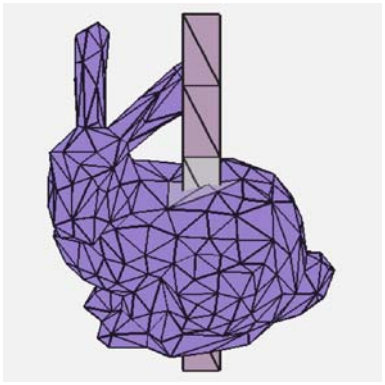
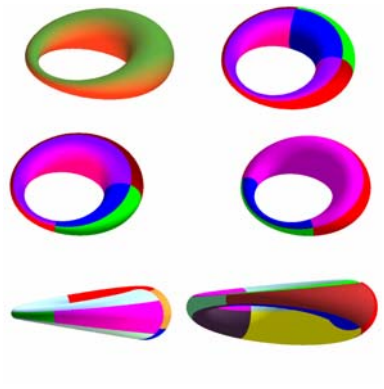
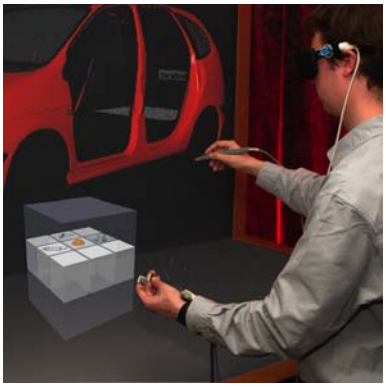
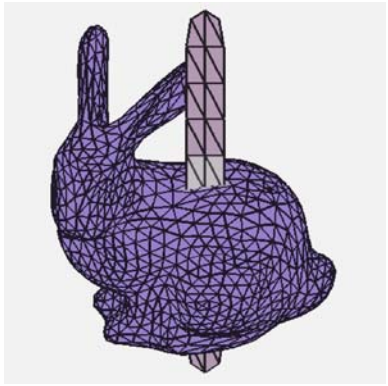
AFIG 2002

Lyon, 9-11 décembre 2002




XV^{èmes} Journées de l'Association Française d'Informatique Graphique

AFIG 2002

Lyon, 9-11 décembre 2002

XV^{èmes} Journées de l'Association Française d'Informatique Graphique

Organisation :

Laboratoire d'Informatique Graphique, Image et Modélisation

Université Claude Bernard Lyon 1

Ministère de la Recherche

Centre National de la Recherche Scientifique

Association Française d'Informatique Graphique

TABLE DES MATIÈRES

Animation

Réalité virtuelle 3

Modélisation 2

Rendu 2

Modélisation 3

LÉGENDES DES IMAGES DE COUVERTURE

En haut à gauche et en bas au centre

Les opérations booléennes entre surfaces sont des problèmes fondamentaux en modélisation géométrique. La première étape lors de ces opérations est de trouver la ou les courbe(s) d'intersection(s) entre les surfaces. Pour les surfaces de subdivision, le nombre de couples de faces à tester augmente très rapidement à chaque itération. Nous proposons donc de construire un graphe mettant en relation les couples de faces à tester afin de réduire considérablement le nombre de tests lors des itérations suivantes. Ce graphe est mis à jour après chaque étape de subdivision.

S. Lanquetin, S. Foufou, H. Kheddouci, M. Neveu. Pages 251–258
Deux algorithmes d'intersection des surfaces de subdivision

En haut au centre

Détail filtré de "l'êglise d'Auvers sur Oise" (Van Gogh). Ce filtre conserve les teintes tout en augmentant la luminosité. L'image parait vive, ensoleillée et plus colorée que l'original.

A. Atencia, J.-J. Bourdin. Pages 111–116
Peinture Virtuelle : modélisation et interaction.

En haut à droite

Conversion des cyclides de Dupin en carreaux de Bézier rationnels biquadriques. Les cyclides de Dupin, inventées en 1822 par le mathématicien français Charles Dupin, permettent d'effectuer des jointures géométriques G^1-continues sans se soucier des problèmes de paramétrisation. Beaucoup de modeleurs utilisent des surfaces de Béziers. Les lignes de courbures des cyclides de Dupin étant des cercles, nous souhaitons les convertir en carreaux de Bézier rationnels biquadriques. Deux approches sont possibles : la méthode de Pratt s'appuyant sur les paramètres de la cyclide de Dupin (une version améliorée permet de convertir toute la cyclide de Dupin) ou la notre, basée sur les propriétés barycentriques des carreaux de Bézier rationnels et les symétries du cercle.

L. Garnier, S. Foufou, M. Neveu. Pages 231–240
Conversion de cyclides de Dupin en carreaux de Bézier rationnels biquadriques.

En bas à gauche

Le "Command and Control Cube" (C^3) est un système de contrôle d'application rapide pour la configuration d'environnement virtuel appelée Plan de Travail Virtuel. Menu 3D évolutif, il s'adapte au niveau d'expérience de l'utilisateur. Inspiré des "Marking Menus" (menus circulaires "pop-up"), le C^3 étend le concept aux trois dimensions de l'espace en exploitant chaque dimension pour la sélection elle-même. L'utilisation de la main non dominante pour contrôler le menu libère la main dominante pour la tache principale.

J. Grosjean, S. Coquillart. Pages 1–12
Contrôle d'application en environnement virtuel : le Command and Control Cube.

En bas à droite

Illustration de la visualisation par surfels des textures volumiques. Cette forêt a été modélisée à l'aide des textures volumiques : chaque arbre correspond à l'instanciation d'un volume de référence sous la forme d'un texel. L'originalité est que le rendu est réalisé par projection de points (surfels), le motif de référence contenant de manière hiérarchique et multi-résolution l'ensemble des surfels représentant la surface de l'arbre.

G. Guennebaud, M. Paulin. Pages 117–128
Visualisation par surfels des textures volumiques.

Contrôle d'application en environnement virtuel : le Command and Control Cube

Jérôme Grosjean & Sabine Coquillart.

i3D, INRIA, Domaine de Voluceau, 78153 Le Chesnay Cedex, France

[Jerome.Grosjean|Sabine.Coquillart]@inria.fr

Résumé : *Les environnements virtuels permettent le développement de puissants outils d'interaction avec des mondes 3D générés par ordinateur. Des configurations comme le Plan de Travail Virtuel (Workbench), sont tout particulièrement adaptées à la manipulation interactive de scènes virtuelles. Un des points forts de ces configurations est la possibilité d'interagir directement et naturellement avec les modèles, l'espace de manipulation et de visualisation étant identiques. Les programmes développés pour ces environnements virtuels nécessitent des interfaces nouvelles, adaptées à leur nature spécifique. Si les premières applications ont principalement réutilisé le concept des menus 2D (métaphore du bureau, fenêtres et menus déroulants), directement portés dans l'espace à trois dimensions, ces solutions ne sont pas optimales. En effet, l'ajout d'une troisième dimension introduit entre autre des contraintes de sélection supplémentaires. Dans ce papier nous proposons un paradigme purement 3D pour le contrôle d'application en environnement virtuel : le "Command and Control Cube" (C^3). Inspiré des "Marking Menus" (menus circulaires "pop-up"), le C^3 étend le concept aux trois dimensions de l'espace en exploitant chaque dimension pour la sélection elle-même. Le C^3 est un menu évolutif, qui s'adapte au niveau d'expérience de l'utilisateur. Il procure un continuum entre un mode débutant avec retour visuel et un mode expert très rapide d'accès, qui peut être utilisé de façon similaire aux "raccourcis claviers". Des tests ont été menés pour évaluer le C^3 dans ses différents modes de fonctionnement. Une version hiérarchique a également été développée pour étendre son usage à un arbre d'options de menus, de taille quelconque.*

Mots-clés : retour d'efforts, architecture logicielle, réalité virtuelle.

1 Introduction

Les configurations d'environnement virtuel (EV) changent notre façon d'interagir avec des objets 3D, de naviguer dans des univers 3D et de contrôler les applications. Certaines de ces configurations comme les casques immersifs, la CAVE^{TM} [7], ou le Plan de Travail Virtuel ou "Workbench" [14, 15, 11] pour n'en citer que quelques-unes sont très prometteuses pour exploiter les potentialités offertes par les mondes 3D générés par ordinateur. Parmi ces configurations, le Plan de Travail Virtuel[1] est une des plus attractives pour la manipulation directe. Elle propose une zone de travail ou les objets 3D peuvent être directement manipulés. Bien que cette configuration soit considérée comme immersive (ou semi-immersive), elle permet aussi aux utilisateurs de ne pas perdre le contact avec le monde réel, leur propre corps ou leurs collaborateurs, ce qui est souvent un des facteurs désorientant des EV.

De nombreuses applications opérationnelles sur stations de travail, comme des modeleurs ou des logiciels de visualisation de données scientifiques, peuvent grandement profiter de cette configuration. Le système de visualisation stéréoscopique, l'enregistrement des mouvements de la tête et l'interaction directe avec la scène virtuelle permettent de simplifier et de rendre plus naturelles des opérations comme déplacer des plans de coupe, sélectionner et manipuler des objets, modéliser en direct, déformer des objets 3D ou encore déplacer des sources de lumière.

Cependant, porter ces applications sur le plan de travail virtuel soulève un problème principal : reformuler pour un environnement 3D les techniques classiques d'interaction pour la manipulation et le contrôle d'application (e.g. changement de mode, envoi de commandes). Cette configuration diffère d'une station de travail sur plusieurs points majeurs. Le système de visualisation du plan de travail virtuel est un affichage stéréoscopique 3D qui permet aux utilisateurs de voir, de tourner autour et de manipuler des objets 3D vus en relief. L'enregistrement des mouvements de la tête permet une superposition de l'espace virtuel et de l'espace physique. En comparaison, sur station de travail, comme avec de nombreuses autres configurations, l'espace de visualisation reste distinct de l'espace de manipulation. Les périphériques d'entrée sont très différents également. Sur station de travail, les utilisateurs communiquent avec les applications à travers des périphériques d'interaction multi-usages : le clavier et la souris. Le nombre de signaux d'entrée est très élevé (102 touches sur un clavier de PC). Les périphériques

[1] Le terme "Plan de Travail Virtuel" est employé ici pour décrire toutes les configurations du type "Workbench" à un ou deux écrans.

d'interaction génériques sur plan de travail virtuel sont souvent conçus pour la manipulation spatiale uniquement (léger et équipable) et ne disposent que de très peu de boutons, e.g. un ou deux seulement pour un stylo repéré dans l'espace.

Une interface de contrôle d'application, basée sur le clavier et la souris, est devenue très populaire sur les stations de travail : l'interface WIMP (Windows, Icons, Menus and Pointing). Ce choix d'interface homme-machine est universellement reconnu aujourd'hui comme un standard de facto pour ces configurations. Pour fournir un système d'accès rapide et léger aux commandes les plus usitées, un système de raccourcis claviers s'est également popularisé. Le clavier est en effet un outil efficace pour déclencher des commandes en une simple combinaison de touches (comme CTRL-S pour sauver un travail) et reste le moyen le plus rapide d'appeler des fonctions sur station de travail.

Dans les environnements virtuels le contrôle d'application est un domaine jeune et il n'existe pas encore de standard confirmé. Dans ce papier nous nous intéressons dans une première partie à la mise en place d'un nouveau système de contrôle d'application appelé "Command & Control Cube" (CCC ou C^3) pour un plan de travail virtuel (cf. Figure 1). Un des objectifs principaux dans le développement de ce nouveau paradigme de contrôle d'application, est de fournir aux utilisateurs un premier système, simple et rapide, pour déclencher un jeu réduit de commandes au sein de l'application. L'idée consiste à mettre en place pour les environnements virtuels, et notamment la configuration appelée plan de travail virtuel, une sorte d'équivalent des raccourcis clavier existant sur les stations de travail.

FIG. 1 – Le C^3 sur le Plan de Travail Virtuel

La première partie de cet article propose une description des travaux antérieurs relatifs aux résultats présentés dans ce papier. La partie suivante décrit le C^3. La troisième partie expose une expérience conduite pour évaluer les performances d'utilisateurs novices avec le C^3 dans ses différents modes de fonctionnement et analyse l'apport de retours d'informations supplémentaires, sous forme sonore et tactile, pour sa manipulation. La dernière partie étend le concept premier du C^3, celui d'un menu rapide d'accès pour un jeu limite de commandes, à une version hiérarchique pouvant contenir un nombre quelconque d'options.

2 Travaux antérieurs

Le contrôle d'application dans les environnements virtuels est un domaine de recherche jeune. Les premières applications pour ces environnements ont immédiatement été confrontées au besoin de développer des interfaces spécifiques pour déclencher des fonctions ou contrôler l'état des variables dans les mondes 3D. Les premières solutions apportées ont suivi une approche pragmatique répondant au cas par cas à des besoins particuliers. A la conférence SIGGRAPH'2001 lors d'un cours sur la conception des interfaces 3D[1], Ernst Kruijff a proposé une classification pour les techniques de contrôle d'application actuelle, influencée par la description de techniques non conventionnelles de MacMillan et al.[23]. Il a notamment divisé les différentes approches en menus graphiques, commandes vocales, interaction gestuelle et outils. Dans la suite, nous exposons une classification différente, s'attachant plus spécifiquement aux menus à interface graphique.

Une première approche naturelle pour les concepteurs d'interface graphique 3D a consisté à porter les interfaces

2D vers le monde 3D. Dans le monde 2D (écran, clavier et souris) l'interface WIMP est maintenant le choix communément accepté pour contrôler les applications. Puisque cette interface est populaire et assez efficace en 2D, porter cette interface a l'avantage de proposer aux utilisateurs un système de contrôle d'application familier donc intuitif.

Deux catégories peuvent être distinguées dans cette approche, selon que les paradigmes 2D ont été directement implantés dans le monde 3D, ou que le concept 2D a été adapté ou étendu au monde 3D.

Enfin, certains concepteurs cherchent à inventer de nouveaux paradigmes d'interaction, fondamentalement et entièrement pensé pour l'interaction 3D en environnements virtuels. Cette troisième voie ne s'inspire pas des paradigmes 2D mais tente au contraire de proposer directement des techniques de contrôle d'application 3D.

2.1 Menus 2D implantés dans le monde 3D

Dans le monde 2D (écran, clavier et souris) l'interface WIMP est maintenant le choix communément accepté pour contrôler les applications. Porter cette interface a l'avantage de proposer aux utilisateurs une interface familière, rassurante et donc intuitive.

Cependant, la sélection d'options de menu dans un environnement 3D est assez différente de la sélection en 2D. Sélectionner un objet parmi un ensemble de fonctions est conceptuellement un choix à une seule dimension. L'ajout de dimensions supplémentaires est inutile et nuit à la simplicité de la tâche. Se déplacer dans un plan pour faire un choix 1D est relativement aisé avec une souris, malgré la seconde dimension. L'ajout de la charge de gestion de la profondeur rend cette tâche beaucoup moins facile et ralentit notablement le processus de sélection.

L'intégration des menus 2D se réalise en implantant les fenêtres ou menus 2D sous forme de plan dans le monde 3D. Leur placement à l'intérieur de l'univers virtuel est différent selon les techniques. Feiner et al. [10] est à ce sujet une source principale d'information sur le placement. Les menus peuvent être placés librement dans le monde virtuel (référence : monde), connectés à un objet virtuel (référence : objet), liés à une partie du corps de l'utilisateur comme la tête ou la main (référence : corps), ou placés en référence à un objet physique comme les bords du plan de travail virtuel (référence : équipement).

Les premiers menus 2D implantés dans le monde 3D se sont contentés de proposer des fenêtres flottant dans l'espace en face de l'utilisateur. La sélection directe d'objets par un outil de pointage non contraint dans l'espace 3D n'est pas optimale [18]. L'utilisation d'un rayon virtuel contrôlé par la main comme outil de pointage peut atténuer cette difficulté [24] en éliminant la contrainte de sélection en profondeur. Cependant la manipulation reste lente, car l'interaction n'est plus directe [21].

Une autre approche basée sur un équipement appelé "virtual tricorder" [26] suggère d'utiliser des menus 2D "ancrés". Le menu 2D est affiché comme un plan, à la position courante de l'équipement tenu dans la main. Le déplacement et la sélection à l'intérieur du menu sont effectués en pressant les boutons de la souris. Une technique de placement similaire propose de matérialiser le plan de sélection en tenant une palette [6, 18] physique transparente et plate dans la main non dominante et de sélectionner les options avec la main dominante. Le menu graphique est affiché directement sur la vitre transparente (à l'aide d'un capteur sur la palette), tirant avantage d'une manipulation basée sur un outil physique offrant un retour d'effort passif et un affichage virtuel. En contrepartie, cette méthode contraint l'utilisateur à garder en permanence un équipement plus ou moins encombrant dans sa main, ou à portée de main, pour pouvoir accéder au menu.

Le placement des menus 2D par rapport à des parties du corps est une dernière approche intéressante, faisant appel au sens proprioceptif[19, 4]. La proprioception[2] procure des avantages importants pour la manipulation directe (contrôle excellent de sa propre main), des indices mnémoniques physiques (retrouver des objets centrés sur son corps) et pour les actions gestuelles (rappel des actions).

2.2 Menus 2D adaptés ou étendus au monde 3D

Les paradigmes du monde 2D peuvent également s'intégrer dans les mondes 3D en s'adaptant à ses contraintes et en évoluant pour offrir une manipulation plus confortable, ou une apparence plus adéquate.

[2]La proprioception est "le sens d'une personne de la position et l'orientation de son corps et de ses membres"

Des menus 2D linéaires ont été portés en 3D sous la forme de menus circulaires ou les options sont disposés sur une bande. Le paradigme de sélection reste à une dimension tout en proposant une apparence 3D et une manipulation basée sur la rotation de la main [17, 25].

Deering à également proposé en 1995 une adaptation 3D et hiérarchique des "pie-menus" [8], des menus radiaux qui apparaissent sur invocation autour de la position courante d'un pointeur. Il utilise la profondeur comme dimension utile pour gérer l'affichage de la hiérarchie des menus.

Le menu C^3 fonctionne notamment comme un menu graphique actionné par la main, adapté du concept 2D des "Marking Menus" [16] pour s'étendre aux trois dimensions de l'espace et en tirer partie.

2.3 Nouveaux paradigmes

Enfin, certains chercheurs se sont penchés sur le développement de nouvelles techniques d'interaction, n'ayant aucun équivalent 2D. En 2001 Bowman et al. [2] ont présenté un nouveau menu appelé TULIP, basé sur le pincement des doigts de la main. Des "Pinch Gloves"^{TM} sont utilisés comme gants pour détecter le pincement des doigts, tandis qu'un casque virtuel permet d'afficher des options de menus comme des étiquettes rectangulaires virtuelles prolongeant les doigts.

2.4 Evaluations

Il y a encore peu de travaux d'évaluation rigoureux portant sur les techniques de contrôle d'application en environnement virtuel. La plupart des études ne fournissent pas de tests formels au-delà des simples premières impressions des utilisateurs. Quelques chercheurs tentent de donner un cadre formel aux évaluations de ces techniques, en tenant compte de la multiplicité des facteurs (types d'utilisateur, équipement d'entrée/sortie, techniques d'interaction, type d'application, contexte de la tâche, etc.). Poupyrev et al. [22] ont proposé un cadre général pour les techniques de manipulation en environnement virtuel. Ils se sont concentrés sur des tâches de manipulations élémentaires, comme la sélection et le positionnement. Cet ensemble de tâches élémentaires est choisi de manière à couvrir la majorité des scénarios de manipulation, afin de pouvoir en tirer des résultats généraux et utiles pour les techniques de manipulation immersive. Bowman et al. ont présenté en 1999 un cadre d'évaluation semblable, basé sur une taxonomie de tâches élémentaires et l'étude de mesures de performances [3]. Le processus d'évaluation du C^3 est construit sur une tâche de sélection simple exposée dans cet article.

3 Description

Le C^3 [13] est construit comme une extension du concept de "Marking Menus" [16] à un univers à trois dimensions. Dans les "Marking Menus", la zone de sélection du menu, c'est à dire l'espace autour du pointeur de la souris est divisé en cadrans identiques, délimitant les différentes portions d'un disque. Pour étendre ce concept en trois dimensions, il faut décider du nombre d'options de menu souhaitées et diviser l'espace 3D en portions homogènes. Selon le nombre considéré, par exemple une dizaine, la division de l'espace environnant en unités élémentaires identiques peut être perturbante et difficile à concevoir. Aussi, une division plus simple a été retenue.

Une première approche consiste à décomposer l'espace autour du pointeur en un ensemble de directions simples (haut, bas, gauche, droite, avant, arrière). Les options sont alors disposées de manière intuitive dans l'espace, et facilement atteignables. Cependant, le nombre d'options disponibles est alors limité à un chiffre maximum de 6. Cela reste assez faible, même pour un système de raccourcis.

Une seconde approche a donc été envisagée, s'appuyant sur cette première idée pour organiser la division de l'espace en un schéma simple à appréhender et retenir, une forme cubique.

L'espace autour du pointeur est vu comme un cube dont il est le centre. Ce grand cube est divisé selon chacune de ses dimensions en trois petits cubes pour un total de 3x3x3 = 27 petits cubes ou *cases*. Le pointeur débute dans le petit cube central de cette structure.

Pour actionner le pointeur, le C^3 propose un périphérique faisant office de souris 3D. Un simple bouton dont la position spatiale est capturée par un capteur électromagnétique est requis pour la manipulation. Le système de

”Pinch Glove” développé par Fakespace [9] ou la ”Ringmouse” [20] sont deux équipements valables. Pour des raisons pratiques, nous n'avons pas utilisé de système commercial mais en avons construit un à partir d'une souris standard et d'un capteur (cf. Figure 2). Les boutons de la souris servent de ”boutons à pincer” et sont placés au sommet de trois doigts de la main à l'aide de bagues en tissu. Ils peuvent être pressés avec le pouce. Le capteur est placé sur le poignet de l'utilisateur. Un seul des trois boutons est effectivement utilisé pour les besoins du C^3.

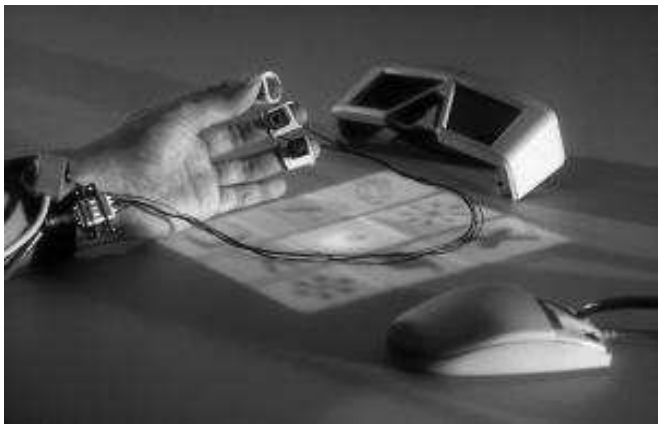


FIG. 2 – Les boutons et le capteur

En position de repos, le menu C^3 est invisible et ne gène pas la vue. Lorsque l'utilisateur presse son pouce contre son index, il pince le bouton et déclenche l'apparition du C^3, c'est à dire la forme cubique, à une distance fixe en avant de sa main, le capteur servant à calculer cette position. Afficher le menu relativement à la main permet de laisser à l'utilisateur le choix du meilleur endroit où faire apparaître le menu. Le C^3 reste à cette position fixe tant que le bouton est maintenu enfoncé.

Un pointeur prenant la forme d'une sphère jaune apparaît également au centre du C^3, dans la case centrale. Tant que l'utilisateur maintient le bouton enfoncé, les mouvements du pointeur reproduisent fidèlement les mouvements de la main avec une correspondance un pour un. Cependant, la sphère ne peut pas quitter le volume cubique du C^3, aussi l'utilisateur peut-il librement effectuer des mouvements larges, la sphère reste contrainte dans le cube.

En déplaçant sa main, l'utilisateur peut placer le pointeur sphérique dans n'importe laquelle des 27 cases du menu C^3. A chaque case est associée une fonction de l'application et une icône placée sur le sommet de chaque petit cube sous forme d'une texture. La visibilité de ces textures est assurée par le fait que le C^3 apparaît à hauteur de la main, donc sous le regard de l'utilisateur en général. Cependant, les étages supérieurs de cases cachent les étages inférieurs à la vue. En conséquence un seul étage de cases est affiché à un instant donné, celui correspondant à la position courante du pointeur sphérique, et ce à tout instant. Pour différencier en un seul coup d'oeil l'étage courant à l'intérieur de la structure cubique, l'enveloppe transparente du cube englobant est également affichée (cf. Figure 3).

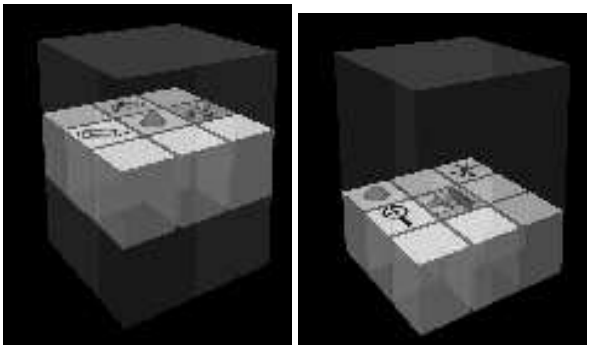


FIG. 3 – Deux positions du C^3

Les 27 cases sont configurables pour accueillir n'importe quelle icône et fonction de l'application, à l'exception de la case centrale, qui est réservée comme case d'annulation pour sortir du menu sans invoquer de commande.

Ainsi, si l'utilisateur presse par inadvertance le bouton, il déclenchera l'option correspondant à la case où débute le pointeur sphérique, la case centrale, et ne déclenchera pas une commande quelconque.

La sélection d'une fonction à l'intérieur du menu est donc réalisée par la simple séquence suivante : pincer les doigts pour faire apparaître le C^3 en face de sa main, maintenir le bouton et bouger la main dans une direction pour placer la sphère dans la case voulue, relâcher le bouton une fois dans cette case pour activer la fonction associée.

La sélection avec le C^3 s'effectue avec la main non-dominante. Cette décision a été prise pour libérer la main dominante, qui est souvent concentrée sur la tâche principale et/ou tient déjà un outil. La main non dominante permet de changer de mode, activer des options, changer l'outil de la main dominante, etc. sans stopper le fil des actions de la main dominante, qui est prioritaire.

Le C^3 propose deux modes de fonctionnement, un mode novice et un mode expert, basés sur la même technique de sélection. Dans les "Marking Menus", un délai est utilisé pour retarder l'affichage du menu circulaire et permettre à un utilisateur expérimenté de réaliser des sélections "en aveugle", sans être perturbé par l'affichage bref du menu. Un utilisateur novice laisse naturellement s'écouler ce délai (de l'ordre du tiers de seconde) et effectue ses sélections en utilisant le retour visuel. Un utilisateur chevronné connaît par coeur la disposition des options dans le menu et peut choisir la direction de son mouvement, avant de voir le menu s'afficher.

Cette méthode a été appliquée dans un premier temps au menu C^3 mais des tests informels avec plusieurs valeurs de délai n'ont pas donné un confort suffisant à la manipulation. Trop court, le délai n'est pas suffisant pour couvrir le temps nécessaire au mouvement de la main ; trop long, le délai devient une gène pour l'utilisateur novice. Une seconde approche a donc été choisie, où le C^3 s'adapte aux besoins de l'utilisateur.

En utilisant les données venant du capteur positionné sur la tête en plus de celles de la main, il est possible de déterminer la direction de regard de l'utilisateur et de vérifier si elle entre dans un cône de sommet la tête et d'axe tête-C^3(cf. Figure 4). Ainsi, lorsque l'utilisateur est concentré sur une tâche en cours et ne se préoccupe pas du menu, celui-ci n'apparaît pas, et lorsqu'il hésite ou décide qu'il a besoin d'une confirmation visuelle, il déplace son regard spontanément et tout naturellement vers l'endroit ou celui-ci apparaît.

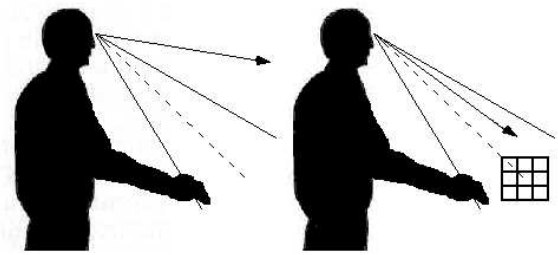


FIG. 4 – Cône de visibilité du C^3

La sélection en aveugle est possible car elle exploite plusieurs indices de positionnement. D'abord elle s'appuie sur la mémorisation d'une direction plutôt que celle d'une distance selon une direction donnée (comme dans les menus déroulants), ce qui a été prouvé plus efficace [5]. Ensuite elle s'appuie sur une organisation simple de l'espace en cases formant une structure cubique autour d'une position de départ. Cette représentation mentale aide l'utilisateur à visualiser le mouvement à réaliser. La réalisation de la sélection elle-même est alors facilitée d'une part par l'alignement des axes du cube avec des directions simples relativement au corps de l'utilisateur (haut, bas, droite, gauche, avant, arrière) lui permettant de bénéficier du sens proprioceptif, d'autre part par l'alignement de ces axes avec ceux de la configuration elle-même, les bords du plan de travail virtuel, ajoutant un indice visuel sous-jacent.

4 Evaluations expérimentales et analyses

L'approche proposée par le C^3 est évaluée [12] ici sur une tâche simple de sélections répétitives d'options à l'intérieur du menu. Du fait de la spécificité spatiale du procédé de sélection du C^3, nous nous intéressons notamment à l'effet de la position des options à l'intérieur du cube sur les performances (vitesse et précision). De plus, nous voulons évaluer les effets des quatre conditions suivantes sur l'interaction. Deux des quatre conditions sont les modes standards de fonctionnement du C^3, i.e. le mode d'apprentissage (avec retour visuel), et le mode expert (sans retour visuel, c'est à dire une manipulation "en aveugle"). Deux conditions additionnelles sont construites

en associant au mode expert des retours sur des canaux sensoriels différents : mode expert augmenté d'un signal sonore (sans retour visuel, avec retour sonore), mode expert augmenté d'un signal tactile (sans retour visuel, avec retour tactile). Ces conditions supplémentaires sont ajoutées dans l'espoir d'améliorer la sélection "en aveugle".

Dans le mode expert augmenté d'un signal sonore, un court bip sonore est émis chaque fois que le pointeur croise la frontière entre deux cases. Le mode expert augmenté d'un signal tactile fonctionne de la même manière en émettant une courte vibration sur le pouce et l'index de l'utilisateur au passage d'une frontière. Un gant "CyberTouch"^{TM} (un gant équipé de vibreurs sur la dernière phalange de chaque doigt et dans la paume) est utilisé pour produire les vibrations. Puisque nous voulons tester le C^3 dans son mode de fonctionnement standard, i.e. sélections par la main non-dominante, le "CyberTouch" utilisé est un gant gauche pour des utilisateurs droitiers.

4.1 Les sujets

La population d'utilisateurs consiste en 23 sujets (18 hommes et 5 femmes). Seulement quatre d'entre eux sont déjà familiers du plan de travail virtuel. Aucun n'a d'expérience préalable avec le menu C^3. Les sujets sont tous droitiers.

4.2 La tâche

La tâche consiste à sélectionner séquentiellement chacune des 26 cases du C^3 dans un ordre de passage aléatoire.

La principale difficulté est de décider de quelle manière informer les utilisateurs de la case à sélectionner, sans leur demander au préalable d'apprendre par coeur un jeu de fonctions et leurs positions dans le cube. L'objectif est de séparer le travail cognitif consistant à reconnaître la case demandée de l'aspect manipulation qui est étudié ici. Il est donc impossible d'utiliser le nom d'une fonction pour désigner une case.

Chaque case étant une combinaison simple de directions spatiales (haut, bas, avant, arrière, gauche, droite) il pourrait être envisagé de décrire oralement la case à sélectionner. Cependant cette solution est rejetée car elle peut favoriser une stratégie de déplacement du pointeur dans le C^3 (mouvement décomposé selon les trois axes) aux dépends des autres possibles (notamment réaliser des mouvements diagonaux).

Pour ne pas influencer le sujet, il est finalement décidé de lui présenter une représentation complète du menu C^3 et de ses 27 cases, et d'illuminer la case à sélectionner. Les sujets savent ainsi immédiatement quelle case ils doivent sélectionner, sans qu'aucune présomption de mouvement ne soit induite. Cette représentation est un objet graphique supplémentaire dans la scène et n'est pas le menu C^3 lui-même, qui reste positionné, lorsqu'il est invoqué, devant la main de l'utilisateur.

Les quatre modes sont testés dans un ordre différent pour chacun des 23 sujets de l'expérience. Pour chacune des conditions expérimentales l'utilisateur porte un gant "CyberTouch", le dispositif nécessaire au C^3 (bouton sur le doigt et capteur), et les lunettes stéréoscopiques. Les 27 cases sont affichées devant les yeux de l'utilisateur. La case courante s'illumine jusqu'a ce qu'une sélection soit réalisée. Le système informe alors l'utilisateur du résultat obtenu en illuminant en vert la case si elle a été correctement atteinte, ou en illuminant en rouge la case sélectionnée par erreur. Apres un délai de deux secondes, la sélection d'une nouvelle case est proposée et le test continue de la même manière.

Toutes les cases sont testées deux fois en tout, à l'exception de la case centrale qui est réservée à l'action d'annulation et n'est pas testée ici.

4.3 Données collectées

Les données enregistrées sont le flot des coordonnées des différents capteurs, la liste ordonnée des cases demandées et les sélections effectivement réalisées par l'utilisateur, et enfin les trois temps de sélection (lorsque la case s'illumine, lorsque l'utilisateur invoque le menu en pressant le bouton pour la première fois, lorsque l'utilisateur termine sa sélection en relâchant le bouton). La vitesse de sélection se déduit directement de ces données.

5 Résultats et discussion

Une analyse de la variance (ANOVA) avec mesure répétée a été réalisée sur les variables précision et vitesse pour les quatre conditions : sans retour (mode aveugle), avec retour visuel, avec retour sonore, avec retour tactile.

La vitesse est significativement affectée par la condition ($F(3,66) = 4.42$, $p < 0.0068$). Des tests supplémentaires (HSD Tukey) indiquent que les utilisateurs réalisent rapidement la sélection dans la condition visuelle (m = 1,0s) comparativement aux autres conditions (aveugle : m = 1,2s ; son : m = 1,3s ; tactile : m = 1,3s). Cette supériorité du mode visuel peut s'expliquer par le fait que les utilisateurs sont tous novices. Ils ont encore besoin du support visuel pour être confiant dans leurs sélections et pour les réussir rapidement.

La précision est aussi significativement affecté par la condition ($F(3,66)= 10,616$, $p < 0.0001$). Le pourcentage moyen de cases correctement sélectionnées est de 92.8% avec retour visuel, 87.0% dans le mode aveugle, 83.5% avec retour sonore et 84.7% avec retour tactile. Un des points d'intérêt de cette étude était de vérifier que les sélections en absence de tout retour sensoriel avec la main non-dominante étaient possibles. Considérant que les utilisateurs découvraient le menu pour la première fois, et pour certains la configuration elle-même, les résultats semblent favorables à cette hypothèse. Les résultats du mode visuel, bien que proche de 100% ne sont pas parfaits. Une explication peut être trouvée dans la contrainte de vitesse qui était demandée aux sujets du test.

Les performances plus faibles des conditions tactiles et sonores peuvent s'expliquer par la nature parfois perturbante de ces signaux. Les sujets ont remarqué qu'ils n'arrivaient pas toujours, pour les mouvement diagonaux à distinguer entre un seul bip (ou vibration) et deux ou trois bips (ou vibrations) successifs. Les sujets qui emploient une stratégie de décomposition de leurs mouvements selon les trois axes n'ont pas rencontré ce problème. Dans la plupart des cas, les sujets utilisent les canaux sensoriels sonores et tactiles pour vérifier la validité de leur mouvement, plus que pour réaliser le mouvement lui-même.

La hauteur d'une case dans le C^3 (position) a aussi un effet significatif ($F(2,44) = 134,058$, $p < 0.0001$). Chaque valeur de précision représente un nombre de sélections correctes réalisées dans un plan (avec un maximum de 18 donc). Le nombre de sélections correctes est plus élevé dans le plan central (m = 17,352), puis dans le plan supérieur (m = 15,587) et enfin dans le plan inférieur (m = 12,859). Les mouvements à hauteur de la main sont les plus aisés. Une explication possible est le moindre effort du bras requis par les mouvements dans le plan central par rapport aux mouvements qui changent de niveau. Il faut noter également que la présence de l'écran horizontal du plan de travail virtuel sous la main de l'utilisateur a pu gêner ou retenir les utilisateurs dans leurs intentions de mouvement vers le bas.

L'interaction double du mode et de la position est également significative ($F(6,132) = 2,684$, $p < 0.0172$). Cette interaction peut être interprétée dans le tableau suivant : Figure 5.

	lower plane									middle plane								upper plane								
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
blind	**32**	39	**35**	**36**	**29**	**35**	40	**34**	45	**46**	**46**	**45**	**45**	43	**44**	**45**	**46**	**44**	40	43	39	**29**	39	41	40	40
visual	**36**	42	43	42	**37**	41	43	43	42	**44**	**46**	**45**	**46**	**46**	**44**	**46**	**44**	42	43	43	39	43	42	**44**	42	42
sound	**29**	**34**	**32**	**29**	**33**	**33**	41	38	**37**	**44**	**44**	**44**	41	**44**	42	41	40	41	41	42	39	**36**	**37**	40	42	**35**
tactil	**29**	**36**	**37**	**32**	**26**	**33**	**37**	38	41	**45**	**46**	**46**	**46**	**45**	42	42	**46**	43	38	40	41	**29**	40	**37**	40	38

FIG. 5 – Position des cellules et précision (cellules gris sombre : 100% correct ; cellules grises : score au-dessus du dernier quartile ; cellules blanches : score en dessous du premier quartile ; cellules gris clair : autres scores)

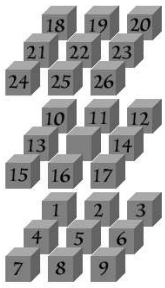


FIG. 6 – Numérotation des cases

Chaque ligne du tableau présente une condition différente, alors que les colonnes sont les cases des trois plans horizontaux, du plus bas au plus haut (cf. Figure 6 pour la numérotation des cases). Les cellules contiennent la somme de toutes les réponses correctes pour les 23 utilisateurs pour une case. La valeur de la médiane est de 41 bonnes réponses. Les cellules avec des résultats en dessous du premier quartile sont sur fond blanc. Celles avec des résultats au-dessus du dernier quartile sont sur fond gris, et celles avec un taux de 100% sont sur fond gris sombre.

La distribution de teinte grise montre quelques tendances générales. Les performances les plus faibles sont obtenues pour des cases dans le plan inférieur. Ce résultat s'explique par la proximité de l'écran du plan de travail virtuel. Dans ce plan, les mouvements qui demandent à l'utilisateur d'allonger son bras reçoivent de moins bons résultats, tandis que ceux moins éloignés du corps obtiennent de meilleurs scores. Les meilleurs résultats sont obtenus dans le plan du milieu, avec des performances légèrement moins bonnes quand la main est trop proche du corps et légèrement meilleures quand elle s'en éloigne pour une distance plus confortable. Il semble d'une manière générale que les utilisateurs réussissent mieux les mouvements demandant un minimum de flexion de leur bras.

6 Version hiérarchique

Disposant d'une solution fonctionnelle pour contrôler un petit jeu de commandes, l'étape suivante consiste à étendre cette solution à une version hiérarchique, sous forme d'arbre de menus et de sous-menus, afin de gérer un nombre arbitrairement grand d'options. Plusieurs approches sont envisageables.

Dans le fonctionnement hiérarchique classique des menus déroulants, la sélection est réalisée par un mouvement continu. Le pointeur suit un parcours allant de menus en sous-menus, franchissant des frontières bien définies entre les menus. Il n'y a pas de zones de transition et les menus ne se chevauchent pas. Le point de sortie du menu père est juxtaposé à l'option choisie, et le point d'entrée dans le menu fils est toujours situé en haut du menu. Il n'y a pas de symétrie. Le segment à traverser pour descendre dans le sous-menu est également souvent très étroit (de la hauteur du texte désignant l'option). Tous ces points concourent à freiner et rendre contraignante la descente et la remontée dans le système hiérarchique.

Un tel système n'est pas forcement souhaitable pour le plan de travail virtuel et le C^3. Notamment, l'idée de réaliser la sélection d'un seul mouvement continu, qui sera ici plus large dans l'espace que le mouvement de la main sur la souris, comporte le risque de faire entrer la main de l'utilisateur en collision avec l'écran horizontal du plan de travail virtuel. Cette situation peut se produire dans le cas d'une succession de sélections de sous-menus positionnés les uns en dessous des autres. D'une manière générale, l'amplitude du mouvement de sélection peut vite prendre des dimensions importantes dans le cas de nombreux sous-menus successifs.

Notre solution s'est donc porté sur l'idée de restreindre la manipulation du C^3 hiérarchique à une zone délimitée de l'espace, qui soit confortable pour l'utilisateur : la position de départ du C^3, qui correspond déjà à son choix preferé. Les menus et sous-menus s'affichent tous à cette position. La géométrie (forme, placement) du C^3 ne change donc pas, mais le contenu sémantique des cubes (fonctions associées et icônes) est adapté à la position du pointeur au sein de l'arbre des sous-menus.

Pour descendre dans un sous-menu associé à une case donnée, il faut sélectionner la case, c'est à dire relâcher le bouton une fois dedans, comme pour une sélection normale. S'il peut être contraignant d'avoir à décliquer pour entrer dans un sous-menu, c'est nécessaire pour distinguer l'entrée dans la case avec le pointeur, du choix de descendre dans le sous-menu. En distinguant ces deux opérations, il est possible de naviguer de cases en cases, sans que les sous-menus remplacent involontairement le contenu courant du C^3 et de n'entrer dans un sous-menu que lorsque l'utilisateur l'a réellement décidé. De plus, cela maintient une cohérence d'interaction vis à vis des cases du C^3.

A l'issue de cette sélection un nouveau contenu du C^3 est affiché, le menu reste visible mais cette fois la main ne presse plus le bouton. La situation ne diffère de la position de repos avant une première invocation du menu que par la visibilité du C^3 en face de l'utilisateur. Dans ce cas de figure, le pointeur sphère est replacé au centre du C^3, fixe et en attente. Tant que l'utilisateur ne réappuie pas sur le bouton, celle-ci ne bouge pas. Dès qu'il appuie, il la rend à nouveau solidaire des mouvements de sa main, comme s'il la saisissait, et la sélection suit toujours le même principe.

La case centrale, qui est une case d'annulation dans le premier menu rencontré, devient dans les sous-menus une case de "remontée" vers le menu père. Si l'action de remontée est choisie, le mécanisme est similaire à la descente dans un sous-menu. Le C^3 reste visible, son contenu change pour correspondre aux 26 options du menu père, et le

pointeur sphère se replace au centre du C^3, en attente d'être saisi.

Pour ajouter un indice visuel supplémentaire permettant de se repérer dans la hiérarchie, chaque fois que l'utilisateur descend dans un sous-menu, la case choisie pour descendre est réaffichée dans une pile verticale au-dessus de la représentation visuelle courante du C^3. Au fur et à mesure de la descente dans les sous-menus, la liste de toutes les cases empruntées pour arriver au cube courant apparaît donc en rappel au-dessus du C^3 (cf. Figure 7).

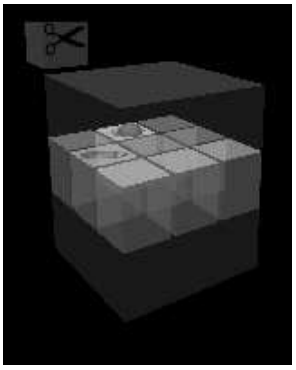


FIG. 7 – Le C^3 hiérarchique

Pour résumer, descendre dans la hiérarchie des sous-menus consiste à enchaîner des sélections simples, selon le même paradigme d'interaction que le C^3 non hiérarchique. Remonter dans la hiérarchie ou quitter le menu sans sélectionner est très simple également, même lorsque le pointeur est descendu profondément dans l'arbre des sous-menus. Il suffit de presser le bouton sans bouger la main. En effet, la sphère est toujours replacée en attente au centre des cubes après chaque action de montée ou de descente. Par conséquent cliquer sans bouger fait choisir la case centrale, i.e. remonter de noeuds en noeuds dans l'arbre jusqu'au noeud racine, puis sortir du menu.

7 Conclusion et travaux futurs

Le C^3 a été développé dans l'intention initiale de fournir un système de menu rapide d'accès sur un petit jeu de commandes usuelles, pour la configuration de réalité virtuelle appelée plan de travail virtuel. Ce système très similaire dans l'esprit aux raccourcis clavier des stations de travail, s'inspire du concept des "Marking Menus" et les étend à trois dimensions en profitant pleinement des trois dimensions de l'espace comme dimensions de sélection.

Le C^3 permet de libérer l'attention de l'utilisateur pour la tâche principale, par son utilisation en main non-dominante. Un soucis tout particulier a été porté aux considérations de placement et d'occupation de l'espace. Le C^3 n'est présent à la vue que sur invocation et il est invocable n'importe où pour ne pas gêner la vue d'une scène. Les mouvements de sélection peuvent être rapides et libres. Ils sont larges dans leur tolérance en amplitude par l'utilisation de parois bloquantes pour le pointeur. L'utilisation de mouvements relatifs à une position de départ permet d'éviter tout problème de calibrage des capteurs.

Des évaluations ont été menées pour tester la viabilité d'une utilisation du C^3 en mode expert "aveugle" et par la main non-dominante. L'ajout de retours sensoriels supplémentaires, sonore et tactile, ne s'est pas révélé satisfaisant dans ce cadre. Les premiers résultats sur des utilisateurs novices sont encourageants et montre un taux de réussite élevé même dans des conditions "aveugle" à priori difficile. Des tests supplémentaires devront être menés pour étudier l'effet de l'apprentissage sur les utilisateurs.

Suite à ces résultats une version étendue du C^3 a été réalisée, pour obtenir un système de contrôle d'application complet par l'ajout du concept de hiérarchie. Le C^3 hiérarchique peut désormais gérer un arbre arbitrairement grand de menus et sous-menus. Des évaluations du C^3 hiérarchique peuvent faire l'objet de travaux futurs, notamment en le comparant aux autres solutions existantes sur ces configurations d'environnement virtuel.

Le C^3 est fonctionnel et actuellement utilisé dans plusieurs applications du plan de travail virtuel, dont par exemple une application de visualisation d'un habitacle de voiture avec une gamme simple d'outils d'exploration, une application de visualisation de modèles fractals ou encore une application de modélisation à partir d'objets mous.

Le C^3 a été développé pour le plan de travail virtuel mais pourrait être étendu simplement à d'autres configurations similaires de réalité virtuelle. Il serait ainsi intéressant d'étudier comment celui-ci s'intègre dans d'autres configurations comme la CAVE ou les casques, qui ne disposent pas d'un référentiel physique pour aligner les axes du cube.

Références

[1] D. A. Bowman, J. LaViola, M. Mine, and I. Poupyrev. Advanced topics in 3d user interface design. In *Course Notes - SIGGRAPH 2001*, 2001.

[2] D.A. Bowman and C.A. Wingrave. Design and evaluation of menu systems for immersive virtual environments. In *Proc. IEEE VR'2001*, 2001.

[3] Doug Bowman, Donald Johnson, and Larry F. Hodges. Testhed evaluation of VE interaction techniques. In Mel Slater, editor, *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST-99)*, pages 26–33, N.Y., December 20–22 2000. ACM Press.

[4] J. Butterworth, A. Davidson, S. Hench, and T. M. Olano. 3dm : A three dimensional modeler using a head-mounted display. In *Proc. 1992 Symposium on Interactive 3D Graphics*, pages 135–138, 1992.

[5] Jack Callahan, Don Hopkins, Mark Weiser, and Ben Shneiderman. An empirical comparison of pie vs. linear menus. In *Proceedings of ACM CHI'88 Conference on Human Factors in Computing Systems*, Menus, pages 95–100, 1988.

[6] S. Coquillart and G. Wesche. The virtual palette and the virtual remote control panel : A device and an interaction paradigm for projection-based virtual environments. In *IEEE VR'99*, 1999.

[7] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-screen projection-based virtual reality : The design and implementation of the CAVE. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 135–142, August 1993.

[8] Michael F. Deering. HoloSketch : a virtual reality sketching/animation tool. *ACM Transactions on Computer-Human Interaction*, 2(3) :220–238, September 1995.

[9] Fakespace. http ://www.fakespacelabs.com/.

[10] S. Feiner, B. Macintyre, and D. Seligmann. Knowlege-based augmented reality. *Communications of the ACM*, 36(7) :53–61, 1993.

[11] B. Frölich, B. Kirsch, W. Krüger, and G. Wesche. Further development of responsive workbench. In M. Göbel, editor, *Virtual Environments '95*, Eurographics, pages 237–246. Springer-Verlag Wien New York, 1995.

[12] J. Grosjean, J.-M. Burkhardt, S. Coquillart, and P.Richard. Evaluation of the command and control cube. In *ICMI'2002*, Pittsburgh, US, October 2002.

[13] J. Grosjean and S. Coquillart. Command & control cube : a shortcut paradigm for virtual environments. In *IPT-EGVE'2001*, Stuttgart, Germany, May 2001.

[14] W. Krüger and B. Fröhlich. The responsive workbench. *IEEE Computer Graphics and Applications*, pages 12–15, May 1994.

[15] Wolfgang Kruger, Christian-A. Bohn, Bernd Frohlich, Heinrich Schuth, Wolfgang Strauss, and Gerold Wesche. The responsive workbench : A virtual work environment. *Computer*, 28(7) :42–48, July 1995.

[16] Gordon Kurtenbach and William Buxton. User learning and performance with marking menus. In Beth Adelson, Susan Dumais, and Judith Olson, editors, *Proceedings of the Conference on Human Factors in Computing Systems*, pages 258–264, New York, NY, USA, April 1994. ACM Press.

[17] Jiandong Liang and Mark Green. JDCAD : A highly interactive 3D modeling system. *Computers and Graphics*, 18(4) :499–506, July–August 1994.

[18] R. Lindeman, J. Sibert, and J. Hahn. Hand-held windows : Towards effective 2d interaction in immersive virtual environments. In *IEEE VR'99*, 1999.

[19] Mark R. Mine, Frederick P. Brooks, Jr., and Carlo H. Séquin. Moving objects in space : Exploiting proprioception in virtual-environment interaction. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 19–26. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[20] Ring Mouse. http ://www.worklink.net/ringmouse.html.

[21] A. Paljic, J.-M. Burkhardt, and S. Coquillart. A study of distance of manipulation on the responsive workbench. In *IPT'2002 Symposium, Orlando, US*, 2002.

[22] I. Poupyrev, S. Weghorst, M. Billinghurst, and T. Ichikawa. A framework and testbed for studying manipulation techniques for immersive VR. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST-97)*, pages 21–28, New York, September 15–17 1997. ACM Press.

[23] McMillan G. Eggelston R. and Anderson T. Nonconventional controls. In *Handbook of human factors and ergonomics*, pages 729–771, 1997.

[24] Ron van Teylingen, William Ribarsky, and Charles van der Mast. Virtual data visualizer. *IEEE Transactions on Visualization and Computer Graphics*, 3(1) :65–74, January – March 1997. ISSN 1077-2626.

[25] G. Wesche and M. Droske. Conceptual free-form styling on the responsive workbench. In *VRST 2000*, 2000.

[26] Matthias M. Wloka and Eliot Greenfield. The virtual tricorder : A uniform interface for virtual reality. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, Virtual and Augmented Realities, pages 39–40, 1995. TechNote.

Visualisation de modèles CAO dans une application de réalité virtuelle immersive

D. Paillot[1], F. Merienne[1], M. Neveu[2], J.P. Frachet[1].

(1) : Institut Image-ENSAM
2, rue Thomas DUMOREY – BP 123
71321 Chalon sur Saône
33 (0)3 85 90 98 60 / 33 (0)3 85 90 98 61
E-mail :
{paillot,merienne,frachet}@cluny.ensam.fr

(2) : Université de Bourgogne – LE2I
UFR Sciences et Techniques – Bât. Mirande
Aile de l'Ingénieur – BP 47870
21078 Dijon Cedex
33 (0)3 80 39 58 45 / 33 (0)3 80 39 58 46
E-mail : mneveu@u-bourgogne.fr

Résumé : *Les applications de réalité virtuelle se développent de plus en plus. La quasi totalité des secteurs industriels peuvent aujourd'hui tirer profit de la réalité virtuelle. Les attentes des utilisateurs sont en augmentation, plus d'interactivité avec la scène graphique, meilleur rendu, meilleure fluidité de l'application,… Bien que les performances des calculateurs continuent d'augmenter, il est toujours nécessaire d'optimiser les scènes 3D. Le principal et premier problème pour optimiser les scènes reste les modèles qui seront utilisés dans l'application. L'origine de ces modèles, leur qualité et leur conversion en un format compatible avec ceux utilisés par les applications de réalité virtuelle restent les problématiques majeures. Nos travaux ont pour objectif de visualiser en temps réel des modèles issus de la CAO dans une salle immersive de type CAVE™. Ces travaux couvrent deux axes de recherche principaux. Le premier concerne la mise en conformité, la simplification des modèles surfaciques en modèles triangulés. Le deuxième axe de recherche se consacre à la visualisation et à son optimisation pour l'utilisation d'un périphérique particulier : salle immersive.*

Mots-clés : Triangulation, simplification, visualisation réaliste, temps réel, salle immersive.

1. Introduction.

Les cartes graphiques actuelles ne savent gérer que des polygones, principalement des triangles. Pour cette raison les applications de réalité virtuelle utilisent des modèles polygonaux plutôt que des modèles surfaciques. Dans la mesure où les modèles doivent être créés pour une application de réalité virtuelle spécifique, ils sont parfois générés directement dans ce format. Mais bien souvent, il est souhaitable d'utiliser des données issues d'autres corps de métiers (design, conception assistée par ordinateur, …). Cette réutilisation évite d'avoir à redessiner les modèles et de bénéficier à tous moments de la dernière mise à jour des modèles. Cependant, cela impose un travail de préparation, de réparation, d'optimisation pour chacun de ces modèles. Cette mise en conformité est nécessaire pour rendre les modèles utilisables par les applications et de les optimiser pour une application de visualisation. Bien que les capacités des calculateurs se soient accrues, il est toujours incontournable de réduire le nombre de polygones d'une scène de réalité virtuelle. Les applications développées sont principalement des applications de revue de projet. La finalité n'est pas d'étudier la fonctionnalité ou la définition structurelle, mais de visualiser les pièces dans leur environnement. La visualisation sera réalisée dans une salle immersive de type MoVE™ (Modular Virtual Environment). Ce périphérique particulier offre une place unique à l'utilisateur. Il est immergé au cœur de la scène 3D, alors que tous les autres périphériques existants placent l'utilisateur en dehors de la scène. L'immersion est totale et réelle. Cette configuration amène a repenser les applications de visualisation. Actuellement, les logiciels offrent des solutions pour les périphériques classiques. Leur fonctionnement pourrait être modifié pour s'implémenter correctement et être encore plus performant dans la gestion du nombre de polygones et de la qualité visuelle offerte. En particulier, la notion de vision périphérique semble être utilisable pour les salles immersives. Comme l'utilisateur est équipé d'un capteur de position, l'application connaît à tout moment ce que regarde l'utilisateur. L'application peut ainsi en temps réel dégrader les modèles situés à la périphérie du champ visuel et affiner ceux situés dans la partie centrale de la vision.

Le chapitre 2 de l'article présentera les enjeux, les problèmes rencontrés et les solutions développées pour apporter une réponse au transfert des modèles. Le troisième chapitre abordera la visualisation et

particulièrement la vision périphérique, ses apports éventuels ainsi que sa mise en œuvre. La conclusion et les perspectives feront l'objet de la quatrième partie.

2. Conversion de modèles surfaciques en modèles polygonaux.

2.1. Problématique.

L'utilisation des modèles issus de la CAO offre l'avantage de travailler avec des modèles constamment en phase avec le cycle de vie du produit. Cependant cette réutilisation n'est pas directe. Il est impossible de visualiser les modèles issus de la CAO dans une application de réalité virtuelle sans les convertir. Les étapes de conversion sont actuellement bien connues :

- Triangulation,
- Cohérence,
- Réduction du nombre de polygones,
- Suppression des éléments non visibles.

Ces quatre étapes sont présentées dans les sous-chapitres suivants.

2.2. Triangulation.

La trinagulation consiste à générer un modèle polygonal à partir d'un modèle surfacique. Pour ce faire, on utilise un critère appelé erreur cordale (cf. Figure 1). Cette erreur porte souvent le nom de SAG dans les logiciels. Elle correspond à la distance maximale autorisée entre la facette générée et la courbe.

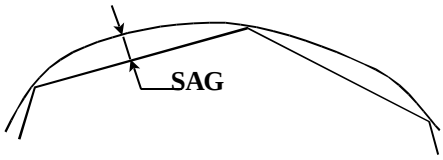
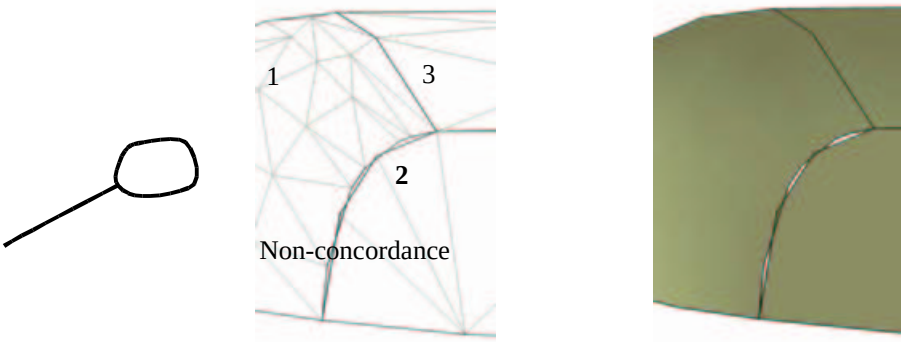


Figure 1 : Erreur cordale.

Cette erreur conditionne le nombre de polygones du modèle final. Plus cette erreur sera faible, plus le modèle polygonal sera fidèle au modèle initial mais plus le nombre de polygone risque d'être élevé. Il faut trouver la valeur optimum entre nombre de polygones et finesse géométrique. Le nombre de polygones peut être sensiblement élevé puisqu'une étape de simplification aura lieu ultérieurement.

2.3. Mise en cohérence.

Lors de la visualisation, il est fréquent de voir apparaître des trous ou discontinuités dans les modèles (cf. Figure 2(b)). La mise en cohérence consiste à rendre le modèle polygonal conforme pour un visualisation correcte du modèle. Ces discontinuités sont dues à la non concordance des sommets entre eux (cf. Figure 2 (a)). Cette non-concordance crée des discontinuités dans le maillage apparaissent comme des trous.

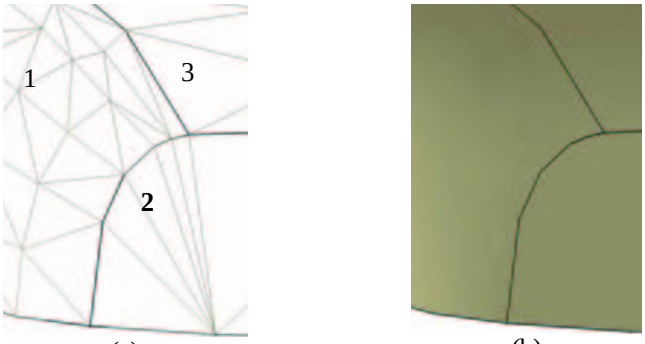


(a) (b)

Figure 2 : Problème de cohérence des modèles triangulés.

Ce problème de discontinué peut être résolu de deux manières différentes.

La première solution consiste à travailler sur le modèle surfacique. Les modèles surfaciques sont un assemblage de carreaux. La Figure 2(a) montre un modèle surfacique avec ses trois carreaux. Ces carreaux ont en commun des frontières, grâce à ses frontières le modèle surfacique possède une unité et permet d'obtenir un rendu de bonne qualité sans trou ou discontinuité.
Ces carreaux sont triangulés un par un. La connaissance des frontières communes est alors perdue. Chaque carreau possède sa propre triangulation. Cela aura pour effet de ne pas avoir de cohérence au niveau du modèle final. Les Figure 2(a) et (b) illustrent le problème de continuité de triangulation le long des frontières. Les carreaux 1 et 2 ont subi chacun leur propre triangulation. Comme la notion de frontière est perdue, la frontière commune entre le carreau 1 et le carreau 2 sera triangulé de deux manières différentes. Les sommets de la frontière appartenant au carreau 1 n'ont aucune raison a priori de concorder avec les sommets de la frontière appartenant au carreau 2. La conséquence de ce manque de cohésion est une apparition de trous et de discontinuités lors du rendu des pièces (cf. Figure 2(b)).
La solution à ce niveau est de coudre les différents carreaux surfaciques entre eux. Cette couture permet de rassembler sous une seule entité les différents carreaux surfaciques initiaux. Cette étape peut se faire dans le logiciel de conception du modèle. A titre d'exemple, une simple pièce plastique d'intérieur de véhicule peut atteindre 200 carreaux. Il n'est pas possible de tous les sélectionner et de les coudre en une seule opération. Il faut bien souvent traiter le modèle morceau par morceau. Cette tâche devient vite longue et fastidieuse. Le résultat de la triangulation présenté sur la Figure 3 montre un exemple de concordance souhaitée pour les sommets ainsi que la qualité de la continuité du rendu.



(a) (b)

Figure 3 : Modèle cousu.

La deuxième solution consiste à intervenir après la triangulation. A partir d'un critère de distance, les sommets et arêtes dont la distance qui les sépare est inférieure à ce critère seront fusionnés. Cette action est plus rapide et moins contraignante à réaliser que la première solution. Cependant, à ce niveau il n'existe plus de référence. Le modèle surfacique initial n'est plus connu. Il est possible d'avoir des dérives non souhaitées sur la géométrie ou topologie du modèle.
Quelle que soit la méthode utilisée, la qualité du modèle triangulé sera primordiale pour la suite. En particulier, le résultat de la simplification sera fonction de la cohérence du modèle initial.

2.4. Réduction du nombre de polygones.

Face à ce problème de réduction du nombre de polygones, de nombreux algorithmes ont été développés. Chacun d'entre eux possède sa spécificité. Plusieurs techniques de génération de niveaux de détails ont été définies par Heckbert et Garland [HG94]. Il est possible de définir trois catégories d'algorithmes pour la création de niveaux de détails. La première catégorie regroupe les algorithmes qui font appel à des simplifications polygonales en générant de nouveaux modèles avec des niveaux de détails différents. La seconde concerne les algorithmes qui remplacent les objets par des textures. Ces algorithmes ne nous seront pas utiles ici, puisque nous nous intéressons à l'immersion 3D. Les textures ne sont pas très appropriées pour la vision en relief. La troisième catégorie représente les algorithmes qui agissent sur la scène et qui remplacent un ensemble d'objets par une représentation plus simple.

Nous ne nous intéressons dans un premier temps qu'à la première catégorie. Dans cette catégorie plusieurs approches existent :

1. Subdivision adaptative (*Adaptative Subdivision*) : cette méthode consiste à construire un modèle de base très simple qui sera ensuite subdivisé. L'algorithme s'arrête lorsque l'écart entre le modèle initial et le modèle subdivisé est inférieur au critère spécifié par l'utilisateur. Cette méthode est peu utilisée à cause de la complexité à définir le modèle de base.
2. Réduction géométrique (*Geometrical Removal*) : cette méthode s'appuie sur le modèle d'origine et supprime des faces ou des sommets. L'algorithme s'arrête lorsque le degré de satisfaction imposé par l'utilisateur est atteint. La majorité des algorithmes respecte la topologie des modèles initiaux. La plupart des algorithmes récents fonctionnent avec cette méthode.
3. Echantillonnage (*Sampling*) : cette méthode effectue un échantillonnage de la géométrie du modèle initial (en choisissant arbitrairement des sommets ou en l'englobant dans une grille tridimensionnelle et en échantillonnant chaque boîte de la grille). L'algorithme essaye de créer un modèle simplifié qui soit proche des données échantillonnées. Le contrôle se fait par le nombre de points ou la taille de la grille.

De manière générale, parmi les méthodes de décimation de polyèdres bon nombre d'entre elles sont contrôlées par des critères d'arrêts directement liés au nombre de sommets désirés par l'utilisateur [SZL92], [HG94], [RO96]. Ce critère n'est dans notre cas d'étude pas satisfaisant. En effet, l'écart géométrique généré doit être connu et même maîtrisé. D'autres méthodes permettent un contrôle de la déviation entre le maillage résultant et le maillage initial [DFMP96], [G96], [PS97]. Ces techniques sont d'utilisation plus délicate ou bien concernent des modèles particuliers (modèles de terrain). Les paramètres de contrôle (distance et angle) ne sont pas faciles à définir par les utilisateurs.
Les principes de simplification de polyèdres reposent sur l'emploi d'opérateurs de fusion d'arêtes [H96] ou bien de remaillage d'un contour après suppression de sommet [PS97], [V97a]. Les opérateurs de fusion d'arêtes n'engendrent qu'un nombre limité de configurations de maillage du contour du nœud supprimé ce qui réduit la qualité de la restitution de la forme du polyèdre simplifié.

2.4.1. Analyse multirésolution de maillage (Eck [EDRDHLS95]).

La méthode consiste à reproduire une représentation à résolutions multiples d'un maillage polygonal quelconque. Les représentations à résolutions multiples sont basées sur une technique introduite par De Rose et al. [DRLW93] appelée analyse multirésolution. Elles consistent en un maillage de base simple et une séquence de corrections locales sous formes de coefficients d'ondelettes.

2.4.2. Maillages progressifs (Hoppe [H96]).

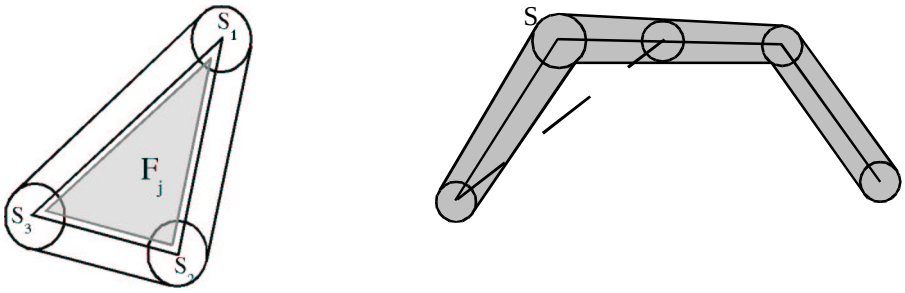
Une représentation en maillage progressif d'un maillage arbitraire M' contient un maillage moins détaillé M^0 et une séquence de *n* enregistrements de détails qui précise comment reconstruire de manière incrémentale M' à partir de M^0 : M' = M^n. Un des points très important de cette technique est que le degré de simplification est contrôlé par le choix du nombre de facettes désiré. De plus, il est possible de générer tous les niveaux de détails en un seul traitement.

2.4.3. Approximation de polyèdre triangulaires (Ronfard [RR96]).

Cet algorithme, présenté à Eurographics'96, est issu des techniques de segmentation d'images. Il utilise un *opérateur de fusion de régions* sur des modèles triangulés. Cet opérateur retire une arête en fusionnant à la fois ses extrémités et plusieurs triangles. Cette méthode est capable de supprimer les détails les plus petits (et ce uniquement à base d'un raisonnement géométrique), elle produit tous les niveaux de détails en une seule passe. En revanche, l'implémentation actuelle est un peu lente.

2.4.4. Simplification multicritères (Véron [V97b]).

Les principes de simplification de polyèdres retenus sont basés sur un critère de distance entre le polyèdre initial et le polyèdre simplifié.
L'approche utilisée est une suppression répétée d'un nœud du polyèdre suivie par un remaillage du contour en fonction des courbures locales du polyèdre.
L'algorithme fonctionne à partir de zones d'erreurs. Ces zones sont géométriquement représentées par des sphères centrées sur les nœuds du polyèdre initial et constituent une enveloppe entourant le polyèdre initial. La Figure 4 représente dans le plan le concept de zones d'erreurs. La Figure 4(a) illustre le placement des sphères par rapport aux sommets ainsi que l'enveloppe d'erreur générée. La Figure 4(b) illustre le principe de suppression, ici, le sommet S ne pourra pas être supprimé car la nouvelle arête est placée en dehors de l'enveloppe.



(a) (b)

Figure 4 : Définition des zones d'erreurs.

La dimension des zones d'erreur (rayon des sphères) est le seul paramètre fourni par l'utilisateur pour effectuer la simplification.

2.5. Suppression des éléments non visibles.

Cette dernière étape dans le processus de conversion est une étape d'optimisation. L'objectif est toujours de limiter le nombre de polygones aux polygones utiles à l'application considérée. Dans notre application, il s'agit de visualisation. Les pièces utilisées sont des pièces issues des bureaux d'études, elles comportent toutes les informations utiles à leur fabrication et à la fonction ultime. Beaucoup d'éléments utiles pour le fonctionnement ne le sont pas pour la visualisation. Par exemple, les nervures et raidisseurs de pièces plastiques, les trous de fonctionnement sont des éléments qui pour leur définition géométrique vont utiliser des polygones mais qui n'ont pas d'apport réel dans une application de visualisation. Afin de diminuer le nombre de polygones dans la scène, il semble nécessaire de les supprimer (cf. Figure 5).

Avec nervures Sans nervure

Figure 5 : Suppression des parties non visibles.

2.6. Mise en œuvre.

La chaîne complète de conversion d'un modèle surfacique en un modèle polygonal peut être décrite comme sur la Figure 6. Cette chaîne représente la première solution, la mise en cohérence est effectuée au niveau du modèle surfacique.

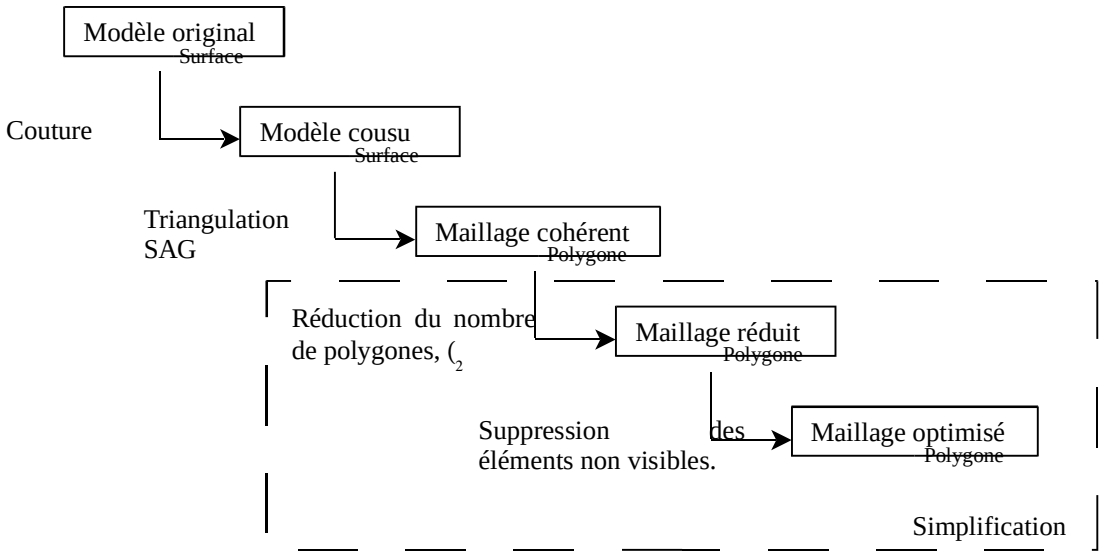
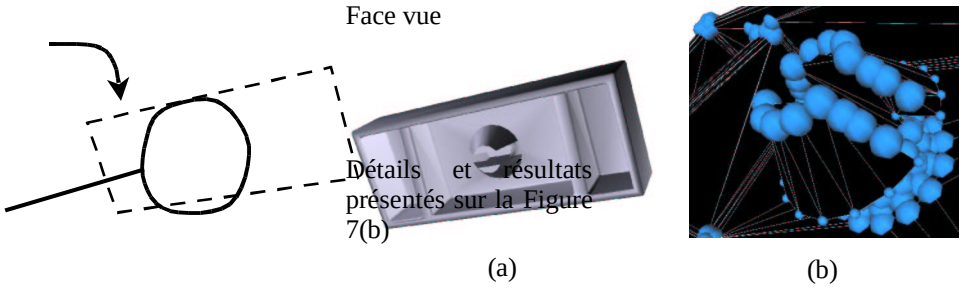


Figure 6 : Chaîne de conversion modèle surfacique / modèle polygonal.

Les étapes de couture et de triangulation sont réalisées par le modeleur CAO. Pour la simplification, nous avons fait le choix de travailler en collaboration avec le laboratoire 3S (Sols Solides et Structures) dirigé par Jean-Claude LEON à Grenoble et la société Géo-Numéric sur la base du logiciel *Simpoly*®. Ce logiciel est le fruit de plusieurs thèses réalisées au sein du laboratoire 3S. Nous avons effectué ce choix suite à l'étude des algorithmes existants. Il s'est avéré qu'ils ne répondaient pas entièrement à notre demande. Principalement parce que nous souhaitions pouvoir piloter la simplification non uniformément sur tout le modèle. Les travaux de P. VERON [V97b-VL98] présentent une approche différente de la simplification. Leur algorithme permet de simplifier de manière non uniforme.

L'objectif de leur outil de simplification est de générer des modèles pour de la simulation de calculs. Ils avaient besoin de simplifier différemment les zones d'un modèle en fonction de leur implication dans les calculs de résistance. Par ailleurs, la majeure partie des algorithmes de simplification existants utilise comme critère de pilotage un pourcentage de réduction. L'utilisateur indique s'il souhaite une réduction de X%, l'algorithme respectera ce critère et générera un modèle. L'écart géométrique, l'erreur commise, sera une conséquence du taux de réduction. *Simpoly*® propose de piloter la simplification avec l'erreur géométrique maximale tolérée par l'utilisateur. Le taux de réduction devient une conséquence du calcul. Dans notre cas, il est impératif de contrôler finement cet écart géométrique. Les modèles ne doivent pas avoir un écart géométrique supérieur au millimètre pour que l'application soit crédible. La simplification non uniforme est pilotée par une carte de taille. Cette carte de taille est une répartition de sphères dans l'espace qui permet d'indiquer pour chaque sommet sa latitude à disparaître ou à se déplacer (cf. 2.4). Actuellement, les modules existants génèrent des cartes de tailles dans le l'objectif d'effectuer des calculs de résistance.

Nous développons les modules qui permettent de piloter l'algorithme de simplification dans l'objectif de générer des modèles pour des applications de visualisation. Notre objectif est de conserver les détails visibles du modèle et de simplifier les zones non visibles. La simplification des zones non visibles peut aller jusqu'à la modification géométrique de certaines parties. Le problème est d'identifier ces différentes zones. Le premier algorithme [PMFNTF02] implémenté travaille à partir de l'angle solide entre les normales des facettes et la direction de vue du modèle. Pour un angle solide compris entre – /2 et + /2 la facette sera déclarée comme visible. La Figure 7 illustre cet algorithme. La pièce traitée est vue par la face dessinée en pointillée sur la Figure 7(a).



(a) (b)

Figure 7 : Orientation des normales.

Cet algorithme n'est pas assez robuste. Le clip de fixation n'a pas été complètement détecté (cf. Figure 7(b)). Nous nous orientons actuellement sur des algorithmes de partitionnement de l'espace ou de profondeur. Nous recherchons à connaître les faces cachées par les autres. Avec cette connaissance, il sera aisé de pouvoir supprimer les sommets des faces non visibles. La priorité dans le traitement des sommets sera faite par l'importance de la concavité entre les faces. Plus la concavité sera grande, plus il semble difficile de voir ce qui se passe à l'intérieur de cette concavité.
Cette préparation des modèles est l'étape incontournable avant la visualisation dans une application de réalité virtuelle. Tout ce travail a pour objectif de pouvoir donner au calculateur le temps de traiter la totalité des polygones contenus dans la scène dans le temps qui lui est imparti. Les applications sont en stéréo et en temps réel. Il faut que le calculateur puisse calculer entre 30 et 50 images par seconde pour un confort maximal.

3. Visualisation et vision.

3.1. Matériel utilisé.

Les applications sont portées dans la salle immersive MoVE™. Cette salle immersive de type CAVE™ est reconfigurable et possède 4 faces (deux faces latérales, la face avant et le sol). Comme le montre la Figure 8, l'utilisateur est physiquement immergé dans la scène 3D.

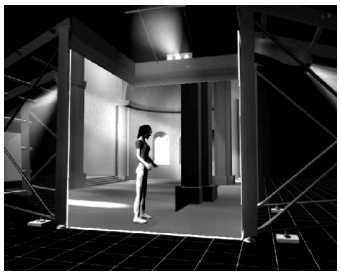


Figure 8 : MoVE.

Habituellement l'utilisateur perçoit les images de l'application sur un écran ou dans un casque, dans le cas d'un écran l'utilisateur a toujours dans son champ visuel des éléments réels (bureaux, meubles, …), dans le cas d'un casque la totalité du champ visuel n'est pas couvert. Dans la salle immersive, l'utilisateur est physiquement au cœur de la scène 3D.

3.2. Système visuel humain.

L'œil humain n'est pas sensible de la même manière à tout son environnement. Le récepteur de l'œil, la rétine, est composé d'une zone, la fovéa, représentant 1% de la surface de la rétine. La densité de cônes dans cette zone est de 180000 par mm² alors qu'elle est de 5000 sur le reste de la rétine. Les cônes servent principalement dans la vue diurne et apportent des informations précises sur la position et la couleur des objets. Les bâtonnets quant à eux sont utilisés pour une vision nocturne ou de faible luminosité. On s'aperçoit donc que la zone où l'information est maximale est faible. Cette faiblesse est compensée par le mouvement rapide des yeux.
L'acuité visuelle est la capacité pour un œil humain à distinguer les détails (i.e. résolution). Elle est évaluée à une minute d'arc. Cette acuité visuelle a été établie pour la ligne de vue qui correspond à la fovéa.

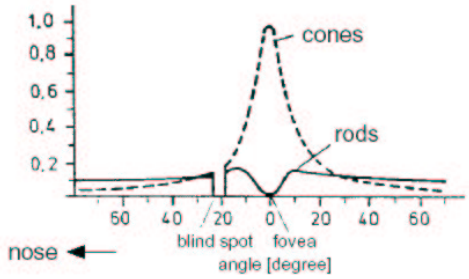


Figure 9 : Résolution spatiale des cônes et bâtonnets en fonction

du rayon angulaire par rapport à la fovéa (source [HVPO]).

La Figure 9 montre que l'acuité visuelle maximale (Av=1) ne représente que 1% du champ de vision, c'et à dire une surface d'environ 2° de rayon angulaire. L'acuité visuelle diminue très vite en fonction de l'écartement de la fovéa, elle est divisée par 2 à 2° du centre de la fovéa, par 4 à 5° et par 10 pour des angles variant de 10 à 20° [SUPELEC-YRR94]. Ces constatations confirment qu'une simplification plus importante des modèles situés à la périphérie du champ visuel ne modifiera pas a priori la perception visuelle de la scène globale.

3.3. Mise en œuvre.

Le fait de simplifier d'avantage les modèles situés sur la périphérie du champ visuel permettra de diminuer le nombre de polygones à calculer pour chaque image et ainsi accroître les performances de l'application. Les logiciels actuels gèrent les niveaux de détails de manière identique. Il est fonction de la distance séparant l'œil de l'utilisateur à l'objet considéré. Dans notre cas, nous avons vu qu'un objet qui serait proche de l'œil mais situé à 15° de l'axe de vision principal serait vu avec 10 fois moins de détails qu'un objet situé sur l'axe. Un pilotage des niveaux de détails ne fonctionnant qu'avec la distance œil-objet ne serait pas satisfaisant. Il serait possible d'optimiser la scène afin d'augmenter les performances de l'application. Nous proposons de définir une autre fonction de choix des niveaux de détails (cf. Figure 10).

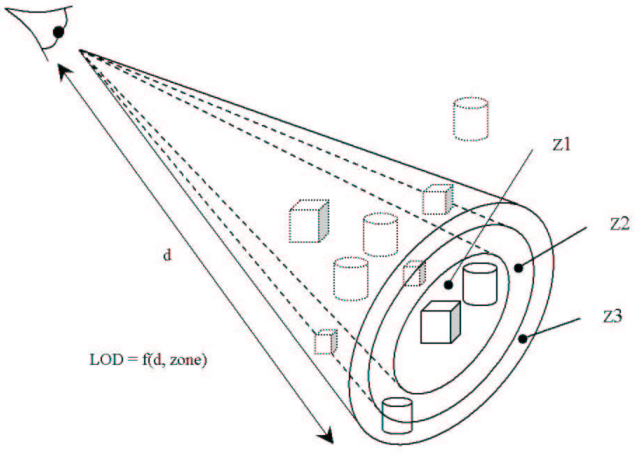


Figure 10 : Cônes de vue.

Le niveau de détail d'un objet ne doit plus être choisi uniquement en fonction de son éloignement par rapport à l'œil, mais aussi en fonction de sa position angulaire.
Cette approche est soumise à la condition que la direction du regard coïncide avec la direction de la tête de l'utilisateur qui est connue (capteur de position). Cette approximation est acceptable puisque l'utilisateur est muni de lunettes stéréoscopiques qui lui obturent une grande partie du champ visuel. On constate que les utilisateurs ont tendance à tourner la tête plutôt que le regard. L'assimilation de la direction de la tête et du regard est dans ce cas valide.
L'approche envisagée actuellement est d'utiliser les arbres binaires de partitionnement de l'espace (*BSP : Binary Space Partition*). Ce partitionnement de l'espace permet de savoir en temps réel en fonction de la direction de la caméra dans quelle zone de l'espace se situe un objet. En fonction de l'appartenance de l'objet à une zone, le niveau de finesse du modèle est choisi et affiché.

4. Conclusions et perspectives.

Les dispositifs de réalité virtuelle imposent des contraintes de temps et de qualité de visualisation très sévères. Aussi, des compromis sont nécessaires afin d'optimiser l'immersion virtuelle de l'utilisateur. L'étude des caractéristiques de la vision humaine montre qu'il n'est pas nécessaire d'afficher une image avec un rendu homogène dans l'espace. La notion de zones d'intérêts doit être développée en fonction de la distance de l'objet virtuel à l'œil mais également en fonction de l'écart à l'axe visuel principal. Par conséquent, il est opportun de prendre en compte ces caractéristiques lors du calcul de rendu des objets pour leur affichage dans la scène virtuelle. L'objet calculé pour l'affichage ne doit tout d'abord comporter que les faces utiles à la visualisation de l'objet. Cette première étape de simplification est déjà traitée par les mécaniciens pour leurs calculs de structures et peut être adaptée à la visualisation. L'objet calculé doit ensuite subir certaines opérations de simplification dont le degré doit être fonction de la distance de l'objet à l'œil ainsi que de la zone d'intérêt de l'objet. Les travaux présentés s'inscrivent dans le cadre du développement de méthodes et outils permettant d'optimiser l'immersion virtuelle eu égard à ces critères. Ces travaux prennent également en compte le contexte technologique fort du dispositif de visualisation MoVE utilisé. Il s'agit alors d'intégrer les méthodes en développement dans les outils logiciels disponibles ou en évaluation rapide sur le marché.

Références :

[DFMP96] L. DE FLORIANI, P. MARZANO et E. PUPPO, *Multiresolution models for topographic surface description*, The visual computer, vol. 12, n°7, pp 317-345, 1996.

[DRLW93] T. D. DE ROSE, M. LOUNSBERY et J. WARREN, *Multiresolution analysis for surface of arbitrary topological type*, report 93-10-05, Department of Computer Science, University of Washington, Seattle, WA, 1993.

[EDRDHLS95] M. ECK, T. DE ROSE, T. DUCHAMP, H. HOPPE, M. LOUNSBERY et W. STUETZLE, *Multiresolution Analysis of Arbitrary Meshes*, *SIGGRAPH '9*5 , 1995.

[G96] A. GUEZIEC, *Surface simplification inside a tolerance volume*, rapport technique RC 20440, IBM research division, T.J. Watson Research Center, USA, Mai 1996.

[HG94] P. HECKBERT et M. GARLAND, *Multiresolution modeling for fast rendering*, Proceedings of Graphics Interface '94, pages 43–50, Banff, Alberta, Canada. Canadian Information Processing Society, 1994.

[H96] H. HOPPE, *Progressive Meshes*, Computer Graphics, Vol. 30, Annual Conference Series, pp. 99--108, 1996.

[HVPO] Human Visual Perception Overview : http://www.stanford.edu/class/ee392c/lectures/chapter05.pdf

[PMFNTF02] D. PAILLOT, F. MERIENNE, J.-P. FRACHET, M. NEVEU, S. THIVENT, L. FINE, *Revue de projet immersive pour le style automobile*, Virtual Concept 2002, Biarritz, pages 92-97, 2002.

[PS97] E. PUPPO et R. SCOPIGNO, *Simplification, LOD and Multiresolution Principles and Applications*, Eurographics'97, 1997.

[RO96] K.J. RENZE et J.H. OLIVIER, *Generalized Unstructured Decimation*, IEEE Computer Graphics and Applications, pp 24-32, Nov. 1996.

[RR96] J.R. RONFARD et P. ROSSIGNAC, *Full-range approximation of triangulated polyhedra*, Technical report RC 20423, IBM research division, T. J. Watson Research Center. Also in Eurographics'96, 1996.

[SUPELEC] http://www.supelec-rennes.fr/ren/perso/jweiss/tv/perception/percept3.html

[SZL92] W.J. SCHROEDER, J.A. ZARGE et W.E. LORENSEN, *Decimation of triangle meshes*, ACM SIGGRAPH 92, Chicago, pp 65-70, 26-31 juillet 1992.

[V97a] P. VERON et J.-C. LEON, *Static polyhedron simplification using error Measurements*, Computer Aided Design, Vol. 29, n° 4, pp 287-298, Avril 1997.

[V97b] P. VERON, *Techniques de simplification de modèles polyédriques pour un environnement de conception mécanique*, thèse de doctorat, INPG, Grenoble, 1997.

[VL98] P. VERON, J.-C. LEON, *Shape preserving polyhedral simplication with bounded error*, Computer & Graphics, Vol. 22, N°5, pp. 565-585, 1998.

[YRR94] Akitoshi YOSHIDA, Jannick P. ROLLAND, John H. REIF, *Design and Applications of a High Resolution Insert Head Mounted Display***,** June 1994.

Drones : Simulateur d'Environnement et Apprentissage

Farès Belhadj

Laboratoire d'Intelligence Artificielle
Université de Paris 8
2, rue de la Liberté
93526 Saint Denis cedex
amsi@ai.univ-paris8.fr

Résumé : *Nous présentons un simulateur d'environnement temps réel pour l'entraînement de pilotes automatiques. Ce simulateur gère le comportement de véhicules de type hélicoptère dans un univers virtuel modélisé à partir de paysages fractals. Un pilote automatique, implanté sous la forme d'un Perceptron Multi-Couches, apprend à voler en rase-mottes et évite les obstacles, statiques ou dynamiques, rencontrés. Cet apprentissage est supervisé et effectué en deux phases : il apprend à voler en observant les réactions de deux superviseurs. Ces phases sont repectivement orchestrées par un programme, l'automate volant, et un utilisateur humain. Ce dernier pilote le véhicule via une interface dans laquelle le rendu de la simulation est effectué en images de synthèse.*

Mots-clés : paysages fractals, simulation, images de synthèse, temps réel, apprentissage, drone, planification de trajectoires, réseau de neurones.

1 Introduction

Les systèmes embarqués sont des entités autonomes capables de percevoir et d'interagir avec leur environnement. La perception est généralement réalisée à l'aide de capteurs. Un algorithme utilisant ces informations prend des décisions et réagit aux événements. Il est impératif de vérifier les réponses de l'algorithme dans diverses situations. Celles-ci peuvent être rencontrées dans des environnements reconstitués ou fictifs. Pour cela, il est nécessaire d'avoir un simulateur d'environnement dans lequel les modèles physiques sont respectés et où tout type d'environnement peut être créé. De nombreux travaux existent dans ce domaine, par exemple [Tho98, DRC+00].

Nous proposons une première approche dans laquelle un environnement est un univers virtuel composé d'un terrain (cartes fractales), d'obstacles et de véhicules de type hélicoptère interagissant avec l'ensemble. Nous synthétisons l'ensemble de notre interface et de ses éléments sous la forme d'une librairie. Chaque module de pilotage peut se connecter à cette librairie et utiliser les entrées/sortie proposées.
Nous décrivons, dans ce qui suit, les méthodes utilisées pour la génération des différents types de paysages, terrains et arbres. Nous donnons une modélisation du type de véhicules importés. Nous détaillons, pour ces véhicules, les possibilités de perception de leur environnement ainsi que le processus de commande. Enfin, nous réalisons et comparons dans ce cadre nos deux implémentations de pilotes automatiques, un *automate volant* et un *Perceptron Multi-Couches*.

2 Les composantes de l'environnement

2.1 La génération de terrains

Nous représentons les terrains selon un maillage régulier donné par une matrice de valeurs d'altitudes. Ces valeurs sont échelonnées sur un intervalle $[0, 255]$ et la matrice résultante produit une carte topographique du terrain. Pour générer ces cartes, nous utilisons au choix trois algorithmes fractals [Man95] (cf. figure 1). L'algorithme du *plasma* [Aud97, Ste90] ainsi que BROWN-GAUSS produisent ce que nous appelons des « nuages fractals ». Enfin, un troisième algorithme, basé sur la méthode du « Quick Union Find » [Sed98], produit des cartes de labyrinthes aléatoires.

Génération de nuages fractals :

Les deux algorithmes proposés sont basés sur un système d'interpolations bilinéaires par déplacement des milieux. Soit une matrice $\mathcal{A}_{(M\times N)}$ dont les valeurs aux quatre coins $p_1 = (0,0)$, $p_2 = (M-1,0)$, $p_3 = (0,N-1)$ et

$p_4 = (M-1, N-1)$ sont prises aléatoirement. Nous calculons récursivement les valeurs aux points intermédiaires. L'interpolation des points varie suivant l'algorithme utilisé.

a. Le plasma : nous calculons à partir du rectangle (p_1, p_2, p_3, p_4) les valeurs aux quatre points des milieux des bords — p_5, p_6, p_7 et p_8 — ainsi que la valeur au point central p_9. Ces valeurs $v(p_i)$ sont obtenues selon :

$$\begin{cases} v(p_5) = v(\frac{M-1}{2}, 0) = \frac{v(p_1)+v(p_2)}{2} + o(L) \;\Big|\; v(p_6) = v(\frac{M-1}{2}, N-1) = \frac{v(p_3)+v(p_4)}{2} + o(L) \\ v(p_7) = v(0, \frac{N-1}{2}) = \frac{v(p_1)+v(p_3)}{2} + o(L) \;\Big|\; v(p_8) = v(M-1, \frac{N-1}{2}) = \frac{v(p_2)+v(p_4)}{2} + o(L) \\ v(p_9) = v(\frac{M-1}{2}, \frac{N-1}{2}) = \frac{v(p_1)+v(p_2)+v(p_3)+v(p_4)}{4} + o(L) \end{cases}$$

où $o(L)$ est un entier relatif aléatoire proportionnel à L avec $L = max(M, N)$.
Chaque valeur $v(p_i)$ est ramenée dans l'intervalle $[0, 255]$ par modulo.

Nous obtenons, après une première itération, quatre nouveaux rectangles de dimensions $\frac{M}{2} \times \frac{N}{2}$: (p_1, p_5, p_7, p_9), (p_5, p_2, p_9, p_8), (p_7, p_9, p_3, p_6) et (p_9, p_8, p_6, p_4). Nous répétons récursivement ce calcul sur chaque sous rectangle obtenu tant que $L = max(\frac{M}{2^n}, \frac{N}{2^n}) > 2$ où n est la profondeur de récursion.
Remarque : Si une valeur a été attribuée à un point, alors cette valeur ne sera plus modifiée.

b. BROWN-GAUSS : de la même manière que précédemment, nous calculons, pour les quatre points initiaux, cinq nouvelles valeurs aux points p_5, p_6, p_7, p_8 et p_9. Les dépendances entre points ne sont plus les mêmes et la variable aléatoire $o(\Delta)$, prise ici, est régie par une distribution gaussienne. Ces valeurs sont calculées selon :

$$\begin{cases} v(p_9) = v(\frac{M-1}{2}, \frac{N-1}{2}) = \frac{v(p_1)+v(p_2)+v(p_3)+v(p_4)}{4} + o(\Delta) \\ v(p_5) = v(\frac{M-1}{2}, 0) = \frac{v(p_9)+v(p_1)+v(p_2)}{3} + o(\Delta) \;\Big|\; v(p_6) = v(\frac{M-1}{2}, N-1) = \frac{v(p_9)+v(p_3)+v(p_4)}{3} + o(\Delta) \\ v(p_7) = v(0, \frac{N-1}{2}) = \frac{v(p_9)+v(p_1)+v(p_3)}{3} + o(\Delta) \;\Big|\; v(p_8) = v(M-1, \frac{N-1}{2}) = \frac{v(p_9)+v(p_2)+v(p_4)}{3} + o(\Delta) \end{cases}$$

où :
- $o(\Delta)$ est un réel aléatoire égal à $\Delta \times Gauss()$.
- $\Delta = \frac{\sigma}{\sqrt{2^n}}$ avec n le numéro de l'itération en cours.
- σ est l'écart-type de la distribution gaussienne (≈ 1).
- $Gauss()$ est une fonction qui renvoie un réel compris entre $\pm 2\sigma$ suivant une distribution gaussienne.

Tant que $L = max(\frac{M}{2^n}, \frac{N}{2^n}) > 2$, nous réitérons cette opération sur chaque sous rectangle obtenu. Pour finir, nous échelonnons la matrice résultante sur l'intervalle [0, 255].

Génération de labyrinthes :

Nous proposons une méthode de production aléatoire de labyrinthes 1-connexes : un unique chemin relie deux positions distinctes (la connexité de ces labyrinthes peut être augmentée en supprimant certains murs). Pour un labyrinthe de dimensions $M \times N$, une matrice $\mathcal{L}$ de dimensions $(2M+1) \times (2N+1)$ est créée. Cette matrice représente les données topographiques du labyrinthe. Elle est initialisée telle que chaque position libre est entourée de murs (cf. exemple pour $M = N = 3$) où les murs sont représentés par la valeur -1. Pour construire le labyrinthe, nous sélectionnons aléatoirement une position murée séparant, en 4-connexité, deux positions libres. Si aucun chemin ne relie ces deux positions, alors le mur est supprimé. Cette opération est répétée jusqu'à ce que toutes les positions libres, à l'initialisation, soient connectées.
Nous donnons un exemple d'initialisation de $\mathcal{L}$ pour $M = N = 3$:

$$\mathcal{L} = \begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & \boxed{0} & -1 & \boxed{1} & -1 & \boxed{2} & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & \boxed{3} & -1 & \boxed{4} & -1 & \boxed{5} & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & \boxed{6} & -1 & \boxed{7} & -1 & \boxed{8} & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

Donc pour M et N quelconques, $\mathcal{L}$ est initialisée par :
– Si la position (i, j) est telle que i ou j pair alors $\mathcal{L}[i][j] = -1$ et (i, j) est murée ;
– Sinon, $\mathcal{L}[i][j] = (M \times \lfloor \frac{i}{2} \rfloor) + \lfloor \frac{j}{2} \rfloor$ et (i, j) est non murée.

Puis nous posons que $\mathcal{L}$ est assimilée à un graphe à $M \times N$ composantes. Nous avons alors un nœud à identifiant unique par composante. Nous produisons, à partir de ce graphe, un graphe à une composante. Pour ce faire, une position murée séparant deux composantes disjointes est sélectionnée aléatoirement ; la position est libérée et le plus petit identifiant est propagé sur la nouvelle composante connexe. Nous posons que deux nœuds sont disjoints si et seulement si leurs identifiants respectifs sont différents. L'algorithme se termine quand les $M \times N$ nœuds du graphe ont un 0 comme identifiant. La matrice résultante est échelonnée[1] sur l'intervalle [0, 255].

[1] Par exemple : 0 pour les positions non murées et 255 pour les positions murées.

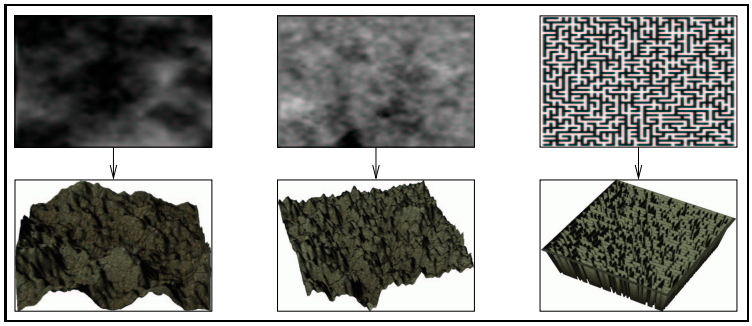


FIG. 1 – Cartes topographiques générées et rendu 3D ; de gauche à droite : plasma, BROWN-GAUSS et labyrinthe.

2.2 Les L-Systèmes

Plus qu'un aspect visuel, les arbres représentent des obstacles supplémentaires dans le parcours des drones. Nous implémentons, pour la génération d'arbres, un interprète de « Bracketed » L-Systèmes à composante stockastique [PL89, PL90, PH92]. Ce module crée les images représentant les différents types d'arbres à partir d'un fichier décrivant les règles de production. Nous utilisons ces images comme des textures 2D que nous plaquons sur deux rectangles perpendiculaires. Les arbres, ainsi créés, sont placés dans le paysage aléatoirement et par groupe de même famille. Un exemple de fichier descriptif ainsi que le rendu 2D de l'arbre obtenu est donné en figure 2.

```
1. Arbre001{                                  // Nom de l'objet
      axiom ++++++++++++X                     // L'axiome ω
      angle 7.2                               // La variation d'angle σ
      angle0 85.0                             // L'angle initial α0
      step 50.0                               // Le pas de déplacement d en pixels
      gen 9                                   // Le nombre de générations
      X=F[@.5+++++++++X]-F[@.4----------!X]@.6X   // Une règle de production
}
```

FIG. 2 – Fichier descriptif et rendu 2D produit par l'interprète de L-Système.

2.3 Les véhicules

Les aéronefs sont placés et évoluent dans les paysages virtuels générés par les algorithmes fractals. Nous allons étudier maintenant comment ces véhicules sont représentés et se meuvent. Nous implémentons un module d'importation d'objets 3D réalisés par modeleur. Ce module permet une représentation tridimensionnelle paramétrable pour chaque type de véhicule, actuel ou à venir. Il créé l'objet *OpenGL* [WNDS99] décrit par des fichiers *A.S.E.* — « Ascii Scene Export » — et le place dans le paysage. Pour un hélicoptère, nous utilisons quatre fichiers *A.S.E.*, chacun décrit respectivement le cockpit, la queue, le disque rotor et le rotor anti-couple. Cette subdivision aide à la construction de l'enveloppe convexe contenant l'ensemble ; nous utilisons cette enveloppe dans la détection de collisions.
Pour obtenir une simulation de la dynamique de vol d'un hélicoptère, nous proposons un modèle physique simplifié du véhicule. Dans notre modèle, la sustentation et la propulsion du véhicule sont assurées par le disque rotor qui produit deux types de mouvements : le tangage et le roulis. Le rotor anti-couple produit une force contraire à la direction de rotation du disque rotor ; elle assure le mouvement en lacet. Seule la variation de cette force est considérée dans notre modèle. La dynamique de vol de l'aéronef est montrée sur la figure 3, elle obéit à :

$$\left\{ \begin{array}{l} \overrightarrow{F} + m \times \overrightarrow{g} = m \times \overrightarrow{a} \text{ projetée sur chaque axe :} \\ \overrightarrow{F} = \begin{pmatrix} F_x \\ F_y \\ F_z \end{pmatrix}, \quad \overrightarrow{a} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} : \left\{ \begin{array}{lcl} a_x & = & F_x \\ a_y & = & F_y - g \\ a_z & = & F_z \end{array} \right. \Rightarrow \left\{ \begin{array}{lcll} a_x & = & \frac{\cos\theta \times ||\overrightarrow{F}||}{m} & (x) \\ a_y & = & \frac{\sin\rho \times ||\overrightarrow{F}||}{m} - g & (y) \\ a_z & = & \frac{\cos\rho \times ||\overrightarrow{F}||}{m} & (z) \end{array} \right. \end{array} \right.$$

Avec $\overrightarrow{F}$ la force produite par le disque rotor, $\overrightarrow{g}$ la gravité, m la masse du véhicule et $\overrightarrow{a}$ son accélération.

Enfin, les différentes projections du vecteur vitesse sont corrigées par l'ajout d'une force de résistance à l'air. Nous obtenons :

$$v_i = v_i - (k \times v_i^2)$$

où k est le coefficient aérodynamique, déterminé pour une vitesse maximale donnée, environ $90m.s^{-1}$ pour ce type d'aéronefs, par :

$$k = \frac{m \times a}{v_{max}^2}$$

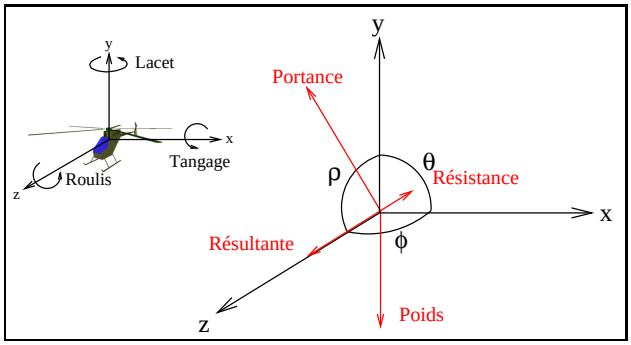


FIG. 3 – Dynamique de vol de l'aéronef.

3 Les éléments de perception et de commande

Chaque programme (pilote automatique) accède via notre librairie à une série de fonctions lui permettant de piloter un ou plusieurs hélicoptères ainsi que de percevoir l'univers virtuel local ou global. Nous décrivons ici ces outils de perception et le processus de commande [LLS+01].

3.1 La perception globale : le *plus sûr chemin*

Notre algorithme du *plus sûr chemin*, noté *P.S.C.*, est une méthode rapide de calcul du chemin discret reliant deux points quelconques sur une carte. Le chemin résultant minimise les altitudes empruntées réalisant le meilleur compromis entre altitude et distance. L'utilisateur de l'interface sélectionne une coordonnée d'arrivée pour chaque véhicule ; le module *P.S.C.* calcule les chemins aux buts.
Cet algorithme est basé sur la méthode de calcul du plus court chemin dans un graphe [Dij59, LS95]. Le graphe des chemins possibles est donné par la matrice, en 8-connexité, des données topographiques du terrain. Les arètes sont pondérées par l'altitude en chaque point ou nœud du graphe. Nous introduisons des modifications de l'algorithme afin d'améliorer les temps de calcul des *P.S.C.* Pour cela, nous découpons la carte des altitudes en sous-régions rectangulaires de dimensions égales. Une moyenne des altitudes est calculée pour chaque sous-région et un *P.S.C. grossier* est produit pour relier l'ensemble des sous-régions empruntées. À partir de là nous raffinons le résultat obtenu en calculant un *P.S.C.* à l'intérieur de chaque sous-région empruntée par le chemin *grossier*. La complexité des graphes représentant chaque sous-région est bien moindre comparée à l'ensemble.
Nous obtenons par cette méthode un *P.S.C.* équivalent à une version non optimisée dans des délais inférieur à la seconde. Un ratio de temps de calcul supérieur à 10^3 a été mesuré entre l'algorithme classique et le nôtre. Cette mesure est donnée pour une carte topographique de 1024×1024 nœuds.

3.2 La perception locale

Tout aéronef créé possède sa propre configuration d'outils de perception. Ces outils sont implémentés sous la forme d'un ensemble paramétrable d'instruments de bord. Nous pouvons les sélectionner dans la liste contenant : des capteurs de distance, des instruments de mesure de l'angle de direction, l'angle de tangage, l'angle de roulis et la mesure de la vitesse. Cette dernière est donnée en *unité-terrain* par seconde, notée $ut.s^{-1}$, et ut est la distance

horizontale séparant, dans le quadrillage du terrain, deux sommets voisins. Enfin, un système radar permet de localiser les différents véhicules présents à proximité de l'appareil.
Nous utilisons, dans nos implémentations de pilotes automatiques, un instrument de mesure de l'angle de direction ainsi que six capteurs de distance. Ces derniers renvoient la distance, en ut, les séparant d'un obstacle. Leur portée maximale est de $10ut$. Quand à l'instrument de mesure de l'angle de direction, il renvoie l'angle permettant à l'aéronef de suivre le chemin discret donné par l'utilisateur via le module du *P.S.C.*

3.3 Le processus de commande

Dans un hélicoptère, les commandes correspondent au *manche de pas cyclique*, il incline le disque rotor afin de modifier la direction de la portance, le *palonnier* contrôle le rotor anti-couple, le *levier de pas collectif* contrôle l'inclinaison des pales et la *manette de gaz* modifie la vitesse de rotation du disque rotor.
Nous émulons les commandes réelles d'un hélicoptère par l'intermédiaire d'un automate à états qui produit une modification des paramètres de l'hélicoptère et génère la dynamique de vol correspondante. De cette manière, toute action produite par le pilote devient accessible en *lecture / écriture*. Nous pouvons ainsi, à tout moment, interroger l'interface sur l'état de chaque commande.
Nous obtenons alors une représentation numérique des actions effectuées par le pilote. Cette représentation nous permet d'introduire la notion *d'apprentissage* par observation. Elle est utilisée pour l'apprentissage du *Perceptron Multi-Couches*. L'observateur apprend à piloter en *copiant* les réaction du superviseur (cf. figure 4).

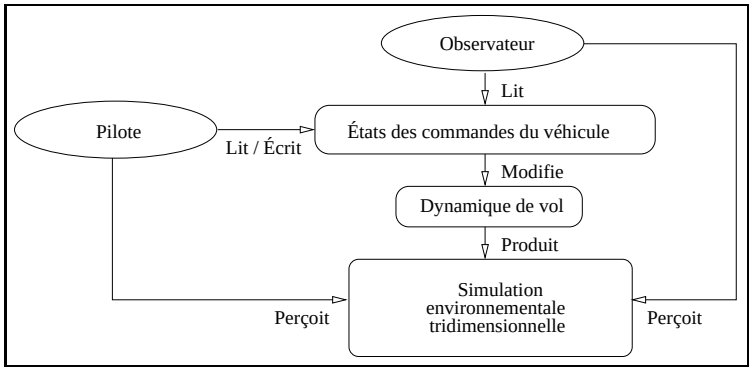


FIG. 4 – Le processus d'apprentissage par observation.

4 Les pilotes automatiques

Comme application de notre librairie de gestion d'environnements virtuels, nous implémentons deux pilotes automatiques, un automate déterministe et un *Perceptron Multi-Couches*, et nous comparons leurs résultats. Pour cette évaluation, la configuration de la perception des hélicoptères est la suivante :

- six capteurs de distance configurés comme suit :
  - cinq capteurs en position avant-centre et orientés respectivement $(\rho = \frac{\pi}{15}, \phi = -\frac{\pi}{15})$, $(\rho = \frac{\pi}{15}, \phi = \frac{\pi}{15})$, $(\rho = -\frac{\pi}{15}, \phi = \frac{\pi}{15})$, $(\rho = -\frac{\pi}{15}, \phi = -\frac{\pi}{15})$, $(\rho = 0, \phi = 0)$;
  - un capteur en position centre-bas et orienté $(\rho = -\frac{\pi}{2}, \phi = 0)$;
- un instrument de mesure de l'angle de direction : il renvoie l'angle ϕ permettant à l'aéronef de rejoindre la prochaine position discrète donné par le *plus sûr chemin* ;
- un instrument de mesure de la vitesse au sol.

4.1 L'automate volant

Ce pilote doit, à partir des informations de perception locale (instruments de mesures et capteurs de distance) et globale (le *plus sûr chemin*), s'orienter dans l'environnement virtuel et arriver jusqu'aux coordonnées spécifiées par l'utilisateur de l'interface. Il est implémenté sous la forme d'un automate déterministe pour lequel, d'une part,

les capteurs définissent les entrées de l'automate et d'autre part, ses différents états donnent les commandes correspondantes pour l'hélicoptère (cf. figure 5). Ce pilote adopte un comportement de *précaution* : il vole à des altitudes moyennes en prenant ses distances par rapport aux obstacles.

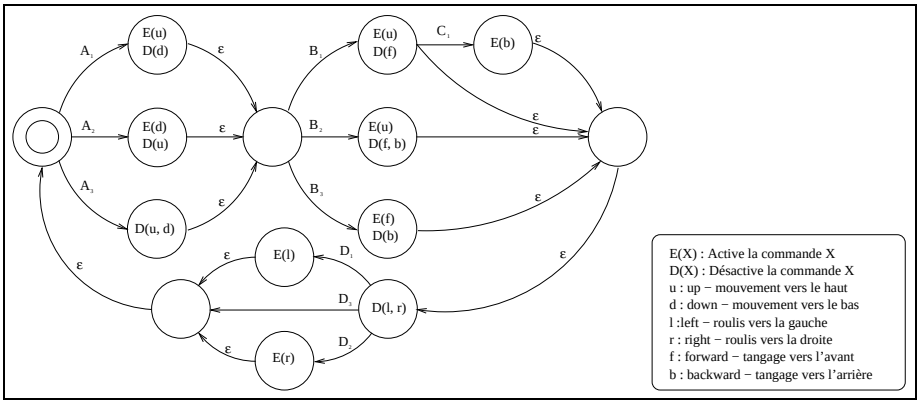


FIG. 5 – La réaction aux événements (perception via les capteurs de distance, de vitesse et d'orientation) de l'automate volant produit un changement dans le comportement de l'hélicoptère.

Nous donnons les conditions de passage d'états sur la figure 6, où $D(cpt_n)$ est la distance, en *unité-terrain*, renvoyée par le capteur n et vH est la projection de la vitesse sur le plan horizontal.

$$
\begin{array}{lcl}
A_1 & \Longleftrightarrow & (D(cpt_6) < 1\ ut) \vee (D(cpt_3) < 3\ ut) \vee (D(cpt_4) < 3\ ut) \\
A_2 & \Longleftrightarrow & \neg A_1 \wedge (D(cpt_6) > 2\ ut) \wedge (D(cpt_3) > 6\ ut) \wedge (D(cpt_4) > 6\ ut) \\
A_3 & \Longleftrightarrow & (\neg A_1 \wedge \neg A_2) \\
B_1 & \Longleftrightarrow & (D(cpt_5) < 3\ ut) \vee (D(cpt_1) < 3\ ut) \vee (D(cpt_2) < 3\ ut) \\
B_2 & \Longleftrightarrow & \neg B_1 \wedge (\ (D(cpt_5) < 6\ ut) \vee (D(cpt_1) < 6\ ut) \vee (D(cpt_2) < 6\ ut)\ ) \\
B_3 & \Longleftrightarrow & (\neg B_1 \wedge \neg B_2) \\
C_1 & \Longleftrightarrow & (vH > 0) \\
D_1 & \Longleftrightarrow & (\delta\phi > 0.5\ rad) \\
D_2 & \Longleftrightarrow & (\neg D_1 \wedge (\delta\phi < -0.5\ rad)) \\
D_3 & \Longleftrightarrow & (\neg D_1 \wedge \neg D_2) \\
\varepsilon & \Longleftrightarrow & \text{vrai}
\end{array}
$$

FIG. 6 – Définition des conditions de changement d'état de l'automate volant.

4.2 Le *Perceptron Multi-Couches*

La dynamique de vol de l'automate, ainsi défini, suit un comportement de précaution. Notre but est d'obtenir un nouveau pilote qui adopte une attitude *dangereuse* mais efficace. Celui-ci devra voller en rase-mottes, à une vitesse soutenue et sans collision. Pour ce faire, nous utilisons l'automate volant à des fins d'entraînement d'un *Perceptron Multi-Couches* noté *P.M.C.*

Le *P.M.C.* est un réseau de neurones [Ros58, DMS+02, MP90] à apprentissage supervisé. Notre implémentation est constituée de trois couches de neurones avec huit neurones en entrée pour une compatibilité avec l'automate et six neurones en sortie pour les six commandes de l'hélicoptère. Lors de la **première phase** d'apprentissage, nous interfaçons l'automate (i.e. le superviseur) aux commande de dix hélicoptères. Le *P.M.C.* apprend par observation (cf. figure 4) et converge vers le même comportement.

En **deuxième phase**, l'utilisateur *humain* prend, via l'interface clavier, les commandes d'un hélicoptère et adopte un comportement plus dynamique et vole au plus près du sol. Comme dans le cas précédent, le *P.M.C.* apprend par observation : cette phase d'apprentissage dure quelques minutes. L'interface permet à l'utilisateur de *redonner la main* au *P.M.C.* et ainsi vérifier l'amélioration de son comportement.

Nous obtenons à la suite des deux phases d'apprentissage un nouveau pilote au comportement dit **amélioré**. Nous comparons les résultats obtenus par l'automate volant aux résultats du *P.M.C.* après chaque phase d'apprentissage (cf. tableau 1). Ces tests sont effectués dans les mêmes conditions, sur un échantillon de dix terrains et pour les mêmes couples de coordonnées (point de départ et point d'arrivée). Nous pouvons voir sur la figure 7 une réduction considérable de la distance par rapport au sol. Le *P.M.C.* **amélioré** effectue un vol en rase-mottes et sa vitesse horizontale est globalement constante.

	Vitesse horizontale	Distance par rapport au sol
Automate volant	$1,344\ ut/s$	$44,784\ ut$
P.M.C.	$1,501\ ut/s$	$49,3375\ ut$
P.M.C. **amélioré**	$1.2522\ ut/s$	$19,3815\ ut$

TAB. 1 – Comparatif entre l'automate volant, le *P.M.C* et le *P.M.C* amélioré. La distance par rapport au sol est donnée est *unité-terrain* et la vitesse horizontale moyenne en *unité-terrain* par seconde. La fréquence de l'échantillon est de 40 itérations par seconde.

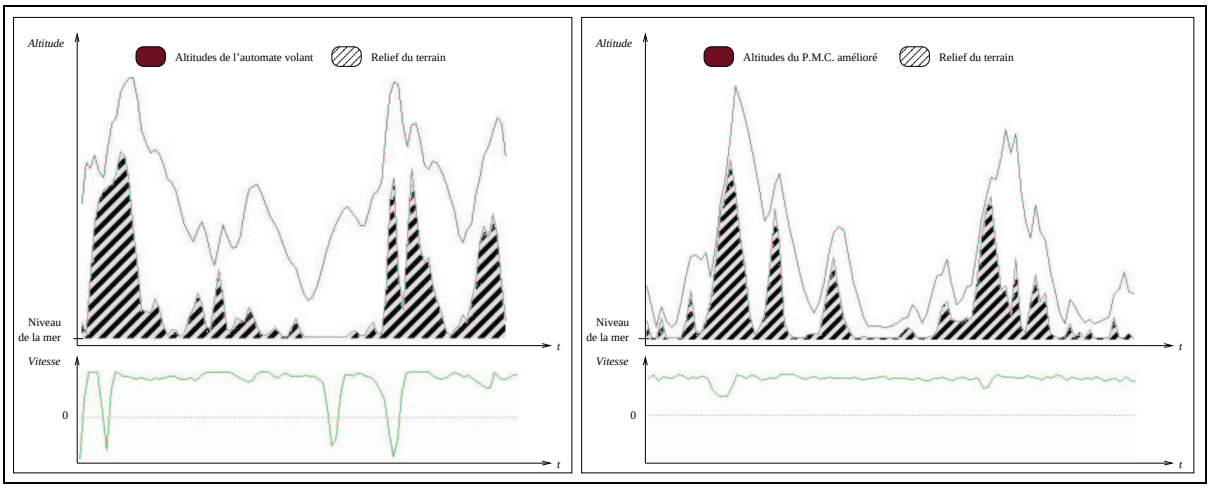


FIG. 7 – Comparaison des courbes des altitudes et de vitesse horizontale de l'automate volant et du *Percepton Multi-Couches* **amélioré**.

4.3 Conclusion et perspectives

Nous avons développé, sous la forme d'une librairie, un simulateur temps réel d'environnement d'entraînement pour drones. L'interface nous a permis de réaliser plusieurs phases d'apprentissage, de visualiser et de comparer les comportements obtenus. Une modélisation plus complexe et plus interactive de l'interface peut être produite en étendant les résultats obtenus par cette première approche.
Dans cette optique, la création et l'intégration à l'environnement d'un modeleur pour *véhicules et capteurs virtuels* permettrait d'élargir le champ applicatif de l'interface. Ce module prendrait à sa charge la dynamique de mouvement des véhicules créés. Une autre perspective est la possibilité de mettre en concurrence plusieurs programmes autonomes et d'observer leur évolution comportementale. Ces programmes communiqueraient entre-eux via l'environnement virtuel et feraient émerger des notions de concurrences ou d'entre aide.

Références

[Aud97] P. Audibert. *Algorithmes et Programmation*. Université Paris 8, 1995-1997.

[Dij59] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1 :269–271, 1959.

[DMS+02] G. Dreyfus, J.-M. Martinez, M. Samuelides, M.B. Gordon, F. Badran, S.Thiria, and L. Hérault Sous la direction de Gérard Dreyfus. *Réseaux de neurones : Méthodologie et applications*. Eyrolles, 2002.

[DRC+00] T. Duval, J. Regincós, A. Chauffaut, D. Margery, and B. Arnaldi. Interactions collectives locales en immersion dans des univers virtuels 3d avec gasp. Actes de la conférence ERGO-IHM 2000, Biarritz, France, Octobre 2000.

[LLS+01] J.-P. Laumond, Florent Lamiraux, Sepanta Sekhavat, Pascal Morin, Claude Samson, Patrick Rives, Michel Devy, Malik Ghallab, Bernard Espiau, and Frank Génot sous la direction de Jean-Paul Laumond. *La robotique mobile*. Hermès, 2001.

[LS95] C.A. Lazere and D.E. Shasha. *Out of Their Minds*. Copernicus Books, 1995.

[Man95] B. Mandelbrot. *Les Objets Fractals*. Flammarion, quatrième edition, 1975-1995.

[MP90] M.L. Minsky and S.A. Papert. Perceptrons. 1990.

[PH92] P. Prusinkiewicz and J. Hanan. *Lindenmayer Systems, Fractals, and Plants*. Springer Verlag, 1992.

[PL89] P. Prusinkiewicz and A. Lindenmayer. Developmental models of multicellular organisms : A computer graphics perspective. In Christopher G. Langton, editor, *Artificial Life volume VI : Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems*. Addison Wesley, 1989.

[PL90] P. Prusinkiewicz and A. Lindenmayer. *Algorithmic Beauty of Plants*. Springer Verlag, 1990.

[Ros58] F. Rosenblatt. The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological Review 65 : 386-408*, 1958.

[Sed98] R. Sedgewick. *Algorithms in C*. Addison-Wesley, third edition, 1998.

[Ste90] R.T. Stevens. *Advanced Fractal Programming in C*. M&T, 1990.

[Tho98] G. Thomas. Représentation d'environnement urbains pour l'animation de piétons. *AFIG'98*, 1998.

[WNDS99] M. Woo, J. Neider, T. Davis, and D. Shreiner. *Le guide officiel à l'apprentissage d'OpenGL, version 1.2*. CompusPress France, 1999.

Conférence invitée : Du Virtuel Au Réel

Claude Ecken

ecken_c@club-internet.fr

La Science-Fiction ne se préoccupe pas du futur : elle parle du présent. Peu importent les événements du quatrième millénaire, nous ne serons plus là pour les voir. Mais le monde change si vite qu'il est nécessaire, pour le comprendre, de le mettre en perspective. Celle tournée vers le passé n'est pas toujours suffisante tant les référents et points de repères ont été bouleversés. En revanche, il est possible d'imaginer les alternatives qui s'offrent à nous et de projeter, dans un futur plus ou moins lointain, les interrogations et les problèmes de nos sociétés.

Il ne s'agit pas de prospective ni de futurologie : la question n'est pas de savoir si l'auteur a prédit l'avenir mais s'il a mené sa réflexion jusqu'au bout. L'ancrage dans un futur indéterminé lui permet justement de s'abstraire du présent, des trop nombreuses interactions qui perturbent l'analyse d'un problème. C'est un peu comme s'il éliminait les parasites, le bruit de fond ambiant, qui gênent son observation. Il se place ainsi dans les conditions d'un chercheur de laboratoire, qui isole un phénomène de son environnement et fait ensuite varier les paramètres qui permettent de le mesurer. Aujourd'hui, on ne parle bien du présent qu'au futur.

Les questions d'ordre philosophique ou métaphysique peuvent s'épanouir dans des futurs lointains, sans rapport avec notre quotidien. Les problèmes sociologiques, centrés sur des préoccupations actuelles, s'illustrent davantage dans des futurs proches qui gardent une continuité historique avec le présent.

La SF ne se préoccupe pas de prévoir l'évolution de la science, ni les prochains développements scientifiques. Elle ne prédit pas, elle commente, pas forcément dans une perspective éthique, mais dans la façon dont l'homme s'approprie les nouveaux outils qu'il a mis au point et les intègre à son environnement, observant plus spécifiquement l'impact social et psychologique que peut provoquer une révolution technologique.

La SF est une mise en scène de la science qui permet d'effectuer une lecture du présent ; elle ressemble, de ce point de vue, à un story-board qui met un scénario en images avant le tournage effectif des scènes. Cela permet parfois de trouver, pour une nouvelle technologie, des utilisations auxquelles on n'avait peut-être pas pensé, et de prévoir les avantages et les inconvénients auxquels on peut s'attendre. De cette mise en scène naît une vision nouvelle du monde. Le miroir déformant de la science-fiction met en évidence des détails qui avaient jusque là échappé à l'attention et qui permettent d'interpréter notre présent. Elle déforme la réalité à la façon d'une caricature qui exagère ou minimise les traits d'une personne afin de mieux la saisir dans son essence.

Du coup, la SF est prompte à placer dans un environnement familier ce qui n'existe encore qu'à l'état d'ébauche dans les laboratoires. Pour les besoins de son histoire, elle est capable d'aller bien au-delà du raisonnable : l'essentiel est qu'elle tire matière à réflexion à travers sa fiction.

Les problèmes techniques qu'il a fallu surmonter pour mettre au point une invention sont souvent gommés au profit de son utilisation. On se contente de voir que ça marche, et ça marche bien : il est rare qu'en SF l'utilisation d'un outil informatique soit ralentie par de gros temps de calcul ou par l'apparition de bugs… sauf si ces complications servent l'intrigue.

La preuve que la SF ne prédit rien est qu'elle n'a pas annoncé la révolution informatique. Ni la miniaturisation de l'ordinateur (on trouve dans certains récits des ordinateurs occupant la surface d'une planète), ni sa diffusion dans le grand public, à l'exception d'une seule nouvelle de Murray Leinster, en 1948, intitulée *Un Logic nommé Joe*. Par contre, quand les premiers PC ont été mis sur le marché, elle s'est aussitôt penchée sur les implications de cette dissémination technologique, ce qui a généré un rameau spécifique de ce genre littéraire : le cyberpunk.

Le terme, titre d'une nouvelle de Bruce Bethke parue en 1983, n'est pas forcément apprécié des tenants du genre mais l'alliance de cybernétique et de punk recouvre assez exactement son contenu. Il s'agit d'univers où la technologie évoluée est omniprésente, disséminée dans les moindres interstices du quotidien, et utilisée par tous, notamment les délinquants, qui pullulent dans ces sociétés déliquescentes, dures et cruelles, où la survie est un enjeu quotidien.

Le premier roman de William Gibson, *Neuromancien*, est considéré comme le livre ayant lancé le mouvement cyberpunk, orchestré par Bruce Sterling, auteur et théoricien du genre. La première phrase du roman, typique de l'ambiance cyberpunk, témoigne bien de l'immersion de l'informatique à tous les niveaux : "*Le*

ciel au-dessus du port était couleur télé calée sur un émetteur hors service."

Dès le début, les perceptions, les couleurs sont décrites à l'aide de référents technologiques. On ne prend plus la nature comme modèle, on la compare au contraire aux images virtuelles fabriquées par l'homme, ce qui est le signe d'une immersion dans une réalité de synthèse.

Le glissement n'est pas nouveau. Il fait même partie de l'histoire de l'humanité. Parler d'images de synthèse, de réalité augmentée ou virtuelle revient à parler de notre désir de façonner le réel. L'homme s'est toujours caractérisé par sa capacité à intervenir sur son environnement, à le modeler selon ses souhaits. Ce que permet la technologie n'est qu'une nouvelle étape du remodelage de la réalité.

Images de synthèse et images virtuelles : moins d'outils, plus de réalisme

Une image de synthèse est une image obtenue en l'absence de support concret, mais qui est capable d'imiter les outils ayant servi à la produire. Le grain du papier, le gras du fusain ou les couleurs pastel de l'aquarelle, les couches plus épaisses de la gouache ou de la peinture à l'huile sont simulées à l'aide d'appareils qui n'ont aucun rapide avec la pratique réelle des arts plastiques. Mais l'image de synthèse ne se limite pas à la simulation : elle produit ses propres images avec des rendus extrêmement longs et difficiles à obtenir par des moyens classiques, quand on y parvient. Un dégradé impeccable, des reflets métalliques ou humides sont obtenus par infographie avec une précision supérieure à l'usage de l'aérographe des hyperréalistes.

Cette substitution ne se limite pas à l'image : la musique électronique a connu la même révolution. Les synthés et autres orgues électroniques simulent depuis longtemps une foule d'instruments et créent également des sons nouveaux, que nul instrument ne saurait reproduire.

L'art, qui imitait la nature, est à présent imité par la machine.

Elle l'interprète également. Les images de synthèse permettent de visualiser des objets invisibles à l'œil, parce que trop petits ou n'appartenant à notre spectre de vision, et d'autres qui n'existent pas en tant que tels mais qui sont construits par l'interprétation d'une certaine somme d'informations. On peut ainsi "voir" de superbes galaxies lointaines aux couleurs chatoyantes alors qu'on n'a fait qu'écouter des bruits dans une certaine longueur d'onde ; un microscope électronique ne voit pas les atomes à l'aide d'un instrument optique mais interprète de même des informations électromagnétiques. L'imagerie médicale, par TEP (Tomographie par Émission de Positons) et IRM (Imagerie par Résonance Magnétique Nucléaire) permet de reconstituer le système digestif d'un patient le long duquel le médecin se promènera ensuite à loisir, comme dans un jeu vidéo. On n'a jamais réellement filmé l'intérieur du corps humain, mais on peut le voir comme s'il avait été ouvert. La plupart des images que nous produisons sont donc loin de la réalité mais intègrent un certain nombre d'informations qui leur donne du sens. Virtuel ne signifie pas irréel mais réel interprété.

En cela, rien de neuf sous le soleil. Les peintures rupestres des hommes des cavernes étaient déjà une interprétation de la réalité, et l'histoire de la peinture témoigne des tentatives pour imiter le réel, avec un rendu toujours plus grand, ou l'interpréter, surtout à partir du moment où la photographie a rendu caduque la représentation platement réaliste. Dans le cas de l'imitation comme de l'interprétation, il y a à la base une intention de truquer le réel.

Au XIIIe siècle, Giotto, alors élève de Cimabue, dessina sur un tableau de son maître qui s'était absenté une mouche à l'apparence si réelle que Cimabue tenta de la chasser quand il la vit. L'anecdote montre que le désir de donner réalité à nos productions, en abusant nos sens, est de tous temps. Par ailleurs, les trompe l'œil ne sont rien d'autre que des leurres dessinés. Plus près de nous, l'hyperréalisme a le détail et le rendu d'un cliché photographique.

Mais on peut aller beaucoup plus loin dans le trucage avec les images virtuelles, qui ne diffèrent des premières que par l'angle selon lequel on les considère : l'image de synthèse se définit par rapport aux moyens utilisés, l'image virtuelle par rapport au contenu.

On sait comme l'industrie du cinéma s'en est emparée pour représenter l'impossible et donner libre cours à tous les fantasmes, à l'imaginaire le plus débridé, avec un saisissant effet de réalisme. Ce qu'on appelle trucage et effets spéciaux n'est pas non plus nouveau : la photographie à peine née a suscité des trucages, certes grossiers pour un œil contemporain, mais qui déjà annonçaient l'ère du faux dans laquelle nous sommes entrés. On allait jusqu'à gratter le film pour remplacer un détail par un autre. Le cinéma, dès Méliès, s'engouffra dans les effets spéciaux avec plus de rapidité encore que la photographie. Les techniques sont devenues toujours plus sophistiquées jusqu'à l'apparition du numérique, dont la puissance de calcul permet de créer des mondes entièrement simulés. *Final Fantasy*, le film, en est un bon exemple.

Les auteurs de science-fiction n'ont pas manqué de se pencher sur les implications de l'usage immodéré de tels moyens. Dans *Remake*, de Connie Willis, Hollywood est toujours aussi puissante mais n'utilise plus d'acteur depuis longtemps ni ne tourne un seul film. On se contente de faire du neuf avec du vieux, la modélisation des interprètes du passé permettant de les placer dans des situations nouvelles. C'est ainsi que Marylin Monroe donne la réplique à Tom Cruise et que Charlie Chaplin tombe amoureux de Sharon Stone. Le numérique annonce la mort de l'acteur, dans la perspective de belles économies pour des bénéfices identiques, à

moins qu'ils ne soient réinvestis dans les moyens techniques. Les majors ne s'en plaindront pas. On sait que les acteurs sont tous des caractériels avec lesquels il est impossible de travailler et qui coûtent cher, en plus. Le film *Simone*, où une créature virtuelle a le premier rôle, témoigne de cette tendance.

Mais il y a fort à parier qu'on donnera ensuite au spectateur la possibilité d'effectuer lui-même les modifications qu'il désire. Voici ce que pense d'un film étranger un personnage de *La Tour des rêves*, de Jamil Nasir : "*C'était une* virtu *– une bande virtuelle, avec des acteurs digitalisés – fauchée, pratiquement pas interactive, avec juste quelques ébauches de scénarios alternatifs et sans fonction "en coulisses" permettant de déshabiller et de programmer soi-même les vedettes, comme dans des* virtus *occidentales.*"

On peut aussi offrir au spectateur d'être le héros d'une histoire : son image est synthétisée de manière à se substituer à celle de l'acteur qui a joué le rôle.

La retouche et la correction d'images, les incrustations, ont depuis si longtemps envahi notre quotidien qu'on se méfie avec raison de ce que l'on voit. Alarmiste, la science-fiction avait mit l'accent sur l'usage totalitaire de tels procédés, qui permettraient à des dictateurs de renforcer leur pouvoir par la désinformation. Dans le *1984* de George Orwell, Winston est employé au Ministère de la Vérité pour maquiller les journaux, réhabilitant ou supprimant des personnages selon la ligne officielle du parti. Sur les photos, des têtes disparaissent et d'autres les remplacent. Les complots sont souvent de nature idéologique : dans le film *Capricorne One*, les astronautes ne se sont jamais posés sur la Lune mais se sont livrés à une simulation filmée visant à faire accroire la suprématie technologique des États-Unis. Il est cependant difficile de se livrer à de tels trucages de la réalité sans se faire attraper. L'usage qui en est fait est davantage commercial et économique que politique ou idéologique. Les top-models des magazines sont toutes retouchées pour que leur corps soit parfait sur le papier glacé, et les incrustations publicitaires pullulent lors des retransmissions d'un match de foot.

Mais si on peut éliminer les défauts physiques, on peut aussi s'attaquer aux problèmes moraux. Le politiquement correct, les tenants d'une éthique rigide se sont toujours autorisés à censurer ce qu'ils jugeaient inacceptable. Ce n'est pas neuf non plus, mais le numérique offre ici aussi de vertigineuses perspectives. Toujours dans *Remake*, on s'attaque à tout ce que la société a prohibé et qui perdure sur la pellicule : l'alcool, le tabac et la drogue sont soigneusement effacés des vieux films par conformité aux codes moraux en vigueur.

On truque également sa propre image. Dans *L'Enfance attribuée* de David Marusek, les personnes se rencontrent de façon virtuelle, par hologrammes interposés, et affichent une apparence qui leur semble flatteuse ou correspond à l'humeur du moment.

Le trucage des images ne posant plus aucun problème, la science-fiction s'est davantage intéressée à multiplier les supports où les incruster. Les lunettes sur lesquelles s'inscrivent des messages n'ont déjà plus rien de futuriste. La SF se contente de remarquer qu'il n'est pas très poli de s'abriter derrière ses lunettes pour lire son journal pendant une conversation : elle invente les nouveaux codes sociaux qui seront en vigueur quand ces pratiques se seront généralisées.

Les murs des appartements sont tapissés de vidéos, qui peuvent interagir avec les occupants. Dans les rues, des bâtiments holographiques décorent la ville et cachent la misère. Les vêtements constituent évidemment un support idéal, bien que cela puisse être gênant quand on en change ou quand le tissu est élimé. Certains auteurs lui préféreront donc le brassard, plus facilement transportable et plus économique. Les images sur les vêtements servent davantage à afficher une publicité ou un motif ornemental, personnalisé, à la façon d'un tee-shirt imprimé. Mais elles peuvent aussi devenir des outils de communication. Voici comment Greg Bear imagine leur fonctionnement dans *Éon* : "*Les symboles lumineux qui apparaissaient entre les deux hommes provenaient de leurs pictotorques. Ces colliers qu'ils portaient autour du cou étaient des appareils capables de projeter le langage parléographique qui s'était imposé au cours des siècles (...) Toller picta l'image déplaisante d'un nid grouillant de créatures qui ressemblaient à des serpents. (...) Toller haussa les sourcils et picta quatre cercles de surprise orangée.*" Autrement dit, le pictogramme est devenu un mode de communication rapide, capable également de transmettre des émotions, à la façon des smileys qui ponctuent les e-mails.

Mais pourquoi s'encombrer de gadgets ou de vêtements dont il faut parfois changer ? Il suffit de se faire greffer des puces sous la peau, alimentées par l'électricité naturelle du corps humain, pour devenir à son tour un logo ambulant. Dans *Métrophage*, de Richard Kadrey, les nouveaux peinturlurés urbains sont très à la page : "*Les traits du garçon étaient d'une pâleur lumineuse, son crâne lisse et chauve. Jonny reconnut sa dégaine : c'était un zombi analytique. (...) L'archétype du zombi. Jonny vit toutefois des taches noires sur le crâne et les mains du garçon, aux endroits où les pixels sous-cutanés avaient cramé ou avaient été détruits. Manifestement, cela faisait plusieurs mois qu'il avait négligé tout entretien sérieux.*" Ce Zombi s'est donc fait "*dermatténuer la peau et pixeliser les couches profondes*" pour afficher sur son corps les images de son choix. Et voici ce qui se passe lorsqu'il est blessé lors d'un combat : "*Le Kid était sur le dos, à demi conscient, la peau couverte de serpents et de phosphènes. Écrasement de fichier, songea Jonny. Toutes les images de son logiciel ressortaient d'un coup, hors de contrôle. Le bras tendu de Kid-la-Glisse se mit à clignoter comme un stroboscope pris de folie : un bras de femme, un reptile, un robot industriel ; des araignées écarlates le couvraient de leur toile ; couleur d'ambre, des caractères alphanumériques défilaient sur son visage déformé par la douleur : Brando, Lee, Bowie, Véga ; le programme tournait en boucle, les visages s'enchaînaient de plus en plus vite, succession*

clignotante fondue en un unique méta-visage imaginaire, incolore, multicolore, pour s'évanouir à l'instant même où il se formait. La Glisse se releva, assis, jeta autour de lui un regard dément et partit d'un grand rire. Un ultime éclat de chiffres binaires, et il s'affala de nouveau."

Les images envahissent donc le réel, profitant du moindre support. Elles contaminent également l'intimité de l'être, au-delà de la peau. Elles sont imprimées directement sur la rétine. Transmises par le nerf optique, elles permettent déjà à un aveugle de voir *a minima* ; en SF, celui-ci voit en technicolor. Le mieux est encore de les diffuser directement dans le cerveau : l'idée à présent très répandue des broches à la base du crâne est presque devenue un lieu commun de la science-fiction, en littérature comme au cinéma.

Réalité augmentée

Les lunettes permettent surtout d'afficher des informations qui se surimposent à la vision normale. Dans *Lumière virtuelle*, de William Gibson, l'héroïne est poursuivie car elle est en possession d'une paire de lunettes qui, pour peu qu'on active sa lumière virtuelle verte, dessine les plans des projets immobiliers des promoteurs de San Francisco.

Cette réalité augmentée, qui trouve des applications dans de multiples domaines, depuis la chirurgie jusqu'à l'orientation dans l'espace, est surtout une réalité commentée, qui ajoute de l'information sur une image. Ce n'est pas neuf non plus : on met de la valeur ajoutée partout. Un plan, une photo sur laquelle on a entouré des bâtiments ou nommé une colline s'apparente déjà à la réalité augmentée. On parle bien, à propos de livres, d'édition annotée et augmentée. Une voix off sur un film augmente également l'information visuelle par un commentaire. Ce désir d'enrichir le réel de façon signifiante est si ancré en nous que nous ne le percevons même plus : à l'époque du cinéma muet, un pianiste illustrait l'action avec ses gammes ; depuis l'ère du parlant, la musique est un contrepoint indispensable à l'image, dont le rôle est de souligner les aspects émotionnels d'une scène. C'est si évident qu'on s'en étonne dans les rares cas où aucun thème sonore n'accompagne le film. La réalité a ensuite été contaminée par ce besoin d'illustration sonore : la musique dans les supermarchés, les apparitions d'artistes au cirque ou dans une émission télévisée jusqu'au candidat politique qui entre en scène, pardon, monte à la tribune, sur le rythme d'une marche triomphale.

Ce qui change, c'est l'interactivité et la simultanéité, tout cela s'effectuant en temps réel.

L'interaction avec l'image fait entrer l'homme dans une nouvelle dimension, où il manipule des objets à distance. Ce pouvoir d'agir à distance fait également partie de la longue marche de l'humanité. Les armes, depuis la lance jusqu'au fusil, en passant par les missiles à tête chercheuse, agissent sur des distances toujours plus grandes. J'avais remarqué, dans mon roman *L'univers en pièce*, que l'expression "voie de communication" était tombée en désuétude. On ouvre à présent des voies de transport. La communication n'est plus liée à la route, elle n'est plus physique mais immatérielle, grâce au téléphone, à la télévision, à Internet, et transporte de l'information, plutôt que des produits. Avant, il fallait se déplacer pour communiquer ; ce n'est plus nécessaire aujourd'hui. Le déplacement est supprimé car, bien qu'on ait besoin d'aller de plus en plus vite, il est presque impossible de se déplacer plus rapidement qu'on ne le fait aujourd'hui, en tout cas, d'aller plus vite que les outils technologiques, ces extensions à nous-même. Si le titre de mon roman s'écrit sans "s" à *pièce*, c'est parce qu'on préfère aujourd'hui faire venir le monde chez soi que sortir pour s'y déplacer.

La commande à distance passait jusqu'alors par une interface : un bouton à presser, une manette à relever. À présent, l'action est exécutée d'un simple geste, qui peut être un clignement de paupière, comme on l'a vu lors de la guerre du Golfe, où un pilote est désormais capable de tirer avec l'œil. Arthur C. Clarke avait stipulé, dans une de ses lois, que toute technologie suffisamment avancée est impossible à distinguer de la magie. Effectivement, un individu du moyen-âge, voire du XIX^{e} siècle, confondrait avec un magicien une personne capable d'allumer des lumières d'un simple claquement de doigt ou de demander l'ouverture d'une porte à l'aide de sa seule voix. Il la considérerait comme quasiment divine s'il la voyait manipuler des objets qui n'existent pas, comme une molécule de carbone. C'est une autre distance dont on s'affranchit ici ; elle n'est pas spatiale mais dimensionnelle. L'homme, qui a toujours cherché à étendre son domaine d'influence partout et à tous les niveaux, de l'infiniment grand à l'infiniment petit, avec des télescopes et des microscopes, change radicalement d'échelle en touchant ce qu'il ne pouvait jusqu'à présent manipuler que par outil interposé.

Cette fois, la réalité augmentée ne dispense pas l'information à l'aide des codes usuels, comme l'écriture ou les pictogrammes mais en se servant des "périphériques humains" par lesquels nous recevons de l'information. Elle sollicite de plus en plus nos sens : après la vue, l'ouïe, puis le toucher et l'odorat en attendant le goût. Les gants à retour d'effort, les casques de vision équipés de lunettes stéréoscopiques sont des outils qui n'agissent plus à distance mais qui agissent sur nous. C'est un fantastique renversement de perspective : le corps devient l'objet que l'on manipule via une interface. C'est lui qui devient un organe augmenté par les sensations qu'il reçoit.

Parler d'objet à propos de corps n'est pas anodin. Il est de plus en plus considéré comme tel. Les chirurgiens ressemblent à des plombiers s'occupant de problèmes de tuyauterie ou à des mécaniciens remplaçant les pièces défectueuses. Des analyses chimiques permettent de sonder cette machine biologique qu'on modifie

ensuite par l'injection de molécules. La nanotechnologie prévoit d'y envoyer des machines servant à évaluer, modifier, réparer des cellules. Le mariage avec la cybernétique permet d'informer en temps réel "l'occupant" de l'état de son "véhicule". Voici comment, en pleine action, un personnage de *L'École des assassins*, roman de Gilles Dumay et Ugo Bellagamba, reçoit dans son cerveau des informations en provenance de son corps : "*Rythme cardiaque: 55 pulsations/minute, maximum enregistré durant la course : 75 pulsations/minute, retour automatique à p.n.m. : 1% - OK, résorption nanomachines : 8 secondes*". Tout son environnement est ainsi analysé en permanence : "*Augmentation de la densité de la peau : maximum tolérable. Identification de la menace : 8 fusils d'assaut AEG cal. 5.56 – cartouches équipées de projectiles full metal jacket – vitesse de la balle à la sortie du canon 1120 m.s*$^{-1}$."

On comprend donc pourquoi Case, le voyou de *Neuromancien*, de Gibson, qui était auparavant "*branché sur une platine de cyberespace maison qui projetait sa conscience désincarnée au sein de l'hallucination consensuelle qu'était la matrice*" se sent devenir une épave quand on le prive de la possibilité de se connecter : "*Pour Case, qui n'avait vécu que pour l'exultation désincarnée du cyberspace, ce fut la Chute. Dans les bars qu'il fréquentait du temps de sa gloire, l'attitude élitiste exigeait un certain mépris pour la chair. Le corps, c'était de la viande. Case était tombé dans la prison de sa propre chair.*"

En fait, Case ne voit plus son corps que comme une interface. Il se déréalise car il ne se sent exister que dans la réalité virtuelle.

Une Virtualité bien réelle

A présent que l'homme a dominé la nature, façonné son environnement, il ne lui reste plus qu'à créer sa propre réalité dans laquelle il s'engouffrera car elle sera à sa (dé)mesure et à son goût. Cette fois, il ne s'agit plus seulement d'éprouver quelques sensations cinesthésiques ou de commander des objets à distance, mais de s'immerger dans le décor ainsi crée.

C'est bien vers ce but que tous les efforts humains ont tendu depuis des siècles. Un intérieur, une architecture, témoignent de la volonté de se doter d'un environnement adapté et non naturel. L'argent n'est rien d'autre qu'une forme de troc où un des éléments d'échange est devenu virtuel, remplacé par un symbole auquel on assigne une valeur. Plus tard, la monnaie est à son tour remplacée par des écritures sur papier dans un premier temps, lettre de change ou chèque, sous forme entièrement virtuelle ensuite, l'argent électronique transitant par carte de crédit ou par des saisies sur un écran d'ordinateur.

Le fait que l'argent soit plus apprécié qu'un objet réel de même valeur est que sa dimension virtuelle le rend, comme une cellule à son stade primitif, totipotente. Il peut se concrétiser en plusieurs objets. Il y a un fantasme de l'argent, dans ce qu'il représente comme possibilité de consommation, et il y a de même un fantasme de la réalité virtuelle. Ce n'est pas un hasard si chaque nouvelle révolution technologique, dont on vante d'abord les vertus pratiques, se développe paradoxalement par l'assouvissement du plus répandu des fantasmes, celui du sexe. C'était déjà le cas à l'époque du Minitel et du téléphone rose. Ça l'est encore avec Internet. Et on y songe pour l'exploration d'univers virtuels.

Cette superposition de sa propre création sur la réalité, avant l'évacuation de cette dernière, est à l'œuvre dans *Le Château des Carpathes*, de Jules Verne. Certes, l'auteur n'y décrit que l'invention du cinéma et du magnétophone, mais l'intrigue qu'il en tire est à nouveau basée sur une supercherie, un leurre : il s'agit de faire prendre pour réelles, dans un but de vengeance, l'image et la voix de Stilla, une chanteuse décédée. L'homme qui l'aimait croit devenir fou quand il voit et entend ce fantôme qu'il poursuit à travers les couloirs du château. Orfanik, l'inventeur, utilise bien ses instruments dans le but de faire prendre pour réel ce qui ne l'est pas. Il est symptomatique aussi de constater que ces inventions très modernes sont mises en scène dans une région passéiste, où survivent mythes et légendes : ce sont bien deux mondes qui se rencontrent.

Les premières manifestations de cette création d'un autre monde sont dans les communautés virtuelles, chats, forums, qui établissent une proximité avec des personnes distantes dans l'espace. Gibson lance le terme de *cyberespace* dans *Neuromancien*, accréditant l'idée qu'un nouveau territoire est à conquérir. Les images de synthèse, après avoir nourri les univers fantastiques des jeux, sont investies dans des villes virtuelles où on se promène, sur écran d'abord, en s'immergeant ensuite.

Les lieux virtuels finissent par être aussi fréquentés que les concrets, à la différence que tout y est plus rapide. Les sociétés y importent ou y adaptent l'organisation qu'elles avaient adoptées dans le monde réel. Dans *Le Samouraï virtuel*, de Neal Stephenson, le Métavers est un univers virtuel établi le long d'une rue d'un milliard de kilomètres de long où l'on travaille et s'amuse sous l'apparence de son choix. Les moins fortunés ne disposent que d'un avatar en noir et blanc et se projettent depuis les cabines publiques aussi répandues que celles du téléphone de nos jours.

Dans *Nécroville*, de Iain McDonald, voici comment une avocate se connecte au tribunal virtuel pour y plaider une affaire devant un juge tout aussi immatériel puisqu'il s'agit d'une Intelligence Artificielle capable de gérer les procès avec l'impartialité et l'objectivité requises, sans jamais se lasser :

"1er *novembre 20 : 30 : 35 : 50. Temps moyen de Greenwich. Affaire numéro 097-0-17956-67-01.*

Dans une chambre en papier proche de Sunset, Yo-Yo Mok enjamba le cadre d'une fenêtre événementielle et arriva sous Zurich, à deux mille mètres de profondeur.

Zwingli II était impressionnant. Ses concepteurs suisses l'avaient conçu pour inspirer du respect envers les procédures quasi divines de la justice. Ils avaient atteint leur but. Elle en restait bouche bée. Chaque fois.

Yo-Yo se retrouvait sur une étroite corniche à un tiers de la hauteur de la face interne d'une pyramide qui, si elle avait été réelle, l'eût surplombée de huit kilomètres. Quatre kilomètres plus bas, sa base eût recouvert la majeure partie du Secteur Métropolitain de la Reine des Anges. Les parois noires de la construction virtuelle frissonnaient et ondulaient de coulées lumineuses colorées : les logs légaux ne pouvaient franchir le portillon de l'arène où seuls des esprits humains avaient l'autorisation de s'affronter. Mais elle les savait derrière elle, et leur présence lui apportait de l'assurance, de l'audace. Redresse toi, Yo-Yo. Du calme. Du calme. Détends toi. Garde la tête aussi froide que les processeurs immergés dans du CO_2 liquide de Zwingli II. Des étoiles scintillaient à l'intérieur du volume démesuré, des constellations qui brûlaient et mouraient. En permanence. L'ordinateur judiciaire traitait simultanément soixante-dix mille affaires.

La peau de la pyramide ondoya sous ses pieds et cracha un pont sur l'abîme miroitant.

Debout. La séance est ouverte.

Elle tendit un doigt gainé de noir et ceint d'argent et fut propulsée sur l'étroite passerelle. Un astre se détacha de l'arrière-plan galactique et vint à sa rencontre en acquérant de la substance et de la netteté.

Mon adversaire. Pas d'outrecuidance, pas d'orgueil mal placé. Ne va surtout pas t'imaginer que deux cents Go de logs légaux *corporadistos* te permettront d'écraser à plate couture ces ploucs en djellaba. Ici, Zwingli est le seul Dieu.

Elle ouvrit la main et descendit au centre du tablier convexe du pont. Le vide au-dessus d'elle. Le vide au-dessous. Des étoiles qui brasillaient. Son collègue était devenu un fantasme de jambes et de bras surmonté d'une tête. Un homme stellaire. Avec la rapidité surnaturelle propre à la virtualité, il se posa devant elle."

La possibilité de recréer un environnement de son choix incite à rhabiller de même le réel. La décoration de la maison abandonne le papier peint pour l'image de synthèse qui autorise les rêves les plus fous, tel celui de Consuela, qui, dans *Les Synthérétiques* de Pat Cadigan, a transformé son appartement en aquarium :

"*Il entra et se retrouva sous l'eau.*

Des rubans d'algues fluorescents de toutes les couleurs dressaient leurs molles ondulations au-dessus du plancher océanique, éclairant d'un feu froid la semi-obscurité. Gabe hésita, laissa la porte se refermer dans son dos puis avança d'un pas. Son pied passa au travers du plancher d'aspect pâle et mou et disparut ; il sentit au-dessous un plancher plus conventionnel mais sans que l'illusion visuelle se dissipe. Consuela se débrouillait comme un chef ; seuls les gens richissimes ou les grosses boîtes comme Alternatives avaient des projecteurs de cette qualité.

Une pieuvre d'un pourpre lumineux rampa au sommet d'un rocher qui lui arrivait à la tête ; l'animal le regarda, faisant mouvoir ses tentacules avec une grâce sensuelle ; un poisson couvert de piquants sortit de l'ombre et lui passa sous le nez comme un dirigeable compassé. Gabe plissa les yeux. Pas tout à fait un poisson : les piquants étaient des aiguilles plantées sur des puces de silicium en guise d'écailles. Les énormes yeux marron l'examinèrent avec une solennité glacée.

« Que voulez-vous ? » lui demanda le poisson d'une voix de contralto féminin, dont le léger accent lui était resté familier.

« Salut, Consuela… C'est moi. Gabe Ludovic. »"

Ce qui n'est encore qu'un décor immatériel prend vite davantage de texture avec l'introduction d'effecteurs propres à apporter des sensations cénesthésiques. La combinaison munie de capteurs s'est bien vite imposée comme le vêtement standard du virtuel. On suppose que ce genre de gadget, dans un premier temps, ne sera pas donné et qu'il sera disponible, comme actuellement Internet dans les cybercafés, dans des lieux réservés à cet usage, ce qui pose tout de même des inconvénients. Pat Cadigan, dans *Vous Avez Dit Virtuel ?*, en a tout de suite pris la mesure : "*– Attendez, répond-elle en agitant les bras pour diffuser l'odeur. D'ici une seconde, on ne sentira plus* rien*. Ce truc-là vous endort le nez. On en utilise des* tonnes *ici, à cause des odeurs corporelles. Les combis* puent*, voyez-vous. (...) On est obligés d'attacher les clients sur des couchettes, sinon ils bousilleraient les combis à force de se rouler par terre et de se jeter contre les murs.*"

Comme quoi, le réel ne se laisse pas effacer comme ça. Les effets, par contre, sont avérés. Même en ce qui concerne les dommages physiques : "*Une fois, il y en a qui s'est blessé avec, sans mentir ! A force de s'exciter, il a réussi à s'entailler avec les sangles. Même qu'il avait des côtes cassées. Et vous savez la meilleure ? (...) La meilleure, c'est qu'au* même *moment, sa persona était prise dans une bagarre et qu'elle s'est cassé les* mêmes *côtes.*"

Cela peut-il aller jusqu'à la mort ? Elle se résume à une déconnexion dans *Le Samouraï virtuel*. Dans *Les Synthérétiques*, une forte émotion peut provoquer une crise cardiaque. J'ai cependant supposé, dans *Petites Vertus virtuelles*, qu'une combinaison bien conçue serait équipée de capteurs qui préviendraient ce risque en arrêtant l'immersion virtuelle à la moindre alerte. Évidemment, le jeu consiste à chercher un moyen de contourner ces précautions. Dans mon roman, l'affaire n'était possible qu'à condition de connaître

l'environnement de l'utilisateur pour l'amener à accomplir des actes dangereux, par le biais d'illusions. C'est ainsi qu'un personnage mord un fil électrique sans le savoir.

Pour éviter de sangler un internaute du futur sur une couchette, il convient de lui allouer un espace dans lequel il peut se déplacer : des salles nues sont conçues à cet effet, qui se rempliront d'un décor virtuel au sein duquel évoluer. Pour se connecter dans un café virtuel, le héros de *L'Univers en pièce* prend la précaution de placer une bouteille réelle à l'emplacement où est censée se trouver celle que le serveur lui apportera, afin de pouvoir réellement trinquer avec un interlocuteur physiquement absent. David Brin, dans sa nouvelle *La Vie naturelle™*, imagine que la pièce repose sur un tapis qui suit les mouvements de la personne et se déforme pour simuler les accidents de terrain.

La simulation totale reste cependant l'immersion, grâce à une connexion neuronale reliant l'homme et la machine. Les sensations n'ont plus besoin d'effecteurs : le rêve a, sur le plan émotionnel, les mêmes effets que l'état de veille. S'immerger, de façon consciente, dans une autre réalité, en activant des zones du cerveau revient à peu de choses près à consommer de la drogue ; les paradis artificiels sont également des substituts à la réalité, à la différence qu'on n'en contrôle pas réellement ses effets. L'avocate de *Nécroville* recevant instantanément les analyses et les sentences du juge virtuel est bien consciente de cette analogie :

"Les données actives des murs fusionnèrent en nœuds serrés à la blancheur stellaire brûlante et s'élancèrent sur l'étroit pont noir. Leur flot traversa Yo-Yo tel un raz-de-marée igné. Nul plaisir terrestre ne pouvait être comparé à cette pénétration d'une micro seconde, au goût de l'omniscience savouré à l'instant où les logs légaux éjaculaient des Go d'arguments dans le système.

Si Yo-Yo était une piètre avocate, elle était une toxicomane hors pair."

Cette connexion n'est pas sans danger : les virus peuvent à présent passer dans le cerveau, les expériences traumatisantes transformer l'utilisateur en légume, à moins que celui-ci ne se perde dans un espace virtuel d'où il ne revient jamais.

Jean-Marc Ligny, dans *Inner City*, distingue la Basse Réalité, le réel, et la Haute Réalité, celle du virtuel, plus une troisième catégorie, la Réalité Profonde, une sorte de no man's land, un abîme virtuel où disparaissent les *Inners* ayant craqué leur console pour empêcher la déconnexion au bout de vingt-quatre heures. Des équipes spécialisées dans la récupération des Inners en détresse s'emploient à les retrouver dans le cyberespace avant la mort du voyageur, par déshydratation, faim ou épuisement. N'oublions qu'en Corée est récemment mort un joueur qui était resté connecté dans un cybercafé 87 heures d'affilée. Les Inners s'égarent au point de plus pouvoir réintégrer leur corps quand ils entrent en résonance avec l'inconscient collectif : "*les inners visualisent leur propre inconscient, leurs fantasmes, leurs désirs, mais aussi leurs angoisses, leurs frustrations, etc.*", ce qui les bloque dans une sorte d'état autistique dont ils ne ressortent pas seuls.

On pourrait penser que ces incursions restent sans incidence sur l'utilisateur soucieux de sa sécurité physique, hormis peut-être une tendance au dédoublement de personnalité à force de vivre entre deux univers, ce qu'illustre *Les Deux Sam*, une nouvelle de Robert Reed où le protagoniste ne parvient pas à choisir entre sa famille, notamment de sa femme atteinte d'un cancer, et son univers simulé dont il est le roi. Existe-t-il un risque de confusion avec le réel ? Le protagoniste de *La Vie naturelle™* est gêné d'éprouver un désir physique pour une créature virtuelle à qui il a sauvé la vie, dans une simulation de la vie préhistorique : "*Puis il y eut la pression de son corps chaud qui se collait au mien, et qui s'avéra beaucoup plus confortable, en certains endroits, que je ne l'avais imaginé. (...) Bientôt, je réalisai que Cheville de Girafe ne s'agrippait plus à moi pour être réconfortée. Elle bougeait, et respirait, d'une façon qui n'avait rien à voir avec le réconfort moral.*" La confusion avec le réel est notée par ce raccourci saisissant : "*Ce n'était qu'un logiciel – des morceaux d'illusion sur une puce de silicium. Et puis, je la connaissais à peine.*" Toute l'ambivalence du virtuel est là, dans les effets réels provoqués par quelques octets. Le problème du narrateur est de savoir, si, en acceptant les avances de cette femme, il commet ou non un adultère. La question est vite résolue : voulant en parler à sa femme, il la trouve dans sa chambre de Virtualité, accroupie sur le sol qui s'est déformé pour dessiner les contours d'un homme. Son épouse avait depuis longtemps résolu la question. Il n'en va pas toujours de même dans la réalité : un Israélien a demandé à divorcer au motif que sa femme le trompait en se connectant sur des sites pornographiques.

Ce risque de confusion est cependant mineur. Dans la plupart des romans, la distinction est toujours bien établie entre le monde réel et l'univers virtuel. Seul un individu aliéné est susceptible de se comporter dans la réalité comme s'il évoluait dans une simulation. Dans *Idoru*, de William Gibson, Rez, chanteur célèbre, projette d'épouser une star du petit écran qui n'est rien d'autre qu'une idoru, c'est-à-dire une créature virtuelle. Mais Rez est lui-même une icône, qui tente d'échapper à ses fans hystériques. Ce "mariage alchimique" n'est pas la preuve d'une confusion mais un défi, une démarche artistique et philosophique, qui vise en même temps à développer l'intelligence artificielle vers une véritable personnalité humaine.

Quand les protagonistes ignorent qu'ils se trouvent dans un univers de synthèse, ce n'est pas par confusion mais parce qu'une machination occulte la vérité. C'est le cas du film *Matrix*, où la machine domine l'homme, et aussi celui de *Dark City*, où des extraterrestres ont falsifié les souvenirs des habitants pour les plonger dans un environnement qui n'est qu'un simulacre dont ils peaufinent l'architecture chaque nuit, à minuit, en arrêtant le temps. Dans ces exemples aussi, une fois admise l'existence d'un réel caché, la distinction entre

univers concret et virtuel est faite.

Mais le fait de faire la distinction n'empêche pas les utilisateurs de préférer l'univers virtuel au point de ne pas vouloir réintégrer le réel, forcément plus plat et plus trivial. Déjà, dans *Futur Intérieur*, de Christopher Priest, certains membres d'une ville virtuelle, qui sert de laboratoire social dans la mesure où on y introduit des problèmes contemporains afin de voir comment la société factice les résout, refusent de revenir dans la réalité. Leurs corps, artificiellement nourris dans des caissons, massés par des intervenants extérieurs, dépérissent lentement. Pour les obliger à se réveiller, il suffit d'agiter devant leurs yeux un miroir ; mais les plus malins parviennent toujours à éviter les agents chargés de les ramener.

Les contempteurs de la réalité virtuelle dénoncent cependant les ravages que peuvent provoquer une mauvaise utilisation des univers virtuels : perte de la communication réelle, enfermement dans des fantasmes, affadissement du réel. A se perdre dans le virtuel, on en vient à oublier que le concret n'a pas que des inconvénients. Dans *Inner City*, des protagonistes égarés dans la Basse Réalité redécouvrent les rugueuses sensations du réel et des plaisirs terrestres, notamment au contact de deux étonnantes grands-mères amatrices d'alcool et de hasch.

Pour Baudrillard, "*l'absence des relations des gens par rapports aux autres, l'absence de soi par rapport à soi-même, la non-identité définitive des choses"* aboutit à une perte de sens car les référents s'effacent. On ne sait plus quelle est la part de réalité derrière les images de synthèse. Baudrillard se défend d'avoir un jugement moral sur des faits de société mais s'insurge contre le mensonge fait à nous-mêmes, le fait que nous nous illusionnions et que nous remettions "*en cause le principe de réalité"*. Ce que Gibson caractérise comme une *hallucination consensuelle* établit une hiérarchie entre les deux univers, qui se fait de plus en plus au détriment du réel.

A cet égard, l'acte de foi de l'avocate de *Nécroville* est éclairant :

"*Le réseau est un domaine. Un potentiel. Un état. Une hallucination. Une zone intermédiaire. Un défi lancé aux définitions spécieuses. Un article de foi. Un credo.*

Je crois en l'inviolabilité des mathématiques pures, appliquées et statistiques, créatrices et nourricières de toutes les connaissances, langage sacré par lequel les réalités de l'univers sont le plus justement exprimées. Je crois en la physique, la chimie, la biologie, la théorie quantique et la relativité générale, l'informatique et le chaos (bien qu'il me soit impossible de faire un choix entre l'indécidabilité de Gödel et les incertitudes d'Heisenberg). Je crois au Saint-Esprit de l'Information, aux journaux télévisés, à mes relevés de compte bancaire, à la musique de ma chaîne hi-fi, aux amis qui apparaissent sur l'écran digital de mon Idcom. Je crois en la résurrection nanotechnologique des corps et en la vie éternelle. Amen.

J'y crois parce que j'ai la preuve que ça marche. Je n'ai nul besoin d'en comprendre les mécanismes. Je sais que c'est efficace. Les gris-gris de la science ont un sérieux avantage sur les autres. La piété et la foi ne sont pas nécessaires pour permettre d'atteindre le but recherché. Il suffit pour cela d'avoir de l'argent. Yahvé a fait tomber la manne avec la rosée du matin pour nourrir les enfants d'Israël, mais à cette exception près ce sont par les réseaux de virtuel-achat qu'on obtient du lait et du miel.

Comme toute croyance, c'est un pur produit de l'esprit. Or, les esprits évoluent et, avec eux, les doctrines sur le mode de fonctionnement du monde. Les modèles changent."

Mettre le réel et le virtuel sur le même plan donne plus de crédibilité à ce dernier. Il n'est pas si évanescent ni dénué de conséquences qu'on veut bien l'affirmer. Citant Nietzsche qui disait que les caméléons changent mais ne *deviennent* pas, Baudrillard affirme qu'un adepte du virtuel ne devient rien de plus quand il revêt plusieurs identités lors de ses connexions. C'est vrai et faux à la fois. Avancer masqué dans le but de tromper autrui sur sa nature ou s'amuser à devenir un autre un peu plus valorisant est une façon de se leurrer effectivement stérile, identique au mythomane qui cherche à impressionner son entourage par des mensonges. Mais revêtir pour un temps une autre personnalité, vivre des expériences qui demeurent virtuelles, ne sont pas sans effet sur l'individu qui en sort changé. Un livre ou un film totalement imaginaire a bien un impact émotionnel, quand bien même on sait qu'on ne réagit qu'à une fiction. Une lecture peut transformer une vision du monde, avoir des résonances qui modifient un individu, dans sa façon de penser et d'être. On s'enrichit également de l'expérience des personnages fictifs. On en revient à la définition de la science-fiction, qui est une "*exploration de la virtualité rationnelle*" pour reprendre l'expression de Gérard Klein : puisqu'on "*se perd en conjectures sur les conséquences à venir des univers virtuels, (...) faute d'expérience et de recul, c'est sans doute à la lecture des textes de Science-Fiction que l'on peut rencontrer les réflexions les plus avisées*", écrit-il dans sa préface à *L'Âge de diamant*, de Neal Stephenson.

Dans *L'Âge de diamant*, justement, un manuel interactif destiné à l'éducation des petites filles aisées tombe entre les mains d'une gamine pauvre, défavorisée par la vie. Ce livre, qui s'adapte à son utilisateur, par les contes métaphoriques qu'il imagine, par des jeux et des exercices, parvient à si bien transformer la fillette qu'elle devient l'un des plus importants personnages de sa société, qu'elle contribuera à remodeler.

Jadis, on partait en quête de soi, de son identité, en voyageant. L'exploration était le moyen de se confronter au réel pour se connaître. À présent le voyage est virtuel, mais garde la même fonction. Sandy Torrès, sociologue, note qu'"*indépendamment des formes qu'elle peut revêtir, la fiction autorise des concrétisations de*

notre «pouvoir-faire»", elle est un lieu où des mondes possibles peuvent être éprouvés. Les fictions, et le virtuel également "*donnent corps à nos désirs aussi bien qu'à nos craintes et permettent ainsi de prendre la mesure de notre liberté et de nos possibilités d'action"*.

Vers l'abstraction

Cependant, mettre le réel et le virtuel sur le même plan revient également, par réciprocité, à "virtualiser" le réel.. Des penseurs et philosophes ont toujours émis des doutes sur la réalité du monde ou interrogé sa nature et la fiabilité de nos perceptions, mais cette fois il s'agit d'une remise en question plus radicale. Dès 1964, Galouye posait le problème dans *Simulacron 3*. Simulacron 3 est un simulateur d'environnement total, qui a crée une société électronique en tous points conforme à la nôtre, afin d'étudier les réactions de la population virtuelle face à certains changements. Les créatures électroniques n'ont aucune conscience d'évoluer dans un simulacre. Le concepteur découvre alors qu'il en va de même pour son univers, recréation d'un méta-univers semblable au sien. "*Au sommet de la colline, une terreur glaciale me paralysa. (...) A une centaine de mètres plus loin, la route s'achevait brusquement. Au-delà, le paysage s'interrompait net, comme tranché au couteau ! De chaque côté du ruban d'asphalte, la terre elle-même sombrait dans une impénétrable barrière de ténèbres absolues.*"

Le vertige provoqué par cette prise de conscience débouche sur un sentiment de néant. Rencontrant une femme de cet univers supérieur, qui a projeté son esprit dans le simulacre afin de l'observer, le protagoniste avoue son désarroi :

"– Mais Jinx, je ne suis rien !

Elle me sourit.

– Moi non plus, en ce moment, je ne suis rien.

– Mais tu es réelle ! Tu as une longue vie physique devant toi ! (...)

– Non, Doug. Rien ne prouve que, même dans mon monde, les choses matérielles aient une substance réelle. Quant à l'esprit, qui a jamais prétendu qu'il dût avoir un support physique à sa mesure ? S'il en était ainsi, un nain ou un amputé en détiendrait moins qu'un géant hyperthyroïdien. Et ce que je dis est valable pour tous les mondes. (...) C'est l'intellect qui compte (...). S'il existe une vie spirituelle, elle n'est pas davantage refusée aux unités de ce monde qu'à celle du simulateur de Fuller ou aux gens "réels" de mon monde."

Tout est dit. Et cette future acception de la conscience augure de la façon dont il faudra un jour considérer les intelligences artificielles, des entités dignes du même respect et des mêmes droits que ses concepteurs.

Il n'en reste pas moins qu'on va vers une spiritualisation du monde, une abstraction qui en efface les contours tangibles. Mais ce mouvement n'est que l'aboutissement de ce à quoi a toujours tendu l'homme. On a vu combien les réalisations du passé, notamment à travers l'art, amorçaient une immersion dans une réalité virtuelle. Dans le domaine de l'image, Yann Minh, plasticien et infographiste, observe que de tous temps les peintres ont cherché à représenter de façon réaliste des événements, des décors et des lieux imaginaires[1]. Sa filiation va de Breughel à Christopher Foss et Manchu en passant par les peintres de la Renaissance. Ce qu'il nomme "*hyperréalisme immersif qualifie une démarche spécifique, une motivation d'auteur, à la fois dans le monde des arts plastiques, comme dans celui du cinéma ou du jeu vidéo, de favoriser l'immersion du spectateur dans une cosmogonie imaginaire et spéculative, en s'efforçant de simuler le plus précisément possible nos modes de perceptions sensuels et sensoriels, tout en investissant l'œuvre d'une charge émotionnelle forte. En particulier, en simulant de la façon la plus réaliste possible nos perceptions visuelles, mais aussi nos perceptions auditives, et parfois tactiles. (...) L'hyperréalisme immersif est l'expression de notre besoin ancestral de pouvoir communiquer et transmettre de l'information, de l'émotion à nos semblables avec le plus d'efficacité et de fiabilité possible. Quoi de plus efficace en termes de transmission de l'information que de pouvoir immerger l'interlocuteur dans une cosmogonie artificielle et maîtrisée qui sera d'un réalisme en tous points semblable à ce que nos sens nous font percevoir de la « réalité » à chaque instant ?*" Pour Yann Minh, l'exploration de la noosphère n'est que le dernier avatar de cette immersion toujours plus totale.

Finirons-nous, comme dans le livre de William Hjortsberg, *Matières grises*, dans des boîtes abritant nos cerveaux, nos consciences, dans l'attente d'un corps ou comme les protagonistes numériques de *Jour de noces*, de Pierre Bordage, qui attendent que la terre soit à nouveau habitable ? Ou encore comme ces entités désincarnées d'*Un Feu sur l'abîme* qui se passent désormais de support physique ? Pour Philippe Quéau, directeur de l'information et de l'informatique à l'UNESCO, "*le virtuel est en train de devenir le paradigme fondamental de notre civilisation"*. "*La véritable réalité de l'homme est dans sa virtualité, dans sa capacité virtuelle à transformer le monde. (...) Le virtuel est au sens propre une réalité intermédiaire"*. S'il nous aide à mieux comprendre le monde, et à nous réaliser, le danger que repère Philippe Quéau est "*de nous désensibiliser, de nous couper de nos racines humaines et sociales les plus profondes"*. Ce risque de désincarnation n'est pas nul, mais peut-être faut-il y voir, plutôt qu'un anéantissement, une étape vers une autre forme d'humanité et de réalité,

[1] Son argumentation est lisible sur son site : http://www.yannminh.com/hyperealism/

un état supérieur permettant également d'accéder à une conscience supérieure.

La Cité des permutants de Greg Egan, présente un individu ayant réussi à réaliser une copie numérique de lui-même. C'est là qu'il constate que l'homme peut survivre sans support informatique : la trame de l'univers devient l'assise de l'esprit. Il accède en quelque sorte à l'immortalité et s'enrichit en proposant le procédé à des milliardaires qui, devenus simulations, vivent dans la cité virtuele parfaite, Permutation City, qu'il a crée à leur effet. À présent, seul ce qui est pensée existe. Egan pose les problèmes d'identité lié à la duplication, qui peut être en grand nombre ; une copie est-elle encore humaine ? Les abîmes métaphysiques qui s'ouvrent sous nos pieds, sous-tendus par des théories cosmogoniques et de physique quantique, donnent le vertige.

Nous ne serions plus que de l'information qui se moque de savoir quel support elle utilise, support qui est d'ailleurs est en passe de se modifier plus vite que prévu par les miracles de la génétique. Cité par Norman Spinrad, Greg Bear avait un jour demandé dans un débat sur le courant cyberpunk : "Combien d'entre-vous pensent-ils que les gens auront un aspect reconnaissable comme humain d'ici cinquante ans ?" En voyant toutes les mains levées, il répondit : "Vous vous trompez tous." Son roman *La musique du sang* présente de l'ADN utilisé comme mémoire vive d'ordinateur. L'expérimentateur s'injectant les noocytes pour éviter qu'o ne détruise ses réalisations jugées dangereuses voit son corps se transformer, les noocytes repérer le siège de l'intelligence, puis contaminer l'univers entier, dissociant les molécules de la matière pour les combiner différemment et intégrer les personnalités de l'humanité dans une seule méta-conscience où chacun garde cependant son individualité. L'esprit supérieur ainsi crée coupe ses liens avec le monde physique pour poursuivre ailleurs son évolution.

Les réalités virtuelles nous entraînent décidément très loin. Mais cette désincarnation ne correspond-elle pas à notre nouvelle perception de l'univers ? Les récents développements de la physique tendent à prouver que la matière est faite d'information et que les paradoxes de la physique quantique se dissipent lorsqu'on considère celle-ci sous l'angle de l'information. De la sorte, nous ne ferions, une fois de plus, que nous conformer à la réalité, à y coller au plus près plutôt que de nous en éloigner.

Le réel est devenu immatériel, voire subjectif comme chez Greg Egan. Faut-il s'en inquiéter ? Le déplorer ? Ou l'accepter comme une évolution inéluctable à laquelle l'humanité accèdera un jour, s'il ne se détruit pas entre-temps. Personnalité électronique évoluant dans un univers virtuel, nous aurons aimé et souffert, éprouvé des sensations, nous aurons échangé de l'information. Nous aurons mis de l'intention, donc du sens quel que ce soit le niveau de réalité et d'abstraction où nous nous situerons. En d'autres termes, nous aurons vécu.

Tout ceci n'est bien sûr que spéculation science-fictive et rien ne permet d'affirmer que ces futurs virtuels se concrétiseront. Mais le simple fait d'évoquer ces fictions qui vont aussi loin que l'imaginaire peut porter permet de poser sur notre présent un regard, qui je l'espère, l'enrichit ou tout au moins le questionne utilement.

Claude Ecken

BIBLIOGRAPHIE DES TITRES CITES :

Éon	Greg Bear	Le Livre de Poche 7162
La musique du sang	Greg Bear	J'ai lu 2355
Jour de noces	Pierre Bordage	in Galaxies 21, été 2001
La Vie naturelle™	David Brin	DLM, Cyberdreams 01
Les Synthérétiques	Pat Cadigan	Denoël, Présence du futur 537 & 538
Vous Avez Dit Virtuel ?	Pat Cadigan	J'ai lu 6407
L'École des assassins	Gilles Dumay, Ugo Bellagamba	Bifrost/Etoiles vives
L'Univers en pièce	Claude Ecken	Fleuve Noir 1521
Petites Vertus virtuelles	Claude Ecken	Baleine, Macno 11
La Cité des permutants	Greg Egan	Livre de Poche 7224
Neuromancien	William Gibson	J'ai lu 2325
Lumière virtuelle	William Gibson	J'ai lu 3891
Matières grises	William Hjortsberg	Pocket 5129
Métrophage	Richard Kadrey	Denoël, Présence du futur 491
L'Enfance attribuée	David Marusek	Bifrost/Etoiles vives
La Tour des rêves	Jamir Nasil	Pocket 5758
1984	George Orwell	Folio 822
Futur intérieur	Christopher Priest	Pocket 5335
Les Deux Sam, in *Chrysalide*	Robert Reed	Imaginaires sans Frontières
L'Âge de diamant	Neal Stephenson	Livre de Poche 7210
Le Samouraï virtuel	Neal Stephenson	Livre de Poche 7221
Le Château des Carpathes	Jules Verne	Livre de Poche
Remake	Connie Willis	J'ai lu 4429

Bibliographie critique :

Roger BOZZETTO, *Science-fiction et ordinateurs : un mariage d'amour ou comment l'imaginaire nourrit-il la technique au point de lui fournir des idées ?*, in *Métaphores* n°20-21-22, *Actes du quatrième colloque international de science-fiction de Nice*, tome 1, Centre d'étude de la métaphore, Nice, 1992.
Jean-Claude DUNYACH, *Science-fiction et ordinateurs*, in *Galaxies* n°1, Nancy, 1996.
Groupe ETI, *Rencontre avec Jean Baudrillard*, in *Zénon* n°6, Toulouse, 2000.
Gérard KLEIN, Préface à *L'Âge de diamant*, Neal Stephenson, Livre de Poche 7210, Paris, 1998
Yann MINH, *L'hyper-Réalisme immersif en art plastique et en SF*, http://www.yannminh.com/hyperealism/, 2002
Louise POISSANT, *Réel ou virtuel : l'art et les nouvelles technologies*, in *Métaphores* n°20-21-22, *Actes du quatrième colloque international de science-fiction de Nice*, tome 2, Centre d'étude de la métaphore, Nice, 1992.
Philippe QUÉAU, *Multiplicités virtuelles*, in *Zénon* n°6, Toulouse, 2000.
Norman SPINRAD, *Les Neuromantiques*, in *Univers 1987*, J'ai Lu 2169, Paris, 1987.
Bruce STERLING, *Préface* à *Mozart en verres miroirs*, Anthologie de Bruce Sterling, Folio SF 49, Paris, 2001
Sandy TORRES, *Espaces du cinéma de science-fiction*, in *Zénon* n°6, Toulouse, 2000.

Je tiens à remercier à Yann Minh pour son aide ainsi que sa relecture attentive et critique.

Validation d'un processus de traitement allant de la capture du mouvement à l'immersion de sujets en réalité virtuelle : application au tir au handball

Benoit Bideau[(1)], Laetitia. Fradet[(1)], Franck Multon[(1)], Stéphane Ménardais[(2)], Richard Kulpa[(1)], Bruno Arnaldi[(2)]

1-Laboratoire de physiologie et de biomécanique de l'exercice musculaire av charles Tillon 350044 Rennes

2-IRISA projet SIAMES campus universitaire de Beaulieu 35042 Rennes

Benoit.Bideau@uhb.fr

Résumé : . *Cet travail consiste à évaluer toute la chaîne allant de la capture du mouvement à l'immersion de sujets en réalité virtuelle. En effet, de nombreux paramètres peuvent faire que le sujet ne réagisse pas comme dans le monde réel : qualité des modèles géométriques, mise à l'échelle de la scène projetée, réalisme des mouvements, comportement des humanoïdes… Dans la littérature, l'impact global de ces paramètres a été étudié en évaluant la présence. Cependant, encore peu d'études ont cherché à évaluer chaque paramètre isolément. Nous proposons une nouvelle méthode pour évaluer un processus particulier permettant d'immerger un gardien de handball dans un environnement virtuel habité de joueurs synthétiques. Une étude préalable de la gestuelle du gardien en situation réelle nous permet de vérifier que ses réactions dans l'environnement virtuel sont réalistes. Dans cet article, nous insistons sur le processus qui a permis d'animer les joueurs synthétiques afin de montrer que ces techniques d'animation produisent des mouvements suffisamment réalistes pour déclencher des réponses réalistes du gardien de but.*

Mots-clés : Evaluation de la présence, capture du mouvement, application sportive, réalité virtuelle, animation

1. Introduction

L'utilisation de la réalité virtuelle est de plus en plus répandue. Cependant, on peut s'interroger sur la manière dont les sujets perçoivent l'environnement virtuel. La présence dénote la sensation subjective d'un sujet d'être dans le monde virtuel. Cette sensation a été souvent étudiée au travers d'études comportementales [SVK01]. L'environnement virtuel doit être le plus proche possible du monde réel, comme la montré Hodgins [HOT 98]: la qualité du model géométrique semble jouer un rôle important dans le réalisme des scènes animées. En plus du réalisme graphique, les humanoïdes synthétiques peuplant l'environnement virtuel doivent agir comme des acteurs réels.

Le mode d'interaction avec l'environnement virtuel a aussi un rôle important à jouer. Différentes formes d'interactions peuvent avoir lieu entre les différents agents. Noser [NPCM96] a expérimenté l'interaction entre un joueur de tennis réel (représenté par son avatar) et un joueur virtuel. Le joueur virtuel était un agent perceptif et interactif guidé par un modèle comportementale. Cependant l'avatar était simplement représenté par une partie du corps composé du bras et de la raquette. Le mouvement capturé du joueur réel était rejoué avec ce bras virtuel sans tenir compte des détails du geste. De même, les effets donnés à la balle ainsi que les déplacements complexes du joueur sur le cours n'étaient pas pris en compte. Molet [MBT96] a réalisé une expérience avec deux joueurs qui interagissent entre eux dans un environnement distribué via VLNET [CPNMT97]. Les deux joueurs voyaient leur propre avatar (bras et raquette) jouer avec l'humanoïde virtuel. Comme dans l'étude précédente, l'utilisateur ne pouvait pas jouer au tennis comme dans le monde réel. Cette forme de jeu virtuel ne peut pas être directement appliquée à l'étude de mouvement sportif d'athlètes de haut niveau.

D'autres applications sont dédiées à l'évaluation et l'entraînement de sportifs. Par exemple la réalité virtuelle a été utilisée pour tester des stratégies en sport collectif en immergeant un entraîneur dans une phase de jeu simulée [MM00]. Dans ce jeu l'entraîneur doit donner des ordres aux joueurs synthétiques pendant que les opposant sont dirigé par un modèle comportementale. Cependant les comportements simulés ne sont pas comparés à une situation réelle et l'entraîneur ne réagit pas comme dans une vrai match mais utilise des métaphores pour diriger son équipe. D'une autre manière des bobsleigheurs se sont entraîné sur un simulateur pour les jeux olympiques d'hiver [HH96]. Dans cette étude le simulateur a été créé pour que le conducteur du bobsleigh réagisse comme dans le monde réel, sans vraiment le vérifier.

Pour résumer, les travaux antérieurs sur la réalité virtuelle sont généralement basés sur des métaphores pour animer les avatars. Toutefois, on peut s'interroger sur le réalisme des réactions des sujets, avec ce type d'interaction. Nous proposons de mettre en place une expérimentation entre un gardien de but de handball réel et un tireur virtuel. Cette expérimentation vise à valider les choix techniques qui sont mis en œuvre pour immerger le sujet dans un terrain virtuel de handball. En particulier, la qualité de l'animation de l'avatar est essentielle pour assurer un bon niveau de présence.

Pour animer les acteurs virtuels, plusieurs techniques peuvent être envisagées. Les techniques de motion warping ont été expérimentées en modélisant les trajectoires dans le domaine temporel [WP95] ou fréquentiel [UAT95]. L'inconvénient principal du domaine fréquentiel est le manque de la contrôlabilité du mouvement résultant. En effet changer le poids d'une harmonique n'est pas intuitif et nous conduit à un processus itératif d'essai-erreur. Le fait d'utiliser des points de contrôle ou d'ajouter des contraintes spatio-temporelles [WK88] est plus intuitif. Les mouvements capturés doivent être corrigés afin de pouvoir être employés pour une telle méthode. Par exemple, le bruit doit être filtré et des corrections anatomiques doivent être exécutées [MBT96][BRRP97]. Pour notre application, l'animation procédurale [Z82][BMT90][BC96] et la simulation dynamique [ADHMT89][HWBO95][MNH99] ne sont pas appropriées. En effet, même pour des modèles précis, les mouvements synthétiques ne peuvent pas être comparés à ceux mesurés dans de vrais phases de match.

Une dernière approche consiste à adapter des mouvements capturés à l'environnement virtuel mais cela pose un certain nombre de problèmes techniques : conserver le contact des pieds avec le sol, s'adapter au squelette synthétique, éviter les glissements sur le sol, gérer une animation complète à partir de plusieurs petites séquences… L'adaptation de mouvements capturés à des squelettes et des environnements synthétiques est, toutefois, une étape préliminaire à toutes les précédentes approches. Gleicher [G98] propose une méthode permettant d'adapter un mouvement capturé à un personnage de morphologie différente et à un environnement différent [GL98]. Le problème de l'adaptation d'un mouvement à une morphologie différente est de respecter les contraintes particulières de ce mouvement. Par exemple, un pointage ou une préhension contraignent la position cartésienne d'un ou de plusieurs éléments du squelette. A l'inverse, un mouvement de gymnastique doit garantir une posture globale répondant à des contraintes sur les angles aux articulations. Les contraintes à respecter doivent être spécifiées pour chaque mouvement étudié et aucune méthode générale ne semble répondre, à l'heure actuelle, à tous les cas de figure.

En plus d'adapter un mouvement élémentaire à un squelette et à un environnement, l'animation de personnages synthétiques nécessite généralement de générer des mouvements complexes. Ceci peut être obtenu en mélangeant des mouvements élémentaires [BBE97]. Dans ce cas, plusieurs problèmes se posent : gérer la continuité du mouvement, assurer que les contraintes géométriques et non-holonomes sont respectées…

Toutes ces techniques d'animation modifient les trajectoires capturées originelles et peuvent donc dégrader le réalisme du mouvement. Un moyen de vérifier la validité des mouvements ainsi modifiés est d'utiliser la réalité virtuelle. Nous proposons, dans une application liée au handball, de vérifier qu'un gardien de but réagit de la même manière à des tirs virtuels ainsi modifiés qu'à ceux (réels) qui ont servi à calculer les mouvements de l'avatar. Dans un premier temps, nous présentons la démarche générale de cette approche. Nous décrivons ensuite les différentes phases utilisées pour animer l'adversaire synthétique du gardien de but. Nous présentons ensuite les résultats de cette expérimentation pour conclure sur l'intérêt de ce type d'expérimentation

2. Organisation générale

La première partie de cette étude est une expérience préliminaire impliquant le gardien de but et le tireur de handball lors d'une rencontre sur un terrain réel de handball. Les deux protagonistes étaient équipés de marqueurs infrarouges qui permettaient de capturer leurs mouvements en 3D. Le système utilisé était un Vicon370 (Oxford Metrics) composé de 7 caméras infrarouges, synchronisées à 60Hz. Le but de cette étude était de capturer le tir au but d'un vrai joueur de handball ainsi que le mouvement correspondant du gardien (cf. figure 1). Par conséquent, nous avons analysé la réaction du gardien face à différents tirs. Pour chaque catégorie de tir, nous avons identifié les contraintes spatio-temporelles qui lient les mouvements du tireur et du gardien.

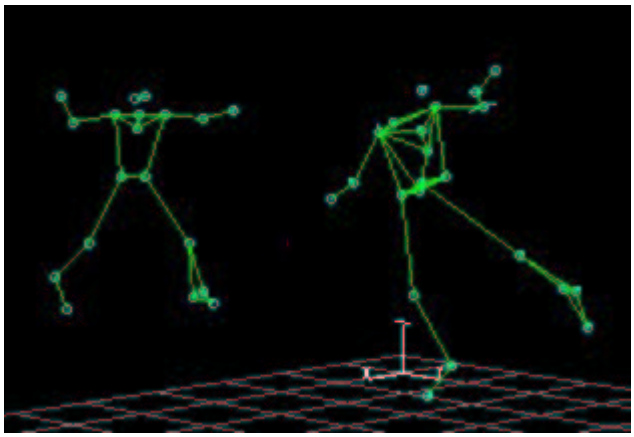


Figure 1 : capture des mouvements d'un gradient de but et d'un tireur.

Une fois les mouvements du tireur capturés, plusieurs traitements sont nécessaires pour animer un joueur synthétique de la même manière :

1. Adaptation des mouvements capturés à des humanoïdes synthétiques,
2. Animation temps réel de joueurs synthétiques de handball.

Notre objectif est de vérifier que les traitements génèrent un mouvement suffisamment réaliste pour engendrer les mêmes réactions chez le gardien de but. Ceci a fait l'objet d'une seconde expérimentation.

En effet, dans la deuxième partie de l'expérience nous avons placé le même gardien de handball dans un stade virtuel afin de jouer contre un tireur virtuel. Nous avons employé un système de réalité virtuelle composé d'une SGI Onyx2 InfiniteReality dont les sorties vidéo sont redirigées vers trois vidéoprojecteurs Barco 1208S synchronisés. L'image était projetée sur un écran semi-cylindrique (avec un rayon de 3,80m, une taille de 2.38m et un champ visuel 135°). Pour obtenir un vrai comportement du gardien nous avons reconstruit un environnement aussi réaliste que possible en reproduisant les repères visuels bien connus du joueur. Un des repères les plus importants était le but qui était physiquement placé au centre de la salle de réalité virtuelle. Une calibration entre l'environnement virtuel et réel a été effectuée. Ainsi, l'image retransmise sur grand écran permettait de reconstruire un stade à échelle 1. Pour étudier les mouvements du gardiens, nous avons une fois de plus utilisé le système de capture du mouvement Vicon370. La plate-forme de réalité virtuelle et le système Vicon n'étaient pas physiquement synchronisés et aucun signal de départ n'était donné au gardien. Celui-ci devait juste essayer de parer les tirs virtuels qui lui étaient proposes: vingt quatre tirs qui ont été choisis de manière aléatoire parmi l'ensemble des tirs capturés lors de la première expérimentation. Ces tirs ont été divisés en trois catégories :

- **Tirs à 6 mètres en appui**, dans cette catégorie nous avons utilisé trois tirs capturés.
- **Tirs à 6 mètres en suspension,** dans cette catégorie nous avons utilisé quatre tirs capturés.
- **Tirs à 9 mètres en appui,** dans cette catégorie nous avons utilisés quatre tirs capturés.

Tous les tirs ci-dessus ont été joués deux fois, mélangés aux autres essais, de manière à éviter que le gardien puisse identifier les tirs. Entre chaque tir, le gardien se repositionne dans son but et attend la prochaine épreuve.

Nous décrivons maintenant les différentes phases qui ont permis de restituer les mouvements du tireur dans l'environnement virtuel.

3. Adaptation des mouvements capturés à des humanoïdes synthétiques

Les humanoïdes de synthèse utilisés pour cette expérimentation utilisent 26 degrés de liberté :

- Trois rotations à l'épaule
- Une rotation au coude,
- Trois rotations au niveau du torse,
- Trois rotations et trois translations au niveau du pelvis (considéré comme l'origine de la hiérarchie),
- Trois rotations à la hanche,
- Une rotation au genou,
- Une rotation à la cheville.

7 caméras infrarouges cadencées à 60 Hz (faisant partie du système Vicon370, Oxford Metrics) ont été placées de manière à couvrir un champ de mesure de 12 mètres par 6 mètres. Pour couvrir cet espace et mieux appréhender le mouvement du tireur et du gardien, nous avons disposé les caméras en cercle autour de la surface de jeu. Les sujets étaient ensuite équipés de 26 marqueurs infrarouges qui permettaient la reconstruction 3-D de repères anatomiques. La figure 2 représente, à gauche, le sujet vu grâce aux marqueurs infrarouges et, à droite, l'humanoïde synthétique utilisé pour l'expérimentation.

Figure 2 : Placement des marqueurs infrarouges et humanoïde synthétique utilisé lors des expérimentations.

A partir de ces marqueurs externes, placés sur des repères anatomiques peu sensibles aux glissements de la peau, nous développons maintenant le processus qui nous permet d'animer au plus juste le joueur synthétique. Ce processus se déroule en 4 étapes pour chaque animation capturée [MM01] :

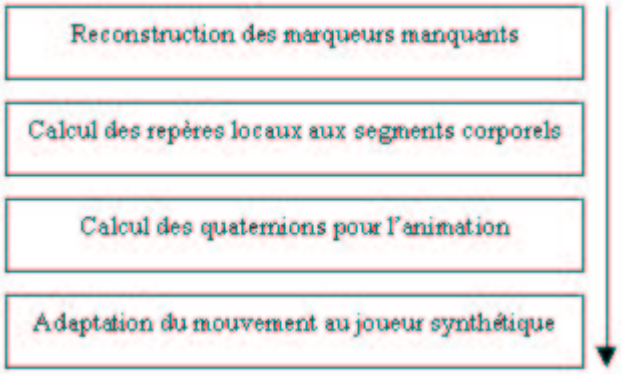


Figure 3 : processus de traitement des mouvements capturés pour l'animation de joueurs synthétiques.

1. à récupérer les points manquants qui sont dus à des occultations pendant la capture. Pour cela, nous utilisons un fichier de paramètres décrivant la structure du squelette humanoïde (mesurée lors de la capture du mouvement grâce à une posture pour laquelle les longueurs de chaque segment sont mesurées). Un graphe non-orienté définit des contraintes de distance entre deux marqueurs appartenant au même segment corporel. Les nœuds de ce graphe sont les marqueurs, les arcs entre deux nœuds représentent une notion de contrainte de distance. Les arcs sont associés à la longueur séparant les deux marqueurs. Ces longueurs, supposées constantes pendant le mouvement, permettent de calculer les points manquants en minimisant l'ensemble des contraintes avec ces voisins dans le graphe. Soient quatre marqueurs V1, V2, V3 présents, et M, à reconstruire. d'1, d'2 et d'3 représentent respectivement les distances séparant M de V1, V2 et V3. M' est une approximation de M, à partir d'une interpolation naïve de M dans le temps. Cette première interpolation ne tient pas compte des contraintes de distance. M est reconstruit en modifiant M' pour qu'il corresponde au mieux (au sens des moindres carrés) aux distances d'1, d'2 et d'3 (cf. figure 4).

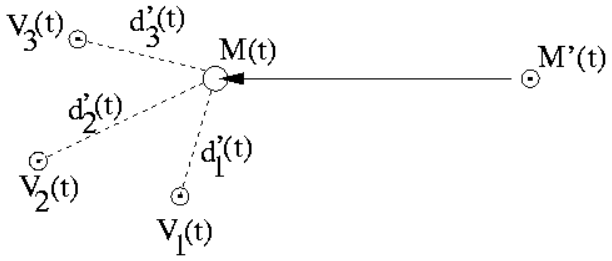


Figure 4 : recontsruction d'un point manquant à partir de ses trois voisins.

2. à reconstruire les repères associés à chaque segment corporel à partir des repères anatomiques. Par exemple, le centre du poignet est retrouvé à partir des deux marqueurs positionnés sur la tête de l'ulna et du radius. Le même type d'hypothèse est utilisé pour retrouver les autres centres articulaires qui composent ainsi les axes principaux des repères liés à chaque segment corporel [D95].

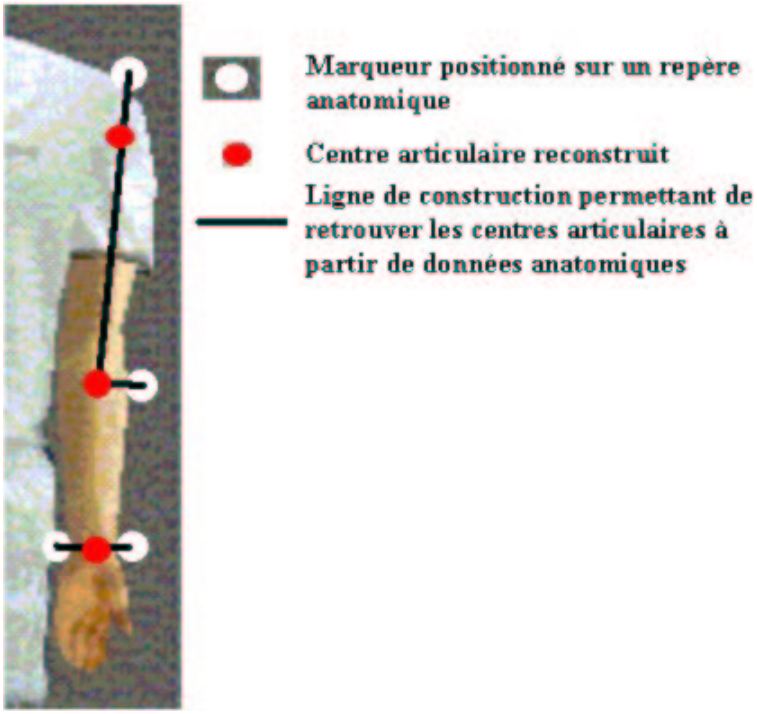


Figure 5 : reconstruction des centres articulaires à partir des marqueurs externes.

3. à calculer les quaternions à chaque pas de temps pour obtenir les trajectoires articulaires nécessaires à l'animation du personnage synthétique.
4. à adapter le mouvement au squelette synthétique qui a, en général une taille différente de celui du sujet [MM01].

En sortie de ces étapes, plusieurs mouvements, adaptés à l'humanoïde synthétique, sont disponibles.

4. Animation temps réel de joueurs de handball

Les étapes présentées ci-dessus doivent être effectuées pour tout mouvement capturé. Afin d'offrir une plus grande liberté de mouvement au joueur synthétique, nous avons capturé des trajectoires de course, de marche et de tir au but. Ces trajectoires sont capturées indépendamment les unes des autres.
Le joueur synthétique doit être capable d'enchaîner et de mélanger ces trajectoires pour obtenir un mouvement complexe qui corresponde à une situation réaliste de jeu. Cette situation inclut le déplacement du joueur, le tir et son replacement sur le terrain. Il est donc nécessaire d'enchaîner ces actions de manière réaliste.
Pour cela, des priorités sont associées à chaque mouvement et évoluent continûment au cours du temps en fonction des besoins de l'animation [M03]. Ces priorités indiquent, pour chaque degré de liberté l'importance relative de chaque mouvement. Prenons l'exemple d'une marche normale, suivie d'une marche fatiguée puis d'une course (figure 6).

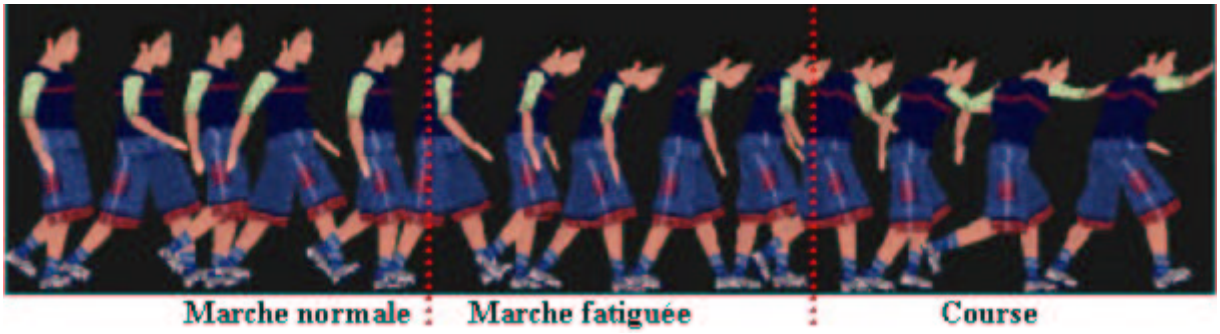


Figure 6 : mélange de mouvements capturés (marche normale, marche fatiguée et course) appliqués à un personnage synthétique.

La locomotion fait intervenir principalement les membres inférieurs. Les trois mouvements de locomotion ont donc une priorité importante sur les articulations de la hanche, du genou et de la cheville. Inversement, ils ont une importance plus relative pour le haut du corps qui peut donc effectuer une autre tâche en parallèle. Supposons maintenant que, pendant la séquence de la figure 6, un lancer de ballon doive être effectué. Le lancer de ballon fait principalement intervenir le haut du corps. Par contre, le bas du corps intervient moins dans ce lancer. En résultat, le bas de corps est principalement animé grâce aux mouvements de locomotion (priorité importante) même si ces mouvements sont légèrement modifiés par le lancer. A l'inverse, sur le haut du corps, le lancer a une priorité importante mais le mouvement est légèrement modifié par la locomotion.
En plus de ce mélange de mouvements, il est possible de séquencer les animations. Pour cela, la priorité de chaque mouvement augmente et diminue continûment respectivement en début et en fin de séquence. Cette augmentation (resp. diminution) continue permet de lisser les transitions entre deux gestes.
Cette technique ne fonctionne que si les postures de début et de fin sont suffisamment proches, sans quoi, même avec une interpolation linéaire, la séquence résultante est totalement irréaliste. Dans notre application, nous séquençons des mouvements proches : différentes locomotions et un tir. Les différentes locomotion sont séquencées en prenant soin que le pied d'appui en fin de mouvement corresponde bien au même appui au débuts de l'autre. Le tir au but s'effectue toujours avec un pied l'un devant l'autre (le pied avant est aussi appelé pied d'appui). De plus, le tir s'effectue généralement après une locomotion si bien que le séquencement d'une marche ou d'une course avec un tir s'effectue sans discontinuité majeure.
Nous aurions pu utiliser des techniques de séquencement de mouvement comme ceux proposés par Gleicher [G98] mais nous avons préféré cette solution qui garantit une réponse rapide de l'humanoïde aux consignes, tout en assurant une fluidité et une continuité dans le geste.
Lors des expérimentations, il a été impossible de capturer correctement le mouvement du ballon sur la totalité du volume de capture. De plus, en réalité virtuelle, nous voulions être capables de modifier la trajectoire du ballon soit de manière artificielle, soit pour répondre à une adaptation du geste du tireur. C'est pourquoi nous avons choisi de modéliser le ballon comme un système mécanique, en négligeant les forces de frottement. Pour ce modèle, les paramètres d'entrée sont :

- la vitesse du ballon mesurée grâce à un radar à main Radargun (V_r),
- la position du ballon au moment du lâcher (X_0),
- et la destination du ballon dans le but (mesurée lors des expérimentations) (X_f).

Avec ces paramètres, il est possible de connaître la trajectoire du ballon à chaque instant :

$$\begin{cases} x = V_0^x t + x_0 \\ y = V_0^y t + y_0 \\ z = -\frac{1}{2} g t^2 + V_0^z t + z_0 \end{cases} \quad (1)$$

où g est l'accélération de la gravité, et $X_0=(x_0, y_0, z_0)$. Le radar était placé dans la direction du tir et mesurait donc $\sqrt{(V_0^x)^2+(V_0^y)^2}$ alors que la balle suivant une trajectoire longue de $l=\sqrt{(x_f-x_0)^2+(y_f-y_0)^2}$ mètres dans le plan horizontal. Ainsi, le temps requis pour atteindre le but est de $\Delta t=\frac{l}{V_r}$. Si on considère t=0 au moment du lâcher, V_0^z est égal à :

$$V_0^z = \frac{z_f - z_0 + \frac{1}{2} g \Delta t^2}{\Delta t} \quad (2)$$

Le modèle de ballon, ainsi que le module responsable de l'animation du tireur synthétique ont été embarqués dans la plate-forme Openmask [DCDK98]. OpenMask est une plate-forme logicielle qui permet des communications et des interactions entre des entités autonomes qui évoluent dans l'environnement virtuel. Le ballon et le tireur sont deux entités autonomes communicantes. Les paramètres initiaux du ballon sont donnés par le module de tireur au moment du lâcher. Avant cet instant, le module de tireur fournit la position de la main qui tient le ballon à ce dernier pour qu'il suive parfaitement cette même trajectoire.

5. Résultats

En résultat, nous présentons les comparaisons entre les mouvements du gardien face au tireur virtuel et face au tireur réel. Dans cette étude nous nous sommes intéressé à la qualité de l'animation du tireur virtuel. Pour cela, nous avons utilisé des données biomécaniques pour évaluer le comportement du gardien. Ainsi, nous avons choisi de nous intéresser au déplacement du centre de masse (CM) du bras du gardien dans les deux situations (réelle et virtuelle). Nous avons choisi le centre de masse du bras car dans les situations étudiées le gardien a paré les tirs avec le membre supérieur.
Pour comparer les trajectoires du CM du bras dans les deux situations, il nous faut tout d'abord normaliser le temps. Pour cela, nous avons choisi de prendre l'instant zéro à un événement particulier que l'on retrouve dans tous les tirs : l'instant où le gardien réagit au tir. Cet instant est calculé à partir du pic d'accélération du bras du gardien dans toutes les situations. Nous définissons une plage d'étude de –0.3s à +0.3s autour de ce pic.

Nous avons étudié 24 tirs divisés en trois catégories. Pour chaque tir, nous comparons les paramètres suivant, pour le tir en situation d'origine (dans le réel) et dans la salle de réalité virtuelle :

- la position initiale du CM du bras,
- la position finale du CM du bras,
- son déplacement,
- la différence en pourcentage entre le déplacement face au tir virtuel et celui lié au tir réel,
- la corrélation point à point entre les deux trajectoires.

Pour l'ensemble des tirs étudiés, les trajectoires étaient identiques aussi bien du point de vue des valeurs que de la forme. Le tableau 1 présente les résultats liés à la trajectoire du CM du bras selon l'axe vertical. Ces valeurs sont obtenues pour le tir où les variations par rapport à la situation d'origine étaient les plus importantes. Ainsi nous pouvons dire que les variations les plus importantes sont de 11,9% et que le coefficient de corrélation le plus faible entre les deux courbes est de 0.96. Il semble important de souligner que lorsque l'on compare deux tirs réels considérés comme quasiment identiques, les variations sont de l'ordre de 20% et le coefficient de corrélation est nettement inférieur (de l'ordre de 0.9).

Mouvement	Position initial du bras (m)	Position finale du bras (m)	Déplacement (m)	Différence par rapport à l'action réelle (%)	R^2
Réelle	0.546	0.706	0.16	**	
Virtuelle 1	0.526	0.667	0.141	11.9	0.96
Virtuelle 2	0.519	0.668	0.149	6.9	0.98

Tableau 1 : Variations cinématique du CM du bras suivant l'axe vertical

6. Discussion

Dans cet article, nous avons présenté une expérimentation virtuelle mettant en jeu un gardien de but de handball réel immerge dans un stade de handball virtuel. Un joueur virtuel effectue une série de tirs que le gardien est censé arrêter, comme lors d'un match réel. Cette application implique que le gardien reconnaisse son environnement et réagisse de la même manière que lors d'un match réel. Plusieurs phénomènes peuvent dégrader la perception qu'a le sujet de son environnement. L'aspect géométrique semble jouer un rôle important puisqu'il intervient, semble-t-il [HOT98], sur la perception qu'a le sujet du mouvement synthétisé. Le calibrage de l'environnement virtuel joue lui-aussi un rôle important puisqu'il fait garantir que l'environnement soit correctement perçu. Nous nous sommes principalement intéressé aux animations produites pour l'environnement virtuel. En effet, un gardien de but est habitué à intercepter des tirs et réagit à des stimuli qui semblent plus liés aux mouvements du tireur qu'à celui du ballon. Ceci est spécialement vrai à haut niveau où les vitesses de balle sont tellement importantes que le temps séparant la perception d'un événement intervenant après le lâcher du ballon et la réaction est trop long. L'animation est donc un élément crucial pour assurer que le gardien se sente effectivement immergé dans le jeu virtuel.

La méthode de validation que nous avons mise en œuvre dans cet travail ne permet pas d'isoler la qualité du modèle d'animation. En effet, nous jugeons de la présence du sujet dans sa globalité : est-ce que le sujet réagit de la même manière dans le réel et dans une représentation virtuelle de la même scène ? La réponse donnée par cette expérimentation pilote est positive. Des travaux sont en cours pour reproduire ce travail sur un plus grand nombre de sujets. Comme nous pouvons conclure que la présence est vérifiée pour cette expérimentation, nous pouvons aussi conclure que l'animation (qui était un élément entrant dans cette expérimentation) est de qualité suffisante.

Cette animation est le produit d'une chaîne de traitements qui vont de la capture du mouvement à sa restitution sur un personnage synthétique de taille différente, en passant par une interpolation des points manquants, par un calcul des repères associés aux segments corporels et par une adaptation du mouvement à un squelette de taille différente. Lors de ces traitements, le mouvement capturé subit un certain nombre de dégradations et de modifications qui auraient pu altérer les réactions du gardien de but. Or, nos résultats tendent à montrer que ce n'est pas le cas. De futurs travaux sont à envisager pour juger de la sensibilité des gardiens aux modifications apportées aux mouvements. Ainsi, nous envisageons de définir un modèle cinématique et dynamique de tir au handball à partir des mesures déjà effectuées. Ce modèle permettrait d'obtenir des tirs proches de ceux capturés et de vérifier si les réactions des gardiens sont altérées.

Le fait que les gardiens de but réagissent positivement à ce type d'expérimentation ouvre un grand champs d'applications de la réalité virtuelle. Il est possible d'envisager des entraînements virtuels, de mener des expériences pour mieux cerner les paramètres pris en compte par le gardien pour réagir…

Remerciements

Ce travail a été soutenu par le Ministère des Sports, la Préparation Olympique et le Conseil Régional de Bretagne. Merci à Dominique Favotti pour son modèle géométrique de joueur de handball.

Références

[ADHMT89] Arnaldi, B., Dumont, G., Hégron, G., Magnenat-Thalmann, N., Thalmann, D. *Animation control with dynamics*. Proceedings of Computer Animation'89, 113-123.(1989).

[BBE97] Boulic R, P Becheiraz , L Emering, D Thalmann. *Intergration of motion control techniques for virtual human an avatar real-time animation.* Actes de VRST '97 111-118 septembre 97 (1997)

[BC96] Bruderlin, A., Calvert, T.*Knowledge-driven, interactive animation of human running*. In Proceedings of Graphics Interface'96, 213-221(1996).

[BMT90] Boulic, R., Magnenat-Thalmann, N., Thalmann, D. *A Global human walking model with real-time kinematic personification.* The Visual Computer, 6(6), 344-358.(1990).

[BRRP97] Bodenheimer, B., Rose, C., Rosenthal, S., Pella J. *The process of motion capture: dealing with the data*. Proceedings of Eurographics Workshop on Computer Animation and Simulation, 3-18.(1997*).*

[CPNMT97] Capin, T., Pandzic, I., Noser, H., Magnemat-Thalmann, N., Thalmann. D. *Virtual Human Representation and Communication in VLNET Networked Virtual Environments*. IEEE Computer Graphics and Applications, Special Issue on Multimedia Highways, 17(2), 42-53, 1997.

[D95] Dempster WT . *Space requirements of the seated operator*. WADC-TR-55-159. Wright-Patterson Air Force Base, Ohio (1995).

[DCDK98] Donikian, S., Chauffaut, A., Duval, T., Kulpa. R. *GASP: from Modular Programming to Distributed Execution.* Computer Animation'98, IEEE, Philadelphie, USA, Juin.(1998*).*

[G98] Gleicher M *Retargetting motion to new characters*. Actes de ACM siggraph 33-42 Juillet (1998)

[GL98] Gleicher M, Litwinowicz *Lconstraint based motion adaptation* . Journal of Visualization and computer animation, 9 (2) 65-94(1998).

[HH96] Huffman, K., Hubbard, M. *A motion based virtual reality training simulator for bobsled drivers.* The engineering of sport, 195-203, Balkema Rotterdam, July (1996).

[HOT98] Hodgins, J., O'Brien, J. , Tumblin, J. *Perception of human motion with geometric models*. IEEE Transaction on Visualisation and Computer Graphics, Vol 4, No. 4, 307-316 (1998).

[HWBO95] Hodgins, J., Wooten, W., Brogan, D., O'Brien, J. *Animating human athletics*. In Proceedings of ACM SIGGRAPH, 71-78.(1995).

[M03] Ménardais *Fusion et adaptation temps réel de mouvements acquis pour l'animation d'humanoïdes synthétiques*. Thèse de l'Université de Rennes 1, janvier 2003 (à paraître).

[MAC99] Molet, T.,. Aubel, A ., Capin, T., Carion, S., Lee, E., Magnenat-Thalmann, N., Noser, H., Pandzic, I., Sannier, G., Thalmann, D. Anyone for tennis? Presence, MIT, Vol. 8, No. 2, 140-156 (1999).

[MBT96] Molet, T., Boulic, R., Thalmann, D. *A real-time anatomical converter for human motion capture*. Eurographics Workshop on Computer Animation and Simulation, 79-94, september(1996).

[MM00] Metoyer, R., Hodgins, J. *Animating athletic motion planning by example*. Proceeding of graphics interface 2000, Montreal, Canada, 61-68, May 15-17 (2000).

[MM01] Menardais S, Multon F *Amélioration des trajectoires acquises par des systèmes optiques pour l'animation de personnages synthètiques.* Revue internationale de CFAO, 99-113, 2001.

[MNH99] Multon, F., Nougaret, JL., Hegron, G., Millet, L., Arnaldi, B. *A software toolbox to carry-out virtual experiments on human motion.* Computer Animation, Genève, 16-23, May 1(1999).

[NPCM96] Noser, H., Pandzic, I.., Capin, T., Magnemat-Thalmann, N., Thalmann, D. *Playing Games through the Virtual Life Network*. ALIFE V, Oral Presentation , Nara, Japan, 114-121(1996).

[SVK01] Mj, Schuemie, P Van der Straaten, M Krijin, C Van Der Mast. *Research on presence in Vr: a survey.* cyberpsychology and behavior. 4(2), 183-202 (2001).

[T.96] Thalmann. D. *A new generation of synthetic actors: the Interactive Perceptive Actors.* Proceedings of Pacific Graphics'96, Taipeh, Taiwan, 200-219 (1996).

[UAT95] Unuma, M., Anjyo, K., Takeuchi, R. *Fourier principles for emotion-based human-figure animation.* Proceedings of ACM SIGGRAPH, 91-96 .(1995).

[W79] Winter, D. A new definition of mechanical work done in human movement. Journal of applied physiology, 46(1), 78-83 (1979).

[WK88] Witkin, A., Kass, M. *Spacetime constraints*. Proceedings of ACM SIGGRAPH, Atlanta, Georgia, 159-168 (1988).

[WP95] Witkin, A., Popovic, Z. *Motion warping*. Proceedings of ACM SIGGRAPH, 105-108.(1995).

[Z82] Zeltzer, D. *Motor control techniques for figure animation*. IEEE Computer Graphics and Applications. 2(9), 53-59.(1982).

Modèle d'animation comportemental de piétons virtuels

A. Ebel, D. Hanon, B. Stanciulescu, P. Pudlo, E. Grislin, F-X. Lepoutre

LAMIH UMR CNRS 8530, Université de Valenciennes et Hainaut-Cambrésis
59313 Valenciennes cedex 9

aurelien.ebel@meletu.univ-valenciennes.fr

Mots-clés : Animation, Capture de Mouvement, Animation Comportementale, Intelligence Artificielle, Agents Autonomes, Planification de Trajectoire

1. Introduction

Ce papier s'inscrit dans le cadre du projet RESPECT (Route Empruntée en Sécurité par le Piéton-Enfant Confronté au Trafic) projet PREDIT, qui s'effectue en partenariat avec l'INRETS LPC, le CRP2C et l'entreprise CORYS TESS.
Ce projet consiste à développer un simulateur réaliste du trafic des piétons et de voitures en ville. L'objectif pédagogique vise à sensibiliser les jeunes enfants de 5-7 ans à se déplacer dans la ville en sécurité. Présenté dernièrement au salon de l'automobile de Paris (septembre 2002) au stand sécurité routière, il sera testé dans sa première version en janvier 2003 dans les écoles primaires.
Cette communication traite des deux principaux aspects que la modélisation de l'enfant-piéton comporte : la modélisation des mouvements du piéton et la modélisation comportementale.

La deuxième partie traite des mécanismes qui gèrent le comportement des piétons par rapport au trafic et aux évènements qui l'entourent. Deux types de comportement sont modélisés : le comportement réactif du piéton face aux événements générés par son environnement dynamique et le comportement général du piéton en fonction du but global de celui-ci. A cette fin, un modèle orienté agent est utilisé. Le contrôle est basé sur une architecture à fusion d'actions généralisée.

Enfin, une démonstration pratique du stade de développement du projet clôturera notre communication (pas utile).

2. Modèle du piéton

Le modèle du piéton virtuel retenu (cf. figure 1) se compose de 18 corps rigides. Les 17 articulations sont de type pivot (1 DDL), cardan (2 DDL) et rotule (3DDL). Le modèle compte 41 degrés de liberté. Il ne correspond pas au modèle exhaustif du groupe H-ANIM (Humanoid Animation Working Group) : il est en restriction.

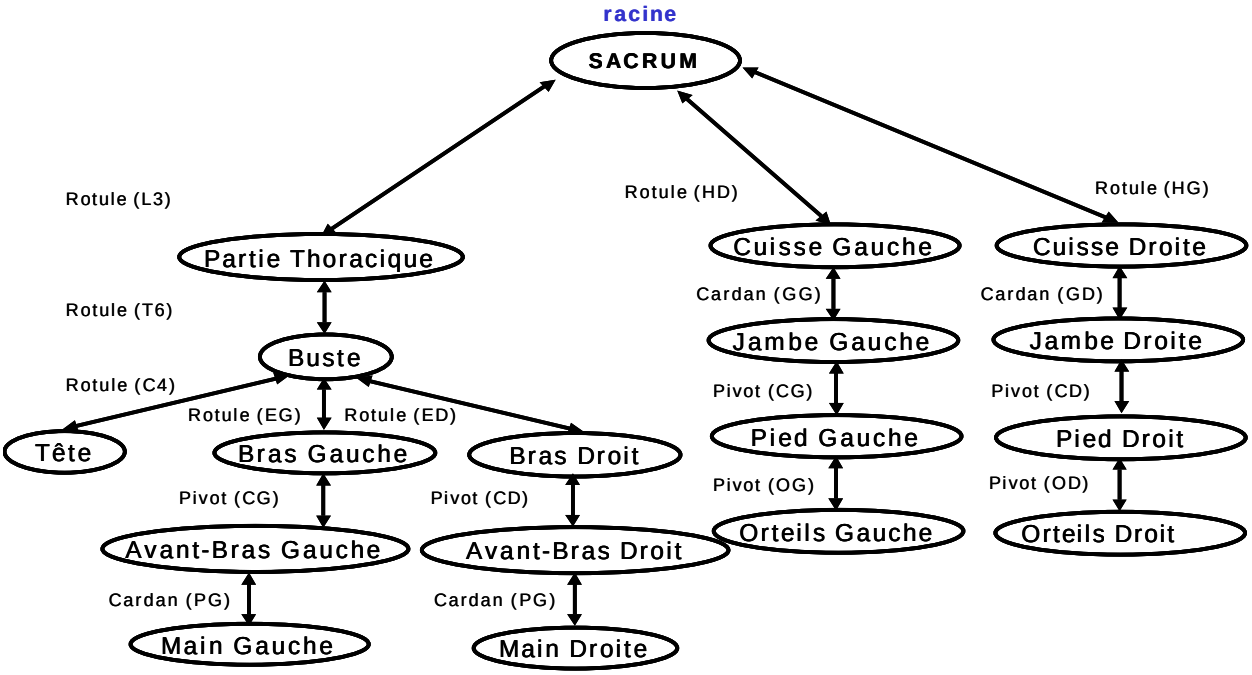


Fig. 1 : Hiérarchie des nœuds (segments corporels et articulations) pour le modèle générique de piéton

3. Obtention des données d'entrée nécessaires à l'animation

3.1. Principaux mouvements à générer

La table 1 présente les mouvements retenus pour l'animation réaliser des piétons virtuels en environnement urbain.

Marcher	**Courir**
Accélérer	**Ralentir**
Arrêter	**Attendre**
Monter un trottoir	**Descendre un trottoir**

Table 1 : Mouvements standards d'animation en environnement urbain

3.2. Capture des mouvements

Les mouvements considérés précédemment sont enregistrés à l'aide du système d'analyse gestuelle VICON, composés de 8 caméras CCD, à une fréquence de 100Hz .

Avant chaque expérimentation, il est demandé au sujet de se positionner dans une configuration enregistrée, dite de référence, coïncidant avec le modèle virtuel à l'état initial.

4. Recalage des données réelles

Les données mesurées expérimentalement doivent permettre d'animer et contrôler le piéton virtuel. Ce piéton peut être positionné " n'importe où " dans l'environnement virtuel.

Une méthode a été mise en œuvre, dite de « recalage », et consiste à identifier les matrices de cosinus directeurs, définissant l'orientation de chaque segment corporel du piéton virtuel par rapport au segment corporel père, à appliquer au modèle pour une animation du piéton virtuel coïncidant au mouvement réel du sujet lors de l'expérimentation. La méthode de « recalage » procède en deux temps : la coïncidence et le recalage .

- La coïncidence exploite l'expérimentation en position de référence afin d'identifier les corrections à appliquer aux données réelles. - Le recalage applique ces corrections à l'expérimentation afin d'obtenir un mouvement du piéton coïncidant au mouvement réel.

5. Animation du piéton virtuel

Le modèle du piéton est hiérarchique et le sacrum en est sa racine. L'animation du piéton virtuel consiste à :

- positionner et orienter le sacrum du piéton virtuel dans l'environnement,
- orienter chaque segment corporel fils par rapport au segment corporel père respectif dans l'espace virtuel 3D.
  l'orientation et la position des segments corporels fils dérivent de la position et l'orientation du sacrum

6. Modèle comportemental du piéton

Les piétons peuplant la simulation sont conçus dans l'objectif d'en faire des acteurs autonomes, capables de raisonner et d'analyser leur situation afin de choisir les actions appropriées en fonction de leurs objectifs.
Ces exigences conduisent naturellement à adopter une modélisation orientée agent. En effet, un agent est une entité réelle ou virtuelle, capable de percevoir, au moins partiellement, son environnement, d'agir sur son environnement et de se comporter de manière « rationnelle » : ses actions sont le produit d'un « raisonnement » (ou du moins, d'un ensemble de règles de comportement) dans le but d'obtenir un état souhaité de l'environnement [MAN 01]. La solution choisie consiste à associer un agent à chaque piéton. Chaque agent contrôle les mouvements du piéton associé.

Le fonctionnement d'un agent peut se résumer en trois sous-fonctions : percevoir, décider, et agir.
La fonction de perception d'un agent de la simulation est caractérisé par un besoin de production d'un comportement crédible. En effet, bien que toutes les informations associées aux objets de l'environnement soient a priori disponibles pour l'agent, il s'agit de restreindre la recherche au sous-ensemble des données compatible avec le champ de perception que possède le personnage dans le monde réel. Les personnages de la simulation sont ainsi dotés de capteurs virtuels [NOS 95].
Actuellement, ces capteurs sont limités au domaine visuel. Chaque capteur virtuel retourne les adresses des objets présents dans le champ de vision de l'agent. L'agent a alors accès à tous leurs attributs, qui décrivent la géométrie de l'objet : dimensions, position, orientation, etc. La figure 3 montre un exemple d'évitement entre deux agents. Chaque agent dispose de deux capteurs de proximité et d'informations sur la distance et la direction du point à atteindre.
L'agent met en œuvre également une forme de perception volontaire (ou "active") lorsqu'il recherche des informations dans un but précis. Par exemple, la perception volontaire s'appliquera lors de la recherche d'information avant la traversée : présence d'un feu tricolore, présence d'un passage protégé, position et vitesse des véhicules, etc.

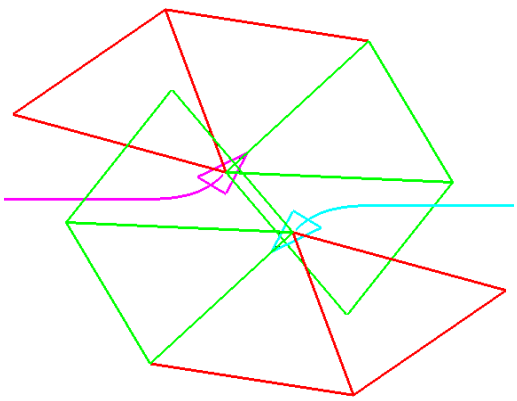


Fig. 3. Exemple d'évitement entre 2 piétons

Dans le cadre de l'animation d'humanoïde, les modèles décisionnels utilisés par les agents servent à déclencher les actions en fonction de leurs intentions et des informations perçues.
Selon Yves Duthen [DUT 02], il y a trois principales manières de programmer le comportement d'un personnage virtuel, ce sont les approches :

- Impératives : ce type d'architecture est généralement basé sur des scripts que l'agent rejoue ;
- Déclaratives : le comportement est déterminé par un système à base de règles. Le raisonnement provient des inférences réalisées ;

- Basées sur l'émergence : cette approche repose sur l'utilisation d'éléments simples qui s'organisent pour arriver à une solution.

D'un point de vue agent la première solution n'a pas de réel intérêt.
La seconde correspond à un agent cognitif (raisonnement sur des symboles abstraits). Cette solution présente les inconvénients liés à ce type de système : il n'est pas simple de constituer un ensemble de règles, celui-ci se limite souvent à un domaine particulier. Ce type d'approche est particulièrement remis en cause par l'approche animat [GUI 99] et la robotique [BRO 98]. Pour gérer les déplacements du personnage avec ce type d'architecture il faut, vraisemblablement, discrétiser l'environnement et contraindre les déplacements du joueur, ce qui n'est pas souhaitable.
La dernière solution envisage la construction d'un agent composé de plusieurs agents simples. Il s'agit d'une architecture à découpage horizontal (opposée à l'architecture verticale des agents cognitifs). Dans cette lignée, nous avons choisi de mettre en oeuvre une architecture basée sur la « fusion d'action généralisée » [ARN 00]. Ses principes sont particulièrement intéressants car ils permettent l'unification de trois architectures de contrôle :

- L'architecture à subsomption de R.Brooks [BRO 86]
- L'architecture à sélection d'actions de P.Maes [MAE 89]
- L'architecture orientée schémas de R.Arkin [ARK 98]

Comme toutes les architectures horizontales elle utilise le comportement comme élément de base. Un comportement est caractérisé par :

- une activité a. La valeur de a est un nombre décimal compris entre 0 (comportement inactif) et 1 (comportement actif). L'activité traduit la présence et l'importance des stimulus relatifs à ce comportement.
- Un vecteur de nombres décimaux R. Chaque élément de R correspond à une consigne sur « les actionneurs » de l'agent. En particulier la vitesse et la direction pour un robot mobile

Le vecteur de consigne appliqué aux « actionneurs » de l'agent est calculé par un réseau de nœuds. Cette notation permettant de décrire l'architecture du système dérive directement de la subsomption et utilise quatre types de nœuds : l'inhibition, la suppression, l'augmentation et le maximum. Les fonctions de transfert des nœuds permettent de réaliser un arbitrage entre les différents comportements et de calculer le vecteur de consignes en sortie du réseau.
Cette architecture est spécialement adaptée aux robots mobiles et peut être implémentée simplement. Néanmoins comme pour les Boids [REY 99], ce modèle nécessite des ajouts afin de gérer des déplacements étendus à l'ensemble d'une ville. Les agents sont construits sur le modèle présenté en figure 4. Les vecteurs de consignes sont : (vitesse, changement de cap).

L'architecture de contrôle reprend les comportements présentés dans [ARN 00] auxquels nous proposons l'ajout d'un comportement cognitif appelé "Contrôle de la traversée". Ce comportement analyse la circulation et inhibe les autres comportements lorsque l'agent doit attendre avant de pouvoir traverser une rue.
L'action de suivre le trottoir se composent de deux comportements qui vont tendre à positionner le piéton au centre du trottoir.
L'évitement de collision est prioritaire par rapport aux comportements "suivre le trottoir" et "aller au but". L'action qu'il propose subsume les actions proposées par les couches inférieures de l'architecture. Les deux comportement qui le composent font dévier le mobile de sa direction. " Éviter objet à droite " permet l'évitement des obstacles se trouvant à droite en proposant un virage à gauche, même chose pour l'évitement à gauche. Un nœud "maximum arbitre ces deux comportements : seule la proposition de comportement le plus actif est donc prise en compte. Les comportements d'évitement peuvent être plus ou moins réactifs, en effet une idée est d'introduire une mémorisation de l'état précédent afin de limiter les oscillations du personnage.

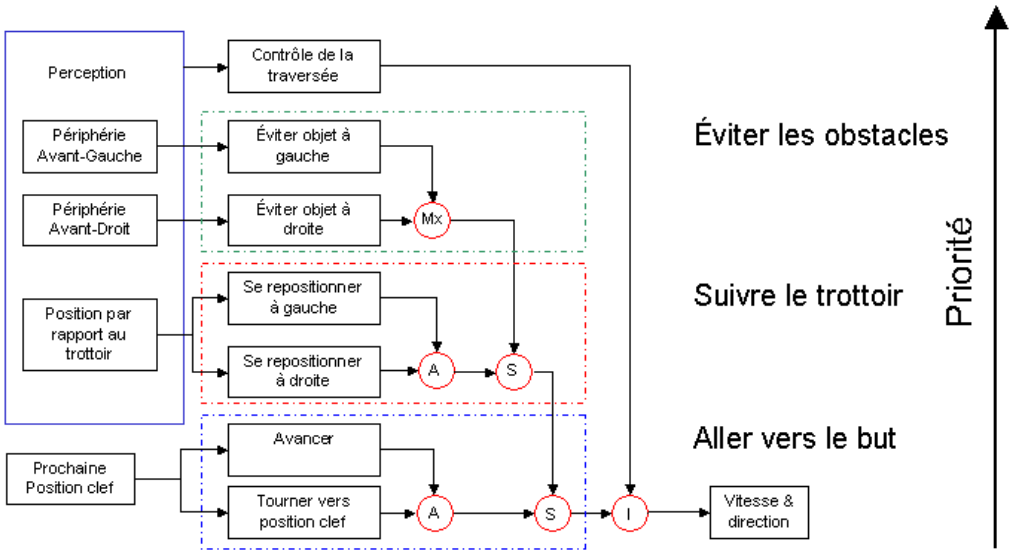


Fig. 4. Architecture

7. Conclusions et Perspectives

Dans ce papier les deux lignes générales du simulateur issues du projet RESPECT ont été présentés, à savoir la partie modélisation de piétons et la partie IA de la modélisation du contrôle de leur comportement.
Dans le cadre de la modélisation de piétons, nous avons introduit le modèle d'animation de l'humanoïde piéton. Ce modèle, dérivé du modèle h-anim[1], s'est révélé suffisant afin d'implémenter les capacités d'animation qu'un piéton réel suppose.
Un ensemble de mouvements d'animation qu'un piéton est supposé présenter a été enregistré par capture de mouvement à l'aide d'un système de capture VICON.
Les données ci-enregistrées ont été traitées de manière qu'elles puissent servir pour l'animation des piétons virtuels. Le premier traitement de données a été le calcul des positions articulaires à partir des positions enregistrées des marqueurs, suivi du calcul des angles articulaires correspondant aux différents types de mouvements et aux différents membres du piéton. Ces angles ainsi enregistrés et stockés sont ensuite injectés après dans le modèle du piéton, afin de réaliser les différents mouvements.
La deuxième partie de ce travail a consisté dans le développement de systèmes de contrôles comportemental du déplacement des piétons dans la ville. Pour cela nous avons choisi l'utilisation des réseaux à fusion d'actions. Le comportement d'un piéton est défini par deux composantes : la composante réactive qui gère l'interaction instantanée du piéton avec le milieu extérieur (évitement d'obstacles, évitement de collision avec un autre piéton, etc.) et une deuxième composante qui gère la stratégie longue-durée du piéton (suivi de trajectoire, traversée de rue, planning de trajectoire, etc.).
Comme perspectives immédiates du projet nous pouvons mentionner les points suivants :

- Au niveau de la modélisation du mouvement du piéton nous envisageons d'introduire des modules de commande adaptatifs en fonction des contraintes externes, au lieu d'utiliser des données préenregistrées pour chaque mouvement possible.
- Au niveau comportemental nous travaillons à l'affinement des stratégies de contrôle à moyen et long terme par l'introduction de composantes plus cognitives que les comportements gérés actuellement;

Références

[ARK 98] R. ARKIN, "Behavior-based robotics", The MIT Press, 1998

[ARN 00] ARNAUD P., "Des Moutons et des robots ", Presse polytechniques et universitaires romandes, Lausanne, 2000

[BRO 86] R. A. BROOKS"a robust layered control system for a mobile robot", IEEE Journal of Robotics and Automation, RA-2/1, pp. 14-23, mars 1986.

[BRO 98] R. A. BROOKS, C. BREAZEAL, R. IRIE, C. C. KEMP, M. MARJANOVIC, B. SCASSELLATI, M. M. WILLIAMSON, "Alternative Essences of Intelligence", AAAI98.

[CAU 96] F. CHAUMETTE, S. BOUKIR, P. BOUTHEMY, D. JUVIN. Structure From Controlled Motion. IEEE Transactions on Pattern Analysis and Machine Intelligence. 18(5): 492–504, May 1996.

[DUT 02] Y. DUTHEN, “Behavioural Simulation and Artificial Life”, 3IA 2002, Limoges.

[FAU95] F. FAURE. Modélisation cinématique du contact pour la dynamique inverse. In 3èmes journées de l'AFIG. pages 15–22, Marseille, Novembre 1995.

[GUI 99] GUILLOT A.,”Pour une approche dynamique des animats”, dans Drogoul et Meyer (Eds.). Intelligence Artificielle Située, Hermes, 1999

[LH96] M. LEVOY, P HANRAHAN. Light Field Rendering. Proceedings of ACM Siggraph’96, pages 31–42, August 1996, New Orleans.

[MAE 89] P. MAES, "The dynamics of action selection", proceedings of the international Joint conference on Artificial Intelligence, IJCAI-89,1989

[MAN 01] R. MANDIAU, E. GRISLIN-LE STRUGEON (2001). Systèmes multi-agents. In TI (Ed.), S 7216, Paris: Les Techniques de l'Ingénieur, pp. 1-17.

[NOS 95] H. NOSER, D. THALMANN : "Synthetic vision and audition for digital actors", Proc.Eurographics’95, 1995, pp.325-336.

[REY 99] C. W. REYNOLDS : "Steering Behaviors For Autonomous Characters", in the proceedings of Game Developers Conference 1999 held in San Jose, California. Miller Freeman Game Group, San Francisco, California. Pages 763-782, 1999.

Une architecture pour le Retour d'Efforts.

T.Meyer, G.Andrade-Barroso & B.Arnaldi.

IRISA 35042 Rennes Cedex

[Tangi.Meyer|Guillermo.Andrade-Barroso|Bruno.Arnaldi]@irisa.fr

Résumé : *Ce document présente l'architecture pour la simulation avec retour d'efforts que nous avons développée. Notre objectif est de proposer des outils facilitant la mise en oeuvre d'applications métier. Nous avons implémenté des interfaces qui permettent une intégration générique des éléments logiciels nécessaires au retour d'efforts. Nous décrivons également les évolutions en cours et les perspectives qu'offre notre architecture.*

Mots-clés : retour d'efforts, architecture logicielle, réalité virtuelle.

1 Introduction

De nombreuses applications de réalité virtuelle exploitent le retour d'efforts depuis que certains périphériques comme le PHANToM (c)[29] ou le Virtuose (c) de Haption[1] sont "sortis des laboratoires" pour être commercialisés. Des domaines comme la CAO (assemblage virtuel, ergonomie, design...), la médecine (chirurgie, rééducation...) ou la formation, bénéficient grandement du développement des techniques de réalité virtuelle et plus particulièrement du retour d'efforts.

OpenMASK(c)[2] est une plate-forme Open Source de développement d'applications pour la réalité virtuelle [10], issue des travaux du projet SIAMES de l'IRISA. De nombreux travaux exploitant cet outil ont été réalisés : simulation d'environnements urbains [14], animation d'humanoïdes [22], manipulation coopérative [28] et distante de données industrielles...

Dans le cadre de l'extension de cette plate-forme, le développement d'applications de réalité virtuelle exploitant le retour d'efforts est un objectif naturel. Dans la logique des travaux de l'équipe SIAMES, nous cherchons à proposer des fonctionnalités et non un outil spécialisé. Celles-ci ont pour objectif de faciliter la mise en oeuvre des outils nécessaires à la simulation avec retour d'efforts. En effet, bien que le retour d'efforts soit un dénominateur commun aux applications sus-citées, il semble irréaliste de vouloir développer un outil universel pour sa simulation tant les besoins en terme de modèles, de formats, sont différents. A partir de cette constatation et dans un soucis d'ouverture d'OpenMASK vers le plus grand nombre d'applications de réalité virtuelle, nous avons cherché à identifier les principaux besoins en terme d'outils logiciels pour la simulation avec retour d'efforts. Nous avons alors défini une architecture logicielle modulaire et ouverte afin que les utilisateurs d'OpenMASK puissent facilement intégrer leur travaux et développer des applications plus spécialisées. Nous présentons ici, les travaux réalisés autour d'OpenMASK dans ce but.

La suite du document s'organise de cette façon : à partir d'un état de l'art, nous identifions les principaux éléments logiciels nécessaires à la simulation avec retour d'efforts. Nous présentons ensuite l'architecture développée et les premiers résultats obtenus. Les évolutions envisagées sont exposées dans la quatrième partie du document. Enfin nous présentons les perspectives qu'offre notre architecture.

2 État de l'art

2.1 Exemples d'outils pour la simulation avec retour d'efforts

De nombreux développements ont déjà été réalisés pour des applications spécifiques. Voxmap Point Shell [31] de Boeing (c) est un logiciel destiné à la navigation interactive dans des environnements de CAO avec un périphérique à retour d'efforts. La scène simulée est représentée sous forme de voxels et l'objet que l'utilisateur manipule sous la forme d'un nuage de points. L'interaction entre la pièce mobile et le reste de l'environnement est réalisé au

[1] http ://www.haption.com

[2] http ://www.openmask.org

travers d'une méthode dite de pénalité appliquée aux points du nuage qui intersectent avec les voxels de plus petite dimension. La configuration à chaque pas de temps est calculée en intégrant les équations de Newton-Euler. Le périphérique haptique est lié à la simulation par un couplage virtuel [20] qui permet de renvoyer un effort vers le manipulateur. Ce principe est également utilisé par D.Baraff [12] pour associer un autre type de périphérique haptique à une bibliothèque logicielle de simulation dynamique pour les corps rigides. Des exercices d'apprentissage simples peuvent être construits à partir de ce système. S.Cotin [25] présente une autre application de la simulation avec retour d'efforts : la chirurgie. Un modèle déformable de foie est simulé et lié à un périphérique haptique spécialisé pour la simulation de chirurgie laparoscopique. L'auteur utilise ici les éléments finis pour calculer une déformation due à l'interaction de l'utilisateur.

Ces exemples mettent en évidence l'étendue des différents domaines et outils logiciels qui sont utilisés pour la simulation avec retour d'efforts.

2.2 Les éléments nécessaires à la simulation avec retour d'efforts

2.2.1 Généralités

Dans la plupart des approches, la scène virtuelle associée à une simulation avec retour d'efforts est un monde contenant des objets dont le comportement est dicté par des lois dans la plupart des cas physiques. Il est possible d'établir des domaines du monde à l'intérieur desquels il règne des lois de comportements bien définis et dont la nature ne change pas au cours du temps. Ces domaines sont par exemple les volumes des objets rigides ou déformables et les avatars des utilisateurs. Ce sont les interactions entre les différentes frontières des domaines qui vont déterminer les conditions aux limites ou de passage d'un domaine à l'autre. Ces conditions aux limites vont modifier l'évolution interne des domaines. Ainsi deux objets rigides se touchant en un point (contact entre deux domaines volumiques) génèrent du frottement entre leurs surfaces (frontières). Ce frottement va définir des forces (conditions aux limites) qui vont modifier les déplacements des objets (comportements).

De ces définitions découlent les éléments nécessaires à la simulation :

– des outils capables de déterminer le lieu des interactions entre les frontières,
– des outils permettant de produire les conditions aux limites associées à ces interactions,
– des outils définissant le comportement d'un domaine à partir de son état actuel et des conditions aux limites associées à ses frontières.

Dans la plupart des travaux sur la réalité virtuelle avec retour d'efforts, ces outils prennent les dénominations respectivement de ***détecteur de collision***, de ***traitement du contact*** et de ***solveur physique*** ou de ***solveur mécanique***.

2.2.2 Détection de collisions.

La détection de collisions a été étudiée dans de nombreux domaines comme la robotique et l'informatique graphique. Les besoins ou objectifs peuvent être très différents (application temps-réels, résultat exact, adaptation des données CAO, déformations, etc.). De ce fait, de nombreux algorithmes ont été développés [19] [30][1][6][26].

Toutefois, les étapes essentielles de leur fonctionnement peuvent être décrites simplement par la figure 1. Après une mise à jour des positions et des vitesses des objets concernés, on réalise le test de détection de collisions. Les données géométriques obtenues en résultat sont traitées afin d'être envoyées au calculateur de réponses au contact.

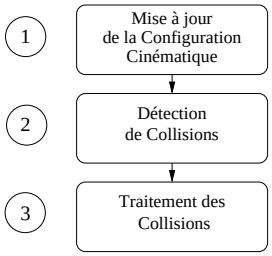


FIG. 1 – Fonctionnement simplifié de la détection de collisions.

2.2.3 Traitement du Contact

Il existe plusieurs traitements possibles de la collision : par une méthode de type pénalités [32], par une méthode impulsionnelle [7] ou par une méthode dite LCP [11]. Chacune produit des informations à traiter par le solveur mécanique. Le résultat de ce traitement est l'apparition de forces de frottement et de non pénétration ou directement de changements dans les champs de vitesses des objets ou dans celui des accélérations. Ces effets seront traités à différents niveaux dans le solveur mécanique.

2.2.4 Mécanique

Il existe de nombreux types de simulateurs physiques. Il sont parfois spécialisés. Ils peuvent être basés sur des algorithmes et des méthodes de résolution très différentes. La robotique est à l'origine d'outils spécialisés dans le traitement des chaînes poly-articulées. SMR [3] par exemple dispose d'un traitement optimisé des chaînes poly-articulées fermées. D'autres logiciels permettent la simulation physique d'ensembles de corps rigides : Dynamo [4], ODE[3], ou encore CONTACT [27]. Vortex de CMLabs[4] a été développé pour les jeux vidéo. Pour animer les corps déformables, Les méthodes retenues sont souvent des modèles mécaniques simplifiés afin de satisfaire les contraintes de l'animation interactive [18].

Bien que d'un point de vue général, ces logiciels permettent de traiter l'animation mécanique, leur variété met surtout en évidence la diversité des problèmes à traiter. Toutefois, comme dans le cas de la détection de collisions, il est possible de décrire les grandes étapes de la résolution du système mécanique (Cf. figure 2). Le système mécanique est construit à partir de l'ensemble des informations dynamiques de la scène. Le système d'équations obtenu lie les forces et contraintes aux accélérations des différents degrés de liberté du système. Une intégration numérique produit alors un nouvelle configuration cinématique.

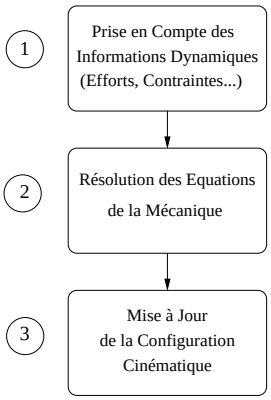


FIG. 2 – Résolution Mécanique simplifiée.

2.2.5 Manipulation interactive haptique

L'immersion de l'utilisateur passe par une manipulation interactive d'objets virtuels. Dans le cas d'une souris, cela consiste à commander directement en position l'objet que l'on manipule. L'introduction du retour d'efforts nécessite une approche plus évoluée : il ne s'agit plus de se contenter de visualiser des déplacements, il faut également renvoyer des informations haptiques à l'utilisateur.

Le principe directeur peut être décrit de la façon suivante : nous disposons de deux représentations d'un même objet, la représentation "réelle" qui est asservie en position et vitesse au périphérique haptique, ainsi qu'une représentation virtuelle dont les position et vitesse sont calculées par la simulation. L'objectif est de faire coïncider ces deux représentations ou tout du moins à les faire tendre l'une vers l'autre. Pour cela, il faut définir un outil bilatéral qui permettra à l'utilisateur de ressentir les efforts nécessaires à cette correspondance (et qui permettra une prise en compte de l'influence de l'utilisateur dans la simulation).

[3]http ://www.q12.org/ode/
[4]http ://www.cm-labs.com/

Plusieurs méthodes ont été proposées : nous avons évoqué le couplage virtuel [20] dans le paragraphe 2.1. Cette technique exploitée par [5], [31] et [12] entre autres, s'accorde très bien avec le principe du rendu d'impédance [9] (mesure de déplacement et renvoi de force, figure **??** (a)). Un lien composé d'un ressort et d'un amortisseur virtuel entre les positions simulée et issue du périphérique permet d'envoyer des informations d'efforts au périphérique et au module de résolution mécanique à chaque pas de temps.

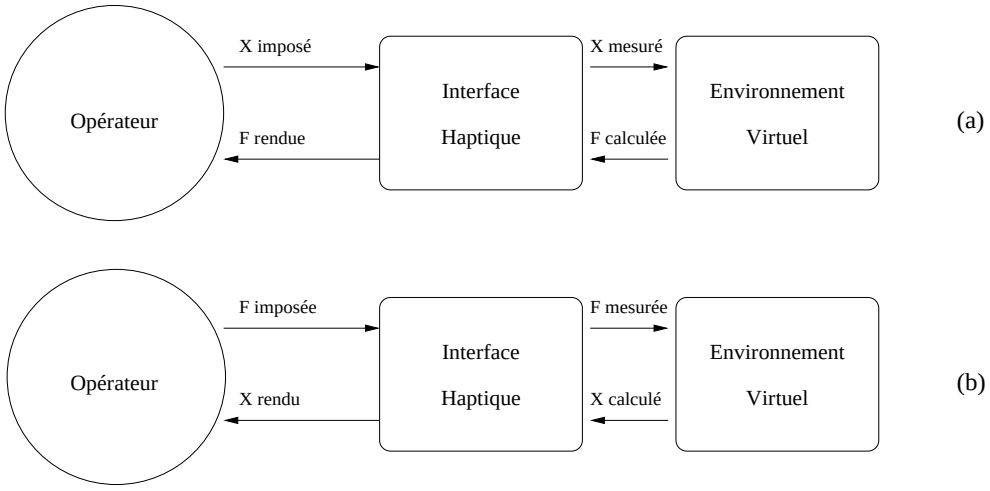


FIG. 3 – Représentation simplifiée du rendu d'impédance (a) et du rendu d'admittance (b).

Une autre approche "par contraintes" a été proposée [8]. Il s'agit d'imposer des contraintes dynamiques entre la position réelle de l'objet manipulé et sa position virtuelle ("God-object"). Le "Proxy" de [13] est associé à une méthode de rendu haptique de type "shading". Ces méthodes permettent d'éviter des incohérences en imposant un suivi plus strict des surfaces. Par exemple, l'utilisateur ne traversera pas un objet lors d'un suivi d'une surface de faible épaisseure.

Comme pour le problème de la détection de collisions et le traitement de la mécanique, la manipulation interactive pour le retour d'efforts peut être résolue de plusieurs façon. La technique la plus adaptée dépend souvent de l'application étudiée.

2.2.6 Identification du besoin

La variété des applications et des techniques sus-citées rend très complexe le développement d'une solution universelle. Cependant, les blocs logiciels représentés par les figures 1, 2 et **??** permettent de faire abstraction de la méthode employée tout en définissant des interfaces communes.

3 Architecture

Notre travail consiste en une formalisation des échanges d'informations pour définir une architecture modulaire ouverte pour la simulation avec retour d'efforts.

3.1 Principe

Le principe de notre architecture est d'offrir des outils génériques pour l'intégration. Nous avons développé plusieurs interfaces abstraites pour les modules logiciels indispensables que nous avons identifiés dans la section précédente.

La figure 5 présente la première étape de nos travaux. Nous l'avons basée sur le fait que les modules mécaniques étaient plus aisément commandables en forces. Ainsi nous considérons que le traitement de la collision est réalisé au travers d'un modèle de pénalité. De la même façon, nous avons retenu le couplage virtuel comme liaison périphérique/simulation. La figure 4 présente le modèle général de nos modules.

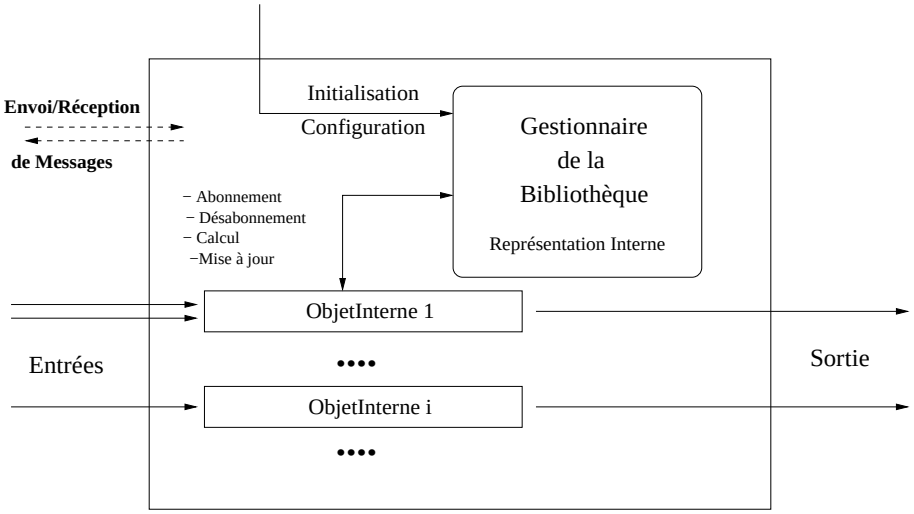


FIG. 4 – Représentation d'un module.

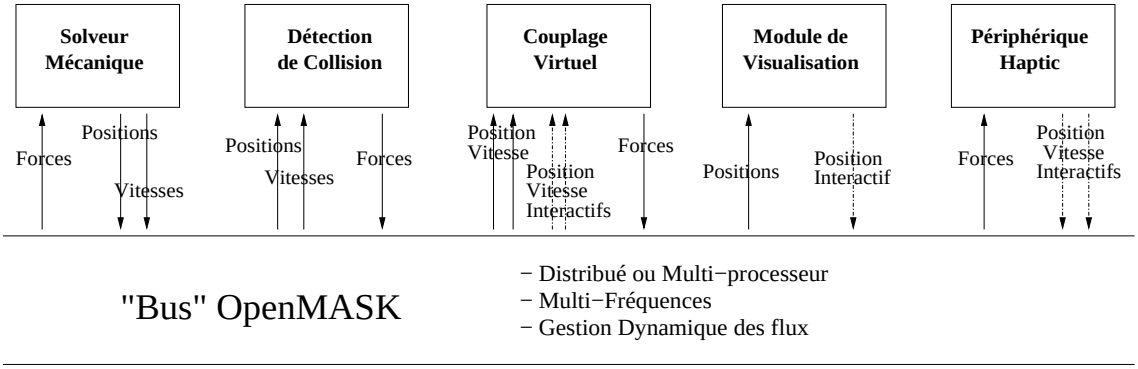


FIG. 5 – Représentation simplifiée de l'architecture.

Notre architecture exploite plusieurs propriétés d'OpenMASK :

– La notion de bus logique et l'architecture modulaire.
– le module de visualisation avancée qui nous évite des travaux complexes concernant le rendu.
– les fonctionnalités distribuées et multi-processeurs.

Mais avant tout, ce sont les outils de communication entre les modules qui nous intéressent. Plus particulièrement, nous exploitons les notions de flux d'entrées et de sorties pour les signaux continus et les messages pour les signaux discrets.

Dans le premier cas, il s'agit pour un module de s'abonner à des informations provenant d'un autre module ou encore de produire des sorties qui seront connectées aux entrées d'un module différent. Nous avons développé des interfaces basées sur les principes présentés par les figures 2 et 1, qui généralisent le type des entrées et des sorties des modules afin de rendre ceux-ci compatibles. Comme le montre la figure 5, un module "solveur mécanique" dispose d'entrées de type forces. Celles-ci permettent la prise en compte d'éventuels contacts[5] ou d'interactions de type couplage virtuel. Après intégration des équations de la mécanique, le solveur propose une nouvelle configuration cinématique par l'intermédiaire de ses sorties de types position et vitesse. Un module de détection de collisions peut se brancher sur ces sorties afin de mettre à jour sa représentation interne du monde. C'est également le cas pour un module de type couplage virtuel ou pour le module de visualisation.

OpenMASK nous permet une gestion dynamique des flux. Pour cela nous exploitons les messages : ils facilitent l'échange d'informations typées entre deux modules. En particulier, nous provoquons l'abonnement ou le désabonnement des entrées à un flux par leur intermédiaire. Dans le cas de l'interaction avec une souris par

[5]Notre premi`ere architecture est bas´ee sur un mod`ele de type p´enalit´e en ce qui concerne le traitement de la collision.

exemple, un couplage virtuel est créé entre la souris et l'objet virtuel dont on prend le contrôle. Celui-ci génère des efforts qui s'exercent sur l'objet manipulé. Le solveur mécanique chargé de l'animation de cet objet doit donc s'abonner à la sortie *force* du couplage virtuel. Dans ce but, un message de type `AddForceStream` est envoyé par la souris au système mécanique qui en réponse crée une entrée de type *force* qui se connecte à la sortie spécifiée. Le désabonnement est réalisé de la même façon grâce à un message `DeleteForceStream`.

3.2 Premiers résultats

Les objectifs de cette plate-forme sont avant tout de faciliter le développement d'applications métier. Pour cela l'interchangeabilité des modules est un élément essentiel. Afin de valider cet objectif, nous avons procéder au développement de simulation basées sur différentes bibliothèques logicielles. En ce qui concerne les solveurs mécanique, Dynamo, SMR et CONTACT ont été portés avec succès. Notre interface s'est avérée fonctionnelle et a permis des développements rapides.

Dans le même but, des travaux sur diverses librairie de détection de collisions ont été réalisés : VCollide, SWIFT++ [24], les résultats étant traités par un modèle de pénalité.

L'architecture nous permet également de mixer dans une même scène la simulation de plusieurs systèmes basés sur diverses librairies.

Nous avons également validé la manipulation interactive de type "couplage virtuel" unilatéral (commande en force de la simulation par la souris). L'interface générique permet une manipulation interactive transparente des objets quelque soit le solveur mécanique qui l'anime.

4 Évolution de l'architecture

4.1 Généralisation du couplage virtuel

Notre architecture permet entre autre de faire coexister plusieurs systèmes de type simulateur dynamique de corps rigides. Commander en position ce type d'outil peut s'avérer délicat en particulier si le système mécanique est constitué d'une chaîne comprenant des contraintes bilatérales telles que des rotules, des pivots... En effet, les manipulations directes risquent d'engendrer des violations de contraintes qui peuvent faire diverger le système lors de l'intégration numérique.

Afin d'éviter ces écueils, il est souhaitable de recourir à un mode de commande plus proche de la dynamique (i.e. du second ordre). Le couplage virtuel présente des caractéristiques intéressantes pour transformer les sorties de type position d'un système en entrées de type force pour un autre solveur mécanique. Par exemple, dans le cas d'une simulation de téléopération (Cf. figure 6), il peut être intéressant de travailler avec deux logiciels différents : le premier issu de travaux en robotique permettra une manipulation du robot virtuel, par contre il sera sans doute moins pertinent pour des travaux de suivi de surface, d'où l'exploitation éventuelle d'un second logiciel. Le couplage virtuel permet un lien transparent entre ces outils : chaque logiciel anime une représentation d'un objet (ici l'extrémité du robot) et elles sont ainsi associées par une liaison bilatérale.

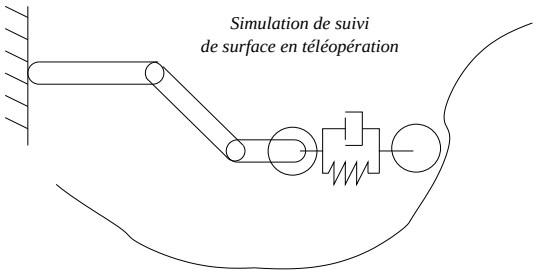


FIG. 6 – Exemple de généralisation du couplage virtuel.

Nous avons donc étendu l'utilisation de la notion de couplage virtuel : il nous permet de faire communiquer des systèmes mécaniques basés sur des logiciels différents.

4.2 Types des Entrées/Sorties

La généralisation du couplage virtuel conduit à rencontrer des systèmes équivalents à la figure 7 où l'objet A et son image sont animés par des simulateurs physiques différents tout en étant liés par un couplage virtuel.

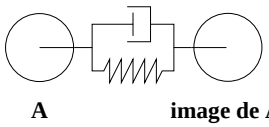


FIG. 7 – Couplage Virtuel généralisé.

La représentation de ce système pour notre architecture est celle de la figure 8. Où le *solveur mécanique* n °1 anime l'objet A et le *solveur mécanique* n °2 son image (X représente ici un vecteur position+vitesse).

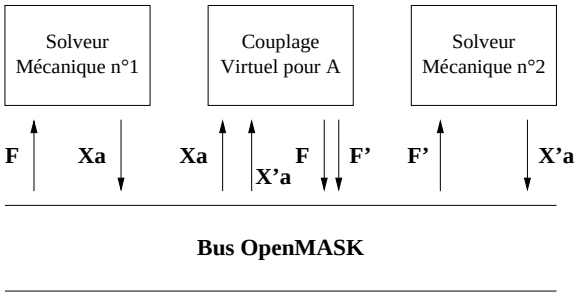


FIG. 8 – Architecture et couplage virtuel généralisé.

Cette configuration peut poser quelques problèmes dans le cas d'une utilisation distribuée d'OpenMASK. Plus particulièrement, les entrées de type forces des solveurs mécaniques risquent de conduire à des instabilités dues à des erreurs de fonctionnement du réseau (latence, perte de données...). Nous avons donc décidé d'instaurer une communication de flux position+vitesse entre les solveurs mécaniques en intégrant le couplage virtuel sous la forme d'un pré-traitement des entrées des solveurs afin d'obtenir une architecture proche de la figure 9.

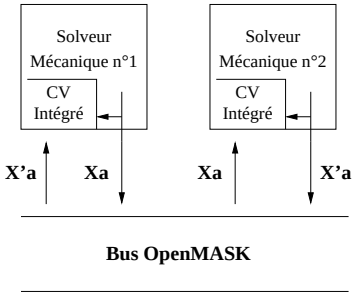


FIG. 9 – Couplage Virtuel intégré.

Nous avons réalisé, avec Simulink (c) [17], une étude comparative de stabilité entre ces deux modes de traitement du couplage virtuel. L'objectif est d'étudier la convergence entre Xa et $X'a$ lorsque le système mécanique est soumis à des perturbations. Les données obtenues[6] mettent en évidence une stabilité accrue et une convergence améliorée lorsqu'on intègre le couplage virtuel au solveur mécanique (modèle de la figure 9).

Nous cherchons à généraliser la communication de type position. En effet, outre les avantages en terme de stabilité, ce modèle permet d'ouvrir le panel des traitements applicables à la collision par exemple. Il devient possible d'appliquer des modèles plus évolués comme des contraintes de type non-pénétration ou contact point/plan. Un

[6] Les r´esultats de cette ´etude feront l'objet d'une publication ult´erieure.

message permettra de spécifier, au moment de la création de l'entrée de type position généralisée, le traitement à appliquer (Cf. figure 10).

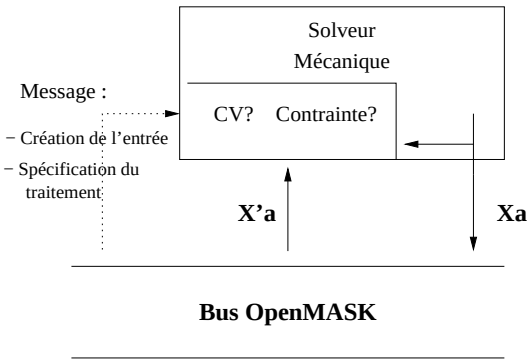


FIG. 10 – Principe d'une entrée généralisée.

5 Perspectives

5.1 Objets déformables

Plusieurs applications du retour d'efforts concernent les objets déformables : la chirurgie [25], la sculpture virtuelle [15] ... Valider notre architecture pour les modèles de corps déformables sera un de nos objectifs futurs. Les particularités d'un solveur mécanique pour ce type de modèle ne paraissent pas incompatibles avec les développements réalisés.

Les intérêts de tels développement sont multiples : il sera possible de coupler la simulation de chirurgie à un simulateur de robotique afin de réaliser une application de téléopération chirurgicale. L'aspect distribué d'OpenMASK nous permettra en outre de répartir les calculs aisément pour satisfaire les besoins en performance.

5.2 Localisation des contraintes logicielles du retour d'efforts

La performance constitue une contrainte forte pour une simulation avec retours d'efforts. En effet, la perception haptique impose des fréquences de fonctionnement élevées pour la simulation : la bande passante de la perception des efforts pour un manipulateur se situe à quelques dizaines de hertz mais celle de la perception tactile est de l'ordre de 300Hz [16] [2].

Les nouveaux types d'environnements de visualisation comme les salles immersives ou les CAVE (c) permettent de travailler sur des modèles de grandes dimensions comme des véhicules à l'échelle 1. Dans la plupart des cas, à un instant donné, le retour d'efforts ne concerne qu'une partie réduite de la scène. Différents travaux sur les modélisations pour la chirurgie à retour d'efforts se rapprochent de la notion de niveaux de détails haptiques [21]. Hayward [23] propose deux niveaux de maillages éléments finis fonctionnant à des fréquences différentes. Debunne [18] fait cohabiter plusieurs maillages de résolutions différentes.

Notre architecture nous permet d'envisager une localisation géométrique des contraintes du retour d'efforts qui se rapprocherait de cette idée de niveau de détails haptiques. En déterminant une zone d'intérêt autour de l'objet que l'utilisateur manipule, il devient possible de séparer les objets de la scène en deux groupes : une partie d'entre eux sera directement concernés par l'interaction haptique (fréquences élevées de simulation), le reste de la scène aura un comportement découplé de l'interaction (contraintes logicielles relachées : fréquences faibles).

5.3 Travail coopératif

Un de nos objectifs est également de développer des applications de travail coopératif. Notre architecture permet la mise en oeuvre de différents périphériques tout en conservant les mêmes interfaces de communication avec la

simulation.

Nous envisageons également d'exploiter les travaux sur l'interaction à distance réalisés sur la plate-forme GASP à l'origine d'OpenMASK. Le réseau VTHD a en effet permis des manipulations interactives entre le projet I3D de l'INRIA Rocquencourt et le projet SIAMES de Rennes. Des démonstrations de manipulations coopératives à distance intégrant le retour d'efforts sont actuellement à l'étude.

6 Conclusions

Nous avons présenté dans ce document les travaux que nous avons réaliser dans le but d'intégrer des applications de réalité virtuelle avec retour d'efforts sur la plate-forme OpenMASK. Nous proposons une architecture modulaire destinée à faciliter l'insertion de bibliothèques logicielles spécialisées. Divers développements nous ont permis de valider la première version de cette architecture.

Actuellement, nous développons des évolutions de nos interfaces afin de rendre plus générique et plus robuste cette architecture. Les perspectives sont relativement nombreuses. L'extention aux simulations avec retour d'efforts pour les corps déformables en est une, tout comme les applications coopératives et distantes.

Références

[1] A.Gregory, M.Lin, S.Gottschalk, and R.Taylor. A framework for fast and accurate collision detection for haptic interaction. In *IEEE Virtual Reality 1999 Conference*, volume 1, pages 38–45, Houston, Texas, USA, March 1999.

[2] A.Lecuyer. *Contribution à l'étude des retours haptique et pseudo-haptique et de leur impact sur les simulations d'opérations de montage/démontage en aéronautique*. PhD thesis, University Paris XI, November 2001.

[3] G. Andrade. *Modélisation et adaptation du mouvement de robots tout-terrain*. PhD thesis, University Paris 6, Paris, France, September 2000.

[4] B. Barenbrug. *Designing a Class Library for Interactive Simulation of Rigid Body Dynamics*. PhD thesis, Technische Universiteit Eindhoven, april 2000.

[5] B.Chang and J.E.Colgate. Real-time impulse-based simulation of rigid body systems for haptic display. In *ASME International Mechanical Engineering Congress and Exhibition*, pages 145–152, Dallas, Texas, USA, 1997.

[6] B.Mirtich. V-clip : Fast and robust polyhedral collision detection. *ACM Transactions on Graphics*, 17(3) :177–208, July 1998.

[7] B.V.Mirtich. *Impulse-based Dynamic Simulation of Rigid Body Systems*. PhD thesis, University of California at Berkeley, Fall 1996.

[8] C.B.Zilles and J.K.Salisbury. A constraint-based god-object method for haptic display. In *IEEE Conference on Conference on Intelligent Robots & Systems*, volume 3, August 1995.

[9] C.R.Carignan and K.R.Cleary. Closed-loop force control for haptic simulation of virtual environnements. *Haptic-e : http ://www.haptic-e.org*, 1(2), February 2000.

[10] David Margery, Bruno Arnaldi, Alain Chauffaut, Stéphane Donikian, and Thierry Duval. Openmask : {Multi-Threaded — Modular} animation and simulation {Kernel — Kit } : a general introduction. In Simon Richir, Paul Richard, and Bernard Taravel, editors, *VRIC 2002 Proceedings*, pages 101–110. ISTIA Innovation, June 2002.

[11] D.Baraff. Fast contact force computation for non-penetrating rigid bodies. In *SIGGRAPH 94 Proceedings*, 1994.

[12] D.Baraff, P.J.Berkelman, and R.L.Hollis. Interaction with a realtime dynamic environment simulation using a magnetic levitation haptic interface device. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 3261–3266, Detroit, Michigan, USA, May 1999.

[13] D.C.Ruspini, K.Kolarov, and O.Khatib. Haptic interaction in virtual environments. In *IEEE International Conference on Intelligent Robots & Systems IROS'97*, Grenoble, France, September 1997.

[14] S. Donikian and G. Thomas. Modeling virtual urban environments for multi-modal driving simulation. In *UM3'99, Int. Workshop on Urban 3D/Multi-Media mapping*, pages 103–110, Institute of Industrial Science (IIS), The University of Tokyo, Japan, 1999.

[15] E.Ferley. *Sculpture Virtuelle*. PhD thesis, Institut National Polytechnique de Grenoble, 2002.

[16] G.C.Burdea and P.Coiffet. *La Réalité Virtuelle.* Hermes, 1993.

[17] G.D.Buckner. Simulink : a graphical tool for dynamic system simulation. In *NCSU ASME Technical Sessions*, October 11 2001.

[18] G.Debunne. *Animation multirésolution d'objets en temps-réel*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, France, December 2000.

[19] J.D.Cohen, M.Lin, D.Manocha, and M.K.Ponamgi. I-collide : An interactive and exact collision detection system for large scale environments. In *ACM Internationnal 3D Graphics Conference*, pages 189–196, 1995.

[20] J.E.Colgate, M.C.Stanley, and J.M.Brown. Issues in the haptic display of tool use. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 140–145, Pittsburgh, PA, USA, 1995.

[21] J.Zhang, S.Payandeh, and J.Dill. Haptic subdivision : an approach to defining level-of-detail in haptic rendering. In *IEEE 10th Symposium On Haptic Interfaces for Virtual Environments & Teleoperator Systems*, Orlando, Florida, USA, March 2002.

[22] N.Courty, S.Menardais, D.Margery, F. Lamarche, S.Donikian, and F.Devillers. Towards believable autonomous actors in real-time applications. In *IMAGINA'02*, Monaco, february 2002.

[23] O.R.Astley and V.Hayward. Multirate haptic simulation achieved by coupling finite element meshes through norton equivalents. In *IEEE International Conference on Robotics & Automation*, volume 1, pages 989–994, Leuven, Belgium, May 1998.

[24] S.A.Ehmann and M.C.Lin. Swift : Accelerated proximity queries between convex polyhedra by multi-level voronoi marching. Technical report, Computer Science Department, University of North Carolina at Chapel Hill, 2000.

[25] S.Cotin and H.Delingette. Real-time surgery simulation with haptic feedback using finite element. In *IEEE International Conference on Robotics & Automation*, volume 3, pages 3739–3744, Leuven, Belgium, May 1998.

[26] S.Redon, A.Kheddar, and S.Coquillart. An algebraic solution to the problem of collision detection for rigid polyhedral objects. In *IEEE International Conference on Robotics & Automation*, San Fransisco, CA, USA, April 2000.

[27] S.Redon, A.Kheddar, and S.Coquillart. Gauss' least constraints principle and rigid body simulations. In *IEEE International Conference on Robotics & Automation*, Washington, DC, USA, May 2002.

[28] T.Duval and D.Margery. Building objects and interactors for collaborative interactions with gasp. In *Proceedings of the Third International Conference on Collaborative Virtual Environments (CVE'2000)*, pages 129–138. ACM, September 2000.

[29] T.H.Massie and J.K.salisbury. The phantom haptic interface : A device for probing virtual object. In *ASME International Mechanical Engineering Congress and Exhibition*, pages 295–300, Chicago, Illinois, USA, 1994.

[30] T.Hudson, M.Lin, J.Cohen, S.Gottschalk, and D.Manocha. V-collide : Accelerated collision detection for vrml. In *VRML'97*, Monterey, CA, USA, 1997.

[31] W.A.McNeely, K.D.Puterbaugh, and J.J.Troy. Six degree-of-freedom haptic rendering using voxel sampling. In *SIGGRAPH*, volume 1, pages 401–408, Los Angeles, California, USA, August 1999.

[32] Y.T. Wang, V. Kumar, and J. Abel. Dynamics of rigid bodies undergoing multiple friction contacts. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 2764–2769, Nice, France, May 1992.

Vectorisation d'une courbe discrète standard 2D

Rodolphe BRETON, Eric ANDRES

Laboratoire IRCOM- SIC,Université de Poitiers, BP 30179,
86962 Futuroscope Chasseneuil Cedex, France

{breton,andres}@sic.sp2mi.univ-poitiers.fr

Résumé : *Une nouvelle méthode de vectorisation est proposée. À partir de l'algorithme de reconnaissance de J. Vittone, nous proposons une reconstruction Euclidienne d'une courbe discrète standard 2D ($4-$connexe) qui est inversible et proche du résultat "intuitif" attendu.*

Mots-clés : Géométrie discrète, vectorisation, polygonalisation, reconstruction

1 Introduction

La vectorisation de courbe discrète est un enjeu important depuis déjà une vingtaine d'années. Une nouvelle approche est poursuivie depuis quelques années en France dans la communauté de géométrie discrète (cf. [IDR92] et [JF96]). Nous présentons ici les premiers résultats 2D pour des courbes standard.

En ce qui nous concerne, la vectorisation s'inscrit logiquement dans un projet de modeleur multi-plongement [EA01] au sein duquel elle constitue une étape majeur. Dans ce modeleur, nous voulons gérer tant des objets réels que discrets, et nous voulons manipuler indifféremment un objet dans un plongement discret ou Euclidien.

Nous nous intéressons ici essentiellement aux courbes discrètes standard 2D ($4-$connexité) (cf. Andres [And00] et [And02] et Reveillès [Rev91]) parce que ce modèle possède de bonnes propriétés mathématiques, par opposition au modèle naïf ($8-$connexité), plus classique. De plus, le modèle standard est particulièrement bien adapté à la modélisation dans l'espace des complexes cellulaires discrets (cf. [Kov93]) ; cette dernière permettant une segmentation efficace d'images en régions. Et enfin, ce modèle est aisément extensible aux dimensions supérieures.

L'enjeu ici est classique : passer du discret au continu. Et plus précisément, en partant d'une courbe discrète standard $2D$ (une suite de pixels), on veut reconstruire une courbe Euclidienne polygonale lui correspondant. C'est-à-dire que si on discrétise la courbe Euclidienne ainsi obtenue, on doit retrouver exactement la courbe discrète de départ (cf. Fig. 1).

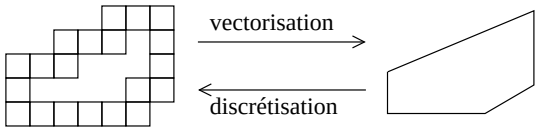


FIG. 1 – Courbe discrète de départ, courbe réelle reconstruite et courbe réelle discrétisée.

Pour vérifier la validité de la méthode, nous avons choisi trois critères[1] qui nous paraissent pertinents :

- l'opération doit être "inversible" (i.e. l'objet réel obtenu peut être discrétisé pour retrouver l'objet discret de départ),
- l'aspect visuel de l'objet réel doit être celui "attendu",
- la méthode doit être la plus générique possible.

Dans une première partie, nous décrirons une première version de notre méthode. Ensuite, dans une seconde partie, nous mettrons en évidence les faiblesses de cette méthode et les améliorations que nous y avons apportées. Et enfin, nous concluerons et envisagerons les futures évolutions de cette technique.

[1] ces critères seront développés dans la section 3

2 Notre méthode

2.1 Rappel sur les droites standard 2D

La discrétisation dans le modèle standard (cf. Andres [And00] et [And02]) de la droite continue $ax + by + c = 0$, avec $a > 0$, est l'ensemble des points (ou pixels) vérifiant la double inéquation suivante :

$$-\omega \leq ax + by + c < \omega \qquad \text{avec} \quad \omega = \frac{|a| + |b|}{2} \quad \text{(épaisseur arithmétique)}$$

Ce qui permet de décrire et de manipuler analytiquement une droite standard.

2.2 Les bases de l'algorithme

Notre méthode de vectorisation d'une courbe discrète consiste à choisir un point de la courbe (une extrémité si celle-ci n'est pas fermée), reconnaître un premier segment et à répéter le processus avec le reste de la courbe.

Parmi les algorithmes de reconnaissance de segments discrets existants, nous avons choisi d'utiliser celui de J. Vittone [Vit99]. Étant originellement prévu pour reconnaître un autre type de droites, les droites naïves ($8-$connexes), nous l'avons adapté au cas standard ($4-$connexité). Ce choix n'est pas le fruit du hasard puisque cet algorithme a l'énorme avantage de donner **toutes** les solutions. C'est-à-dire qu'à partir d'un segment discret, on obtient l'ensemble de toutes les droites réelles qui, discrétisées sur cet intervalle, coïncident parfaitement avec le segment discret de départ. De fait, on peut interpréter cet ensemble de solutions comme une classe d'équivalence. Celle-ci est obtenue sous la forme d'un polygone convexe à trois ou quatre sommets (cf. [ML93]), dans l'espace des paramètres (α, β), qu'on nommera $\mathcal{P}$.

Dans d'espace $\mathcal{P}$, une droite d'équation $y = \alpha x + \beta$ est représentée par un point de coordonnées (α, β). C'est ainsi qu'on fait correspondre à un polygone à trois (resp. quatre) sommets de $\mathcal{P}$, trois (resp. quatre) droites dans l'espace "classique", qu'on appellera $\mathcal{C}$. Dans une première approche, la solution choisie sera la *droite médiane solution*, i.e. la droite passant au centre de l'ensemble de solutions. La figure 2 illustre les cinq formes générales que peut prendre le polygone solution dans $\mathcal{P}$, leur correspondance dans $\mathcal{C}$, ainsi que la droite médiane solution.

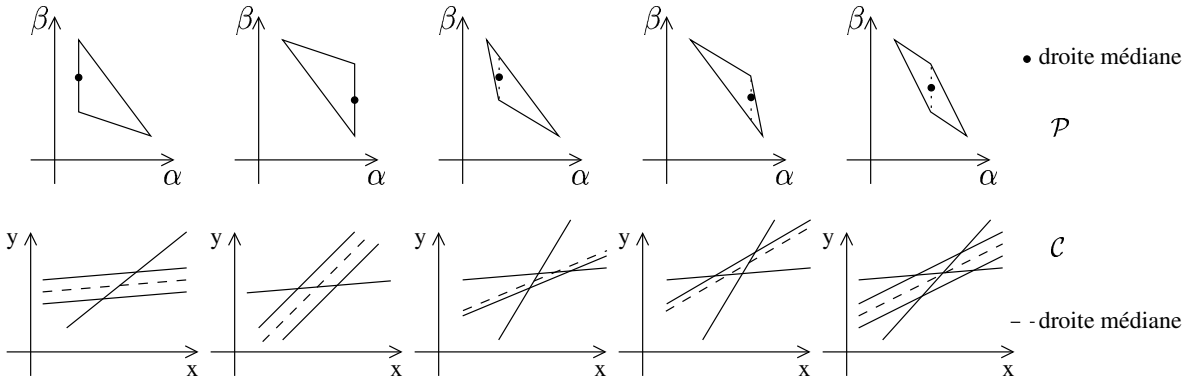


FIG. 2 – Les cinq formes possibles de l'ensemble de solutions et la solution choisie dans chaque cas.

La figure 3 montre un exemple complet de reconnaissance avec cet algorithme. En figure 3 *a.*, on peut voir le segment discret à reconnaître. Une fois l'algorithme déroulé, on obtient un ensemble de triplets d'entiers :

$$\{(2, -1, 4), (4, -5, 6), (4, -3, 6), (6, -7, 8)\}$$

À chaque triplet (a, b, c) on fait correspondre un point $(\frac{a}{c}, \frac{b}{c})$ dans $\mathcal{P}$ et une droite $ax - cy + b = 0$ dans $\mathcal{C}$. On a ainsi une correspondance $1-1$ entre un point dans $\mathcal{P}$ et une droite dans $\mathcal{C}$. D'où, le polygone solution (en figure 3 *b.*) défini par les points suivants : $\{(\frac{1}{2}, -\frac{1}{4}), (\frac{2}{3}, -\frac{5}{6}), (\frac{2}{3}, -\frac{1}{2}), (\frac{3}{4}, -\frac{7}{8})\}$. On peut également observer l'ensemble des solutions représentée par quatre droites (en figure 3 *c.*), l'ensemble étant en fait l'enveloppe convexe de ces quatre droites.

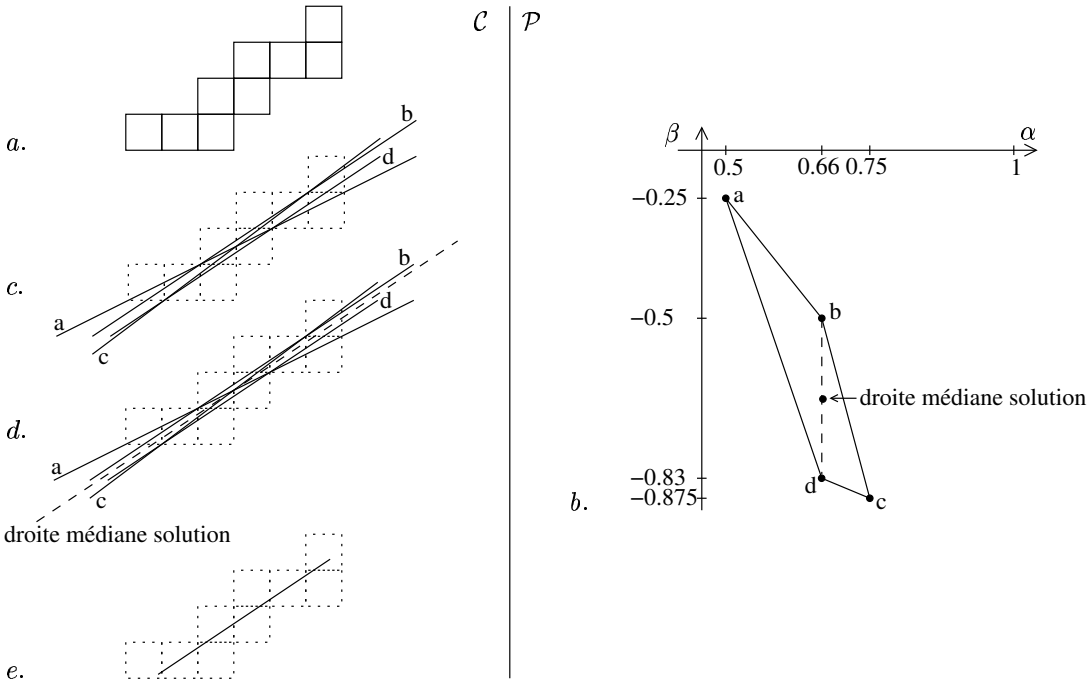


FIG. 3 – Résultat de l'algorithme de Vittone sur un segment.

Ensuite, une fois le segment reconnu, on sélectionne la droite médiane solution dans sa classe d'équivalence (en figure 3 *d.*) et on en déduit enfin les coordonnées des sommets réels (en figure 3 *e.*).

Dans le cas d'un segment isolé, on prend comme extrémités du segment réel, deux points de la droite solution appartenant chacun à un pixel extrémité, et dans le cas d'une ligne polygonale, on prend le point d'intersection des deux droites réelles obtenues successivement. Mais dans ce dernier cas, le point d'intersection n'appartient pas toujours à la courbe discrète, comme nous allons le voir.

La figure 4 montre le cas de figure le plus simple. On a reconnu deux segments discrets s_k et s_{k+1} ayant un pixel en commun et les droites solutions retenues s'intersectent dans ce pixel commun. Il suffit donc de prendre ce point d'intersection comme fin du premier segment réel et comme début du second.

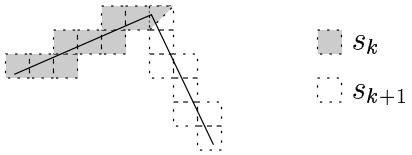


FIG. 4 – Intersection triviale.

Mais on peut très bien être dans un cas où les droites s'intersectent à l'extérieur de ce pixel (cf. Fig. 5), voire ne s'intersectent pas du tout (cf. Fig. 6). Dans ces derniers cas, on est contraint d'ajouter un petit segment pour joindre les extrémités des deux segments réels. On appelle un tel segment un *joint*.

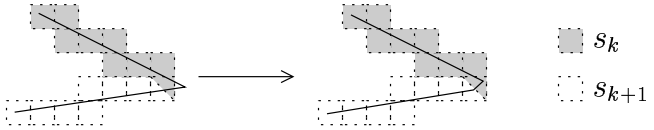


FIG. 5 – Intersection hors du pixel commun aux deux segments ⇒ ajout d'un joint.

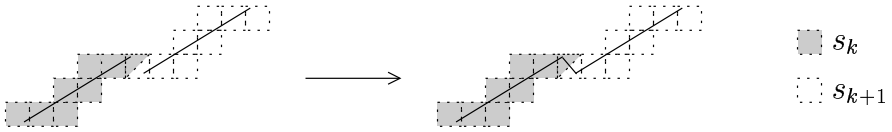


FIG. 6 – Intersection hors du pixel commun aux deux segments ou aucune intersection ⇒ ajout d'un joint.

En itérant ces opérations de base sur toute la longueur de la courbe, on obtient au final une ligne polygonale continue correspondant à l'objet discret initial.

Nous allons maintenant présenter l'algorithme global de reconnaissance, mais auparavant, introduisons quelques notations.

Notations :
Soit une suite ordonnée de n pixels $p_1 \dots p_n$ représentant un segment discret s_k, on note $s_k = [p_1, p_n]$ (avec $k > 0$). On note également d_k la droite réelle choisie comme solution Euclidienne de s_k. Et on note enfin r_k le segment réel solution, porté par d_k et dont les extrémités appartiennent respectivement à p_1 et p_n.

2.3 Première version de l'algorithme

Initialisation :
- au départ, on a une courbe discrète, représentée par une suite ordonnée de n pixels : $p_1 \dots p_n$

Étape 1 : Reconnaissance
- on note le segment courant s_k (au début $k = 1$)
- on note le pixel courant p_i (au début $i = 2$)
- on utilise l'algorithme de J. Vittone pour reconnaître un segment discret :
  - on injecte le pixel p_i dans s_k
  - si s_k ainsi augmenté est toujours un segment discret, on continue : $i = i + 1$
  - sinon s_k s'arrête en p_{i-1} et ce pixel devient le point de départ du nouveau segment : $i = i - 1$ et $k = k + 1$
- jusqu'au dernier pixel $(i = n)$
- la courbe est alors entièrement reconnue et polygonalisée en k segments discrets, chacun associé à une classe d'équivalence représentant toutes les solutions continues valides

Étape 2 : Reconstruction
- pour chacune de ces classes d'équivalence, on prend la droite médiane solution de l'ensemble d_k
- il reste à construire les segments réels r_k portés par les droites d_k
- pour cela, on commence par fixer la première extrémité du premier segment réel r_1 en prenant un point de d_1 appartenant à p_1 (le premier pixel de la courbe)
- puis, on commence une boucle sur l'ensemble des droites d_k trouvées précédemment :
  - on regarde si d_k et d_{k+1} s'intersectent bien dans le pixel commun aux deux segments s_k et s_{k+1}
  - si tel est le cas, ce point d'intersection devient la seconde extrémité de r_k et la première de r_{k+1}
  - sinon (intersection à l'extérieur ou aucune intersection), on crée un petit joint et dans ce cas, la seconde extrémité de r_k est le premier sommet du joint et la première extrémité de r_{k+1} est le second sommet du joint
- on a alors une suite de segments réels r_k (décrits chacun par deux points réels) formant ainsi une ligne polygonale, fermée ou non, et dont la discrétisation standard coïncide parfaitement avec l'objet discret de départ

3 Améliorations de l'algorithme

La méthode de base marche bien mais les résultats obtenus ne sont pas toujours de très bonne qualité "visuelle". On obtient en particulier des lignes polygonales "très brisées" (cf. Fig. 6). Voici plusieurs améliorations pour pallier à

ce type de problème.

• **Jonction premier-dernier segment :** (amélioration de la reconnaissance)

Un premier problème apparaît dans le cas où l'objet discret est fermé, quand le premier point de la reconnaissance se situe au milieu d'un segment discret. Dans ce cas, après la reconnaissance, le premier et le dernier segment auraient pu être fusionnés en un seul. D'où une première amélioration de notre méthode qui consiste à essayer de prolonger la reconnaissance du dernier segment de la courbe avec les premiers pixels de la courbe.

• **Retrait systématique du dernier pixel reconnu :** (amélioration de la reconnaissance)

Pour le moment, lors de la reconnaissance d'un segment discret, on essaye d'aller le plus loin possible, i.e. on décide que la fin du segment n'est atteinte que lorsque le dernier pixel trouvé ne peut plus s'y ajouter. Cependant, une telle règle entraîne des configurations peu esthétiques comme illustré sur la figure 7.

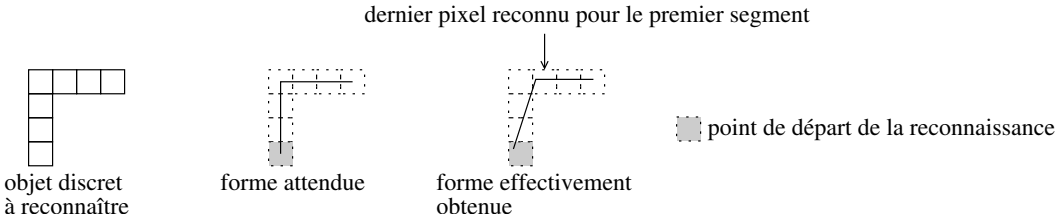


FIG. 7 – Problème de la reconnaissance maximale.

C'est pourquoi, on peut décider de retirer systématiquement le dernier pixel de chaque segment reconnu. Mais cette nouvelle règle a un autre défaut : elle rend la reconnaissance encore plus dépendante du sens de parcours de la courbe discrète (cf. Fig. 8). D'où l'amélioration décrite ci-après.

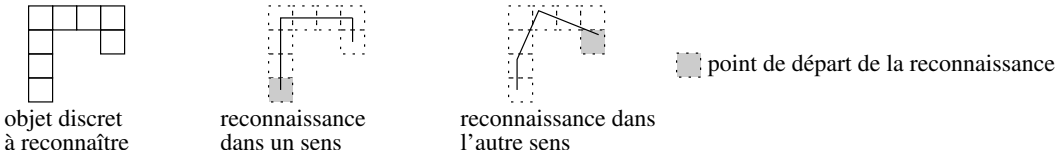


FIG. 8 – Problème du retrait systématique du dernier pixel reconnu.

• **Points de rebroussement et retrait intelligent :** (amélioration de la reconnaissance)

Un *point de rebroussement*, au sens mathématique, est un point d'une courbe où celle-ci admet deux tangentes distinctes à gauche et à droite de ce point. Nous allons introduire la notion de *point de rebroussement discret* définie ainsi : un point d'une courbe discrète est un *point de rebroussement discret* si le segment constitué de ce point, des deux points précédents et des deux points suivants, n'est pas un segment discret (cf. Fig. 9). De fait, un tel point ne peut pas se situer en plein milieu d'un segment, mais seulement à une extrémité (à un pixel près). Et donc, si on repère ces points sur une courbe, on s'aperçoit qu'ils se trouvent systématiquement à la jonction de deux segments discrets. On peut donc les utiliser comme des sorte d'"aimants" destinés à être en priorité des points de départ et de fin de segments.

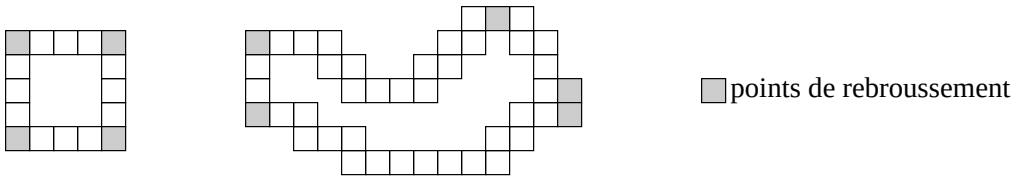


FIG. 9 – Exemple de points de rebroussement sur deux courbes.

Le code de Freeman du voisinage d'un point nous permet de déterminer très simplement si ce point est un point de rebroussement discret (cf. Fig. 10).

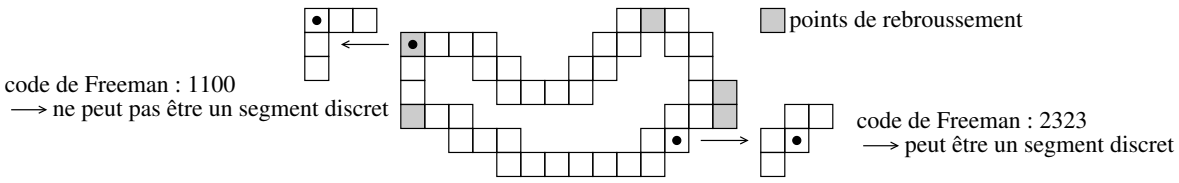


FIG. 10 – Détermination des points de rebroussement.

Utilisation des points de rebroussement :
Lors de la reconnaissance d'un segment s_k, on peut maintenant choisir de garder ou non le dernier pixel trouvé p_i, selon les trois cas de figure suivants :

- p_i est un point de rebroussement : s_k s'arrête sur p_i,
- p_{i-1} est un point de rebroussement : s_k s'arrête sur p_{i-1}, afin de mieux "coller" à l'allure globale de la courbe,
- p_{i+1} est un point de rebroussement : s_k s'arrête sur p_{i-1} ; en effet, par définition, un segment discret ne pouvant contenir un point de rebroussement qu'à une de ses extrémités (à un pixel près), si s_k finit en p_i, le prochain segment aura une longueur maximale de trois pixels et de manière générale, les segments trop courts correspondent assez peu au résultat souhaité,
- dans tous les autres cas : s_k s'arrête sur p_i.

Les points de rebroussement permettent de régler le cas de la figure 11 où on voit très bien que si les deux courbes réelles correspondent bien à la courbe discrète, la première est plus pertinente.

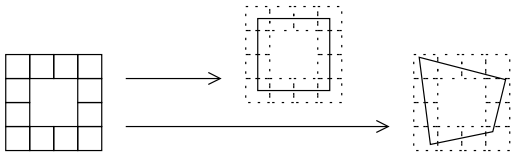


FIG. 11 – Plusieurs courbes réelles correspondent à une même courbe discrète.

De plus, pour s'assurer de respecter l'allure générale de la courbe et éviter le cas illustré par la figure 12, on peut maintenant décider de ne commencer la reconnaissance que sur un point de rebroussement (dans le cas d'une courbe fermée).

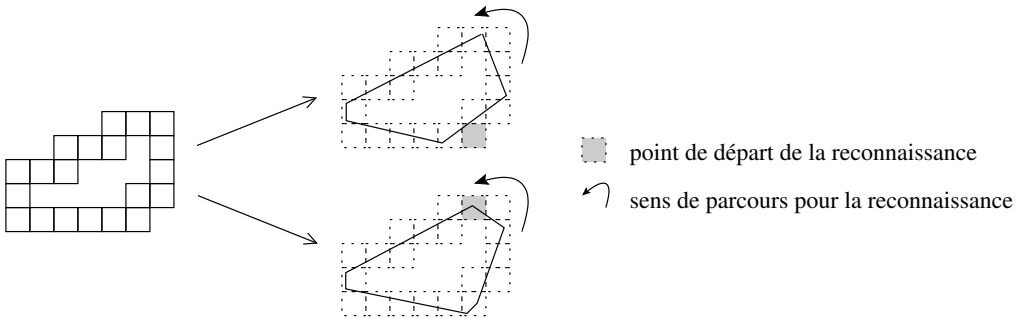


FIG. 12 – La même courbe discrète, reconnue en partant de deux points différents.

• **Élargissement de la zone possible d'intersection :** (amélioration de la reconstruction)

Jusqu'à maintenant, si le point d'intersection des deux droites continues se situait en dehors du sommet discret, nous ajoutions un joint. Or, on s'aperçoit que si l'intersection se situe dans l'un des deux pixels voisins (nous travaillons en $4-$connexité), on peut également conserver ce point d'intersection comme extrémité du futur segment réel.

Preuve :
Prenons deux segments s_1 et s_2 ayant le pixel p_i en commun, avec p_{i-1} appartenant à s_1 et p_{i+1} à s_2. Nommons d_1 et d_2 les deux droites réelles solutions de la reconnaissance, telles que d_1 et d_2 sont sécantes en I. Les deux cas étant symétriques, nous allons considérer que le point d'intersection I se situe à l'intérieur du pixel p_{i+1} (cf. Fig. 13). Puisque p_{i+1} appartient à s_2, si le segment réel correspondant à s_2 débute en I, on perd juste le pixel p_i par rapport au segment discret reconnu. Cependant, la droite d_1 passe par p_{i+1} (puisqu'elle y intersecte d_2). Ce qui implique qu'en fait, on peut prolonger le segment discret s_1 jusqu'en p_{i+1}. Et donc, p_{i+1} devient le nouveau pixel commun aux deux segments discrets.

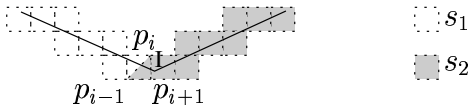


FIG. 13 – Intersection dans un pixel voisin du pixel commun aux deux segments discrets.

• **Reconnaissance inverse :** (amélioration de la reconstruction)

On peut éviter l'ajout de certains joints en effectuant, à certains endroits, une reconnaissance de Vittone dans le sens opposé au parcours choisi pour la reconnaissance.

Exemple :
Soient deux segments discrets s_1 et s_2. Après la reconnaissance, on connaît leurs pixels extrêmes : $s_1 = [p_a, p_b]$ et $s_2 = [p_b, p_c]$. On a également obtenu deux droites d_1 et d_2 ne s'intersectant ni dans p_b, ni dans p_{b-1}, ni dans p_{b+1} (cas précédent). On devrait donc avoir un joint. Mais on peut éventuellement l'éviter. Il suffit de tenter une reconnaissance en sens inverse entre les pixels p_c et p_a.
Appelons s'_2 le nouveau segment discret obtenu, p_d le pixel où s'est arrêtée la nouvelle reconnaissance, avec $a < d \leq b$ (on a donc $s_2 = [p_d, p_c]$), et d'_2 la droite solution correspondante. On a alors deux cas. Soit la reconnaissance s'est à nouveau arrêtée sur $p_b = p_d$ et le joint est alors inévitable puisque $d'_2 = d_2$. Soit la reconnaissance est allée au-delà de p_b (cf. Fig. 14) et on pourra éviter le joint si d_1 et d'_2 s'intersectent dans l'un des pixels de $[p_d, p_b]$. En effet, on constate que les pixels situés entre p_d et p_b appartiennent tous aux deux segments discrets $[p_a, p_b]$ et $[p_d, p_c]$ puisque ces deux segments se chevauchent. Donc, si la nouvelle intersection se situe dans cet intervalle, on peut prendre ce point comme sommet commun aux deux segments réels.

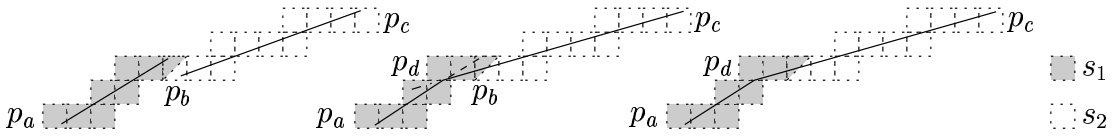


FIG. 14 – Elimination d'un joint grâce à une seconde reconnaissance.

• **Adoucissement des joints :** (amélioration de la reconstruction)

Et enfin, quand le joint est inévitable, on peut quand même s'arranger pour qu'il s'intègre le mieux possible dans la courbe, contrairement à ce que nous pouvons voir sur la figure 6. En effet, supposons qu'on a reconnu deux segments discrets $s_k = [p_a, p_b]$ et $s_{k+1} = [p_b, p_c]$ et qu'on soit obligé d'ajouter un joint entre ces segments car les deux droites solutions d_k et d_{k+1} ne sont pas sécantes (cf. Fig. 15 à gauche). On aurait normalement un joint en p_b, illustré sur la figure 15 au milieu.

Mais en fait, on peut construire un autre joint. Il suffit de prendre comme point de départ, l'intersection entre d_k et le segment commun à p_{b-2} et p_{b-1}, et comme point d'arrivée, l'intersection entre d_{k+1} et le segment commun à p_{b+1} et p_{b+2}. Il est évident que les deux segments réels ainsi obtenus décrivent les segments discrets $[p_a, p_{b-2}]$ et $[p_{b+2}, p_c]$. Et de même, le joint ainsi construit décrit les trois pixels manquants, à savoir $[p_{b-1}, p_{b+1}]$.

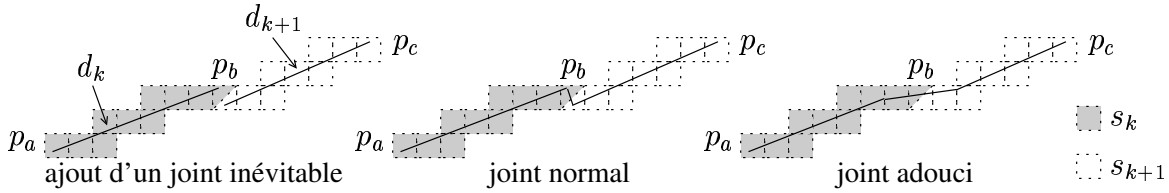


FIG. 15 – Cas d'un ajout de joint inévitable.

Ces six points constituent de bonnes améliorations de l'algorithme précédent qui peut alors être réécrit comme suit.

3.1 L'algorithme amélioré

Initialisation :
- au départ, on a une courbe discrète, représentée par une suite ordonnée de n pixels : $p_1 \dots p_n$

Étape 0 : Rercherche des points de rebroussement
- on répertorie les points de rebroussement de la courbe (cf. Fig. 10).

Étape 1 : Reconnaissance
- on note le segment courant s_k (au début $k = 1$)
- on note le pixel courant p_i (au début $i = 2$)
- on utilise l'algorithme de J. Vittone pour reconnaître un segment discret :
  - on injecte le pixel p_i dans s_k
  - si s_k ainsi augmenté est toujours un segment discret, on continue : $i = i + 1$
  - sinon s_k s'arrête en p_{i-1} et en appliquant la règle établie précédemment, soit p_{i-1}, soit p_{i-2} devient le point de départ du nouveau segment : $i = i - 1$ (ou $i = i - 2$) et $k = k + 1$
- jusqu'au dernier pixel $(i = n)$
- si la courbe est fermée, alors on poursuit la reconnaissance jusqu'au prochain point de rebroussement et on fusionne éventuellement le dernier et le premier segment
- la courbe est alors entièrement reconnue et polygonalisée en k segments discrets, chacun associé à une classe d'équivalence représentant toutes les solutions continues valides

Étape 2 : Reconstruction
- pour chacune de ces classes d'équivalence, on prend la droite médiane solution de l'ensemble d_k
- il reste à construire les segments réels r_k portés par les droites d_k
- pour cela, on commence par fixer la première extrémité du premier segment réel r_1 en prenant un point de d_1 appartenant à p_1 (le premier pixel de la courbe)
- puis, on commence une boucle sur l'ensemble des droites d_k trouvées précédemment :
  - on regarde si d_k (segment $s_k = [p_a, p_b]$) et d_{k+1} (segment $s_{k+1} = [p_b, p_c]$) s'intersectent bien dans p_b, p_{b-1} ou p_{b+1}
  - si tel est le cas*, ce point d'intersection devient la seconde extrémité de r_k et la première de r_{k+1}
  - sinon (intersection à l'extérieur ou aucune intersection), on refait une reconnaissance entre p_c et p_a et on a deux cas :
    - ▷ on a toujours les deux mêmes segments s_k et s_{k+1}, le joint est inévitable, et dans ce cas, la seconde extrémité de r_k est le premier sommet du joint et la première extrémité de r_{k+1} est le second sommet du joint
    - ▷ s_{k+1} a été allongé et l'intersection entre d_k et la nouvelle droite solution permet d'éviter le joint ; on retombe alors dans le cas normal (cf. $*$)
- on a alors une suite de segments réels r_k (décrits chacun par deux points réels) formant ainsi une ligne polygonale, fermée ou non, et dont la discrétisation standard coïncide parfaitement avec l'objet discret de départ

4 Conclusion

4.1 Résultats

L'algorithme présenté ici répond en partie à nos attentes, à savoir, il permet d'obtenir une courbe continue à partir d'une courbe discrète standard 2D et la courbe réelle ainsi obtenue peut être discrétisée pour obtenir de nouveau la courbe discrète. De plus, l'allure générale de la courbe calculée correspond relativement bien à notre intuition. Cependant, même si nous avons optimisé la reconnaissance, notamment grâce aux points de rebroussement, cette méthode est encore un peu dépendante du point de départ de la reconnaissance et du sens de parcours.

La figure 16 montre deux exemples obtenus en affichant sur une même courbe discrète, les différentes courbes réelles obtenues en faisant varier le point de départ de la reconnaissance.

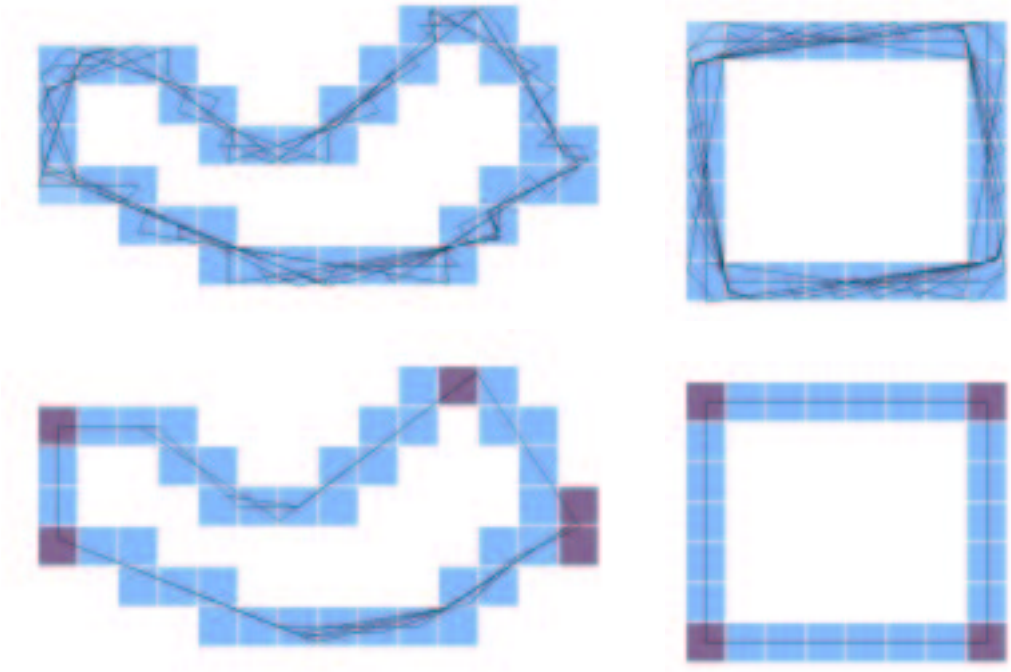


FIG. 16 – En haut, résultats obtenus avec la première version de l'algorithme, en bas, avec la version actuelle. En foncé, les points de rebroussement.

4.2 Perspectives

Le travail est loin d'être terminé et plusieurs améliorations peuvent encore être apportées.

Dans un premier temps, nous allons regarder du côté des points de rebroussement. En effet, si nous trouvons plusieurs points de rebroussement consécutifs, cela "déconcerte" l'algorithme de reconnaissance ; il faudrait donc en éliminer. À l'inverse, peut-être faut-il ajouter de nouveaux points de rebroussement là où, manifestement, il y aura une jonction entre deux segments mais où le critère de détection n'est pas rempli.

Ensuite, nous envisageons d'adapter notre algorithme à divers domaines applicatifs, et notamment à la segmentation d'images où on doit reconnaître plusieurs régions différentes. Le problème surviendra lorsque deux régions auront une frontière commune, puisque pour le moment, la reconnaissance n'est pas unique.

Et enfin, une fois cette méthode éprouvée sur le cas 2D, il est prévu de l'adapter au cas 3D.

Références

[And00] Eric Andres. Modélisation analytique discrète d'objets géométriques, 2000. Université de Poitiers, Habilitation à diriger des recherches.

[And02] Eric Andres. Defining discrete objects for polygonalization : the standard model. In J.-O. Lachaud A. Braquelaire and A. Vialard, editors, *Discrete Geometry for Computer Imagery 2002*, volume 2301 of *Lecture Notes in Computer Science*, pages 313–325, Bordeaux, France, april 2002. Springer.

[EA01] Pascal Lienhardt Eric Andres, Rodolphe Breton. Spamod : design of a spatial modeling tool. In Atsushi Imiya Gilles Bertrand and Reinhard Klette, editors, *Digital and Image Geometry, Advanced Lectures*, volume 2243 of *Lecture Notes in Computer Science*, pages 90–107. Springer, 2001.

[IDR92] J.-P. Reveillès I. Debled-Rennesson. Un algorithme linéaire de polygonalisation des courbes discrètes. In *Discrete Geometry for Computer Imagery 1992*, Grenoble, France, septembre 1992.

[JF96] M. Tajine J. Françon, J.-M. Schramm. Recognizing arithmetic straight lines and planes. In *Discrete Geometry for Computer Imagery 1996*, volume 1176 of *Lecture Notes in Computer Science*, pages 141–150, Lyon, France, novembre 1996.

[Kov93] V. Kovalesky. Digital geometry based on the topology of abstract cell complexes. In *Discrete Geometry for Computer Imagery 1993*, pages 259–284, Université Louis Pasteur, Strasbourg, France, septembre 1993.

[ML93] A. Bruckstein M. Lindenbaum. On recursive, $o(n)$ partitioning of a digitized curve into digital straight segments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9) :949–953, september 1993.

[Rev91] J.-P. Reveillès. Géométrie discrète, calcul en nombres entiers et algorithmique, décembre 1991. Université Louis Pasteur, Strasbourg, Thèse d'état.

[Vit99] Joëlle Vittone. *Caractérisation et reconnaissance de droites et de plans en géométrie discrète*. PhD thesis, Université Joseph Fourier - Grenoble 1, décembre 1999.

Vers une approche plus intuitive des systèmes de C.A.O.

E. Malik*, Y. Gardan*, E. Perrin**

I.F.T.S.* / Université de Metz**
Equipe C.M.C.A.O.
erwan.malik@univ-reims.fr gardan@infonie.fr
perrin@sciences.univ-metz.fr

Résumé : *Le projet DIJA propose, en plus d'être dédié à Internet, une approche synthétique de la conception de solides contrairement aux méthodes de conception classiques plus constructives. Cette méthode a l'avantage d'être adaptée au design et d'éviter certaines imprécisions de calcul pouvant survenir lors d'une conception par assemblage. La mise en œuvre des déformations demande de trouver un modèle spécifique qui s'adapte à notre représentation des objets. Ceux-ci sont constitués d'éléments de dialogue couplés à des règles de construction. Nous distinguons deux types de déformations : les déformations directes qui peuvent modifier directement les éléments de dialogue et les déformations nécessitant l'utilisation d'un modèle intermédiaire. Le modèle de déformation doit donc nous permettre de modifier la forme d'un objet en utilisant des actions simples et naturelles (étirement, torsion, compression, . . .) tout en respectant une cohérence physique. Nous avons répertorié différentes méthodes en analysant, pour chacune d'elles, la manière dont elles pourraient s'adapter à nos besoins. Notre étude porte sur les éléments finis, les masses–ressorts, les particules et les smoothed particle hydrodynamics . En conclusion, nous abordons les orientations actuelles du projet pour le contrôle des déformations en nous intéressant à l'interface homme–machine. Nous montrons comment une interaction plus qualitative peut aider l'utilisateur dans les étapes conduisant à la définition de la forme souhaitée.*

Mots-clés : C.A.O., Interface Homme–Machine, Eléments finis, Particules, Masses–Ressorts, Déformation, Logique floue

1 Introduction

Traditionnellement, les systèmes de C.A.O. sont bâtis autour d'une approche constructive. Pour créer un objet, l'utilisateur ajoute des éléments les uns aux autres, par exemple par combinaison booléenne, jusqu'à ce qu'il obtienne la forme désirée. Cette approche nécessite d'avoir une vision précise de l'enchaînement des étapes nécessaires à la réalisation de celui-ci. Or, cette condition n'est pas toujours vérifiée. Un utilisateur novice dans le maniement des systèmes de C.A.O. n'aura sans doute pas assez d'expérience pour espérer concevoir une pièce aux formes complexes. L'inconvénient majeur survient quand on s'intéresse aux premières phases de conception, celles concernant le design de l'objet. L'important est alors d'avoir un point de vue global concernant sa future forme et non la manière dont celui-ci va être construit [GPD+02].

Dans ce domaine, le projet DIJA souhaite apporter une approche innovante. Nos objectifs principaux sont de rendre la création assistée par ordinateur plus intuitive, que ce soit pour permettre une prise en main plus facile pour l'utilisateur néophyte ou pour permettre au *designer* de se concentrer sur la forme de son objet plutôt que sur les outils proposés par le système. Pour y parvenir le système DIJA est basé sur une méthode synthétique, par opposition aux méthodes constructives. L'utilisateur commence par choisir une forme s'approchant de celle qu'il désire puis il la déforme successivement jusqu'à aboutir à la forme souhaitée. Il est à noter que le fait de choisir une forme initiale éloignée de la forme souhaitée n'a aucune incidence sur notre méthode ; cela ralentit seulement le processus.

Cette démarche place les déformations au cœur de l'architecture du système DIJA. En effet, d'une part il faut un modèle de déformation qui soit puissant (comportement réaliste, possibilités de déformation variées, . . .) et facile d'utilisation (paramètres aisés à appréhender et à manipuler). En outre, le modèle se doit de tirer le maximum parti des particularités de DIJA qui sont principalement liées à la structure des objets. Celle-ci est basée sur des éléments de dialogue (fibre, ligne caractéristique, contour caractéristique) [DDGP02]. Par exemple un cylindre peut êrte composé de deux contours caractéristiques définissant sa face supérieure et inférieure, d'une fibre qui aura

le rôle d'axe de révolution et d'une autre servant de profil (cette représentation n'est pas la seule). Ces éléments sont liés ensemble par des règles de construction dont l'application permet de construire le solide. Notre objectif est d'étudier l'outil de déformation de DIJA dont l'action se situe au niveau des éléments de dialogue. Naturellement, des problèmes de cohérence peuvent survenir lors d'une déformation ce qui peut induire des difficultés pour reconstruire un objet valide une fois un élément de dialogue déformé. Malgré cela, et bien que la reconstruction soit une étape très importante dans notre processus, nous n'aborderons pas cette problématique dans ces pages.

Nous nous proposons dans cet article de passer en revue différents modèles de déformation classiquement utilisés. Nous essayons de dégager les avantages et les inconvénients de chacun d'eux pour déterminer les possibilités d'adaptation à notre projet. Nous continuons en exposant notre approche concernant le contrôle de ces déformations via l'interface de DIJA.

2 Positionnement du Problème

Les déformations dans le projet DIJA sont basées sur l'architecture exposée dans la figure 1. Les objets sont construits à partir d'éléments de dialogue couplés à des règles de construction. Nous faisons la distinction entre deux types de déformations. D'une part, il y a les déformations qui sont directement applicables aux éléments de dialogue que nous appelons déformations directes. D'autre part nous trouvons des déformations, comme celle résultant d'une action telle que "bomber", qui ont besoin de s'appuyer sur un modèle intermédiaire afin d'avoir un comportement cohérent. En effet, il peut être utile d'appliquer des déformations à une surface, ou à un volume calculé à partir de l'élément que l'utilisateur souhaite manipuler. La déformation est, dans ce cas, calculée sur le volume ou la surface qui sera ensuite traité pour extraire l'élément de dialogue final. Afin de clarifier la situation, nous appelons "entité" indifféremment un élément de dialogue (dans le cas d'une déformation directe) ou le volume ou la surface utilisée par le modèle de déformation intermédiaire.

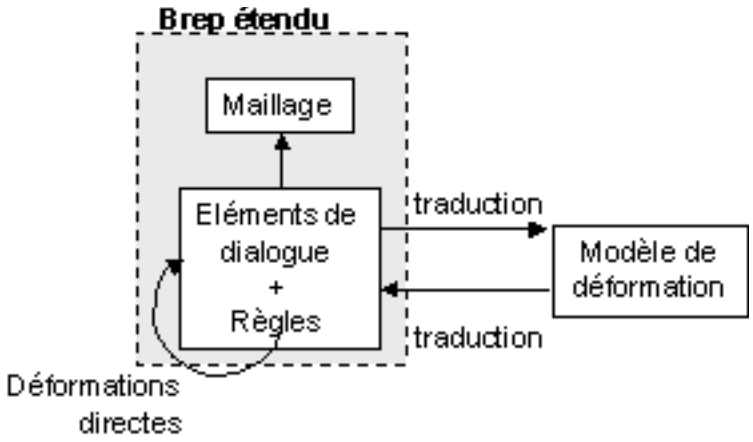


FIG. 1 – *Architecture des déformations*

Les critères qui vont nous servir à étudier les méthodes de déformations sont :

- La souplesse d'utilisation :
  - La possibilité de modifier n'est pas une restriction.
  - Le nombre et la complexité des paramètres est à prendre en compte.
- La stabilité numérique.
- Les contraintes temporelles.

3 Modèles de déformation

3.1 Eléments finis

Les éléments finis sont traditionnellement utilisés en physique pour simuler le comportement des matériaux soumis à des forces [Cot97, TC97]. Comme on ne peut pas résoudre les équations qui régissent le comportement d'un matériau en milieu continu, l'entité est discrétisée pour obtenir une représentation sous forme de maillage de l'entité initiale. Les forces de déformation sont ensuite calculées en chacun des nœuds [Gay89]. L'objectif est de minimiser l'énergie contenu dans le système afin de trouver l'état d'équilibre. Pour cela, on cherche à résoudre un système d'équations afin de trouver comment les points de notre réseau vont se déplacer sous l'effet des contraintes. Cette technique fait donc appel à une méthode de résolution implicite.

La méthode a l'avantage d'être très précise, par contre, elle est relativement complexe à mettre en œuvre. Son inconvénient majeur reste son coût algorithmique prohibitif. En effet, chaque étape de déformation nécessite une inversion de matrice dont la taille varie en fonction du carré du nombre de nœuds. De plus, ce système d'équation est basé sur les relations topologiques qui existent entre chaque nœud du réseau. Il faut donc prendre en compte les problèmes liés aux auto-intersections ce qui alourdit encore le coût en temps de calcul [HFS+01].
Néanmoins, [Cot97] propose une approche originale avec une formulation explicite de la méthode des éléments finis. Son objectif est de définir un modèle anatomique déformable en temps réel. Les méthodes basées sur l'utilisation interactive de déformation via les éléments finis effectuent des pré-calculs en fonction de l'entité qui doit être déformée.

Bien que cette méthode offre des résultats très réalistes et très précis, son coût de calcul en fait une méthode inadaptée à nos objectifs. En outre l'utilisation de pré-calculs ne nous convient pas non plus car la forme de notre entité est définie de manière dynamique par l'utilisateur. Par ailleurs, il est difficile de changer la topologie de l'entité au cours du temps.

3.2 Masses - Ressorts

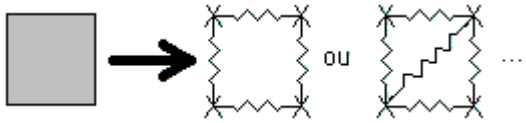


FIG. 2 – *Transformation d'une entité continu en un système masses - ressorts*

Il s'agit d'une méthode qui est utilisée depuis longtemps dans le domaine de l'animation (comme par exemple pour l'animation d'un visage chez [PB81]). On discrétise l'entité afin d'obtenir une représentation sous forme d'un ensemble de points auxquels on associe une masse ponctuelle (cf. figure 2). Chacun de ces points est relié à n de ces voisins. Pour décrire le comportement de l'objet, on exprime le bilan des forces appliquées à chaque point. Ce bilan s'exprime via l'équation suivante [Deb00] :

$$\overrightarrow{f_i} = \sum_{voisinsj} k_{i,j}(l_{i,j} - l_{i,j}^{initiale})\frac{\overrightarrow{IJ}}{l_{i,j}} \tag{3.1}$$

$k_{i,j}$ représente la raideur du ressort entre le point i et le point j, $l_{i,j}$ est la distance séparant les deux points. Cette équation décrit le comportement d'un ressort simple mais on peut aussi utiliser des ressorts plus complexes (avec amortissement, ressort en torsion, ...).
La méthode de résolution classique est une méthode explicite (méthode de résolution itérative) ce qui signifie que

les résultats dépendent du choix d'un intervalle de temps. A chaque pas de temps, on calcule le bilan des forces pour l'ensemble des points du réseau. La valeur des déplacements est ensuite modifiée en fonction de l'intervalle temporel (seule une fraction du déplacement initial est conservée). Les points sont ensuite effectivement déplacés. Au terme de cette phase, on teste la stabilité du nouveau système ; si le système n'est pas stable, le processus est répété jusqu'à satisfaction de cette contrainte.

Le principal avantage de cette méthode réside dans sa simplicité d'utilisation et dans son caractère intuitif en grande partie du à la méthode de résolution explicite. De plus, en prenant certaines précautions, elle peut converger rapidement vers une solution acceptable. Toutefois cette méthode n'est pas exempte de défauts. En effet, dans sa version explicite, elle dépend grandement de la rigidité du système étudié. Plus le système est rigide, plus le pas de temps d'intégration doit être faible entraînant ainsi un nombre d'itérations grand afin d'éviter des divergences éventuelles. On peut aussi ajouter des tests chargés d'empêcher des changements trop brutaux dans notre système. Malgré cela, l'instabilité n'est pas levée : on ne peut garantir le bon fonctionnement de la méthode quelle que soit la rigidité du système. Une solution serait de changer la méthode de résolution et d'opter pour une méthode implicite comme celle proposée par [BW98] pour modéliser le comportement avec une rigidité élevée.
On trouvera une étude comparative portant sur les différentes méthodes de résolutions dans [VMT01]. Celle-ci analyse les points forts et les inconvénients de la méthode explicite du point milieu (*explicit midpoint method*), de la méthode explicite de Runge–Kutta et de la méthode d'Euler inverse qui est une méthode implicite.
La méthode des masses–ressorts est bâtie sur un réseau préétabli (les points et leurs liaisons), ce qui rend difficile un changement de topologie de l'objet.

La méthode est simple et peut être efficace si on prend soin de limiter la rigidité du système. Nous pouvons envisager de l'utiliser pour déformer des éléments de dialogue en appliquant des forces de déformations à un ensemble de points reliés (cas où on ne connaît pas l'élément final), ou en déplaçant certains points d'un élément de dialogue de façon à obtenir les répercutions sur la globalité de l'objet.
En outre, déterminer la raideur associée à chaque ressort de façon à obtenir l'effet souhaité n'est pas aisé. Cette méthode offre des possibilités limitées concernant les changements de topologies. En effet, une fois celle-ci fixée, il est difficile d'en changer au cours d'une déformation sous peine d'aboutir à une forme non cohérente.

3.3 Particules

Il s'agit d'une extension du modèle masses–ressorts [Deb00]. Il faut discrétiser l'entité afin d'obtenir un ensemble de points la caractérisant. Ici les liens entre les masses ponctuelles ne sont plus préétablis mais évoluent dynamiquement [WH94]. Chaque particule peut interagir avec l'ensemble des autres particules.
Les forces qui régissent ces interactions sont de type attraction–répulsion. La figure 3 représente la force exercée par une particule i sur une particule j en fonction de la distance les séparants (d) :

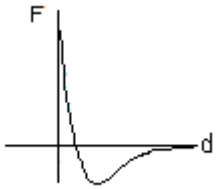


FIG. 3 – *Force d'interaction de type Lennard–Jones*

Comme avec les masses–ressorts, pour connaître le comportement de l'objet, on exprime le bilan des forces appliquées à chaque particule puis on intègre les résultats au système afin d'obtenir la nouvelle position des particules dans l'espace. Le processus peut être stoppé une fois atteint un état stable. Il s'agit donc d'une méthode explicite. Voici l'algorithme utilisé pour cette méthode :

– *Tant que non(Etat Stable)*

- *Pour chaque particule i*
  - *Pour chaque particule j*
    Calculer la force exercée par la particule j sur la particule i
    Mettre à jour la somme des forces appliquées sur la particule i
    Calculer la force exercée par la particule i sur la particule j
    Mettre à jour la somme des forces appliquées sur la particule j
  - *Fin pour*
- *Fin pour*
- *Mise à jour des positions des particules en fonctions des forces qui leurs sont appliquées.*
- *Fin Tant que*

Une nouvelle fois, l'intérêt de cette méthode réside en partie dans sa simplicité. A l'aide d'une loi de comportement simple au niveau atomique, on obtient un comportement global complexe. Cette méthode autorise les changements de topologie en fonction du choix de l'utilisateur.
[JV02] ont développé une méthode permettant de passer d'un modèle BRep définissant l'objet à un modèle physique à particules (en réalité, leur modèle se situe entre les systèmes masses-ressorts et les systèmes à particules) sur lequel sont effectuées les déformations pour revenir au BRep au moment de l'affichage.
Dans [ST92], les particules sont orientées et sont uniquement réparties sur la surface de l'objet. Les forces qui les régissent sont basées sur des critères géométriques locaux. Les centres des particules sont ensuite maillés comme un nuage de points mais en tenant compte de l'orientation des particules pour diminuer les incohérences topologiques. Leur méthode permet des déformations de la surface avec changement de topologie à l'aide d'outils virtuels.
Par contre, la complexité des méthodes à particules peut s'avérer un problème pour un traitement en temps réel. Dans sa version de base (tel qu'il à été exposé précédemment) l'algorithme a un coût en $O(n^2)$ (n étant le nombre de particules). En effet, chaque particule peut théoriquement interagir avec n'importe quel autre de ces voisins. En réalité, la force d'interaction tend vers zéro lorsque la distance devient grande. Une technique revient alors à attribuer un rayon d'action à chacune des particules au delà duquel il n'y a plus d'interaction possible. On peut ainsi partitioner l'espace de façon à limiter le coût de l'algorithme en ne travaillant qu'avec les voisins "utiles" d'une particule ramenant ce coût à une complexité quasi linéaire.

Le changement de topologie est l'avantage principal de cette méthode vis à vis des systèmes masses-ressorts. Par contre, cela se fait au prix d'une plus grande complexité de l'algorithme. Les particules n'ayant pas de relations topologiques privilégiées entre elles, il est difficile d'assurer la conservation des arrêtes vives au cours des déformations, ou de conserver certaines distances. De plus, cela signifie que déformer un élément de dialogue (qui par définition est un élément de dimension un) implique l'utilisation d'un modèle intermédiaire dans lequel la méthode s'appliquera sur une surface ou un volume.

3.4 Smoothed Particle Hydrodynamics

La première étape consiste à discrétiser le volume de l'objet en un ensemble de particules. Chacune de ces particules représente un point d'échantillonnage du volume qui l'entoure [Des97, Mon92]. On peut alors exprimer les forces s'exerçant sur une particule comme suit :

$$F_i^{\nabla P} = -m_i \sum_{voisinsj} m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2}\right) \nabla_i W_h^{i,j} \tag{3.2}$$

La particule i a un rayon d'influence noté h. W_h est un filtre de lissage, $\nabla_i W_h^{i,j}$ représente le gradient de $W_h(X_i - X_j)$ par rapport à la position de la particule i. P_k est la valeur du champ de pression au point k. Ce champ de pression dépend directement de l'équation d'état du matériau. Nous pouvons l'écrire sous la forme :

$$P = k(\rho - \rho_0) \tag{3.3}$$

Ce qui revient à appliquer des forces de pressions quand la densité du matériau diffère de sa densité initiale (ρ_0). Enfin, pour calculer la densité associée à chaque particule on utilise l'équation suivante :

$$\rho_i = \sum_{voisinsj} m_j W_h(X_i - X_j) \tag{3.4}$$

Il nous reste à déterminer la forme du filtre de lissage W_h dont le rôle est d'atténuer les hautes fréquences pouvant perturber l'intégration. Généralement, ce filtre est caractérisé par une approximation spline de la Gaussienne avec un support fini de rayon $2h$.
On pourra aussi ajouter des forces dissipatives au système afin d'améliorer la stabilité du système.

Le modèle SPH offre les mêmes avantages que le modèle à particules standard (principalement, il rend possible le changement de topologie). En plus, il permet d'assurer une gestion de l'équation d'état qui garantit un plus grand réalisme lors des déformations de l'objet [Des97, Deb00].
De même, il est nécessaire, pour appliquer les déformations, d'utiliser un modèle intermédiaire dans lequel on traduit les éléments de dialogue afin de pouvoir travailler sur une surface ou un volume avant d'obtenir l'élément de dialogue final. Malheureusement cette méthode s'adapte mal aux frontières définies (elle a été développée pour des gaz) [Mon92, MPT93]. Or, c'est le cas pour toutes les entités que nous souhaitons déformer ! Le problème vient du fait que la densité chute sur les bords de l'objet ce qui provoque donc un mouvement des particules situées à l'intérieur du solide vers les bords de celui-ci (le répartition devient non-homogène). Pour contrer ce phénomène [Des97] utilise une autre équation pour exprimer la densité dans le matériau (équation de continuité basée sur la conservation de la masse) qui permet alors de disposer d'une densité valide dans l'objet.
La première modification apportée se rapporte au calcul de la densité associée à une particule. L'équation 3.4 calcule cette densité en fonction des voisins de la particule, or près des bords leur nombre chute. Il convient donc d'exprimer différemment celle-ci. Pour cela, il utilise les propriétés de l'équation de continuité (hydrodynamique) en exprimant la variation de la densité au niveau local sous la forme :

$$\dot{\rho}_i = -\rho_i \, div(v)_i \tag{3.5}$$

où $div(v)$ représente la divergence de la vitesse avec :

$$div(v)_i = \frac{1}{\rho_i} \sum_{voisinsj} m_j (v_j - v_i) \cdot \nabla_i W_h^{i,j} \tag{3.6}$$

Il suffit alors de définir la densité ρ_i initiale pour chaque particule et d'intégrer cette relation au cours du temps. On dispose ainsi d'une densité correcte même au niveau des bords de l'objet.
La deuxième modification concerne la définition du filtre de lissage qui devient :

$$W_h(X) = \frac{15}{\pi(4h)^3} \begin{cases} (2 - \frac{\|X\|}{h})^3 & si 0 \leq \|X\| \leq 2h \\ 0 & si \|X\| > 2h \end{cases} \tag{3.7}$$

Ce qui permet d'avoir un gradient non nul quand la distance entre deux particules tend vers zéro (si le gradient était nul, comme c'était le cas précédemment, on pourrait assister à des fusions de particules).

On retrouve dans cette méthode les caractéristiques générales des systèmes à particules. Comme elles, cette méthode est plus adaptée aux déformations appliquant des forces à certains points. Ces caractéristiques nous encouragent à utiliser les SPH pour déformer des entités via un modèle intermédiaire, en permettant les changements de topologie et où nous envisageons un système qui se raffinerait en fonction des forces appliquées (particule de grande dimension dans les zones à faible déformation et de petite dimension là où les déplacements sont plus importants).

3.5 Bilan

Après analyse, la modélisation par éléments finis nous apparaît trop coûteuse (algorithmiquement) malgré le fait qu'il existe des méthodes visant à accélérer le processus. Malheureusement ces méthodes ne sont pas applicables dans notre cas ce qui nous pousse à rejeter cette approche. Les trois modèles suivant qui font tous partie de la famille des interactions entre masses [Des97] nous semblent plus prometteur. Le modèle masses–ressorts offre un bon compromis entre possibilités et simplicité (surtout à travers sa version explicite). Malheureusement, cela s'accompagne de quelques restrictions concernant la rigidité associée aux ressorts ainsi qu'à la construction même du réseau ; c'est de lui dont dépendra la cohérence du comportement de la déformation. Il faut noter cependant que ce modèle est celui qui semble s'adapter le mieux à l'architecture du projet DIJA. En effet, il est relativement facile d'exprimer les éléments de dialogue en un réseau de masses–ressorts afin de leur appliquer directement les déformations, ce qui n'est pas le cas des modèles suivants. Le deuxième modèle de la catégorie des interactions entre masses, le modèle à particule, est le premier modèle à offrir de réelles possibilités de changement dynamique de topologie. Malgré cela, cette méthode souffre d'un manque de rigueur dans les résultats dû à sa loi de comportement arbitraire (dans le sens où elle n'est pas basée sur les propriétés réelles d'un matériau). Pour conclure, nous avons abordé le formalisme des SPH. Ce modèle apporte une plus grande rigueur grâce à une gestion réaliste de l'équation d'état du matériau. Malheureusement, pour l'utiliser, il faut passer par un modèle intermédiaire afin de travailler sur une surface ou un volume avant de pouvoir obtenir un élément de dialogue déformé. Toutefois, sa puissance et ses résultats réalistes font de ce modèle un candidat idéal dans le cadre des déformations reposant sur un modèle intermédiaire. Concernant les déformations applicables directement, nous menons des recherches afin de développer un nouveau modèle de déformation qui répondrait d'une meilleure façon à nos critères.

4 Déformations floues

4.1 Introduction

La plupart des logiciels de CAO actuels utilise la géométrie comme le modèle idéal de représentation d'un objet. Ce faisant, cette représentation cristallise les idées de l'utilisateur car elle représente une forme exacte [LL97]. Malheureusement, cela s'avère un frein très important pour les stylistes qui manipule des formes dont le rôle est de véhiculer des idées plus que des informations purement géométriques [HSZ01, DG97, LLD01]. D'autant plus que ces formes peuvent avoir des caractéristiques imprécises ou qualitatives. Heureusement, il existe des moyens de travailler avec de tels concepts. En effet, la logique floue permet de manipuler des informations qui sont qualitatives, imprécises, incertaines et/ou incomplètes [Zad97]. Avec DIJA, notre première contribution à l'utilisation de mots et de notions floues comme vecteurs d'information se situe au niveau des déformations. Nous souhaitons effectivement proposer à l'utilisateur le moyen d'exprimer avec des termes en langage naturel les déformations qu'il désire réaliser. Nous avons donc intégré des actions telles que "bomber", "étirer", ... Ces termes font références à des connaissances préalablement introduites dans le système par un expert. Au stade actuel de nos travaux, une déformation comme "bomber" s'applique sur un contour en deux dimensions d'un objet de révolution.

4.2 Exemple de déformation

Par exemple, l'utilisateur peut souhaiter "bomber" un élément de dialogue qu'il aura préalablement sélectionné. Premièrement, il choisit parmi un ensemble proposé une loi de comportement. Cette loi va caractériser le comportement de la déformation au cours du temps. Ensuite, il doit définir qualitativement l'action "bomber" grâce à des adverbes tel que : "plus", "moins" ou à des combinaisons d'adverbes : "beaucoup moins", "beaucoup plus". Une fois ces paramètres réglés, les informations sont communiquées à un moteur d'inférence flou dont le rôle est de supprimer les paramètres flous pour en permettre une représentation graphique.

4.3 Observations

Malgré le fait que notre système soit pour le moment extrêmement simplifié, nous voyons apparaître un certain nombre de difficultés se rapportant au comportement des déformations ainsi que d'autres d'ordre géométrique. Par

exemple, nous pouvons bomber une forme à la manière d'un ballon dans lequel on souffle, mais aussi à la manière d'un emboutissage avec un objet circulaire. Si nous appliquons ce style de déformation à un segment, nous le verrons se déformer progressivement jusqu'à prendre la forme d'un demi-cercle. Passé ce stade, le système doit proposer des choix à l'utilisateur afin que celui-ci définisse le comportement de sa déformation.

5 Conclusion et perspectives

Le projet DIJA est un système de CAO dont les objectifs sont de rendre la création d'objets plus intuitive grâce à une approche synthétique. Cette approche est une approche modificatrice, ce qui signifie que l'utilisateur part d'une forme initiale puis la déforme jusqu'à obtenir l'objet souhaité. Cette démarche permet à l'utilisateur de se focaliser sur des questions d'aspect (directement lié au design) plutôt que sur des plans de construction comme c'est le cas avec la méthode constructive. Les déformations sont donc un élément situé au cœur du projet. Pour les réaliser, nous avons défini des éléments de dialogue sur lesquels nous nous basons pour construire la silhouette de l'objet. La déformation d'un élément de dialogue implique une déformation locale ou globale de la forme de l'objet. Il est donc nécessaire de proposer une interaction pour que l'utilisateur puisse manipuler simplement les éléments de dialogue. Le modèle sur lequel s'appuient ces déformations doit être suffisamment puissant, pour permettre un grand nombre d'interactions, et réaliste pour offrir des comportements cohérents lors des déformations. Dans cet optique, nous nous sommes premièrement intéressé à plusieurs modèles couramment utilisés afin d'en dégager les caractéristiques principales. En conclusion, nous avons retenu le modèle SPH pour ses possibilités et sa robustesse (déformations utilisant un modèle intermédiaire). Toutefois, nous menons activement des recherches pour développer un modèle de déformation qui soit capable de manipuler directement nos éléments de dialogue tout en palliant aux différents problèmes rencontrés avec l'utilisation des masses–ressorts. Dans un deuxième temps, nous avons évoqué le rôle des mots dans le domaine de la conception assistée par ordinateur. Traditionnellement, la CAO fait appel à des concepts purement géométriques qui sont caractérisés principalement par la précision, la complétude et l'aspect quantitatif. Or, des études ont montré que de telles notions étaient éloignées des considérations de l'utilisateur lors des phases de design. Effectivement lors des premières phases de la conception la forme d'un objet véhicule avant plus de sens que son aspect purement géométrique. Notre premier pas dans ce sens a été de proposer une interface homme–machine pour contrôler certaines déformations basées sur des notions floues. L'utilisateur souhaitant modifier un objet suivant une de ces déformations choisit une loi de comportement puis sélectionne un ou des termes modificateurs qui donnent la force de la déformation. Les résultats que nous avons obtenus jusqu'à présent sont prometteurs notamment suite à l'exposition d'une maquette à l'occasion du salon international de la CAO de 2002.

Nos recherches actuelles se portent sur la réalisation d'une interface homme–machine intuitive capable de prendre en compte des déformations impliquant des changements de topologie dans l'objet. De plus, nous poursuivons nos études sur les déformations à caractère flou. Nous souhaitons à présent disposer d'un modèle de l'objet imprécis bâti en parallèle du modèle géométrique.

Références

[BW98] D. Baraff and A. Witkin. Large steps in cloth simulation. *Computer Graphics*, 32(Annual Conference Series):43–54, 1998.

[Cot97] S. Cotin. *Modèles Anatomiques Déformables En Temps-Réel*. PhD thesis, INRIA Sophia Antipolis, Novembre 1997.

[DDGP02] F. Danesi, L. Denis, Y. Gardan, and E. Perrin. Basic components of the DIJA project. In *Proceedings of the Seventh ACM Symposium on Solid Modeling and Applications*. ACM Press, 2002.

[Deb00] G. Debunne. *Animation Multirésolution D'objets Déformables En Temps-Réel, Application À la Simulation Chirurgicale*. PhD thesis, Institut National Polytechnique de Grenoble, Décembre 2000.

[Des97] M. Desbrun. *Modélisation et Animation de Matériaux Hautement Déformables En Synthèse d'Images*. PhD thesis, INP Genoble, Décembre 1997.

[DG97] T.H. Dani and R. Gadh. Creation of concept shape designs via a virtual reality interface. *Computer-Aided Design*, 29(8):555–563, August 1997.

[Gay89] Gay. *Une Approche Simple Du Calcul Des Structures Par la Méthode Des Éléments Finis*. Hermes Sciences, 1989.

[GPD+02] Y. Gardan, E. Perrin, F. Danesi, L. Denis, N. Gardan, F. Heschung, E. Malik, M. Reimeringer, and R. Stock. First operational systems based on the DIJA project. In *Applied Modelling and Simulation AMS'2002*, A paraître, 2002.

[HFS+01] G. Hirota, S. Fisher, A. State, C. Lee, and H. Fuchs. An implicit finite element method for elastic solids in contact. In *Proceedings of Computer Animation 2001*. CA. IEEE Computer Society, 2001.

[HSZ01] J. Liu H. Shu and Y. Zhong. A preliminary study on qualitative and imprecise solid modelling for conceptual shape modelling. *Engineering Applications of Artificial Intelligence*, 14(2):255–263, April 2001.

[JV02] J. Jansson and J.S.M. Vergeest. A discrete mechanics model for deformable bodies. *Computer-Aided Design*, 34(12):913–928, October 2002.

[LL97] B. Lawson and S. Ming Loke. Computers, words and pictures. *Design Studies*, 18(2):171–183, April 1997.

[LLD01] S. W. Lim, B. S. Lee, and A. H. B. Duffy. Incremental modelling of ambiguous geometric ideas (i-MAGI): Representation and maintenance of vague geometry. *Artificial Intelligence in Engineering*, 15(2):93–108, April 2001.

[Mon92] J.J. Monaghan. Smoothed particle hydrodynamics. *Annu. Rev. Astron. Astrophys.*, 30:543–574, 1992.

[MPT93] T.J. Martin, F.R. Pearce, and P.A. Thomas. An owner's guide to smoothed particle hydrodynamics. Available on Sissa, 1993.

[PB81] S. Platt and N. Badler. Animating facial expressions. In *Proceedings of SIGGRAPH'81*, pages 245–252, July 1981.

[ST92] R. Szeliski and D. Tonnesen. Surface modeling with oriented particles. In *SIGGRAPH'92*, pages 185–194, 1992.

[TC97] J. Turner and C. Chaillou. Utilisation des éléments finis pour simuler les interactions aiguille-tissus humains. In *Actes Du 5ème Séminaire Du Groupe de Travail Animation et Simulation*, pages 101–107, mars 1997.

[VMT01] P. Volino and N. Magnenat-Thalmann. Comparing efficiency of integration methods for cloth animation. In *Proceedings of CGI'01*, Hong-Kong, July 2001.

[WH94] A. Witkin and P. Heckbert. Using particles to sample and control implicit surfaces. In *SIGGRAPH'94*, volume 28, pages 269–278, 1994.

[Zad97] L.A. Zadeh. Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. *Fuzzy Sets and Systems*, 90(2):111–127, September 1997.

Une Méthode d'Appariement Topologique d'Entités dans les Modèles Géométriques Paramétriques

Dago Agbodan
David Marcheix
Guy Pierra
Christophe Thabaud

Laboratoire d'Informatique Scientifique et Industrielle (LISI)
Ecole Nationale Supérieure de Mécanique et d'aérotechnique (ENSMA)
Téléport 2 — 1 avenue Clément Ader
BP 40109
86961 Futuroscope Chasseneuil cedex — France
(+33/0) 5 49 49 80 63

{agbodan | marcheix | pierra | thabaud}@ensma.fr

Résumé : *De nos jours, de nombreux systèmes commerciaux de CAO proposent la modélisation history-based, constraint-based et feature-based. Malheureusement, la plupart de ces systèmes échouent lors de la phase de réévaluation lorsque divers changements topologiques se produisent. Ce problème est connu sous le nom de nomination persistante. Cela consiste à identifier des entités géométriques dans un modèle paramétrique initial puis à apparier ces entités avec celles du modèle réévalué. Cet article propose une méthode complète fondée sur la topologie sous-jacente pour identifier et apparier n'importe quel type d'entités. La méthode d'identification est basée sur la structure invariable de chaque classe de caractéristiques de forme (feature) et sur l'évolution de la topologie. La méthode d'appariement confronte l'historique topologique initial avec l'historique topologique réévalué. Pour chaque étape de construction, l'appariement se déroule en deux étapes. Lors de l'étape locale, deux mesures de similitude topologique sont calculées entre des couples d'entités appartenant respectivement aux modèles initial et réévalué. Lors de l'étape globale, l'appariement final est défini comme une relation binaire qui maximise la similitude topologique entre les entités des deux modèles. La méthode de nomination et d'appariement a été mise en œuvre avec la plate-forme de développement d'application 3D Open CASCADE.*

Mots-clés : CAO, CAD, conception paramétrique, nomination persistante.

1. INTRODUCTION

Les systèmes de modélisation solide statiques (B-Rep, CSG, etc...), largement utilisés dans le domaine de la conception assistée par ordinateur (CAO), sont de plus en plus remplacés par des systèmes de modélisation dynamique (connus sous le nom de modeleur history-based, constraint-based et feature-based) qui permettent d'exprimer et d'enregistrer le processus de conception et les intentions de conception. Ces systèmes de modélisation dynamique sont souvent rassemblés sous le terme de *modeleurs paramétriques*. Un modèle paramétrique se compose de la représentation d'un objet, d'un ensemble de paramètres (caractérisant l'objet) et d'une liste de contraintes (des équations ou des fonctions) appliquées à l'objet. Par extension, un modeleur paramétrique est un système pour la conception géométrique qui préserve non seulement la géométrie explicite de l'objet conçu (appelé objet paramétrique ou instance courante), mais également l'ensemble des gestes constructifs employés pour le concevoir (appelé processus de conception ou spécification paramétrique). Cette structure de données duale permet la modification rapide par réévaluation. Cependant, quand la réévaluation entraîne des modifications topologiques, il est difficile de retrouver les références des entités utilisées par les gestes constructifs dans le nouveau contexte, donnant des résultats différents de ceux prévus. Un système de *nomination persistante*, robuste aux modifications topologiques, s'avère nécessaire pour préserver, d'une réévaluation à l'autre, les références sur les entités topologiques. Ce problème est connu sous le nom de *nomination persistante* ou *nomination topologique* [8,4]. Cet article est structuré comme suit. Dans la section 2, nous donnons un exposé détaillé des principales difficultés liées à la nomination dans un modeleur paramétrique.

La troisième section présente quelques travaux existants ; essentiellement les deux principaux travaux sur la nomination topologique. Ces travaux ne répondent pas complètement au problème de la nomination persistante. Nous présentons, dans la section 4, une approche alternative.

2. PROBLEMATIQUE

Le problème principal pour la réévaluation paramétrique est de caractériser les entités géométriques et topologiques d'un modèle paramétrique. Caractériser des entités consiste à leur donner un nom lors de la conception et à retrouver à quoi correspond ce nom lors de la réévaluation (c.-à-d. apparier les entités du modèle initial avec les entités du modèle réévalué.) Prenons l'exemple de la figure 1 pour illustrer ce problème. Dans l'exemple ci-dessous, la spécification paramétrique contient quatre gestes constructifs successifs. Le quatrième consiste à arrondir l'arête *e*. Si le modèle initial est sauvegardé après cette quatrième étape, l'instance courante ne contient plus l'arête *e* : elle a été détruite par l'opération d'arrondi. Ainsi, l'opération d'arrondi, qui a pour paramètre d'entrée l'arête *e*, ne peut plus être représentée dans la spécification paramétrique du modèle. Par conséquent, un nom est nécessaire pour représenter les entités référencées dans les spécifications paramétriques lorsqu'elles n'existent plus dans l'instance courante. De plus, chaque geste constructif crée un certain nombre d'entités. Ces entités doivent être distinguées et donc nommées, pour pouvoir être référencées par des gestes constructifs ultérieurs ; même si le nombre d'entités est le même dans toute réévaluation possible (aucun changement topologique). Par conséquent, chaque entité devrait être nommée de manière non-ambiguë et unique lors de la phase de construction. Le problème est bien plus complexe lorsque le nombre d'entités change dans le modèle paramétrique d'une réévaluation à l'autre.

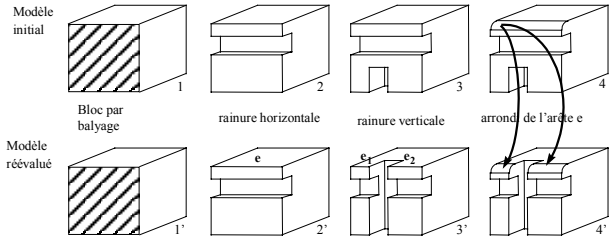


Figure 1: Nomination persistante

Revenons à l'exemple ci-dessus, mais cette fois considérons le modèle réévalué. Nous notons qu'à l'étape 3 l'arête *e* a été coupée en deux arêtes *e1* et *e2*. Ainsi, à l'étape 4, le problème est de déterminer quelle(s) arête(s) doit(doivent) être arrondie(s). Le problème est d'identifier, c.-à-d. apparier, l'arête *e* avec les arêtes *e1* et *e2* en dépit des changements topologiques. Ainsi, quand la réévaluation entraîne des changements topologiques, la difficulté supplémentaire est d'apparier deux structures différentes. Le mécanisme de nomination devrait être assez puissant pour effectuer un appariement robuste lors de la réévaluation.

3. ETAT DE L'ART

Au cours des dernières années, à la suite des premiers travaux de Hoffmann et Juan [6], plusieurs auteurs ont analysé la structure de données interne des modèles paramétriques, proposant des représentations éditables [6,10,14,11,9], discutant les structures mathématiques fondamentales [10,12], décrivant les difficultés liées à la sémantique des opérations [6,5,1] ou à la gestion des contraintes [3]. La plupart de ces travaux ont abordé la modélisation paramétrique en terme de création mais peu en terme de réévaluation. Plusieurs méthodes de nomination persistante et d'appariement ont été proposées. En particulier Kripac [8] et Chen [5] ont proposé des solutions pour résoudre certains des problèmes mentionnés dans la section précédente. Kripac a essentiellement développé un algorithme d'appariement tandis que Chen s'est concentré sur la nomination persistante non ambiguë d'entités.

3.1. L'approche de Chen

Chen [5] propose un modèle qui se compose de deux représentations. Pour la première, il définit une représentation éditable, appelée Erep [6], qui est une représentation de haut niveau, générative, textuelle, indépendante de tout noyau de modeleur et non-évaluée. Elle abstrait les opérations de conception, contient la spécification paramétrique et stocke les entités sous forme de nom. La deuxième représentation, évaluée et dépendant du modeleur, contient la géométrie (l'instance courante). Le lien entre ces deux représentations est obtenu par un schéma de nomination qui met en correspondance les entités du modèle géométrique et les noms génériques (persistants) de la représentation non-évaluée.
Chen définit une structure précise pour la nomination des entités résultant d'une opération d'extrusion ou de révolution. Chaque entité résultant de l'extrusion est nommée par référence à l'entité source correspondante du contour 2D et au geste constructif. Il propose également une technique d'identification pour les entités générées par collision qui est fondée sur la composition des contextes topologiques (les voisinages topologiques plus ou

moins étendus) et sur l'orientation de la feature. Chacune de ces entités est décrite par son origine, soit une entité source soit une intersection de faces sources, son plus petit contexte topologique non ambigu et l'orientation locale dans le modèle B-Rep [4][5]. Pour assurer également l'unicité des noms dans le domaine non linéaire, une information additionnelle, basée sur la géométrie, est ajoutée à l'information topologique précédente : l'orientation de n'importe quelle arête par rapport à la direction d'extrusion de la feature à laquelle elle appartient. L'appariement d'une entité est réalisé par une comparaison locale des voisinages topologiques. Par exemple, dans le cas des faces, la face qui doit être appariée est comparée à l'ensemble des faces issues de la même face invariante (*ensemble préliminaire – preliminary set*). À chaque étape de construction, les faces contingentes héritent du nom de leur face parent ce qui permet de construire l'ensemble préliminaire. Un *indice* est associé à chaque face de cet ensemble préliminaire. L'indice pour chaque face candidate est le nombre d'arêtes frontières mises en correspondance. La face est conservée si cet indice dépasse un seuil.
Dans son étude, Chen s'est limité à trois types de features : balayage (extrusion et révolution), arrondi et filet. Pour ces features, il a montré qu'il était possible d'identifier dans la plupart des cas pratiques (c.-à-d., quand il n'y a pas trop de symétries dans le modèle) sans ambiguïté les entités topologiques des modèles définis par l'attachement successif de telles features, même lorsque les faces sont gauches,. Un algorithme d'appariement, supportant un certain degré de changements topologiques dans le modèle réévalué, est aussi proposé. Cependant, l'utilisation du contexte réduit dans cet algorithme n'est pas détaillée. De plus cet algorithme emploie des seuils et aucune précision n'est donnée sur les valeurs raisonnables possibles. Finalement, l'algorithme d'appariement est local à l'entité à rechercher (cf. 4.2). Dans le cas de la figure 4, et selon le seuil utilisé, *F2* serait probablement appariée avec *Fx*.
Le modèle proposé apporte deux notions importantes dans le domaine : d'une part, deux concepts principaux pour l'identification topologique des entités (contexte topologique et orientation de feature qui seront employés ensuite par plusieurs approches), et d'autre part une étude très précise des cas d'ambiguïté.

3.2. L'approche de Kripac

Kripac [8] s'est concentré sur l'appariement des entités. Il propose un API (Interface de Programmation d'Application) encapsulant son système topologique d'identification et garantissant la nomination persistante des entités en utilisant une table de correspondance entre une entité du modèle initial et une ou plusieurs entités du modèle réévalué. Il propose une structure de graphe intéressante pour l'identification de toutes les entités topologiques basées sur l'historique des faces (créations, scissions, fusions et suppressions des faces) et un algorithme d'appariement complexe. Lors de chaque réévaluation, toutes les faces, comme toute entité référencée dans la spécification paramétrique, sont appariées avec les nouvelles entités. En plus de la structure de graphe des faces, l'approche de Kripac est novatrice car le mécanisme d'appariement proposé est global. La robustesse et la fiabilité induites par le caractère global de la méthode d'appariement entraîne un surcoût dans la complexité spatiale (maintien de deux structures parallèles) et temporelle (plus d'entités à comparer). Le modèle de Kripac ne permet pas d'enregistrer précisément la qualité d'un appariement car il emploie une métrique discrète. Cela induit fortement les appariement ultérieurs et mériterait d'être pris en considération. De plus, aucune explication n'est donnée sur la façon de représenter et d'exploiter cette relation entre graphes pour les opérations suivantes. Son algorithme d'appariement est très sensible à la subdivision du voisinage topologique. Par exemple, comme illustré dans la figure 2, nous appelons γ_{Fi} le voisinage topologique de la face *Fi*, ainsi les voisinages topologiques des faces F_a et F_b lors le la phase de construction sont: γ_{F_a}*={F1,F2,F3,F4,F5,F6,F15,F14}* et γ_{F_b}*={F7,F8,F9,F10,F11,F12,F13,F16}*. Lors de la réévaluation, la scission génère deux nouvelles faces *Fx* et *Fy* dont les voisinages topologiques sont : γ_{F_x}*={F1,F15,F10,F11,F12,F13,F14}* et γ_{F_y}*={F2,F3,F4,F5,F6,F7,F8,F9,F16}*. L'algorithme proposé par Kripac tente d'apparier ces nouvelles faces avec les faces initiales en analysant les voisinages topologiques. L'analyse consiste à trouver le plus long cycle de faces communes (ici *{F2,F3,F4,F5,F6}* et *{F10,F11,F12,F13}*) dans les voisinages topologiques.
Malheureusement, comme on peut remarquer avec cet exemple, les faces F_a et F_b sont respectivement appariées avec les faces F_y et F_x et pas avec les faces F_x et F_y. Une opération ultérieure avec F_a en paramètre d'entrée, aura Fy pour paramètre en réévaluation.

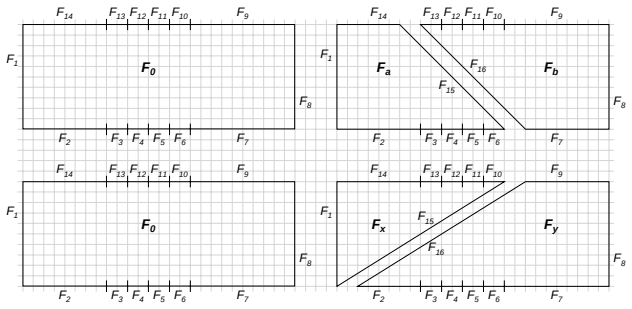


Figure 2 : Vue de dessus d'un bloc avec rainure : construction et réévaluation

Un autre problème important de cette approche est la perte de morceaux pendant la réévaluation. L'algorithme d'appariement consiste en une recherche en arrière-en avant dans le graphe et une analyse croisée. Plus précisément, à partir d'une face donnée, une recherche en arrière est faite dans le graphe réévalué, jusqu'à atteindre une face appariée avec une face de l'ancien graphe. Puis, à partir de cette face appariée, une recherche vers l'avant est faite dans l'ancien et le nouveau graphe, traitant toutes les branches et récupérant les feuilles (des faces). Une analyse croisée est faite sur les faces. L'appariement entre les deux faces est fait approximativement. Par conséquent, il est possible de ne pas prendre en compte toutes les faces qui devraient être analysées. La figure 3 illustre ce problème. L'appariement des faces *F* avec *T* et *G* avec *U* est fait à la quatrième étape de réévaluation. L'analyse croisée est effectuée seulement entre les faces issues de *G* et les faces issues de *U*. En particulier, dans cet exemple, seules les faces *K* et *L* seront croisées avec les faces *X* et *Y*. L'algorithme *oubli* la face *J* qui peut être considérée comme une partie de la face *X*. En conclusion, dans son approche, Kripac préserve une copie des modèles géométriques à chaque étape du procédé de construction. Ceci accélère la réévaluation mais nécessite un espace mémoire qui n'est pas compatible avec la taille des modèles effectivement rencontrés en CAO.

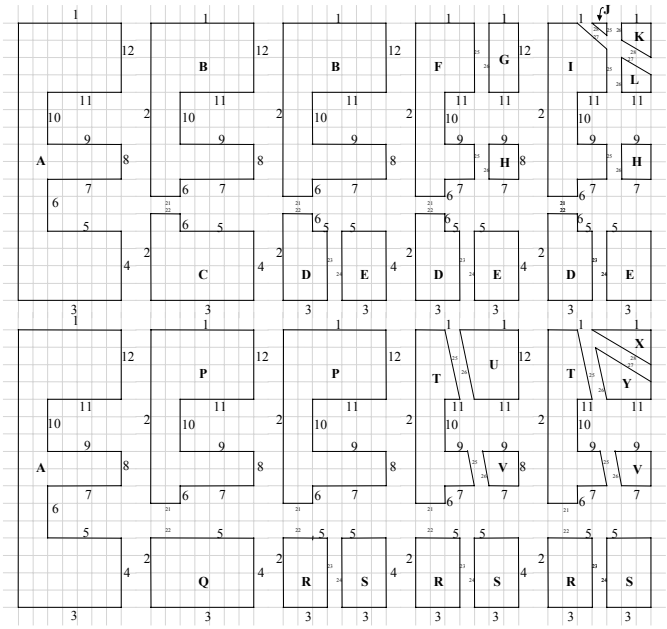


Figure 3: Perte de face lors d'un appariement (J correspond à X ?)

4. PRINCIPE DE NOTRE APPROCHE

Pour définir des noms robustes permettant de résoudre les problèmes précédents, nous proposons de distinguer deux types d'entités géométriques et topologiques [1] :

- ***Entités invariantes***

Une entité invariante est une entité géométrique et topologique qui peut être, complètement et sans ambiguïté, caractérisée par la structure d'un geste constructif et ses paramètres d'entrée, indépendamment des valeurs impliquées. Sur la figure 1, les entités invariantes incluent la face extrémité du bloc balayé, la coque latérale de la rainure horizontale avec ses faces initiale et finale (qui peuvent exister ou pas), la face résultant du geste

d'arrondi, etc. Pour caractériser, c.-à-d. nommer, de telles entités, des modèles de caractérisation doivent être définis pour relier ces entités aux gestes constructifs et à leurs paramètres d'entrée.

- ***Entités contingentes***

A côté de ces entités invariantes, existent des entités qui dépendent du contexte d'un geste constructif. Nous appelons entité contingente, une entité géométrique et topologique qui résulte d'une interaction entre le modèle géométrique courant et les entités invariantes résultant d'un geste constructif particulier. Par exemple, sur la figure 1, le nombre de faces latérales de la rainure horizontale est différent dans le modèle initial (étape 3) et dans le modèle réévalué (étape 3'). Un mécanisme de nomination est également nécessaire pour nommer ces entités contingentes.

La méthode que nous avons développée est basée sur le modèle proposé dans [1]. Ce modèle permet d'identifier, d'une manière unique et non-ambiguë, les entités invariantes, puis les entités contingentes à l'aide des entités invariantes.

4.1.1. Le graphe des faces

Le but est de suivre l'évolution des faces afin de pouvoir, pendant la conception, identifier les faces impliquées, puis, pendant la réévaluation, identifier les faces effectives (dans l'instance courante) correspondant aux faces référencées.

La figure 4 présente un exemple de construction avec le graphe de faces associé. Chaque geste constructif peut être décomposé en deux étapes. La première étape est la spécification approximative de la feature. Elle correspond à la structure invariante (six faces du premier bloc). Cette première structure invariante représente les entrées du graphe des faces. La deuxième étape correspond à l'interaction avec le modèle ce qui introduit des entités contingentes. Ces entités résultent de l'évolution de la structure initiale. L'évolution des faces est décrite par des liens historiques. Sur la figure 4, nous pouvons voir la structure de graphe partielle liée à deux rainures sur un bloc. Dans cet exemple, la face supérieure du bloc est coupée en deux faces par la première rainure, puis en quatre faces par la deuxième rainure. Le graphe des faces représente l'historique (création, rainure, suppression) de la face supérieure. Notez que le graphe initial et le graphe réévalué ne sont pas identiques.

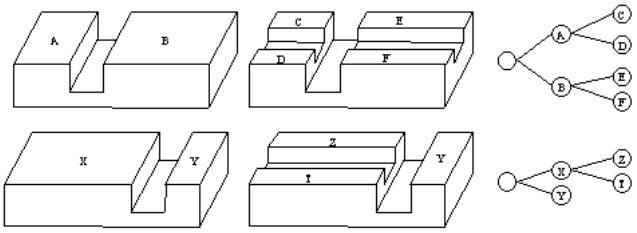
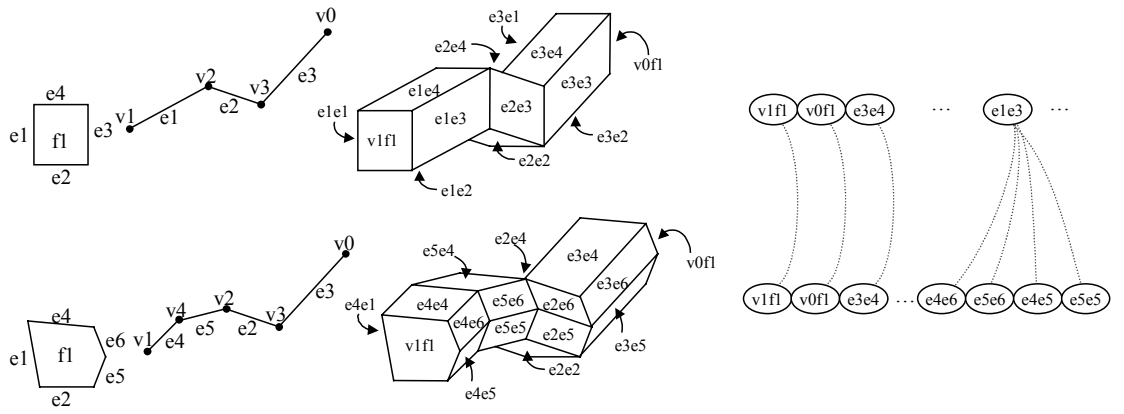


Figure 4 Exemple de graphe de faces. Objet initial et réévalué avec les graphes de faces correspondant (face supérieure seulement)

Chaque face est identifiée par un nom unique qui est défini à la fois par un parcours des entités topologiques unique (les entités invariantes), et par un nombre itératif (entités contingentes) (cf. [2]). Chaque nœud représente une face, qui existe ou a existé dans le modèle. Toutes les faces sans lien historique sortant existent dans la géométrie.

4.1.2. Nomination des entités

L'identification des entités (sommets, arêtes, chemins etc...) est faite en se référant aux faces. Il est ainsi nécessaire de pouvoir nommer ces faces de manière unique et déterministe. D'une façon générale, l'identification d'une entité est fondée sur les éléments inchangés qui la caractérisent d'une manière unique. Dans un modèle paramétrique, ce qui ne change jamais c'est le processus de construction (nous considérons la modification du processus de construction comme édition du modèle et non comme réévaluation du modèle). Par conséquent, la nomination des faces est faite au moyen de l'étape de construction (ordre de création) et au moyen d'un identifiant qui caractérise chaque face de manière unique. Le problème est de définir cet identifiant pour effectuer une caractérisation unique à chaque étape de construction.



Contour	Réévaluation	e1	e2	e5	e6	e4
	Construction	e1	e2	e3	e3	e4

Chemin	Réévaluation	e4	e5	e2	e3
	Construction	e1	e1	e2	e3

Figure 5: Nomination des faces invariantes

Pour chaque étape de construction, nous considérons qu'il y a deux phases. Premièrement, la création de la feature où toutes les faces doivent être nommées. Deuxièmement, la feature est placée dans la géométrie existante. Cette interaction avec la géométrie existante entraîne la modification, la suppression de faces existantes et la création de nouvelles faces (contingentes). Ces faces contingentes doivent également être nommées. Par conséquent, il y a deux types de nom à définir : un pour les faces invariantes et un pour les faces contingentes.

4.1.2.1. Faces invariantes

Selon la taxonomie des features proposées dans [7,2], les faces invariantes du graphe sont créées par quatre types de features (primitive, transition, extrusion et révolution). Pour les deux premiers types, la nomination des faces invariantes est assurée par un parcours topologique unique de l'objet (cf.[2]).
Dans une opération d'extrusion, un *profil générateur* est balayé le long d'un *chemin directeur*. Chaque entité topologique correspond au produit cartésien entre une entité topologique du profil et une entité topologique du chemin. Par exemple sur la figure 5, la face *e1e4* de l'objet extrudé correspond au produit cartésien de l'arête *e1* du chemin directeur et l'arête *e4* du profil générateur. D'une manière analogue, la face interne *v2f1* correspond à *v2* (chemin directeur) et *f1* (profil générateur). La nomination robuste de chaque entité du profil et de chaque entité du chemin est donc fondamentale pour permettre la nomination robuste des faces dans le graphe. Par conséquent, un appariement est fait entre le profil générateur et le chemin directeur du modèle initial et le profil générateur et le chemin directeur du modèle réévalué, pour assurer une nomination persistante. Chaque nom de face est construit comme suit :

<étape, entité du profil générateur, entité du chemin directeur>

Pour simplifier l'écriture, les numéros d'étape ont été omis sur la figure 5. Dans cet exemple, lors de la réévaluation, le sommet entre les arêtes *e1* et *e4* (du profil) a été déplacé. Cette modification, comme toute modification géométrique, n'a aucune influence sur la nomination topologique du profil générateur, ni du chemin directeur, et n'a donc aucune influence sur la nomination des entités invariantes. Toute modification topologique du profil (par exemple dédoubler l'arête *e3*) ou du chemin (par exemple dédoubler l'arête *e1*) doit être tracée pour assurer une nomination robuste. La table d'appariement représentée sur la figure 5 permet d'enregistrer la relation entre le profil et le chemin du modèle initial et le profil et le chemin du modèle réévalué. Ainsi, en dépit des déformations géométriques et des subdivisions topologiques (arêtes *e3* et *e1*) du profil et du chemin, la face *e3e4* est identifiée de façon unique dans la construction comme dans la réévaluation. De façon analogue, les faces *e4e6*, *e5e6*, *e4e5* et *e5e5* seront identifiées comme subdivision de la face *e1e3*. Nous obtenons une relation d'identification (arc en pointillé sur la figure 5) entre les faces invariantes du graphe de faces en construction (appelée AG) et celles du graphe de faces en réévaluation (appelée NG) (cf. section 4.2.1.1 pour plus de détails).

4.1.2.2. Faces contingentes

Le nom des faces contingentes se compose du numéro d'étape et d'un numéro itératif (un numéro arbitraire mais unique pour chaque étape de construction). Pour les faces contingentes, ceci n'est pas suffisant pour permettre un appariement ultérieur. Par conséquent, des informations relatives au voisinage topologique sont associées à chaque face contingente dans le graphe. Ainsi, le nom d'une face contingente est constitué d'un numéro d'étape,

d'un numéro itératif et d'une information supplémentaire pour permettre un appariement ultérieur (voir la section 4.2).

4.2. Méthode d'appariement des entités contingentes

Apparier des entités consiste à associer *n* entités du modèle initial avec *m* entités du modèle réévalué afin de déterminer si chacune des *n* entités correspond à une ou plusieurs entités du modèle réévalué, et réciproquement si chacune des *m* entités correspond à une ou plusieurs entités du modèle initial. L'appariement peut être réalisé en exploitant la géométrie et/ou le voisinage topologique des entités à mettre en relation. L'utilisation de la topologie permet d'obtenir une méthode d'appariement robuste par rapport aux variations géométriques importantes et aux petites variations topologiques. Cependant, dans quelques cas particuliers, lorsque le modèle contient des entités non linéaires, les voisinages topologiques, même étendus [8], sont ambigus et ne permettent pas de caractériser de manière unique une entité du modèle. Ainsi, il serait judicieux d'employer un mécanisme additionnel reposant sur la géométrie (orientation de feature, etc...) permettant de lever les ambiguïtés [5].
La qualité de l'appariement est très relative et dépend généralement des opérations et de la sémantique que le concepteur veut exprimer. Par exemple, la face *J* sur la figure 3 peut être appariée de deux manières différentes selon la sémantique donnée à l'opération :

- On peut considérer que la face *J* est un *morceau* de la face *X* en raison de la similitude topologique et de l'ancêtre invariant commun (face *A*).
- Ou on considère que la face *J* est issue de la division de la face *F* par la quatrième rainure. La face *F* est appariée avec la face *T*, ainsi *J* ne peut qu'être appariée avec une face issue de la division de la face *T*. En conséquence, dans cet exemple, *J* ne serait appariée avec aucune face.

Notre approche adopte la première sémantique qui s'avère être plus générale et permet d'éviter la perte d'appariement d'une face telle que *J*. Comme nous le verrons, cette perte d'appariement est fortement liée avec le type d'appariement qui peut être représenté dans un modèle. Ainsi, le choix d'une représentation peut s'avérer trop restrictif. En effet, la deuxième sémantique est plus restrictive car elle ne tient pas compte du fait qu'apparier une entité avec une autre signifie que les deux entités sont géométriquement et topologiquement semblables mais pas nécessairement identiques.
Notre approche consiste à calculer un coefficient de ressemblance pour les faces du graphe. Les autres entités (arêtes, sommets) sont nommées selon l'appariement des faces (voir la section 4.2.2). Notre méthode d'appariement des faces est composée de deux étapes : le *calcul générique de recouvrement* permet d'évaluer les recouvrements topologiques entre les faces de AG et les faces de NG (voir la section 4.2.1.1), et le *calcul d'appariement spécifique* permet de déterminer un appariement effectif lié à la sémantique des opérations (voir la section 4.2.1.2). Cette décomposition en deux étapes est fondamentale car elle permet de distinguer la partie générique et la partie spécifique d'une méthode d'appariement. Une telle approche offre de nombreux intérêts comme par exemple la possibilité de définir un système qui propose une méthode d'appariement par défaut que l'utilisateur pourra spécialiser si elle ne lui convient pas. Par ailleurs, la méthode de calcul de recouvrement est une méthode globale d'appariement topologique entre deux ensembles de faces, qui peut être employée dans d'autres domaines utilisant la reconnaissance de forme comme l'identification et l'extraction de feature, etc.

4.2.1. Appariement de faces contingentes

4.2.1.1. Calcul générique de recouvrement

Lors de l'étape de réévaluation, nous calculons un recouvrement qui consiste à évaluer la ressemblance topologique entre *p* faces de AG et *q* faces de NG. Ainsi, nous parlons de *croisement* fondé sur les voisinages topologiques des faces. Pour chaque face *F*, on note $\gamma_F=\{o_0, o_1, .. o_n\}$ le circuit d'arêtes orientées $(o_i)_{i=0..n}$ du bord de *F*. Le résultat du croisement est un ensemble de liens inter-graphes entre des faces de AG et des faces de NG.
Soit $\gamma_{F_{ag}}=\{o_0, o_1, .. o_n\}$ et $\gamma_{F_{ng}}=\{o_0', o_1', .. o_m'\}$ les circuits associés aux faces F_{ag} de AG et F_{ng} de NG. Nous définissons $\Gamma_{F_{ag}}$ et $\Gamma_{F_{ng}}$ les ensembles de *sous-chemins partiels* de $\gamma_{F_{ag}}$ et $\gamma_{F_{ng}}$; un sous-chemin partiel d'un circuit est un sous-chemin du circuit où des arêtes orientées ont été supprimées.
On notera qu'une arête orientée ne peut pas apparaître dans deux circuits de faces distincts dans un modèle orienté. Si une arête orientée apparaît dans le circuit de la face *F* et dans les circuit de la face *G* alors cela signifie que *F* et *G* ont des orientations opposées : le modèle n'est pas orienté. Ainsi, pour chaque arête orientée *o*, il existe une et une seule face dont le circuit fait apparaître *o* et nous appelons *face adjacente voisine* de *o*, la face adjacente à l'arête associée à *o* qui ne fait pas apparaître *o* dans son circuit.
Pour quantifier la ressemblance topologique, nous définissons une relation d'équivalence $\sim_{Adj}$ entre deux circuits de faces γ et γ', par : $\gamma \sim_{Adj} \gamma' \Leftrightarrow \exists (o_i)_{i=0..n}$ *et* $(o_i')_{i=0..n}$ / $\gamma=o_0,..o_n$, $\gamma'=o_0',..o_n'$ *et* $\forall i \in \{0..n\}$, *la face invariante ancêtre de la face adjacente voisine de* o_i *est aussi la face invariante ancêtre de la face adjacente voisine de* o_i'.
En d'autres mots, quand on parcourt γ et γ' et que l'on considère seulement les faces invariantes ancêtres des

faces adjacentes voisines, on obtient la même liste circulaire de faces invariantes autour des faces dont les circuits sont γet γ'.

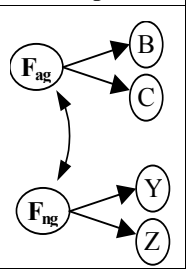
Voisinages topologiques	δ_0	δ_1	**Graphe**
$\gamma_{F_{ag}}$ est égal à $\gamma_{F_{ng}}$	1	1	
$\gamma_{F_{ng}}$ est inclus dans $\gamma_{F_{ag}}$	1	]0,1[	
$\gamma_{F_{ag}}$ est inclus dans $\gamma_{F_{ng}}$	]0,1[	1	
$\gamma_{F_{ag}}$ et $\gamma_{F_{ng}}$ d'intersection non vide	]0,1[	]0,1[	
$\gamma_{F_{ag}}$ et $\gamma_{F_{ng}}$ d'intersection vide	0	0	

Table 1: Relations inter-graphes

Ainsi, on peut définir $\Gamma_{F\cap G}$ l'ensemble des éléments de Γ_F qui sont équivalents à un élément de Γ_G selon la relation précédente. De cette façon, $\Gamma_{F\cap G}$ contient tous les sous-chemins partiels de γ_F tels qu'il existe au moins un élément de γ_G dont la liste circulaire des faces adjacentes voisines, en termes de faces invariantes, est identique. Puis, pour répondre au problème de la subdivision des voisinages topologiques illustré dans la figure 2, nous proposons d'introduire un coefficient permettant de pondérer l'influence des arêtes dans le voisinage topologique selon la longueur de l'arête. Nous introduisons alors trois fonctions :

- π telle que pour toute arête (orientée ou non) e, $\pi(e)$ est la longueur de e,
- Π telle que pour tout circuit $\gamma=\{o_0,o_1,..o_n\}$, $\Pi(\gamma)=\Sigma_{i=0..n}\pi(o_i)$,
- Θ telle que pour tout élément γ de $\Gamma_{F\cap G}$, $\Theta(\gamma)=max\{\ \Sigma_{i=0..n}\ min(\pi(o_i),\ \pi(o_i'))$ *avec* $(o_i)_{i=0..n}$ *et* $(o_i')_{i=0..n}$ / $\gamma=o_0..o_n$ *et* $o_0..o_n\sim_{Adj}o_0'..o_n'\}$.

$\Theta(\gamma)$ peut être interprété comme le poids maximal commun entre γ et un élément équivalent de Γ_G.
Enfin, on définit $\sigma=max\{\Theta(\gamma), \gamma\in\Gamma_{F\cap G}\}$.
σ est la somme de longueurs d'arêtes maximale que l'on peut extraire du bord de F_{ag} et F_{ng} telle que les arêtes apparaissent dans le même ordre dans le bord orienté de F_{ag} et F_{ng}.
Nous calculons deux ratios : $\delta_0=\sigma/\Pi(\gamma_G)$ et $\delta_1=\sigma/\Pi(\gamma_F)$. δ_0 est le ratio d'inclusion de $\gamma_{F_{ag}}$ dans $\gamma_{F_{ng}}$ et δ_1 est le ratio d'inclusion de $\gamma_{F_{ng}}$ dans $\gamma_{F_{ag}}$. Comme le montre la table 1, δ_0 et δ_1 sont compris dans l'intervalle [0,1] selon la ressemblance de leur voisinages topologiques pondérés.
Observons l'exemple de la figure 2. Nous devons croiser deux faces de AG (*Fa*, *Fb*) avec deux faces de NG (*Fx*, *Fy*). La table 2 illustre une étape du calcul de δ_0 et δ_1.
Les calculs précédents permettent d'évaluer de façon individuelle les ratios δ_0 et δ_1 d'inclusion mutuelle des faces F_{ng} et F_{ag} et ainsi la ressemblance topologique entre ces deux faces. Cette approche locale ne prend pas en compte la ressemblance des autres faces à croiser. Une fois que δ_0 et δ_1 sont calculés, nous devons définir une méthode permettant d'évaluer de façon globale les recouvrements entre les faces à croiser. Cette méthode consiste à traiter, de façon itérative, la table de toutes les cellules dans l'ordre de ressemblance décroissante. Pour cela, nous appliquons l'algorithme suivant :

- Trouver une cellule non traitée dont la somme $\delta_0+\delta_1$ est maximale (s'il existe plusieurs cellules ayant la valeur maximale, en prendre une quelconque). Supposons que cette cellule correspond au croisement des faces F_{ng} et F_{ag},
- Décrémenter les poids des arêtes de $\gamma_{F_{ag}}$ et $\gamma_{F_{ng}}$ selon le poids correspondant à chaque arête d'un élément $o\in\Gamma_{F\cap G}$ qui donne σ maximal ; en fait, une fonction de pondération temporaire remplace π et fait apparaître les arêtes 'raccourcies' car une certaine longueur est devenue indisponible pour les calculs de cellules suivants,
- Pour les cellules non traitées, calculer le numérateur σ de δ_0 et δ_1 avec les poids restants,
- Marquer la cellule traitée,
- Réitérer jusqu'à ce que toutes les cellules soient marquées.

Notons que traiter une cellule dont les coefficients δ_0 et δ_1 valent zéro ne modifie aucune cellule de la table. Ainsi, lorsque les deux coefficients d'une cellule valent zéro, on peut considérer la cellule traitée donc marquée ; on notera de plus que les coefficients δ_0 et δ_1 ne peuvent que décroître lors du traitement de la table.

Sous-chemin partiel des circuits des faces F_a et F_y maximisant σ (ici σ=14).

Voisinage topologique pondéré de F_y.

Faces du graphe initial / Faces du graphe réévalué	F_a	F_b
	F_1 F_2 F_3 F_4 F_5 F_6 F_{15} F_{14} 10 8 2 2 2 2 14.1 6	F_7 F_8 F_9 F_{10} F_{11} F_{12} F_{13} F_{16} 8 10 10 2 2 2 2 14.1
F_x F_1 F_{15} F_{10} F_{11} F_{12} F_{13} F_{14} 10 18.9 2 2 2 2 8	F_1 F_{15} F_{14} 10 14.1 6 $\delta_0 = 30.1/44.9, \delta_1 = 30.1/46.1$	F_{10} F_{11} F_{12} F_{13} 2 2 2 2 $\delta_0 = 8/44.9, \delta_1 = 8/50.1$
F_y F_{16} F_2 F_3 F_4 F_5 F_6 F_7 F_8 F_9 18.9 6 2 2 2 2 10 10 8	F_2 F_3 F_4 F_5 F_6 6 2 2 2 2 $\delta_0 = 14/60.9, \delta_1 = 14/46.1$	F_7 F_8 F_9 F_{16} 8 10 8 14.1 $\delta_0 = 40.1/60.9, \delta_1 = 40.1/50.1$

Table 2: Croisement

Observons le résultat de cette méthode sur l'exemple de la figure 3. Nous pouvons voir, à la deuxième étape, que la cellule grisée est sélectionnée car la somme des coefficients est maximale. Les poids des arêtes (via une fonction de pondération temporaire) de γ_X et γ_A sont nuls car chaque arête a été utilisée en totalité. Les coefficients de la ligne et de la colonne sont de nouveau calculés. Le résultat est zéro car *X* et *A* n'ont plus rien en commun. Les coefficients étant nuls, les cellules sont considérées traitées (cellules hachurées). A la troisième étape, une seule cellule reste à traiter. Aucun calcul des coefficients δ_0 et δ_1 n'est alors nécessaire car toutes les cellules de la ligne et de la colonne sont traitées.

A chaque étape de construction, comment déterminer quels ensembles de faces doivent être croiser. Ce problème est fondamental car le croisement d'un ensemble de faces de AG et d'un ensemble de faces de NG est d'une part très coûteux et d'autre part cela peut engendrer des pertes d'appariement, comme le montre la figure 3.

Les croisements permettent de savoir à l'étape *i* de la réévaluation quel ensemble de faces de AG et quel ensemble de faces de NG doivent être croisés. Ces ensembles sont déterminés selon les croisements obtenus à l'étape précédente. A l'étape *i*, les faces à croiser sont les feuilles de AG et de NG apparues à l'étape *i*. Les faces feuilles et leurs pères sont liés par des *liens de recouvrement* dont la valeur excède un seuil choisi. Pour cela, seuls les recouvrements apparus au niveau des feuilles de NG sont nécessaires pour déterminer quelles faces sont à croiser. Le seuil $\varepsilon \in [0,1]$ définit la précision du recouvrement. Au moins un des coefficients δ_0 et δ_1 doit être supérieur à ε pour représenter un lien de recouvrement inter-graphe valide. Un seuil ε=0 signifie que tous les recouvrements sont représentés et qu'ainsi la perte d'appariement est évitée durant la réévaluation. Réciproquement, un seuil proche de ε=1 signifie que seuls les liens correspondant à une ressemblance quasi-totale sont conservés.

Observons l'évolution de la réévaluation au cours de différentes étapes de la figure 3. Nous choisissons dans cet exemple ε=0.15, ce qui permet d'éliminer les croisements qui ne sont pas significatifs. Le choix de ce coefficient dépend de la précision souhaitée pour l'appariement topologique. Initialement, à la première étape de la réévaluation, une identification entre les entités invariantes est effectuée (voir section 4.1.2.1) et est symbolisée par le lien en pointillé entre les faces *A* de AG et de NG (voir figure 6).

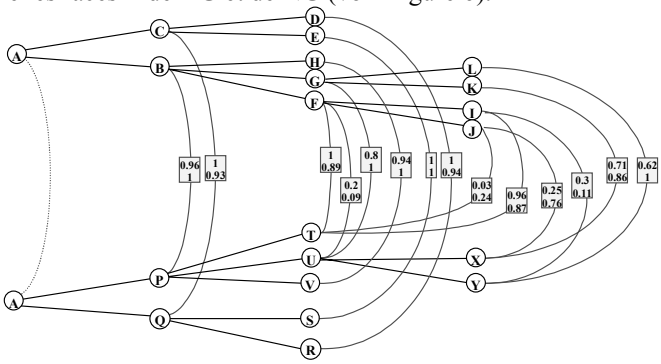


Figure 6: Liens de recouvrement après réévaluation

A la deuxième étape de la réévaluation, la face *A* est scindée en deux nouvelles faces *P* et *Q*. La face *A* de NG, ancêtre de ces faces, est liée par *un lien de recouvrement* (lien d'identification dans ce cas particulier car cette face est invariante) à la face *A* de AG dont les feuilles, apparues à la deuxième étape, sont les faces *B* et *C*. Les faces *P* et *Q* doivent dont être croisées avec les faces *B* et *C*. Le croisement de ces faces donne le résultat suivant. Les coefficients δ_0 et δ_1 correspondant au calcul de recouvrement des faces *P* et *C* sont inférieurs au seuil ε=0.15. Le lien de recouvrement entre *P* et *C* n'est donc pas représenté. Seuls les recouvrements apparus au niveau des feuilles de NG seront nécessaires pour déterminer quels ensembles de faces seront à croiser à l'étape suivante. Le lien entre les faces *A* de AG et NG peut donc être supprimé. Les liens de recouvrement obtenus

après la deuxième étape de réévaluation sont représentés dans la figure 6 par des arcs valués par le couple (δ_0,δ_1) entre les nœuds *B*, *C* et *P*, *Q*.
A la troisième étape de réévaluation, la face *Q* est scindée en deux nouvelles faces *R* et *S*. La face *Q*, ancêtre de ces deux faces, possède un lien de recouvrement avec la face *C* de AG dont les feuilles, apparues à la troisième étape, sont les faces *D* et *E*. Les faces *R* et *S* doivent donc être croisées avec les faces *D* et *E*.
Enfin, la totalité du graphe, obtenu après la quatrième étape de réévaluation, est présenté dans la figure 6.

Step 1	B 6 7 8 9 10 11 12 1 2 21 1 5 2 5 3 5 4 7 10 2	C 2 3 4 5 6 22 5 7 4 5 1 2
P 6 7 8 9 10 11 12 1 2 21 2 5 2 5 3 5 4 7 10.7 2	$\delta_0 = 44/45.7$ $\delta_1 = 44/44$	$\delta_0 = 6/45.7$ $\delta_1 = 6/24$
Q 2 3 4 5 22 4.3 7 4 5 2	$\delta_0 = 4.3/22.3$ $\delta_1 = 4.3/44$	$\delta_0 = 22.3/22.3$ $\delta_1 = 22.3/24$

Step 2	B 6 7 8 9 10 11 12 1 2 21 0 0 0 0 0 0 0 0 0 0	C 2 3 4 5 6 22 5 7 4 5 1 2
P 6 7 8 9 10 11 12 1 2 21 1 0 0 0 0 0 0 0 0.7 0	$\delta_0 = 44/45.7$ $\delta_1 = 44/44$	$\delta_0 = 1.7/45.7$ $\delta_1 = 1.7/24$
Q 2 3 4 5 22 4.3 7 4 5 2	$\delta_0 = 0/22.3$ $\delta_1 = 0/44$	$\delta_0 = 22.3/22.3$ $\delta_1 = 22.3/24$

Step 3	B 6 7 8 9 10 11 12 1 2 21 0 0 0 0 0 0 0 0 0 0	C 2 3 4 5 6 22 0.7 0 0 0 1 0
P 6 7 8 9 10 11 12 1 2 21 1 0 0 0 0 0 0 0 0.7 0	$\delta_0 = 44/45.7$ $\delta_1 = 44/44$	$\delta_0 = 1.7/45.7$ $\delta_1 = 1.7/24$
Q 2 3 4 5 22 0 0 0 0 0	$\delta_0 = 0/22.3$ $\delta_1 = 0/44$	$\delta_0 = 22.3/22.3$ $\delta_1 = 22.3/24$

Step 4	B 6 7 8 9 10 11 12 1 2 21 0 0 0 0 0 0 0 0 0 0	C 2 3 4 5 6 22 0 0 0 0 0 0
P 6 7 8 9 10 11 12 1 2 21 0 0 0 0 0 0 0 0 0 0	$\delta_0 = 44/45.7$ $\delta_1 = 44/44$	$\delta_0 = 1.7/45.7$ $\delta_1 = 1.7/24$
Q 2 3 4 5 22 0 0 0 0 0	$\delta_0 = 0/22.3$ $\delta_1 = 0/44$	$\delta_0 = 22.3/22.3$ $\delta_1 = 22.3/24$

4.2.1.2. Calcul d'appariement spécifique

Le calcul de recouvrement précédent est générique dans la mesure où il évalue et quantifie différents appariements possibles tout en laissant à une méthode plus spécifique le choix d'un appariement particulier répondant aux besoins propres à une application.
Les deux méthodes présentées dans cette section sont des exemples de calcul d'appariement spécifique fondés sur les recouvrements génériques.
A chaque étape de réévaluation, nous calculons, selon les liens de recouvrement, l'appariement des entités apparues à cette étape. Si nous considérons l'ensemble *E* des arcs valués représentant les liens de recouvrement entre l'ancien et le nouveau graphe à cette étape, nous obtenons un graphe bipartite *G={AG, NG, E}*. Un appariement spécifique correspond alors à un graphe bipartite *G'={AG, NG, E'}* où *E'* est un sous-ensemble de *E* et les arcs de *E'* représentent l'appariement spécifique des nœuds.
Une méthode, permettant de déterminer quels arcs de *G* doivent être conservés dans *G'*, consiste à maximiser la somme des coefficients δ_0 et δ_1 présents sur les arcs de *E'*. En effet, plus la somme est grande, plus l'appariement correspond à une identification topologique exacte. Pour cela, nous affectons à chaque nœud *i* de

G' un coefficient $\delta^i = \sum_{\substack{\text{j= tous les liens} \\ \text{du noeuds i}}} (\delta_0^j + \delta_1^j)$. Pour le nœud *i*, ce coefficient correspond à la qualité de l'appariement de son voisinage topologique. Ensuite, le graphe *G'* maximisant la somme $\Phi = \sum_{\substack{i=nodes\ of \\ the\ graph\ G'}} \delta^i$ correspond au meilleur appariement réalisable.

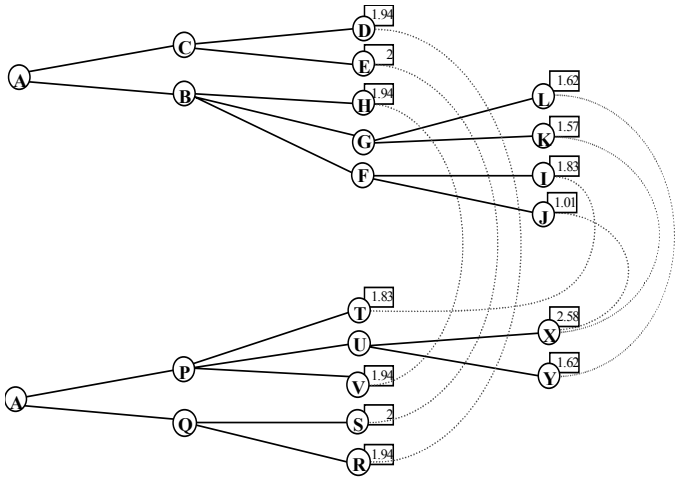


Figure 7: Liens de recouvrement après réévaluation

Un exemple d'appariement spécifique reposant sur la méthode précédente consiste à construire G' tel que tous les chemins de E' sont de longueur inférieure ou égale à un. Cela signifie que l'appariement fera correspondre à une face au plus une face.
Un autre exemple d'appariement spécifique reposant sur la méthode précédente consiste à construire G' tel que tous les chemins de E' sont de longueur inférieure ou égale à deux. D'un point de vue sémantique, cela signifie qu'une face de AG (resp. NG) peut être appariée sur plusieurs faces de NG (resp. AG). Ce choix est mutuellement exclusif. Utilisons cette méthode avec l'exemple de la figure 3. Pour les nœuds apparus dans le graphe à la dernière étape de réévaluation, la maximisation de Φ conduit au graphe *G'* de la figure 7, où les coefficients δ^i de chaque nœud sont représentés et où les liens en pointillé sont des liens d'appariement.
Ces relations d'appariement sont enregistrées dans le graphe bipartite à chaque étape de réévaluation. On remarque que les faces *K* et *J* de AG sont appariées avec la même face *X* de NG.

4.2.2. Autres méthodes d'appariement

L'appariement des faces étant robuste, les autres entités (loops, arêtes, sommets, etc.) peuvent être nommées en termes de faces et d'ensembles de faces. La caractérisation de ces entités peut être effectuée de façon analogue à la méthode proposée par Chen [5]. Par exemple, une arête sera caractérisée par ses deux faces adjacentes et par les listes ordonnées des faces aux extrémités, ainsi qu'une orientation dépendant de la feature permettant de lever certaines ambiguïtés topologiques.

5. CONCLUSION

Nous proposons un mécanisme de nomination persistante associé à une structure hiérarchique permettant d'enregistrer l'évolution historique des invariants facilement identifiables à chaque geste de construction. La méthode d'appariement proposée utilise une pondération des voisinages topologiques pour caractériser précisément chaque entité. Cette méthode se décompose en deux étapes fondamentales : premièrement, le *calcul des recouvrements génériques* permet d'évaluer les recouvrements topologiques entre les faces de l'ancien graphe (AG) et les faces du nouveau graphe (NG), et secondement, le *calcul d'appariement spécifique* permettant de déterminer un appariement adapté à la sémantique d'une opération.
Cette décomposition est fondamentale car elle permet de distinguer la partie générique et la partie spécifique d'une méthode d'appariement. La méthode de calcul générique des recouvrements présente de nombreux avantages. En premier lieu, il s'agit d'une méthode globale d'appariement topologique dans la mesure où elle met en jeu deux ensembles de faces pour déterminer le meilleur appariement pour toutes les faces. De plus, cette méthode permet à chaque étape de déterminer quelles sont les faces qu'il est nécessaire de croiser. Enfin, elle répond au problème de la perte d'appariement qui est étroitement lié à un appariement spécifique.

REFERENCES

[1] Agbodan, D., Marcheix, D., Pierra, G. *A Data Model Architecture For Parametrics* in Journal for Geometry and Graphics, Vol.3, N°.1, pp.17-38, 1999.

[2] Agbodan, D., Marcheix, D., Pierra, G. *Persistent Naming for Parametric Models* in WSCG'2000, Vol., pp.17-38, 2000.

[3] Bouma, W., Fudos, I., Hoffmann, C.M., Cai, J., Paige, R. *Geometric constraint solver* in Computer-Aided Design, vol. 27, n° 6, pp 487-501, June 1995.

[4] Capolylas, V., Chen, X., Hoffman, C.M. *Generic naming in generative, constraint-based design* in Computer-Aided Design Vol. 28 pp. 17-26.

[5] Chen, X. Representation, *Evaluation and Editing of Feature-Based and Constraint-Based design*. Ph.D. thesis, Department of Computer Sciences, Purdue University, West Lafayette, Indiana, 1995.

[6] Hoffmann, C.M., Juan, R. *EREP: an editable high-level representation for geometric design and analysis* in Technical Report CER-92-24, Department of Computer Sciences, Purdue University, West Lafayette, Indiana, 1993.

[7] ISO 10303-224: 1999*, Industrial Automation Systems and Integration - Product Data Representation and Exchange - Part 224: Application protocol: Mechanical product definition for process planing using machining features*, ISO, Geneva, 1994.

[8] Kripac, J. *A mechanism for persistently naming topological entities in history-based parametric solid models (Topological ID System)* in Proceedings of Solid Modeling'95, Salt Lake City, Utah USA, pp.21-30, 1995.

[9] Laakko, T., Mäntylä, M. *Incremental constraint modeling in a feature modeling system* in Computer Graphics forum, Vol.15, N°3, EUROGRAPHICS'96, Poitiers, France, pp.366-376, 1996.

[10] Pierra, G., Potier, J.C., Girard, P. *The EBP system: Example Based Programming for parametric design*, Workshop on Graphic and Modeling In Science and Technology, Coimbra, Springer Verlag. 27-28 June 1994.

[11] Pierra, G., Ait-Ameur, Y., Besnard, F., Girard, P., Potier, J.C. *A general framework for parametric product model within STEP and Part Library* in European Conference Product Data Technology, London, 18-19 April 1996.

[12] Raghothama, S., Shapiro, V. *Boundary Representation Variance in Parametric Solid Modeling* in Report SAL 1997-1, Spatial Automation Laboratory, University of Wisconsin-Madison, 1997.

[13] Shah, J.J., Mäntylä, M. *Parametric and feature-based CAD/CAM : Concepts, Techniques, Applications*, John Wiley and Sons Inc., july 1995

[14] Solano, L., Brunet, P. *Constructive Constraint-based model for parametric CAD systems* in Computer-Aided Design, Vol.26, N°8, pp.614-621, 1994.

Textures de dilatation pour la génération de plis

Jean Combaz **Fabrice Neyret**

iMAGIS, Laboratoire GRAVIR
INRIA Rhone-Alpes
38334 Saint Ismier Cedex
Jean.Combaz@imag.fr, Fabrice.Neyret@imag.fr
http://www-imagis.imag.fr/Membres/Jean.Combaz/

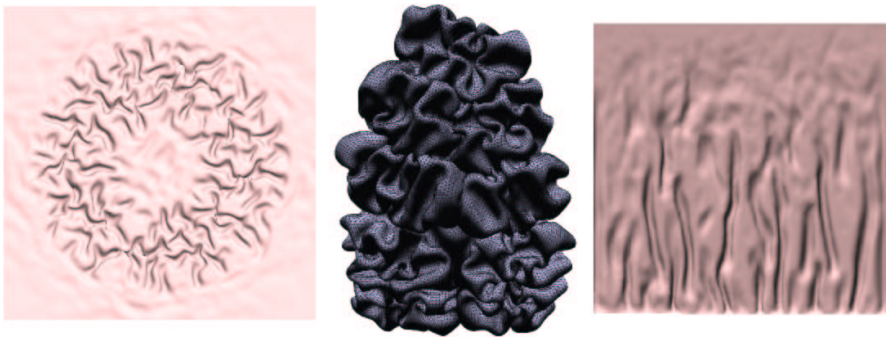


FIG. 1 – *Exemples de quelques plissements générés par notre outil.*

Résumé : *Nous introduisons ici les* **textures de dilatations** *pour ajouter des détails à une surface. Dans cet article nous nous intéressons plus particulièrement aux plis. L'utilisateur peint les attributs de dilatation sur la surface (l'intensité et la direction de dilatation, la longueur d'onde et la régularité des plis) à l'aide d'outils interactifs ou procéduraux. le système génère alors les plis, fronces ou cloques qui en résultent en calculant un nouvel équilibre de la surface. Les résultats montrent que cet outil permet au graphiste de contrôler facilement l'aspect des plis et drapés en ajoutant localement de la surface, ce qui est proche de la manière de penser des sculpteurs.*

Mots-clés : modélisation de surfaces, interface utilisateur, modélisation procédurale , plis, croissance, détails, imperfections.

1 Introduction

Les drapés et les plis peuvent se former dans de nombreuses situations (voir Figure 2), depuis l'action de la gravité ou des frottements sur les tissus jusqu'à la croissance de surfaces élastiques contraintes (par exemple de vieilles couches de peinture, le développement de surfaces biologiques ou géologiques). La séquence des actions qui a pu générer une surface plissée donnée peut être très complexe, voir inconnue (comme pour un lit défait) et l'état initial de la surface très artificiel (comme pour un vêtement). En conséquence, la simulation physique de ces objets, qui suppose la connaissance de l'état initial et des forces agissantes, est souvent difficile à mettre en oeuvre.

Pourtant les artistes traditionnels savent peindre et sculpter ces drapés, sans pour autant avoir à définir état intitial et forces ni à simuler la physique. De plus différentes situations peuvent conduire à des formes similaires. ce qui autorise les artistes à se reposer sur de nombreuses intuitions pour interpréter les formes qu'ils ont dans l'esprit ou devant les yeux. Par exemple les artistes créent de nouvelles formes ou en modifient des existantes en ajoutant ou en supprimant de la matière plutôt qu'en considérant un état initial idéal et appliquant une série de forces pour la modéliser.

Notre but est alors de faciliter le travail du graphiste en lui permettant de 'peindre' des déformations comme les plis ou les drapés. Notre approche consiste à partir d'un état initial, qui est une approximation de la forme que l'on veut créer (comme un cylindre dans le cas de la nappe de la Figure 2, ou un corps pour des vêtements). Cela ressemble à la première ébauche du sculpteur ou du peintre. Puis nous introduisons un nouveau concept: ajouter de la matière pour la modélisation de surface en considérant des opérateurs de dilatation ou contraction, isotropes

ou anisotropes. Le contrôle peut se faire soit de manière interactive, soit procéduralement. Notre système génère alors des plis en trouvant un nouvel équilibre pour la surface, ce qui est facilité par le fait que l'état initial est géométriquement proche de l'état final.

Notre système possède des similitudes avec un simulateur physique; il n'a cependant pas besoin de simuler la dynamique. De plus, nous raffinons la surface et effectuons les calculs seulement là ou c'est nécessaire, c'est à dire là où l'utilisateur ajoute des plis. Dans cet article nous montrons plusieurs applications de ce principe afin d'en illustrer l'utilité.

Notre article, qui est une reprise de notre article paru à Pacific Graphics cette année (cf. [CN02]), se compose de cinq sections. Le paragraphe 2 passe en revue les travaux existant sur la modélisation de détails et la simulation des plis. Puis nous décrirons notre concept de *texture de dilatation* du point de vue de l'utilisateur dans la section 3 ainsi que du point de vue technique dans la section 4. Nous discuterons alors nos résultats dans la section 5 puis nous conclurons.

2 État de l'art

Il existe de nombreuses manières de créer les détails géométriques nécessaires au réalisme des objets de synthèse. Cela inclut deux aspects: comment définir et contrôler ces détails, et comment les représenter.

La définition de ces détails peut se faire par:

- des outils interactifs qui permettent à l'utilisateur de définir explicitement les détails, comme les déformations de forme libre (FFD) [SP86, Coq90] ou de peindre directement ces détails sur la surface [HH90];
- des outils procéduraux qui permettent à l'utilisateur de contrôler les paramètres d'un générateur automatique de détails. Ce générateur peut être alors soit générique [Per85, EMP+94, FF80] soit spécialisé [FLCB95, PHM, Pru93, BB90, WNH97];
- des outils de simulation qui reproduisent des lois physiques. Ils sont particulièrement utilisés pour les vêtements [TF88, BHW94, BW98] ou la création de certaines structures biologiques [FMP92, WK91, Tur91].

Ces détails peuvent alors être représentés par:

- les moyens classiques pour représenter une surface comme les maillages polygonaux;
- les cartes de déplacement (littéralement displacement maps), i.e. textures qui encodent le relief et qui peuvent être converties en géométrie au moment du rendu [WMF+00, GSS99];
- les textures volumiques [KK89, Ney98] et les hypertextures [PH89] qui n'encodent pas les détails d'une manière surfacique;
- les cartes pour le plaquage de normales (littéralement bump maps) [Bli78] qui modifient l'aspect de la géométrie en ne changeant que l'illumination.

Des transitions entre ces différentes représentations sont définies dans [BM93, COM98].

L'idée de simuler une croissance a déjà été introduit dans le cadre d'objets biologiques [WFM01, PHM, Pru93]. La forme de surfaces élastiques comme les vêtements est généralement obtenue au moyen de simulations physiques [TF88, BHW94, BW98], mais certains outils géométriques peuvent être utilisés pour imiter la physique [DKT98]. Comme D'Arcy Thompson [Tho17] le suggère il y a de nombreuses approches possibles pour expliquer une forme donnée.

Les outils interactifs comme Maya Artisan sont pratiques pour l'utilisateur mais ils demandent la modélisation explicite des formes ce qui peut être très rébarbatif pour les détails. Les outils de simulation génèrent des formes réalistes, mais cela peut parfois prendre beaucoup de temps pour obtenir la forme finale. De plus l'utilisateur doit définir un état initial, les paramètres physiques du matériau ainsi que les forces agissantes, ce qui n'est pas toujours connu: un sculpteur reproduisant le modèle qu'il a dans son esprit ou devant les yeux ne peut pas le décrire facilement en temps qu'expérience de mécanique. Les outils procéduraux permettent un haut niveau de contrôle ce qui est très pratique pour l'utilisateur; par contre de tels outils ont surtout été proposés dans le cadre des textures [Per85, EMP+94] et seulement peu d'entre eux sont utilisés pour créer de la géométrie.

Notre but est de créer des formes géométriques ressemblant au résultat d'une simulation physique, tout en laissant à l'utilisateur un haut niveau de contrôle des détails générés, comme s'il utilisait un outil procédural. Nous allons surtout nous concentrer sur la génération de ces détails. Notre implémentation est pour le moment

FIG. 2 – *Quelques drapés et plis rencontrés dans la réalité: des vêtements réels ou sculptés, un chouchou, des plissements de lave, une nappe, de la peau, un vieille couche de peinture, des plis de macadam et une bâche en plastique posée sur le sol.*

limitée à une représentation sous forme de maillages polygonaux, mais on pourrait adapter notre modèle à d'autres représentations comme les cartes de déplacements ou de normales. Dans cet article, nous allons nous intéresser aux drapés et aux plis qui sont très utilisés dans dans l'art classique, mais aussi très difficiles à modéliser avec les outils actuels de la synthèse d'image.

3 Les *Textures de dilatation* du point de vue de l'utilisateur

Le principe est de permettre à l'utilisateur de contrôler les paramètres de haut niveau, soit globalement soit en peignant leurs variations sur la surface. Le solveur (qui est une partie de notre système de modélisation) modifie alors le maillage pour générer un nouvel équilibre de notre surface, en ajoutant des détails -des plis-. Ce résultat est calculé soit interactivement, soit en différé, selon la complexité de la tache. La Figure 3 illustre une session interactive.

Les premières poignées de contrôle correspondent à la dilatation: l'utilisateur contrôle la magnitude et la direction de la dilatation, et fournit aussi d'autres informations comme la longueur d'onde désirée ainsi que la régularité des plis. La dilatation est donnée par un champ de tenseurs, que l'on peut représenter localement par une ellipse (dilatation d'un cercle en ellipse). Dans le cas particulier d'une dilatation unidirectionnelle, on peut représenter ce champ de tenseurs par un champ de vecteurs.

Une autre série de poignées contrôle les degrés de liberté de la surface. Dans notre implémentation nous pouvons fixer la position de certaines zones de la surface (selon un axe, selon un plan ou totalement), rajouter des forces, prendre en compte la collision avec d'autres objets pour contraindre la surface.

Les paramètres qui correspondent aux premiers moyens de contrôle sont assez semblables aux shaders des outils de rendu puissants. Ils peuvent être spécifiés de différentes manières suivant qu'ils soient uniformes ou non sur la surface, et suivant la façon dont l'utilisateur préfère les contrôler. De plus ces paramètres peuvent être scalaires, vectoriels ou tensoriels. Dans notre implémentation nous laissons l'utilisateur les contrôler interactivement, procéduralement ou par l'intermédiaire de cartes (textures). Les cartes qui n'encodent pas des paramètres scalaires peuvent être considérés comme une collection de cartes scalaires (que l'on pourra éditer avec les logiciels classiques de dessin), ou comme des cartes vectorielles. Tout comme avec les shaders, le contexte d'utilisation

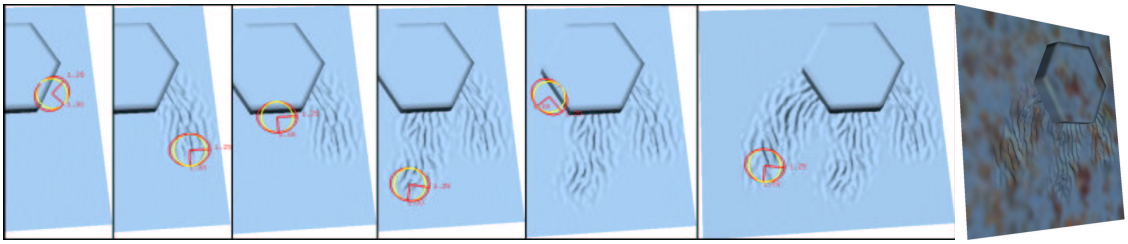


FIG. 3 – *dessin interactif de plis sur une couche de peinture autour d'un écrou (voir Figure 2). L'ellipse foncée représente la magnitude et la direction de la dilatation qu'applique l'outil à la surface. L'orientation de l'outil suit le chemin de la souris.*

est conditionné par la complexité et la qualité désirée. Une simple édition de la surface peut-être faite interactivement alors que les formes assez complexes ont intérêt à être construites grâce à des textures ou une définition procédurale.

4 Les *Textures de dilatation* du point de vue technique

Comme cela a été mentionné dans l'introduction, définir l'équilibre d'une surface connaissant les contraintes qui s'y exercent est un problème standard de mécanique (théorie des coques) pour lequel de nombreuses solutions existent pour sa résolution (par exemple grâce aux éléments finis). Nous décrivons ici notre implémentation d'un modèle simplifié qui utilise des techniques déjà existantes.

Premièrement nous utilisons un maillage triangulaire pour représenter la surface sur laquelle on veut rajouter les détails. Nous définissons un *état de référence* par une longueur l_0 pour chaque arête, et la courbure moyenne κ_0 pour chaque point. On considère les valeurs l_0 comme les longueurs au repos des arêtes, que l'on va modifier selon la croissance. Ceci est donc un état *virtuel* de référence, car on peut seulement l'atteindre localement (rien ne garantit l'existence réelle d'une telle surface). Après avoir appliqué la dilatation ou la contraction définie par l'utilisateur sur les l_0, un solveur itératif se charge de calculer un nouvel équilibre de la surface. Chaque itération déplace les points pour faire décroître les tensions dans la surface. Ces tensions -ou contraintes- sont obtenues en comparant localement la présente configuration de la surface avec l'état de référence. Par la suite nous noterons l la longueur d'une arête et κ la courbure moyenne mesurée sur la surface. Notre surface peut être orientée, autorisant une croissance selon une face privilégiée (par exemple si la surface est sensée être une frontière délimitant un volume). Nous définissons alors une normale orientée $\vec{N}$ en chaque point de la surface.

Voici un aperçu de notre algorithme qui sera décrit en détails dans les sections suivantes :

- application de la dilatation (ou de la contraction) par la modification des longueurs à vide l_0;
- optimisation locale du maillage virtuel (possibilité d'ajouter et de supprimer des arêtes);
- itérer des petits déplacements $\delta\vec{F}$ qui vont diminuer les contraintes.

Dans le cas de dilatations importantes, cet algorithme doit être adapté. Nous en reparlerons dans la section 4.5.

4.1 Dilatation

La dilatation agit sur la longueur au repos des arêtes: soit un tenseur de dilatation donné τ, alors l'arête $\vec{l_0}$ voit sa longueur multipliée par un facteur $\sqrt{\vec{l_0}^t \tau \vec{l_0}}$. En pratique on a un champ de tenseur de dilatation $\tau(u,v)$ et on intègre ce tenseur le long de l'arête pour avoir une valeur moyenne. On peut aussi utiliser une représentation MIP-mapping de la texture de dilatation.

4.2 Optimisation du maillage

Alors que le maillage initial peut être grossier, la génération de petits détails demande un maillage relativement fin. Nous subdivisons alors le maillage suivant la longueur d'onde des plis requise, ainsi que le gradient des paramètres locaux. De plus les triangles trop allongés sont éliminés afin de maintenir une triangulation de bonne

qualité. Ceci est important en pratique pour des raisons de stabilité et de performance dans le solveur. Nous avons implémenté la permutation, la subdivision et l'effacement d'arêtes (voir Figure 4) en utilisant un critère inspiré des maillage de Delaunay appliqué aux surface gauches (cf [WW94]). On permute les arêtes pour rendre maximum le plus petit des angles des triangles. On subdivise les arêtes dont la longueur ou l'angle entre les normales de ses sommets excède un seuil (ces seuils sont définies par l'utilisateur), de plus on insère des points là où les gradients de dilatations sont trop importants afin de garder une bonne précision dans la simulation. L'effacement des arêtes est effectué dans le cas contraire et en utilisant des seuils plus petits afin d'assurer un hystérésis entre ces deux opérations (cela permet de garder une certaine stabilité dans la configuration du maillage).

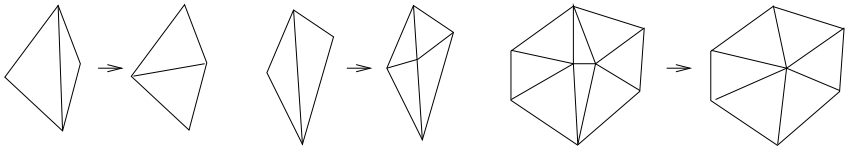


FIG. 4 – *Permutation, subdivision et effacement d'arêtes.*

4.3 Évaluation des contraintes

On calcule tout d'abord la déformation subie en passant de l'état de référence à l'état courant de la surface, l'état de référence étant définie par les longueurs à vide et l'état courant (déformé) par la position des points. On utilise pour cela le tenseur des déformations de Green-Lagrange (4.1) plutôt que celui de Cauchy car il est plus adapté aux grandes déformations mais reste assez simple à évaluer (cf. [DDCB01] pour la discussion des ces choix). Nous évaluons ce tenseur pour chaque triangle. Il est représenté par une matrice 2×2:

$$(\epsilon)_{ij} = \left(\frac{\partial x}{\partial \Omega_i} \cdot \frac{\partial x}{\partial \Omega_j}\right) - \delta_{ij} \tag{4.1}$$

où δ est la fonction de Kronecker[1] et (Ω_1,Ω_2) est un repère local de l'élément. Ce tenseur donne la déformation entre le triangle dans l'état de référence (défini par ses longueurs au repos) et l'état présent.

Le tenseur des contraintes σ peut alors en être déduit en utilisant la loi de Hook (en supposant que l'on a affaire à un matériau élastique linéaire et isotrope), permettant d'obtenir les forces en chaque sommet:

$$\sigma = \lambda tr(\epsilon) I + 2\mu\epsilon \tag{4.2}$$

où μ représente la rigidité de la surface et λ mesure son incompressibilité. Pour plus de détails, voir [OH99] où le problème a été exposé en 3D (éléments finis explicites). Puisque nous avons une surface gauche, nous nous sommes restreints à un problème à deux dimensions, puis l'avons adapté à des surfaces courbes. Les forces qui expriment la déformation tangentielle de la surface sont déduites de σ A chaque noeud i on somme les contributions $\vec{F_{ki}}$ des triangle k qui ont le sommet i en commun, où:

$$\vec{F_{ki}} = -\frac{a_{0_k}}{2} \sum_{j=1}^{3} \vec{x_j} \sum_{\alpha=1}^{2} \sum_{\beta=1}^{2} (L^{ki})_\alpha (L^{kj})_\beta (\sigma)_{\alpha\beta} \tag{4.3}$$

a_{0_k} est l'aire du triangle k, L^{kj} la représentation sous forme de vecteurs de sa fonction de base linéaire associée au sommet j (ce vecteur dépend uniquement des longueurs au repos) et $\vec{x_j}$ la position 3D du sommet j. La somme des $\vec{F_{ki}}$ est alors projetée dans le plan tangent.

Puis nous ajoutons des forces de courbure $\vec{F_\kappa}$ et des forces normales de contrainte $\vec{F_N}$. $\vec{F_\kappa}$ est une force de rappel qui tend à lisser la surface en limitant la différence de courbure avec l'état de référence. $\vec{F_N}$ est une force créatrice de plis qui traduit la compression de la surface en un déplacement suivant la normale. C'est aussi grâce à cette force que l'on contrôle la forme des plis. Afin d'éviter la complexité de la théorie des coques, nous avons utilisé des simplifications inspirées par [DMSB99]. Nous avons choisi :

$$\vec{F_\kappa} = -k_\kappa.(\kappa - \kappa_0)\vec{N} \tag{4.4}$$

$$\vec{F_N} = (k_p \mathrm{f}(\kappa - \kappa_0) + k_{p_i}) C_a \vec{N} \tag{4.5}$$

1. δ_{ij} est égale à 1 si i=j et à 0 sinon.

κ et $\vec{N}$ sont calculés en utilisant la même interpolation des points voisins que [DMSB99]. k_κ and k_p sont deux constantes. k_{p_i} est un biais qui permet de pousser la surface dans une direction privilégiée ($\vec{N}$ or -$\vec{N}$). Si aucune direction n'est préférée, k_{p_i} peut être égale à zéro, mais il est préférable de la fixer à de faibles valeurs aléatoires pour chaque points: cela évite des problèmes dans le cas de surfaces rigoureusement plates. C_a est le taux de compression surfacique de la cellule entourant le point considéré. On a:

$$C_a = \frac{\sum_{cellule} a_{0_i} - a_i}{\sum_{cellule} a_{0_i}} \tag{4.6}$$

où a_0 et a sont les aires des triangles dans l'état au repos et courant (déduites des longueurs des arêtes l_{0_i} et l_i). $f(\kappa)$ est une fonction qui contrôle la forme des plis; nous expliquons comment la choisir dans la section suivante.

Notons que toutes ces forces sont relativement indépendantes de la discrétisation: elles sont évaluées plus précisément dans le cas de triangles plus petits et presque équilatéraux, mais il n'y a pas de biais. Si une surface en équilibre est subdivisée, sa forme va être conservée (si l'on excepte quelques petites erreurs numériques).

4.4 Contrôle de la forme des plis

On introduit $\kappa^*(0) = \kappa - \kappa_0$ et $\frac{\partial \kappa^*}{\partial t} = L(\kappa^*)$ où L est le noyau d'un filtre de diffusion anisotrope non uniforme (cf [DF95] pour plus d'information sur les filtres différentiels). On choisit alors $f(\kappa - \kappa_0) = \sigma(\kappa^*)$ où $\sigma()$ est une sigmoïde[2]. Pour des κ^* assez grands nous voulons juste savoir dans quelle direction la surface doit être poussée. Maintenant illustrons le choix du filtre en 1D en prenant

$$L(u) = \frac{\partial u^2}{\partial x^2} g(u + \nu^2 \frac{\partial u^2}{\partial x^2}) \tag{4.7}$$

où $g(x) = \frac{x^2}{1+\alpha x^2}$. La fonction g permet de lisser plus ou moins un signal u suivant sa fréquence (c'est un filtre passe bande: le signal n'est pas lissé s'il a pour pulsation ν). A l'équilibre κ^* reste stationnaire, donc $L(\kappa^*) = 0$. Cela arrive si $\kappa^* + \nu^2 \frac{\partial^2 \kappa^*}{\partial x^2} = 0$, i.e. pour un signal harmonique de pulsation ν. Ainsi si nous voulons des plis d'une longueur d'onde λ, on doit choisir $\nu = \frac{\lambda}{2\pi}$. Remarquons que si κ^* est nul, il est aussi stationnaire, d'où la nécessité de prendre k_{p_i} non nul pour initier un premier mouvement et créer de la courbure (et rendre ainsi κ^* différent de zéro). En fait nous utilisons un opérateur anisotrope, c'est à dire que nous appliquons le même filtre (4.7) mais dans deux directions et avec deux valeurs de ν différentes. Dans une première direction (celle du plis) $\nu_1(\Lambda)$ va contrôler sa régularité, et dans la direction orthogonale $\nu_2(\lambda)$ contrôlera la longueur d'onde des plis (cf Figure 5).

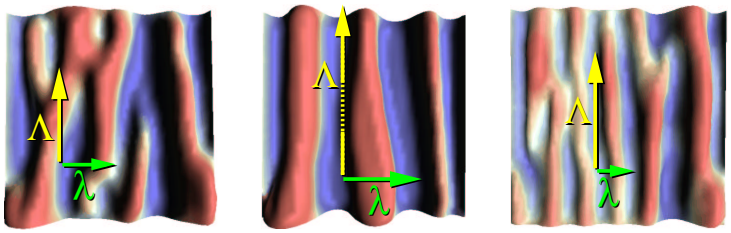


FIG. 5 – *Réglage de la longueur d'onde λ et de la régularité Λ.*

4.5 Modification de l'algorithme dans le cas de grandes dilatations

Si le taux de dilatation est raisonnable, augmenter la longueur au repos des arêtes en un seul pas est suffisant. Sinon il vaut mieux itérer, c'est à dire appliquer successivement des dilatations partielles. Et dans le cas de très grandes dilatations (pour la morphogénèse, ce qui dépasse le sujet de cet article) il est nécessaire d'ajouter de la plasticité afin d'éviter que des contraintes qui n'arrivent pas à se dissiper ne s'accumulent. La plasticité revient à atténuer la mémoire que l'on avait de l'état initial. Cela est réalisé en relaxant la longueur au repos: à chaque fois que l'on obtient un équilibre, l_0 est remplacée par $(1 - \varepsilon)l_0 + \varepsilon l$. Cela peut aussi être utile d'ajouter de la plasticité dans le cadre de la modélisation de détails, surtout si la texture de dilatation est trop complexe pour que les contraintes ne se dissipent, sinon les plis peuvent sembler trop distordus.

2. Une sigmoïde est une fonction monotone croissante variant entre -1 et 1 avec une transition rapide en 0.

4.6 Optimisations

Dans notre implémentation nous utilisons plusieurs classes d'optimisations:

- Comme nous l'avons mentionné dans la section 4.2, nous adaptons le maillage au besoin de la simulation.
- Notre solveur utilise différents pas de déplacements pour chaque points, ceci afin de consacrer du temps de calcul seulement là où c'est nécessaire. Cette adaptation a été faite dans l'esprit de [DDCB01]: on calcule le pas maximum tolérable pour chaque point. Soit δ le plus petit d'entre eux. Si un point nécessite un pas $\delta_i \in [2^n\delta, 2^{n+1}\delta[$, nous ne recalculons ses forces que pour les itérations dont le numéro est un multiple de 2^n.
- Dans le cas d'une utilisation interactive, nous définissons un morceau de surface actif en dehors duquel aucun calcul n'est effectué: l'outil interactif dilate ou contracte une surface circulaire limitée; Nous considérons alors une portion de surface circulaire légèrement plus grande dans laquelle l'aspect de la surface pourrait être affecté. On active alors la simulation que dans cette zone. Un décompteur est attaché à chaque point de la zone et l'on désactive les points actifs dont le compteur a atteint zéro.

5 Résultats

Dans la Figure 6, nous montrons une dilatation unidirectionnelle uniforme d'une surface carrée dont les bords gauche et droite sont attachés. Nous avons fait l'expérience avec une petite longueur d'onde et une petite régularité, et avec une plus grande longueur d'onde et régularité. En comparaison, on peut voir sur la gauche une bâche en plastique réelle. Sur la droite, des fronces ont été simplement obtenues en peignant d'étroites bandes dans la texture de dilatation. Dans l'image de droite de la Figure 1 nous avons utilisé un taux croissant de dilatation du haut vers le bas et introduit une force de gravité ainsi qu'une détection de collision avec le sol (un plan horizontal), tous les bords étant attachés: cela produit des plis positifs, séparés de zones relativement plates.

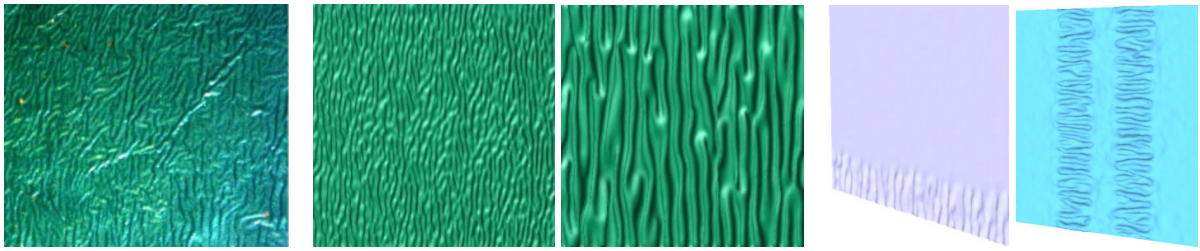


FIG. 6 – *Plis réguliers (à gauche une image d'une bâche en plastique réelle); Fronces.*

Dans la Figure 7, une dilatation circulaire unidirectionnelle est appliquée à une couronne de surface (la texture de dilatation est montrée à gauche), avec plusieurs longueurs d'ondes et différentes régularités: sur la gauche la longueur d'onde et la régularité sont pratiquement nulles, ce qui donne un motif presque aléatoire. Sur la droite elles sont plus importantes et le motif montre alors des plis beaucoup plus organisés.

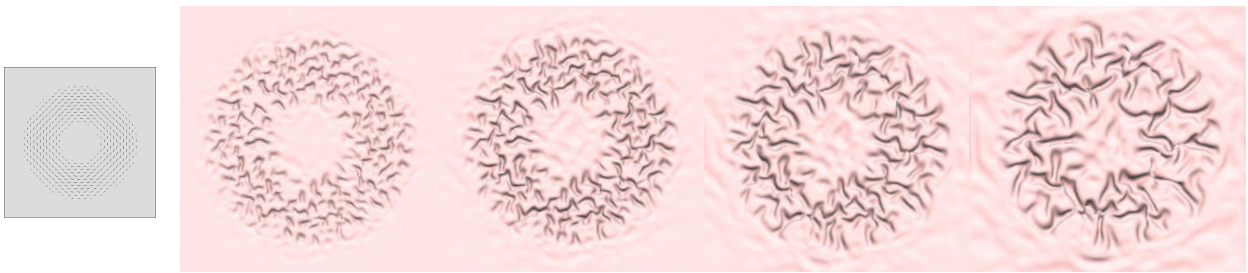


FIG. 7 – *Plis sur une couronne de surface*

La Figure 8 montre un chouchou calculé avec différents taux de dilatation. Il est obtenu en dilatant un tore comme indiqué dans la Figure de gauche. Comme dans la réalité, la surface doit être contrainte pour éviter d'obtenir un simple tore plus grand: dans la réalité un anneau en caoutchouc est insèré à l'intérieur; Nous réalisons l'équivalent en rajoutant un tore rigide à l'intérieur de notre surface torique mobile (i.e. une détection de collision empêche le grand tore de trop s'agrandir au lieu de plisser). D'autres solutions peuvent être utilisées dans

le même but: dans l'image du milieu de la Figure 1 nous avons défini plusieurs bandes rigides sur la surface (ce qui est plus ou moins équivalent avec ce qui se fait avec les vrais vêtements). On peut remarquer que malgré le fait qu'il n'y ait pas de tests d'auto-collision, le lissage de la courbure suffit à éviter les collisions locales (une déformation purement géométrique comme avec les textures de déplacement n'aurait pas pu le faire). Mais si l'on ne fait aucun test, des plis distants peuvent néanmoins s'intersecter. Cependant cela n'apparaît que quand le taux de dilatation dépasse un facteur de 3 (cf Figure 9). Une dilatation plus importante du chouchou dans la Figure 8 aurait probablement montré des auto-collisions.

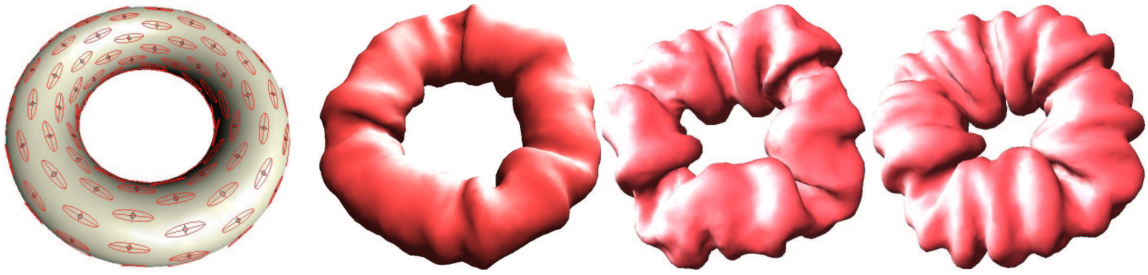


FIG. 8 – *Un chouchou obtenu à partir d'un tore*

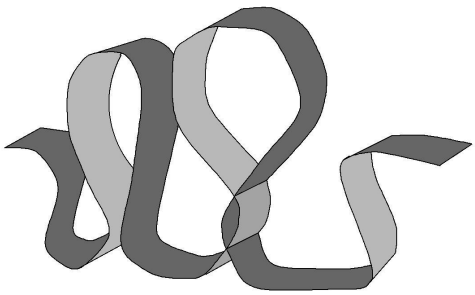


FIG. 9 – *Pour des dilatations importantes (typiquement supérieur à 3) des plis voisins peuvent s'intersecter.*

Les temps de calcul avec notre implémentation vont de quelques secondes dans le cas d'une édition interactive (cf Figure 3) à plus d'une demi-heure sur un Pentium III à 700 MHz pour l'image du milieu de la Figure 1 (le maillage compte à peu près 50000 points). De meilleurs performances devraient être obtenues avec un solveur plus efficace, par exemple basé sur des méthodes implicites (voir [BW98, DMSB99]).

Bien que les objectifs de cet article soient limités à la modélisation de détails sur une surface, nous avons expérimenté des textures de dilatation plus fantaisistes, qui affectent la forme de la surface dans sa globalité. Dans nos travaux futurs, cela nous permettra d'aller vers des méthodes de morphogénèse pour modéliser des formes complexes. Nous présentons ici quelques résultats préliminaires: la Figure 10 montre une forme ressemblant à un cerveau obtenue à partir d'une dilatation isotrope sur une sphère. Une texture (montrée à droite) contenant des taches de dilatation isotropes est appliquée à un carré, et donne l'ensemble des cloques au milieu de la Figure 10. Un autre exemple de résultat montre dans la Figure 11 un carré se courber après avoir été soumis à une dilatation isotrope limitée à certaines zones (La texture correspondante est visible au milieu).

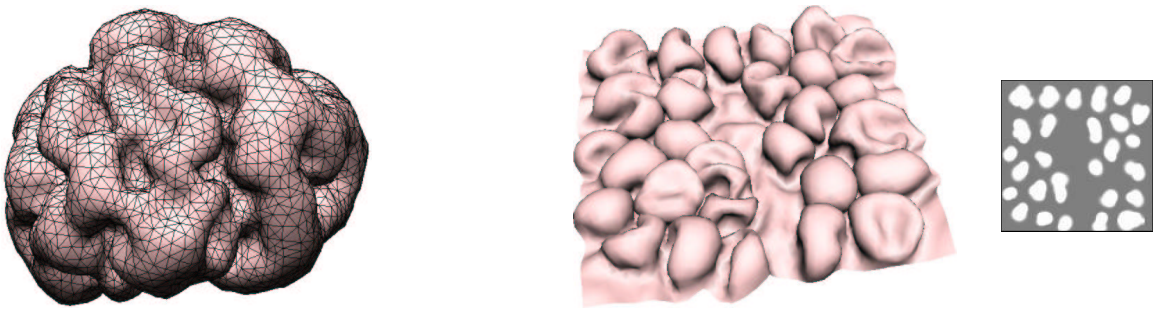


FIG. 10 – *Circonvolutions et cloques*

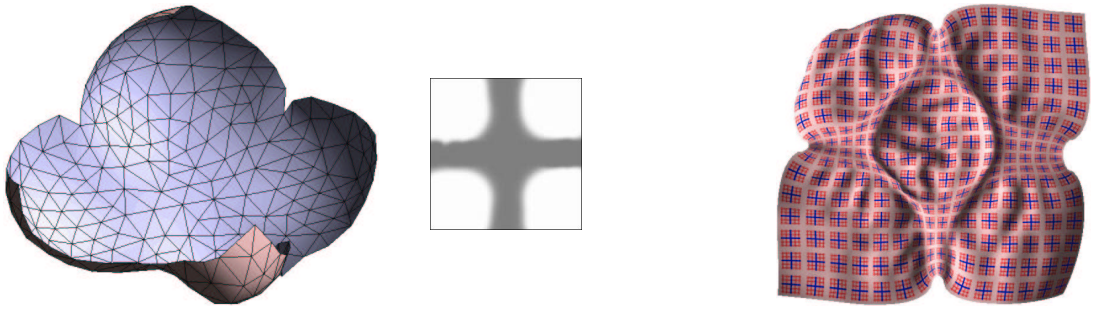


FIG. 11 – *Croissance non uniforme d'un carré*

6 Conclusion

Nous avons introduit le concept des *textures de dilatation*, lesquelles permettent à l'utilisateur de spécifier l'aspect des plis sur une surface avec un contrôle à haut niveau (i.e. sans avoir à définir explicitement les plis comme cela serait fait avec des cartes de déplacement ou une modélisation directe). L'utilisateur spécifie les zones de plis, un taux et une direction de dilatation et les possibles contraintes qui peuvent s'exercer sur la surface. L'utilisateur peut utiliser un outil interactif de dessin des plis, ou passer par une carte des dilatations (qu'il définit soit explicitement, soit procéduralement). Bien que le calcul de l'équilibre possède beaucoup de similitudes avec les solveurs de simulation physique, notre approche est plus compatible avec les connaissances et les souhaits que peut avoir un artiste: l'utilisateur n'a besoin de connaître que le genre de forme qu'il veut obtenir, et non pas l'historique des forces qu'il faudrait appliquer en réalité pour obtenir le résultat (beaucoup de sculptures baroques ont d'ailleurs des plis très exagérés qui ne sont probablement pas physiques, et même des surfaces non développables).

En ce qui concerne les travaux futurs, nous allons inclure les auto-collisions dans notre implémentation pour empêcher les plis de s'intersecter. Nous voudrions aussi étudier la génération direct de textures pour le placage de normales (afin d'éviter d'obtenir une géométrie trop lourde, si l'objet doit être observé de loin). Notre but à plus long terme est d'expérimenter la morphogénèse, c'est à dire le modélisation de formes qui sont principalement le résultat de phénomènes de croissance. Comme cela implique des taux de dilatation d'un autre ordre, nous sommes particulièrement intéressé par la définition procédurale de textures de dilatation, qui auraient par exemple des propriétés fractales.

Références

[BB90] Norman I. Badler and Welton Becket. Imperfection for realistic image synthesis. *Journal of Visualization and Computer Animation*, 1(1):26–32, August 1990.

[BHW94] David E. Breen, Donald H. House, and Michael J. Wozny. Predicting the drape of woven cloth using interacting particles. In *Proceedings of SIGGRAPH 94*, pages 365–372. ACM SIGGRAPH / ACM Press, July 1994.

[Bli78] James F. Blinn. Simulation of wrinkled surfaces. In *Computer Graphics (SIGGRAPH '78 Proceedings)*, volume 12(3), pages 286–292, August 1978.

[BM93] Barry G. Becker and Nelson L. Max. Smooth transitions between bump rendering algorithms. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 183–190, August 1993.

[BW98] David Baraff and Andrew P. Witkin. Large steps in cloth simulation. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 43–54. ACM SIGGRAPH / Addison Wesley, July 1998.

[CN02] Jean Combaz and Fabrice Neyret. Painting folds using expansion textures. In *Pacific Graphics 2002 Proceedings*, pages 176–183, October 2002.

[COM98] Jonathan Cohen, Marc Olano, and Dinesh Manocha. Appearance-preserving simplification. *Proceedings of SIGGRAPH 98*, pages 115–122, July 1998.

[Coq90] Sabine Coquillart. Extended free-form deformation: A sculpturing tool for 3D geometric modeling. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 187–196, August 1990.

[DDCB01] Gilles Debunne, Mathieu Desbrun, Marie-Paule Cani, and Alan H. Barr. Dynamic real-time deformations using space & time adaptive sampling. In *Proceedings of ACM SIGGRAPH 2001*, pages 31–36. ACM Press / ACM SIGGRAPH, August 2001.

[DF95] Rachid Deriche and Olivier Faugeras. Les EDP en traitement des images et vision par ordinateur. Technical Report RR-2697, Inria, Institut National de Recherche en Informatique et en Automatique, 1995.

[DKT98] Tony D. DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 85–94. ACM SIGGRAPH, July 1998.

[DMSB99] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of SIGGRAPH 99*, pages 317–324. ACM SIGGRAPH, August 1999.

[EMP+94] David Ebert, Kent Musgrave, Darwyn Peachey, Ken Perlin, and Worley. *Texturing and Modeling: A Procedural Approach*. Academic Press, October 1994. ISBN 0-12-228760-6.

[FF80] Alain Fournier and Don Fussell. Stochastic modeling in computer graphics. *Computer Graphics (Proceedings of SIGGRAPH 80)*, 14(3):108, July 1980. Held in Seattle, Washington.

[FLCB95] Kurt W. Fleischer, David H. Laidlaw, Bena L. Currin, and Alan H. Barr. Cellular texture generation. *Computer Graphics*, 29(Annual Conference Series):239–248, 1995.

[FMP92] Deborah R. Fowler, Hans Meinhardt, and Przemyslaw Prusinkiewicz. Modeling seashells. In Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 379–388, July 1992.

[GSS99] Igor Guskov, Wim Sweldens, and Peter Schröder. Multiresolution signal processing for meshes. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 325–334, Los Angeles, California, August 1999. ACM SIGGRAPH / Addison Wesley Longman.

[HH90] Pat Hanrahan and Paul E. Haeberli. Direct WYSIWYG painting and texturing on 3D shapes. In Forest Baskett, editor, *Computer Graphics (SIGGRAPH 90 Proceedings)*, volume 24, pages 215–223, August 1990.

[KK89] James T. Kajiya and Timothy L. Kay. Rendering fur with three dimensional textures. In Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23(3), pages 271–280, July 1989.

[Ney98] Fabrice Neyret. Modeling animating and rendering complex scenes using volumetric textures. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):55–70, January–March 1998. ISSN 1077-2626.

[OH99] J.F. O'Brien and J.K. Hodgins. Graphical modeling and animation of brittle fracture. In *SIGGRAPH'99 Conference Proceedings*, pages 137–146. ACM SIGGRAPH, 1999.

[Per85] Ken Perlin. An image synthesizer. In B. A. Barsky, editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19(3), pages 287–296, July 1985.

[PH89] Ken Perlin and Eric M. Hoffert. Hypertexture. In Jeffrey Lane, editor, *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23(3), pages 253–262, July 1989.

[PHM] Przemyslaw Prusinkiewicz, Mark Hammel, and Radomír Mech. Visual models of morphogenesis: A guided tour. http://www.cpsc.ucalgary.ca/Redirect/bmv/vmm-deluxe/.

[Pru93] Przemyslaw Prusinkiewicz. Modelling and visualization of biological structures. *Graphics Interface '93*, pages 128–137, May 1993. Held in Toronto, Ontario, Canada.

[SP86] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In David C. Evans and Russell J. Athay, editors, *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 151–160, August 1986.

[TF88] D. Terzopoulos and K. Fleisher. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *SIGGRAPH'88 Conference Proceedings*, pages 269–278, 1988.

[Tho17] D'Arcy Wentworth Thompson. *On Growth and Form*. Cambridge University Press, Cambridge, 1917.

[Tur91] Greg Turk. Generating textures for arbitrary surfaces using reaction-diffusion. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 289–298, July 1991.

[WFM01] Marcelo Walter, Alain Fournier, and Daniel Menevaux. Integrating shape and pattern in mammalian models. In *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 317–326. ACM Press / ACM SIGGRAPH, August 2001.

[WK91] Andrew Witkin and Michael Kass. Reaction-diffusion textures. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 299–308, July 1991.

[WMF+00] Xiaochuan Corina Wang, Jérome Maillot, Eugene L. Fiume, Victor Ng-Thow-Hing, Andrew Woo, and Sanjay Bakshi. Feature-based displacement mapping. *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, pages 257–268, June 2000.

[WNH97] Tien-Tsin Wong, Wai-Yin Ng, and Pheng-Ann Heng. A geometry dependent texture generation framework for simulating surface imperfections. In *Eurographics Rendering Workshop 1997*, pages 139–150, St. Etienne, France, June 1997. Eurographics / Springer Wien. ISBN 3-211-83001-4.

[WW94] William Welch and Andrew Witkin. Free-form shape design using triangulated surfaces. In *Proceedings of SIGGRAPH 94*, Computer Graphics Proceedings, Annual Conference Series, pages 247–256. ACM SIGGRAPH / ACM Press, July 1994.

Peinture Virtuelle : modélisation et interaction

Anaïs Atencia, Jean-Jacques Bourdin

Laboratoire d'Intelligence Artificielle
Université Paris 8
2, rue de la Liberté
93526 Saint-Denis Cedex, FRANCE
anais@ai.univ-paris8.fr, jj@ai.univ-paris8.fr

Résumé : *Nous proposons un modèle de peinture virtuelle intégrant non seulement la simulation physique mais aussi le geste du peintre et son positionnement. Ce modèle inclut les déplacements (aller-retour) indispensables à la peinture impressionniste. Nous verrons que les effets de zooms liés à des capteurs améliorent l'interactivité du modèle.*

Mots-clés : rendu non photoréaliste, peinture virtuelle, haute résolution, palette graphique.

1 Introduction

Un des axes de recherche en rendu non photoréaliste (NPR) [GG01] est de simuler les styles de peinture en s'inspirant des effets des peintures impressionnistes [Hae90, Her98, Mei96, Lit97]. De nombreux travaux de peinture virtuelle ne modélisent qu'un effet comme par exemple la peinture à l'eau [CAS+97]. Nous présentons un modèle de peinture virtuelle plus général qui permet d'intégrer les différents arts graphiques : huile, aquarelle, encre, graphite... Nous utiliserons néanmoins le terme générique de peinture car il correspond à notre exemple récurrent et au mode le plus général. L'application intègre aussi la simulation de l'action de peindre. Peindre, c'est une succession de gestes fins et mesurés, de grands coups de brosses [Her98], de déplacements et de changements de perspectives et d'éclairage. Nous recréons le mouvement de recul que peut avoir un peintre pour visualiser son œuvre. Ce mouvement est modélisé à l'aide d'une succession de zooms arrière ou avant. Ces zooms sont commandés via une interface basée sur la position et les mouvements de l'utilisateur. Notre but est que l'utilisateur puisse visualiser l'image et travailler de la même manière que dans son atelier. En utilisant une tablette *wacom* avec visualiation (de type cintiq [Wac]), de même que sur la toile, il peint directement sur l'image. Les différentes étapes de ce travail sont donc : une simulation physique de la peinture, un système de changement de point de vue et une interface. Nous présenterons dans un premier temps le modèle physique, puis l'interface homme machine et enfin nous détaillerons les techniques utilisées pour la détection de la position de l'utilisateur et pour les zooms.

2 Modèle physique

Dans cette partie, nous détaillons le modèle 3D de peinture crée par Sobczyk et al. [SBB02b] que nous utilisons. Ce modèle de peinture virtuelle utilise des images en très haute résolution [SBB02a, Cai00]. Il s'inspire des méthodes de Curtis [CAS+97] et Buchanan [SB99a, SB99b, SB99c, SB00]. Fredo Durand [Dur02] a noté que la peinture virtuelle était plus qu'une simple projection 3D d'une image 2D. La peinture a une épaisseur. Le modèle de Sobczyk prend en compte trois éléments de la peinture : le support, les instruments et le médium.

- Le **support** est l'objet sur lequel on peint. Nous avons défini différents supports utilisés par un peintre : le bois, le métal, le papier ou la toile. À chaque support correspond plusieurs caracteristiques comme la couleur et le taux d'absorption de la peinture. Chez Sobczyk, le support est représenté comme un objet 3D. Il est constitué de cellules. Une scène comporte 5×10^7 cellules.
- Le modèle gère les différents **instruments** utilisés par un peintre comme le pinceau ou le couteau. À chaque outil on applique une direction, une force et une quantité de peinture.
- Le **médium** correspond à ce qui est appliqué : charbon du crayon, pâte à peinture, vernis, encre... Il est défini par une couleur et un facteur de viscosité.

La palette prend en compte l'interaction entre le support et le médium. En fonction de la force appliquée aux pinceaux, le support peut être altéré. Sur le support, nous appliquons plusieurs couches de peinture. Ces différentes couches augmentent l'épaisseur de la cellule. Chaque cellule est donc caracterisée par une épaisseur variable.

Le tableau est alors un paysage et est codé comme un objet *OpenGL*. Nous pouvons visualiser la peinture selon différents point de vue ou avec différents éclairages.

Pour passer d'un détail à l'image dans son ensemble, nous avons besoin d'outils et de techniques spécifiques.

3 Interface homme machine

Une fois que la toile virtuelle est créée comme un paysage 3D nous devons fournir les outils permettant de la visualiser en prenant en compte le positionnement et les mouvements de l'utilisateur.
Nous avons choisi un système non intrusif c'est-à-dire un système qui laisse l'utilisateur libre de ses mouvements et ne nécessite pas de porter sur la tête le materiel de capture [Col99]. Il est en effet difficile de réaliser des gestes naturels et spontanés si l'on doit porter un équipement pour les mesures. Un sytème non intrusif permet de laisser l'utilisateur libre de ses mouvements. Il est aussi important que l'utilisateur ne soit pas gêné dans son activité par du bruit ou les eventuels déplacements de l'outil de capture.

Notre but est que l'utilisateur puisse visualiser l'image de la même manière qu'il regarderait un tableau. Plus la peinture est éloignée de l'observateur et plus les contours s'estompent. Par exemple, lorsqu'on regarde un tableau impressionniste en prenant du recul, les contours n'apparaissent plus et ce qui ne semblait n'être que des taches de rouge et de bleu devient un violet. Avec l'éloignement, le cerveau opère la synthèse et il restitue les couleurs qui sont juxtaposées sur la toile selon les lois optiques de leur complémentarité.

Les premiers visiteurs des expositions impressionnistes mettaient le nez sur les tableaux, comme ils le faisaient pour les peintures miniatures de Meissonier [Gom90]. Ils ne pouvaient percevoir qu'un mélange confus de couleurs posées au hasard. Il faudra un peu de temps pour que le public comprenne que pour apprécier une peinture impressionniste il est necessaire de prendre du recul. De loin, un tableau comme [Sis73] est quasiment photoréaliste. Sur la même idée, avec notre palette l'utilisateur doit pouvoir passer de manière très simple de l'image dans son ensemble à un détail de cette image. Par exemple dans *le Bal du Moulin de la Galette de Renoir* [Ren76], l'utilisateur doit pouvoir passer du tableau dans son ensemble au couple au fond sur le banc. L'utilisateur pourra alors se demander si ce couple est en phase de dispute ou de séduction.

Les images sur lesquelles nous travaillons sont en très haute résolution (30 pixels représentant 1 millimètre) et en très haute définition (6000×9000) selon les souhaits exprimés à Eurographics [ABB+01]. On ne peut donc visualiser sur un écran d'ordinateur qu'une partie de l'image. Pour pouvoir visualiser les images très haute définition dans leur ensemble sur l'écran, on applique des zooms arrières sur une partie ou sur l'ensemble de l'image.
Nous avons besoin d'une interface simple et intuitive pour utiliser les zooms. L'un des moyens les plus naturels est de retranscrire le mouvement de l'utilisateur devant la tablette. En fonction du mouvement avant arrière et de la distance parcourue par l'utilisateur, on applique un zoom.

Le principe de cette interface est :
– quand l'utilisateur s'approche de l'écran, on effectue un zoom avant sur l'image.
– quand l'utilisateur recule, on effectue un zoom arrière sur l'image.
Nous allons maintenant présenter cette interface.

4 Technique de zoom

Le système de zoom est composé de deux parties : la mesure de distance et le calcul du zoom adéquat. C'est dans cet ordre que nous traiterons ces deux parties.

4.1 Détection de la position

Nous utilisons des télémètres à ultrasons pour déterminer la position de l'utilisateur. Ces télémètres sont basés sur la mesure du temps écoulé entre l'émission d'un ultrason et le retour de l'écho. L'onde ultrason se propage à la vitesse du son. Dès qu'un obstacle est rencontré, l'écho revient. Le chronomètre s'interrompt à la réception du signal. L'utilisation des télémètres à ultrasons pose certains problèmes :
– précision. Les mesures ne sont pas très précises.

– interférence. Si l'utilisateur fait des gestes brusques ou passe sa main devant son visage, les mesures sont parasitées.
– obstacles. Quand nous mesurons des grandes distances, rien ne doit se trouver entre les télémètres et l'utilisateur.
– minima. Nous ne pouvons détecter précisement les positions très proches de l'écran car les télémètres ne vont pas mesurer la position des yeux mais celle du front.

Nous avons positionné ces télémètres sur la partie supérieure de la tablette. Afin de s'affranchir des perturbations exterieures, nous lançons une série de cinq mesures sur deux télémètres puis nous effectuons un calcul de moyenne. Même ainsi le problème du minima reste posé, nous effectuons donc un calibrage. L'utilisateur détermine les positions "proche" et "loin" et il calibre ses propres positions "proche" et "loin". Ensuite on calcule l'image à afficher en fonction du calibrage de l'image, du calibrage de la position de l'utilisateur, du mouvement et de la distance parcourue par l'utilisateur. Le ratio de distance obtenue sert de valeur d'entrée du zoom. Dans un premier temps, en considérant la scène à afficher, nous avons utilisé les variations de point de vue d'*OpenGL*.

Pour l'instant, notre interface a seulement besoin de connaître le positionnement de l'utilisateur par rapport à l'écran. Mais nous envisageons de mettre en place des systèmes de capture du regard à l'aide d'une caméra [SD02, SYW96] pour traiter seulement la partie de l'image que l'utilisateur regarde.

FIG. 1 – Zoom *aléatoire* sur un détail du Bal du Moulin de la Galette

4.2 Zooms

Les applications de zooms déjà existantes sont essentiellement basées sur un calcul de moyenne. Elles ont un système d'antialliassage qui semble préjudiciable au traitement qu'on effectue sur les images. Par exemple, un lissage sur l'Église d'Auvers sur Oise de Van Gogh [Gog90] dénature la toile. Dans ce tableau, le rapprochement des couleurs vert et bleu pour l'herbe, bleu vif et noir pour le ciel, bleu et rouge pour le toit créent des effets de saturation et de stridence générateurs de malaise. Avec un zoom qui lisse cette image, le contraste des couleurs est atténué et la violence qui traverse la toile originale disparait, les couleurs apparaissent uniformes, l'étirement nerveux de la touche est estompé. Pour éviter cela, nous avons écrit nos propres zooms. Ces zooms prennent en compte les caractéristiques de différents modèles de couleurs (HSV, RGB…)

Principe de ces zooms arrières : un zoom arrière 4 représente la division par 4 des dimensions de l'image.

FIG. 2 – Zoom *moyenne* sur un détail du Bal du Moulin de la Galette

Soient I_O l'image d'origine et I_Z l'image zoomée. Le zoom z est une transformation :

$$I_Z = z(I_0)$$

16 pixels de I_O sont représentés par 1 pixel de I_Z. Pour chaque groupe de 4×4 pixels de I_O, sont appliqués des algorithmes pour obtenir la valeur la plus représentative. Pour le moment, nous appliquons ces algorithmes sur des matrices disjointes.

Nous avons classé nos zooms en catégorie :
- les zooms qui sélectionnent les données de manière arbitraire. En prenant la valeur centrale, une valeur aléatoire ou une valeur différente pour chaque matrice. Avec ces zooms, l'image a un aspect flou et granuleux (cf. image 1).
- les zooms qui recherchent des valeurs communes. On calcule la valeur moyenne, la valeur médiane qui partagent la matrice en deux parties d'égale fréquence ou la valeur la plus fréquente présente dans une matrice. Les images resultantes de ces zooms sont moins contrastées que les images originales (cf. image 2).
- les zooms qui sélectionnent les valeurs extrêmes. On garde la valeur maximum, la valeur minimum ou on calcule la moyenne extrême pour chaque matrice. Ces zooms modifient les couleurs. En prenant les valeurs maximums, on augmente le contraste (cf. image 3).

On choisit ces diffèrents zooms en fonction des caractéristiques qu'on veut faire ressortir sur l'image. En prennant les valeurs moyennes de l'image, nous diminuons le contraste alors qu'en sélectionnant les données extrêmes nous modifions les couleurs. Par exemple pour l'Église d'Auvers sur Oise de Van Gogh, nous utiliserons un zoom qui renforce l'opposition des couleurs. Nous sélectionnons dans le système de couleur HSV, les valeurs moyennes de H et les valeurs maximums de S et de V. L'image 4 présente le résultat ce zoom appliqué sur un détail du tableau de l'Église d'Auvers sur Oise de Van Gogh. La version papier de cet article étant imprimée en noir et blanc, le lecteur trouvera la version couleur des images en [Ate02].

5 Conclusion

L'utilisateur de notre système peut appliquer ses touches de peinture au plus près et en s'éloignant de l'écran obtenir une vue globale du tableau. Nous avons vu comment est détectée la position de l'utilisateur à l'aide de télémètres

FIG. 3 – Zoom *maximum* sur un détail du Bal du Moulin de la Galette

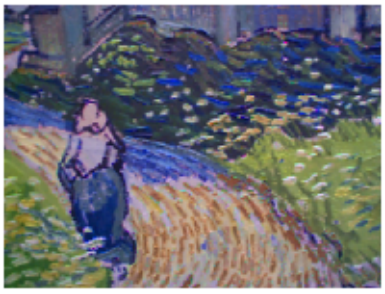


FIG. 4 – Zoom sur un détail de l´Eglise d'Auvers sur Oise

à ultrasons et de quelle manière sont appliqués les zooms. Les télémètres à ultrasons ne sont pas très précis et comportent de nombreux inconvénients. Nous allons prochainement tester notre système avec des télémètres infrarouges pour avoir des mesures plus précises. Nous travaillons sur la mise en place de zooms en 3D qui permettront de visualiser le volume de l'image. Un système de capture du regard doit également être expérimenté. Ce système nous permettra d'appliquer des zooms sur la partie regardée par l'utilisateur. Un de nos axes de recherche est d'améliorer la visualisation des images, en modélisant les effets de craquelure ou de vernis que nous percevons lorsque nous regardons une toile.

Références

[ABB+01] A. Atencia, J.-J. Bourdin, V. Boyer, T. Pissard, and D. Sobczyk. Scalable impressionist rendering. In *Eurographics 2001*, 2001.

[Ate02] Anaïs Atencia. Page web, 2002.
http ://www.ai.univ-paris8.fr/ anais.

[Cai00] R. Caillou. Principles of computer graphics : the experience of a class a user. *Proceedings of EUROGRAPHICS'00*, September 2000.

[CAS+97] C.J. Curtis, S.E. Anderson, JE. Seim, KW. Fleischer, and DH. Salesin. Computer-generated watercolor. *Proceedings of SIGGRAPH'97*, pages 421–430, August 1997.

[Col99] C. Collet. Capture et suivi du regard par un système de vision dans le contexte de la communication homme-machine. Master's thesis, Ecole Normale Supérieur de Cachan, September 1999.

[Dur02] F. Durand. An invitation to discuss computer depiction. *NPAR 2002 : Second International Symposium on Non-Photorealistic Animation and Rendering*, June 2002.

[GG01] B. Gooch and A. Gooch. *Non-Photorealistic Rendering*. A K Peters, 2001.

[Gog90] Vincent Van Gogh. L'Église d'Auvers sur Oise, 1890.
http ://www.abcgallery.com/V/vangogh/vangogh43.html.

[Gom90] E. Gombrich. *Histoire de l'art*. Flammarion, 1990.

[Hae90] Paul E. Haeberli. Paint by numbers : Abstract image representations. *Proceedings of SIGGRAPH'90*, August 1990.

[Her98] A. Hertzmann. Painterly rendering with curved brush strokes of multiples sizes. *Proceedings of SIGGRAPH'98*, July 1998.

[Lit97] P. Litwinowicz. Processing images and video for an impressionist effect. *Proceedings of SIGGRAPH'97*, pages 407–414, August 1997.

[Mei96] B. Meier. Painterly rendering for animation. *Proceedings of SIGGRAPH'96*, pages 477–484, August 1996.

[Ren76] Pierre-Auguste Renoir. Bal du Moulin de la galette, 1876.
http ://www.abcgallery.com/R/renoir/renoir86.html.

[SB99a] M.C. Sousa and J.W. Buchanan. Computer-generated graphite pencil rendering of 3d polygonal models. *Proceedings of EUROGRAPHICS'99*, September 1999.

[SB99b] M.C. Sousa and J.W. Buchanan. Computer-generated pencil drawing. *Western Computer Graphics Symposium*, March 1999.

[SB99c] M.C. Sousa and J.W. Buchanan. Observational model of blenders and erasers in computer-generated pencil rendering. *Proceedings of Graphics Interface*, June 1999.

[SB00] M.C. Sousa and J.W. Buchanan. Observational model of graphite pencil materials. *Computer Graphics Forum*, 19(1), March 2000.

[SBB02a] D. Sobczyk, V. Boyer, and J.-J. Bourdin. Impressionist rendering, a high resolution approach. *3DD : International Workshop on 3D Digitization*, February 2002.

[SBB02b] D. Sobczyk, V. Boyer, and J.-J. Bourdin. Virtual painting : Model and results. *ICCVG'02*, September 2002.

[SD02] A. Santella and D. DeCarlo. Abstacted painterly renderings using eye-tracking data. *NPAR 2002 : Second International Symposium on Non-Photorealistic Animation and Rendering*, pages 75–82, February 2002.

[Sis73] A. Sisley. Chemin de la Machine, Louveciennes, 1873.
http ://www.abcgallery.com/S/sisley/sisley14.html.

[SYW96] R. Stiefelhagen, J. Yang, and A. Waibel. A model based gase tracking system. *Proc. of IEEE International Joint Symposia on Intelligence and Systems*, pages 304–310, November 1996.

[Wac] Wacom.
http ://www.wacom.com.

Visualisation par surfels des textures volumiques

G. Guennebaud, M. Paulin
[guenneba|paulin]@irit.fr

IRIT-UPS 118 route de Narbonne 31062 Toulouse - Cedex 4

Résumé : *Dans cet article, nous présentons une méthode permettant la visualisation en temps réel de scènes complexes ayant un caractère répétitif. De telles scènes peuvent être avantageusement représentées par des textures volumiques. Malheureusement la technique de visualisation temps réel de celle-ci (adaptation de l'algorithme de Lacroute [LL94]) ne permet pas de gérer des motifs très précis (mémoire limitée des cartes graphiques). De plus, des artefacts apparaissent lors des changements de direction des tranches. Nous avons donc adapté à la visualisation des textures volumiques, une technique de rendu hybride entre le rendu à base d'images et le rendu à base de points, . Le motif de référence est alors représenté par un LDC Tree. Une technique permettant de visualiser une telle structure de données a déjà été proposée, mais la déformation des texels implique de revoir l'algorithme de projection ainsi que les différents tests de visibilités. L'intérêt d'une telle approche est de permettre la manipulation de motifs de références beaucoup plus précis, tout en accroissant la qualité du rendu.*

Mots-clés : Rendu à base d'images, Rendu à base de points, Textures volumiques, Temps réel.

1 Introduction

Quels que soient les outils utilisés lors de la modélisation d'un objet ou d'une scène, il est courant de passer par une représentation polygonale pour l'affichage. La principale raison est que ce type de représentation est directement exploitable et affichable par le matériel graphique. Par contre, le nombre de polygones générés est souvent très impressionnant pour des surfaces complexes (de l'ordre du million pour des arbres par exemple). A partir de cette remarque, il devient impossible de pouvoir visualiser en temps réel une forêt complète.

Un maillage de polygones est en fait une représentation très lourde à manipuler lorsque ceux-ci tentent d'approcher un objet complexe. En effet, la géométrie est donnée de manière beaucoup trop explicite et de plus, il existe un lien de connectivité beaucoup trop fort entre les primitives représentant l'objet. Cela rend la réduction du nombre de polygones très délicate dès que l'objet représenté devient trop complexe. Un algorithme de niveau de détails devient alors impraticable. Il devient donc intéressant de chercher une représentation alternative de ces objets complexes afin de rendre leur manipulation plus compacte et d'accélérer leur visualisation.

Quelques solutions ont déjà été proposées, avec en particulier le rendu à base d'images. L'objet est alors représenté par un ensemble d'images contenant au minimum une information colorimétrique mais aussi une information de profondeur et parfois même une information sur le matériau. A mi-chemin entre les représentations à base d'images et les représentations polygonales, on trouve ce que l'on appelle le rendu à base de points. Ici, l'objet est représenté par un ensemble d'échantillons ponctuels appartenant à sa surface ; on parle de surfels. On peut également citer les textures volumiques, qui permettent de représenter efficacement les scènes présentant un caractère répétitif, comme une forêt. Malheureusement la technique de visualisation temps réel de celle-ci présentent de nombreux inconvénients (voir section 3).

Afin de pallier à ces inconvénients, nous proposons une nouvelle manière de visualiser ces textures volumiques. Cela passe bien sûr par une nouvelle représentation des texels. Le choix s'est porté sur une représentation à base de points qui suscite un grand engouement depuis quelques années (voir section 2.1).

2 Travaux antérieurs

2.1 Rendu à base de points

La possibilité d'utiliser des particules pour rendre un objet solide a été initialement suggérée par M. Levoy et T. Whitted en 1985 dans [LW85] où ils présentent les problèmes fondamentaux comme la reconstruction de la

surface, la visibilité et le rendu de surfaces semi-transparentes. Mais ce n'est qu'en 1998 que J.P. Grossman et Dally reprirent cette idée [GD98] et formalisèrent pour la première fois l'utilisation de points comme primitive de rendu.

Les objets sont représentés par un ensemble d'échantillons ponctuels appartenant à leur surface. En fait, ces échantillons sont communément appelés surfels pour *"élément de surface"* (*"surface element"* en anglais). Ce terme a été introduit mathématiquement par Herman en 92 [Her92], mais Pfister et al. proposent une nouvelle définition plus adaptée à notre cas. D'après Pfister, un surfel est un n-tuple de dimension 0 avec des attributs de forme et de matière qui approxime localement la surface d'un objet. De plus, les surfels ne contiennent aucune connectivité explicite avec leurs voisins, ce qui en fait une représentation très souple à manipuler.

Pour ce qui est des structures de données, on peut citer la hiérarchie de sphères englobantes présentée avec le système QSplat dans [RL00] et le LDC Tree présenté avec les surfels dans [PZvBG00]. Ces deux structures possèdent deux caractéristiques fondamentales. La première est qu'elles sont multi-résolutions, ce qui permet d'ajuster le nombre de surfels à projeter en fonction de l'éloignement de la caméra. La deuxième est que les points sont stockés par blocs et de manière hiérarchique, ce qui rend ces tests de visibilité encore plus efficaces. Ces tests sont au nombre de trois et ont été proposés par Grossman [GD98] :

1. Un test classique de fenêtrage.
2. Un test basé sur les *"cônes de visibilité"* (équivalent de l'élimination des faces arrières).
3. Un test basé sur les *"masques de visibilités"* (gérant les problèmes d'auto occlusion).

Il existe deux manières d'aborder la reconstruction de la surface. Une première approche est de travailler dans l'espace image [GD98, PZvBG00]. Avec ce type d'algorithme, il est possible de prendre en compte des surfaces semi-transpentes (utilisation d'un A-buffer), par contre seule une implémentation logicielle est actuellement possible. Pour bénéficier d'une accélération par le matériel graphique il est nécessaire d'exprimer cette reconstruction dans l'espace objet [RL00, KV01]. Cependant, aucune des techniques précédentes ne supporte un anti-aliassage pour les modèles ayant une texture complexe. Pour répondre à ce manque, Pfister et al. reprennent le concept du filtrage pondéré elliptique de Heckbert [GH86] nommé EWA (Elliptical Weighted Average), et l'appliquent au splatting de surface en formulant les noyaux de reconstruction dans l'espace image [ZPvBG01]. Récemment Liu Ren et al. [RPZ02] parviennent à exprimer ces noyaux dans l'espace objet et tirent ainsi bénéfice d'une implémentation possible avec OpenGL.

2.2 Textures volumiques

Introduites par Kajiya et Kay en 1989 [KK89], puis développées par F. Neyret [Ney95], les textures volumiques utilisent 3 niveaux différents pour représenter l'information :

1. Les grandes variations, comme la surface d'une colline ou le dos d'un animal, sont codées par une description géométrique classique (mailles de polygones, carreaux de Bézier, surface NURBS, ...).
2. Le niveau de détail moyen, comme l'herbe ou les poils, qui sont concentrés au voisinage de la surface, est codé en utilisant un volume de référence stocké une seule fois et plaqué répétitivement, à la manière d'une texture 2D. Une instance de ce volume de référence est appelée texel.
3. Le niveau de détail fin, comme les microscopiques variations de chaque objet, sont codées par un modèle de réflexion stocké dans chaque voxel. Ce niveau correspond au niveau du pixel.

Il s'agit d'un modèle multi-échelle. Les textures volumiques peuvent être vue comme une extension des textures 2D traditionnelles. Au lieu de plaquer une image, forcément plane, on va plaquer une donnée volumique appelée texel. Ce plaquage est donc paramétré par les classiques coordonnées de texture 2D et par un vecteur de hauteur. Ce vecteur permet de déformer les texels en les coiffant par rapport à la surface par exemple (voir figure 1). Les texels forment une véritable couche sur la surface de la géométrie sous-jacente.

Au départ, la visualisation d'un texel passait par un algorithme de lancé de rayon. Le volume de référence étant modélisé par un octree, le rendu d'un texel ressemble beaucoup au rendu volumique. Mais, contrairement au rendu volumique, les texels ne contiennent pas une densité mais plutôt une probabilité d'occultation ainsi qu'un modèle de réflectance. Par la suite, la méthode de visualisation interactive de volume développée en 1994 par Lacroute et Levoy [LL94] a été adaptée aux textures volumiques [MN98]. Cette approche consiste à stocker le volume de référence par tranches. Trois séries de tranches sont extraites du volume dans trois directions orthogonales. Lors du rendu, une des piles de tranches est sélectionnée en fonction de la direction de visée. Les tranches sont ensuite

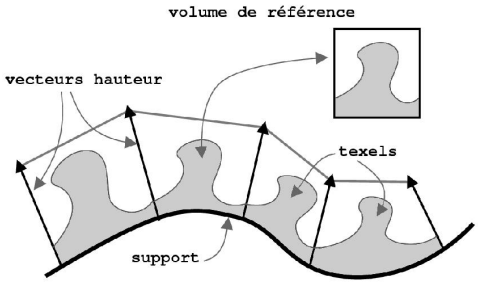


FIG. 1 – Spécification des textures volumiques

rendues par la carte graphique comme des polygones texturés. L'éclairage dynamique peut être pris en compte en stockant également des cartes de normales et en utilisant la même technique que pour le bump-mapping. L'utilisation du mipmapping permet de conserver le caractère multi-résolution des textures volumiques. Remarquons que les dernières cartes graphiques permettent l'utilisation de texture 3D. Il n'est donc plus nécessaire de stocker trois fois le volume de référence.

Un dernier point concerne l'animation des textures volumiques. En effet, il est possible d'animer les texels de trois manières différentes (et indépendantes) :

1. Déformer la surface sur laquelle se trouvent les texels.
2. Déformer l'orientation des vecteurs de hauteur se trouvant sur chaque sommet de la surface (par exemple pour simuler du vent dans l'herbe).
3. Animer le volume de référence à la manière d'un dessin animé (pré-calcul d'une série de volumes de référence).

Par contre, l'animation du volume de référence reste quand même très coûteuse en mémoire.

3 Une nouvelle représentation des textures volumiques

L'idée de proposer une nouvelle méthode de visualisation temps réel des textures volumiques vient du fait que la méthode initialement proposée présente quelques limites : artefacts lors des changements de tranches, impossibilité d'obtenir un ombrage correct, animation du texel de référence quasi impossible. Mais surtout, le volume de référence est stocké entièrement, et même plusieurs fois. La mémoire texture des cartes graphiques étant limitée, on est restreint à des volumes de 128^3 voir 256^3 (en compressant les données stockées).

Il nous faut donc trouver une nouvelle représentation plus compacte, tout en conservant l'aspect multi-résolution. Nous avons choisi une approche par surfels qui semble bien appropriée puisque seuls les points appartenant à la surface de l'objet représenté sont conservés. De plus, en utilisant une structure de données tel que le LDC Tree nous avons une représentation complètement multi-résolution de notre motif de référence. Le choix du LDC Tree est d'autant plus judicieux qu'il permet d'avoir une approche incrémentale de la projection des surfels, ce qui est impossible avec la hiérarchie de sphère englobante.

Nos techniques d'aquisition et de rendu d'un LDC Tree sont largement inspirées des travaux de Pfister et al. et de ceux de Grossman. Nous allons donc nous contenter de présenter ces deux phases dans les grandes lignes (sections 4 et 6) afin de nous concentrer sur l'adaptation aux textures volumiques (section 5).

4 Le LDC Tree et son acquisition

4.1 Contenants de base : les LDI

LDI est une abréviation pour « Layered Depth Image » [SGS98]. Comme son nom l'indique, il s'agit d'images stockées en couches, où chaque pixel contient une information de profondeur. Cette information de profondeur est relative au modèle de la caméra liée au plan image. Dans notre cas il s'agit d'une caméra à projection parallèle.

Un LDI est donc une matrice 2D (une image) où chaque pixel stocke la liste des intersections entre le rayon lui correspondant et la scène (figure 2). D'où la notion de couche. Dans notre cas, il s'agit d'une liste de surfels.

Les avantages de ce stockage sont multiples. Tout d'abord, certains attributs des surfels deviennent implicites, comme leurs coordonnées (x,y) et leur diamètre, ce qui entraine une certaine **compacité**. Un second point concerne l'**efficacité**. En effet, le stockage des surfels dans une grille régulière permet d'avoir une approche incrémentale de la projection [Gro98].

4.2 Pour un meilleur échantillonnage : le LDC

En fait, une seule LDI ne permet pas de représenter correctement tout un objet. En effet, les zones de la surface tangentes par rapport à la direction z de la LDI sont très mal échantillonnées. Une solution possible est d'utiliser trois LDI orthogonales entre elles. Cet arrangement est appelé un LDC, abréviation de « Layered Depth Cube », et est illustré figure 2. Un LDC est aussi appelé bloc.

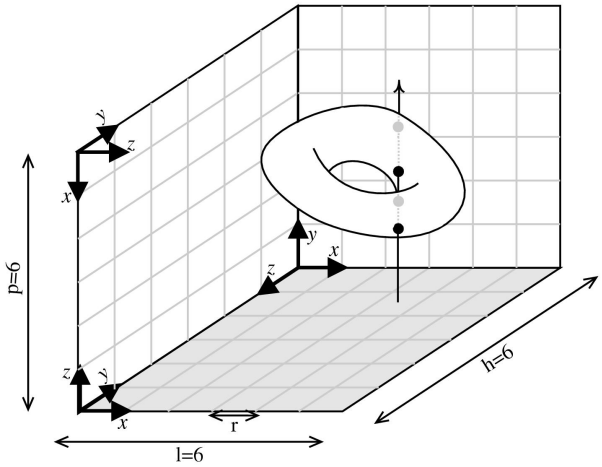


FIG. 2 – Un LDC composé de 3 LDI avec leur repère (i, j, k) respectif. Le rayon associé au pixel $(3, 4)$ de la LDI grisée intersecte 4 fois l'objet, ce pixel contient donc 4 surfels. Sont également mentionnées la largeur l, la hauteur h ainsi que la profondeur p du bloc. Une des trois LDI joue donc un rôle particulier : son repère est aussi celui du bloc.

4.3 Hiérarchique et multi-résolution : le LDC tree

Le LDC tree peut être vu comme une sorte d'octree ou chaque noeud est un LDC (ou plus généralement un bloc). Dans cette section nous allons voir comment construire une telle hiérarchie.

Pour la construction d'un LDC tree, nous partons d'un grand LDC contenant entièrement l'objet voulu. Ce LDC est subdivisé en un certain nombre de petits blocs de taille b^3 (on suppose, pour simplifier, que les blocs sont cubiques, donc $l = h = p = b$). Ces blocs forment les feuilles de l'octree. Le niveau supérieur est construit en fusionnant 8 blocs adjacents (fils) pour former un seul bloc (père) de même taille b^3 mais dont l'espace r entre les pixels est double. Pour la fusion des fils, notre approche est différente de celle de Pfister. Rappelons qu'ils se contentaient de recopier les adresses des listes de surfels d'un pixel sur deux. C'est pour cela qu'ils utilisent plusieurs niveaux de texture. Dans notre cas, un bloc temporaire de taille $(2b)^3$ est créé par concaténation des huit fils. Puis, la réduction de la résolution est réalisée, comme pour une image, en faisant la moyenne des pixels des LDI quatre par quatre. Mais ici les pixels contiennent une liste de surfels. Les surfels des quatre listes à fusionner sont d'abord mis en correspondance (en comparant leur valeur z) puis fusionnés en faisant les moyennes de leurs attributs (profondeur, couleur, normale). Les niveaux supérieurs de la hiérarchie sont construits ainsi de suite jusqu'à obtenir un seul bloc, la racine, représentant à lui seul tout l'objet mais à une très faible résolution.

5 Des surfels aux textures volumiques

Dans cette section nous allons voir comment adapter le rendu classique d'un LDC Tree lorsque celui-ci représente un texel. Pour cela nous proposons une nouvelle manière d'exprimer la déformation des texels (section 5.1) qui va nous permettre d'intégrer cette déformation au sein de notre propre algorithme de projection incrémentale (section 5.2). Enfin, nous aborderons le problème de la visibilité en essayant d'adapter les tests déjà proposés (section 5.3).

5.1 Déformation des texels

La forme et position d'un motif appliqué à la surface d'un objet est complètement définie par la donnée de la position des 4 coins du texel en contact avec la surface de l'objet et des 4 vecteurs hauteurs en ces points (voir figure 3).

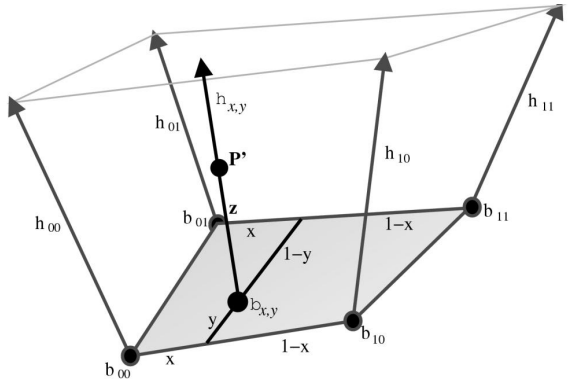


FIG. 3 – Paramétrisation de la déformation d'un texel par les quatres sommets $b_{i,j}$ et les vecteurs hauteurs $h_{i,j}$.

Cette déformation ne peut pas être exprimée à l'aide d'une matrice de transformation. En effet, on ne peut la décomposer en une succession de rotations, mises à l'échelle et translations. Nous allons donc voir comment elle peut être exprimée. Soit p un point du motif de référence de coordonnée (x, y, z) exprimées dans le repère local du texel. Nous supposons de plus que le motif est normalisé, c'est à dire de largeur 1, de hauteur 1 et de profondeur 1. Soit $\widetilde{p}$, l'image de p par la déformation. Le calcul de $\widetilde{p}$ peut être effectué de la manière suivante :

Calcul de $b_{x,y}$ (resp. $h_{x,y}$) résultat de l'interpolation bilinéaire des $b_{i,j}$ (resp. $h_{i,j}$), $(i,j) \in [0..1]^2$), par les coefficients x et y :

$$\begin{aligned} b_{x,y} &= lerp\left(y, lerp\left(x, b_{00}, b_{10}\right), lerp\left(x, b_{01}, b_{11}\right)\right) \\ h_{x,y} &= lerp\left(y, lerp\left(x, h_{00}, h_{10}\right), lerp\left(x, h_{01}, h_{11}\right)\right) \end{aligned} \tag{5.1}$$

Où $lerp$ est la fonction d'interpolation linéaire : $lerp(t, A, B) = (1-t)\,A + t\,B$. On a alors :

$$\widetilde{p} = b_{x,y} + z\,h_{x,y} \tag{5.2}$$

A ce stade, plusieurs remarques s'imposent. Tout d'abord, le calcul de $b_{x,y}$ et $h_{x,y}$ peut être réalisé par la géométrie support des texels, dans le cas de surface paramétrique notamment. Dans ce cas, le résultat d'un motif déformé ne correspond pas à la figure 3 et le motif épousera complètement la surface. D'autre part, dans le cadre de la figure 3, il est possible de considérer le carreau $(b_{00}, b_{10}, b_{01}, b_{11})$ comme un carreau de Bézier à partir duquel il est facile de calculer $b_{x,y}$ et $h_{x,y}$. Cela a pour conséquence de lisser la surface de l'objet. Dans la suite, nous allons rester dans le cadre de la figure 3, les deux possibilités précédentes pouvant être facilement adaptées par la suite.

5.2 Algorithme de projection

Commençons par remarquer que la forme d'un bloc au sein du motif déformé est similaire à celle du motif tout entier représenté figure 3. La position des quatres coins $b_{i,j}$ et les vecteurs hauteurs $h_{i,j}$ d'un bloc sont obtenus

facilement à partir de ses paramètres (position au sein du motif et dimensions) et des équations 5.2 et 5.1. Nous sommes donc en mesure de projeter les surfels d'un bloc : soit (x, y, z) les coordonnées du $k^{ième}$ surfel $S_{i,j,k}$ stocké en (i, j) dans la LDI. A partir de l'équation 5.2, on obtient $(\widetilde{x}, \widetilde{y}, \widetilde{z})$, coordonnées de $S_{i,j,k}$ dans le repère objet et après déformation. Il ne reste plus qu'à appliquer la transformation de modélisation et la projection en perspective conique pour obtenir les coordonnées (u, v) de $S_{i,j,k}$ dans l'espace image. Par contre, l'implémentation directe de cette méthode n'est pas vraiment envisageable, puisqu'elle demande un nombre bien trop important d'opérations par surfel à projeter.

Dans les sections précédentes, nous avons vu qu'il était possible de tirer bénéfice de la cohérence spatiale entre les surfels due à leur stockage dans une grille régulière. Nous allons donc tenter la même approche que Grossman, mais en tenant compte de la déformation du bloc. De plus, nous allons considérer uniquement le LDI des blocs dont la référence correspond au carreau $(b_{00}, b_{10}, b_{01}, b_{11})$. Pour les autres, il suffit de faire subir une rotation à la figure 3, le carreau $(b_{00}, b_{10}, b_{01}, b_{11})$ ne se retrouvant plus appliqué à la surface de l'objet.

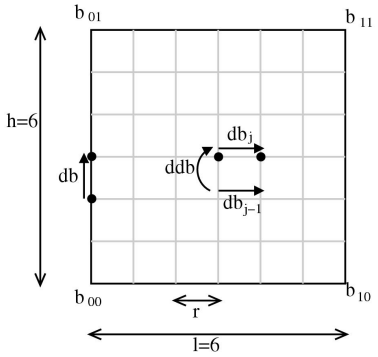


FIG. 4 – Illustration des différents incréments. Ce shéma s'applique aussi bien au vecteur b, qu'aux vecteurs h, q et r.

Afin de simplifier l'écriture des formules, nous allons nous placer dans une ligne j et considérer un seul surfel par "pixel" (i, j). Nous considérons donc le surfel S_i, de coordonnées (x_i, y_i, z_i) dans le bloc :

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = o + r \begin{bmatrix} i \\ j \\ \delta_{i,j,k} \end{bmatrix} \tag{5.3}$$

Où, o est l'origine du bloc courant dans le repère du LDC Tree, r la distance entre deux pixels et $\delta_{i,j,k}$ la profondeur du $k^{ième}$ surfel stocké en (i, j). Il est donc intéressant d'insérer la translation par o et la mise à l'échelle par r dans la matrice de modélisation active ; soit M la matrice 4x4 résultante que l'on décompose en une matrice 3x3 orthogonale A (représentant une rotation et mise à l'échelle) et un vecteur de translation T. Pour simplifier l'écriture, posons : $b_i = b_{x_i,y_i}$ et $h_i = h_{x_i,y_i}$. Dans le repère de la caméra, on a :

$$\begin{bmatrix} x'_i \\ y'_j \\ z'_j \end{bmatrix} = A \begin{bmatrix} \widetilde{x}_i \\ \widetilde{y}_i \\ \widetilde{z}_i \end{bmatrix} + T = A\,(b_i + z_i\,h_i) + T \tag{5.4}$$

Soit (u_i, v_i) les coordonnées de S_i dans l'espace image :

$$u_i = \left\lfloor \frac{L}{2}\left(1 - \frac{x'_i}{z'_i}\right)\right\rfloor \qquad v_i = \left\lfloor \frac{L}{2}\left(1 - \frac{y'_i}{z'_i}\right)\right\rfloor \tag{5.5}$$

Où, L est la taille de l'image et $\lfloor . \rfloor$ renvoie la partie entière. En insérant 5.2 dans 5.5 et en posant :

$$q_i = A\,b_i + T \qquad \text{et} \qquad r_i = A\,h_i$$

On obtient finalement :

$$u_i = \left\lfloor \frac{L}{2}\left(1 - \frac{(p_i)_x + z_i\,(q_i)_x}{(p_i)_z + z_i\,(q_i)_z}\right)\right\rfloor \qquad v_i = \left\lfloor \frac{L}{2}\left(1 - \frac{(p_i)_y + z_i\,(q_i)_y}{(p_i)_z + z_i\,(q_i)_z}\right)\right\rfloor \tag{5.6}$$

Lors du parcours d'une ligne j (voir figure 4, on a donc :

$$q_i = q_{i-1} + \mathrm{d}q_j \qquad r_i = r_{i-1} + \mathrm{d}r_j \tag{5.7}$$

Où les deux incréments introduits sont égaux à :

$$\mathrm{d}q_j = A\,\mathrm{d}b_j \qquad \mathrm{d}r_j = A\,\mathrm{d}h_j$$

L'indice j des incréments est là pour signifier qu'il dépend de la ligne parcourue. Par la suite, on introduira donc des incréments de saut ligne pour ces incréments. $\mathrm{d}b_j$ (resp. $\mathrm{d}h_j$) est l'incrément permettant de passer de b_{i-1} à b_i (resp. de h_{i-1} à h_i). Eux aussi dépendent de la ligne parcourue et nous verrons qu'il n'est pas nécessaire de les calculer explicitement. Maintenant, voyons comment passer d'une ligne à la suivante. Tout d'abord, que se passe-t-il pour les vecteurs q et r :

$$q_{0,j} = q_{0,j-1} + A\,\mathrm{d}b \qquad r_{0,j} = r_{0,j-1} + A\,\mathrm{d}h \tag{5.8}$$

Où, $\mathrm{d}b$ (resp. $\mathrm{d}h$) est l'incrément permettant le calcul de $b_{x,y}$ (resp. $h_{x,y}$) lors du passage d'une ligne à la suivante avec $i = 0$, c'est à dire :

$$b_{x_0,y_{i-1}} = b_{x_0,y_i} + \mathrm{d}b \qquad h_{x_0,y_{i-1}} = h_{x_0,y_i} + \mathrm{d}h$$

Avec :

$$\mathrm{d}b = \frac{b_{01} - b_{00}}{h} \qquad \mathrm{d}h = \frac{h_{01} - h_{00}}{h}$$

Où, rappellons le, h est la hauteur du bloc. Il ne reste plus qu'à incrémenter les incréments $\mathrm{d}q_j$ et $\mathrm{d}r_j$:

$$\mathrm{d}q_j = \mathrm{d}q_{j-1} + A\,\mathrm{dd}b \qquad \mathrm{d}r_j = \mathrm{d}r_{j-1} + A\,\mathrm{dd}h \tag{5.9}$$

Ici, $\mathrm{dd}b$ (resp. $\mathrm{dd}h$) est l'incrément de l'incrément $\mathrm{d}b_j$ (resp. $\mathrm{d}h_j$). Pour illustrer tous ces incréments on peut se référer à la figure 4. Ces derniers sont obtenus par :

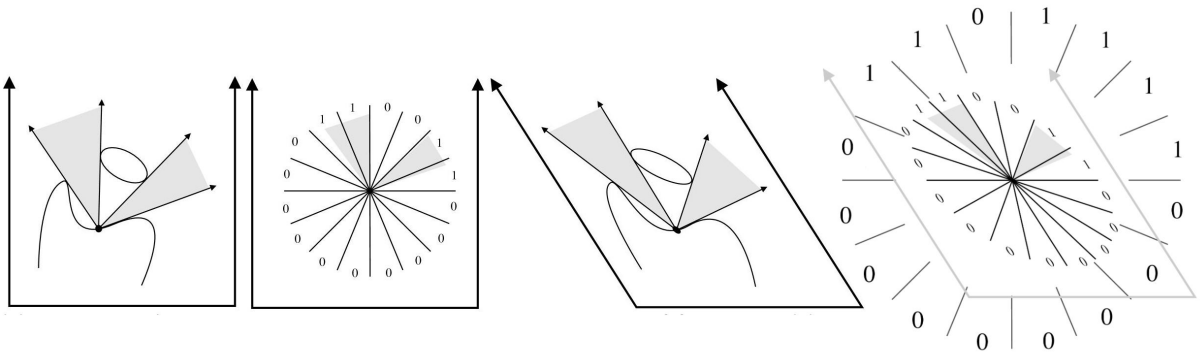
$$\mathrm{dd}b = \frac{b_{11} - b_{01} - b_{10} + b_{00}}{l\,h} \qquad \mathrm{dd}h = \frac{h_{11} - h_{01} - h_{10} + h_{00}}{l\,h}$$

Pour résumer, après initialisation des différents constantes et variables, le rendu d'un bloc s'effectue très simplement et à peu de frais. Un incrément selon i est réalisé par les équations 5.7, (2 additions vectorielles). La projection d'un surfel, equations 5.6, nécessite 2 multiplications vectorielles, 1 inversion et 2 additions vectoriels. Le passage d'une ligne à suivante est réalisé par les equations 5.8 et 5.9 pour un total de 4 additions vectorielles. Bien sùr, la prise en compte de la déformation ajoute un coût de calcul supplémentaire mais qui, au final, reste tout à fait raisonnable, d'autant plus qu'il s'agit de calcul vectoriel et donc directement optimisable par l'utilisation des instructions flottantes SIMD des derniers processeurs PC.

5.3 Visibilité

Nous venons de voir que la prise en compte de la déformation du LDC Tree nécessite de revoir à la base la projection des surfels. Maintenant qu'en est-il des tests de visibilité ?
Pour le fenêtrage, pas de problème, il suffit de calculer la boite englobante du bloc une fois déformé.
Pour ce qui est des masques et cônes de visibilité, les choses se compliquent. Ici nous allons nous contenter de présenter le problème pour les masques de visibilité, le cas des cônes de visibité étant semblable. Une description détaillé de leur fonctionnement peut être trouvé dans [Gro98], mais rapellons tout de même qu'un masque de visibilité est un masque de n bits (typiquement $n = 128$) où chaque bit correspond à un triangle résultant de la subdivision régulière de la sphère en n triangles. Un masque est calculé pour chaque bloc de tel sorte que $k^{ième}$ bit vaut 1 si et seulement si le bloc est visible (à partir d'un point de vue hors de l'enveloppe convexe de l'objet) d'une direction correspondant au $k^{ième}$ triangle. Au moment du rendu, un masque est calculé pour le volume de visualisation. Il suffit alors de réaliser un ET logique entre les deux masques pour savoir si le bloc est visible ou non.

La difficulté est qu'il est tout à fait possible que, pour une position et orientation de la caméra données, une partie du motif soit caché par lui-même et devienne tout à fait visible suite à la déformation. Cette partie, équivalente



(a) (b) (c) (d)

FIG. 5 – Illustration de la déformation des masques de visibilité (en 2D). (a) motif non déformé avec en grisé les directions à partir desquelles le bloc matérialisé par un point est visible. (b) masque de visibilité associé au bloc. (c) déformation du masque de visibilité et reprojection sur la partition de l'espace des directions. (d) une fois le motif déformé, les directions valides pour le bloc correspondent bien au masque précédemment calculé. Masque avant la transformation : 0011 0000 0000 0011 Masque après la transformation : 1111 0000 0000 0110

à un bloc, risque alors d'être éliminée par le test des masques de visibilité. Cela vient du fait que l'espace des directions est aussi perturbé par la déformation. Il faut donc faire subir une transformation au masque avant le test. Cette transformation peut être calculée de la manière suivante. Tout d'abord, la pseudo-sphère représentant les 128 partitions de l'espace des directions est positionnée au centre du bloc. Des 128 triangles approximant la sphère, seuls ceux correspondant aux bits du masque égaux à 1 sont conservés. Puis, la déformation (équations 5.2 et 5.1) est appliquée aux sommets des triangles qui sont ensuite reprojetés sur la pseudo-sphère afin d'en déduire le nouveau masque (figure 5).

Si en théorie tout cela semble fonctionner, le passage à la pratique reste problématique car la démarche qui vient d'être exposée est beaucoup trop lourde en terme de coût de calcul. Un test de visibilité doit être extrêmement rapide à exécuter pour être valide, sinon il ne fait que ralentir le processus de rendu. L'idée pourrait être de prendre en compte la cohérence temporelle de la déformation des texels. En effet, il est raisonnable de faire l'hypothèse que la forme d'un texel varie lentement au cours du temps. Il est alors possible de stocker, pour chaque bloc de chaque texel, le nouveau masque de visibilité et de les mettre à jour seulement lorsque cela est nécessaire. On se retrouve alors avec un problème de coût mémoire dès que le nombre de texels est grand. Rappelons que dans la pratique seul un texel servant de référence est stocké puis instancié et déformé lors du rendu.

Une implémentation efficace des masques de visibilité prenant en compte la déformation du motif serait d'autant plus souhaitable que les textures volumiques permettent d'en exprimer toute la puissance. En effet, prenons l'exemple d'une forêt modélisée avec l'aide des textures volumiques. Il est alors astucieux de construire un motif représentant plusieurs arbres. Par exemple, un au centre et un deuxième réparti dans les quatre coins. En calculant les masques de visibilité sur un tel motif, ceux-ci sont alors capables de gérer les interactions avec les arbres voisins et pas seulement pour l'arbre lui-même.

6 Rendu

6.1 Calcul de l'éclairage

Le calcul correct de l'éclairage nécessite tout d'abord d'appliquer la déformation également aux normales. Exprimer quelle est la transformation que subit la normale d'un surfel quelconque du texel est loin d'être évident. Nous prosons donc une astuce qui consiste à prendre un point situé sur la normale du surfel courant et à une distance ϵ (ϵ étant suffisement petit) de celui-ci. Soit p les coordonnée du surfels courant, $\vec{N}$ sa normale et p_ϵ le point correspondant. On a :

$$p_\epsilon = \epsilon \vec{N} + p \qquad \text{d'ou :} \qquad \widetilde{p_\epsilon} = b_{\epsilon N_x + p_x, \epsilon N_y + p_y} + (\epsilon N_z + p_z)\ h_{\epsilon N_x + p_x, \epsilon N_y + p_y} \tag{6.1}$$

Le calcul direct de $\widetilde{p_\epsilon}$ avec les équations 5.1 et 5.2 est possible mais couteux. Cela dit, comme pour l'algorithme de projection, il est possible d'avoir une approche incrémentale de ce calcul :
En remarqant que :

$$lerp(t+\epsilon, A, B) = lerp(t, A, B) + \epsilon\,(B - A) \text{ et } lerp(t, A+a, B+b) = lerp(t, A, B) + lerp(t, a, b) \quad (6.2)$$

On arrive à :

$$\widetilde{p_\epsilon} = \widetilde{p} + R_b + \epsilon\, N_z\, h_{p_x, p_y} + (p_z + \epsilon\, N_z)\; R_h \quad (6.3)$$

Avec :

$$R_b = \epsilon\, N_y\, lerp\,(p_x, (b_{01} - b_{00}), (b_{11} - b_{10}))\, \epsilon\, N_x\, lerp\,(p_y, (b_{10} - b_{00}), (b_{11} - b_{01})) + \epsilon^2\, N_x\, N_y\; (b_{11} + b_{00} - b_{01} - b_{10}) \quad (6.4)$$

R_h étant calculé de la même manière. Le vecteur normal $\widetilde{n}$ non normalisé est alors égal à :

$$\widetilde{\vec{n}} = \widetilde{p} - \widetilde{p_\epsilon} = R_b + \epsilon\, N_z\, h_{p_x, p_y} + (p_z + \epsilon\, N_z)\; R_h \quad (6.5)$$

Au premier abord cette expression parait beaucoup plus compliquée que le calcul direct de $\widetilde{p_\epsilon}$ avec les équations 5.1 et 5.2. Mais en fait, de nombreux termes sont des constantes qui peuvent être précalculées et les interpolations linéaires apparaissant dans les expressions de R_b et R_h peuvent être calculées de manière incrémentale. De plus, le calcul de h_{p_x, p_y} est déjà réalisé lors de la projection incrémentale du surfel (section 5.2) et les $3^ième$ terme en ϵ^2 dans les expressions de R_b et R_h sont négligeables. Finalement, le nombre de multiplication a été divisé par 2 par rapport au calcul direct de $\widetilde{p_\epsilon}$. Il reste le problème du choix de la valeur à donner à ϵ. Dans la pratique, prendre ϵ égale à la distance entre deux pixels du bloc courant semble être un bon choix.

Le calcul de l'éclairage est alors effectué une fois que tous les surfels ont été projetés, mais avant la reconstruction comme dans [GD98]. En fait, n'importe quel modèle d'illuminaton peut être utilisé, tout dépend de la manière dont les matériaux sont représentés. Pour faire simple et efficace, nous utilisons un modèle de Phong. Pour ce qui est des ombres portées, celle-ci peuvent être calculées très simplement par la technique des *"shadows maps"* [WTRDHS87].

6.2 Reconstruction de la surface

Pour ce qui est de la reconstruction de la surface, nous avons simplement adapté la méthode de Grossman à notre problème. Si cette méthode est loin d'être la meilleure en terme de qualité, elle a comme avantages d'être simple et très rapide. Ici, les surfels sont projetés sur un seul pixel de l'image. La détection des trous est alors réalisée par une hiérarchie de Z-buffer à résolution décroissante. Lors de la projection d'un bloc, le z-buffer ayant une résolution suffisement faible pour qu'il n'y ait pas de trou est activé et utilisé conjointement avec le z-buffer qui est à la résolution de l'image. Une fois que tous les surfels sont projetés, cette hiérarchie de z-buffer va permettre de filtrer les pixels situés en avant plan de ceux situés en arrière plan (en comparant la profondeur du pixel avec la profondeur correspondante stocké par les z-buffer de résolution inférieure). Le résultat est un poids compris entre 0 et 1 qui est affecté à chaque pixel. Ce sont ces poids qui vont permetre la reconstruction des *« trous »* par interpolation, celle-ci étant réalisée par un algorithme dit "pull-push".

6.3 Intégration avec une scène OpenGL

Puisqu'il ne semble pas raisonnable de modéliser une scène complète uniquement avec une représentation ponctuelle (que ce soit par l'intérmédiaire des textures volumiques ou non), et que l'algorithme de rendu présenté est indépendant de toute bibliothèque graphique, il serait intéressant de pouvoir intégrer nos objets dans une scène rendue par OpenGL. La solution est de récuperer le tampon de profondeur après le rendu OpenGL et avant la projection de nos surfels. Il serait également possible de récuper en même temps le tampon chromatique, mais il est plus efficace de mettre à jour une texture OpenGL à partir de l'image résultant de la reconstruction, puis de l'afficher en rendant un polygone recouvrant tout l'écran. Lors de ce dernier rendu et selon les capacités de la carte graphique, il est possible de faire réaliser par cette dernière de nombreuses opérations couteuses :

– Calcul de l'éclairage dans le cas où nos objets sont diffus et que la source de lumière est située à l'infini (cas d'une forêt en plein jour) : utilisation d'une deuxième texture représentant la carte des normales (reconstruite en même temps que la carte colorimétrique) et des *pixels shaders*).
– Normalisation des normales par une texture cubique et l'utilisation des *"textures shaders"*.
– Gérer la visibilité entre nos objets et les objets rendu avec OpenGL en utilisatant les *"textures shaders"* et une autre texture représentant notre carte de profondeur. Il n'est alors plus nécessaire de récupérer le tampon de profondeur calculé par OpenGL (opération coûteuse).
– Calcul de l'éclairage avec des matériaux non nécessairement diffus et des sources ponctuels grace aux *"fragments shaders"* disponible sur la futur NV30.

7 Résultats

Les tests ont été réalisés sur un Atlhon 1Ghz disposant de 256Mo de mémoire et d'une carte graphique GeForceII MX. Pour ce qui est de l'implémentation, notons que seule la partie concernant la projection incrémentale d'une LDI (et la transformation et normalisation des normales) a été optimisée par l'utilisation du jeu d'instructions 3DNow !. L'algorithme de reconstruction traitant des données vectorielles (couleurs RGBA et normales), on peut estimer que l'utilisation du jeu d'instructions SSE diminurait les temps de reconstruction par 4. Les images de la figures 6 ont été obtenues à partir d'un motif contenant un seul arbre et d'une résolution de 512^3 (les feuilles du LDC Tree sont au nombre de 64^3 et contiennent trois LDI d'une résolution de 8^2). Cet arbre a été habillé avec la méthode présentée par Maritaud dans [MDG00] et est représenté par 1.2 millions de surfels. Au niveau du coût de stockage, ce motif nécessite à peu près 30Mo d'espace disque (2 millions de surfels et 40 milles noeuds pour le LDC Tree entier). La scène complète (2000 arbres) nécessitent de 0.3s à 1s (selon le point de vue) de temps de calcul réparti de la manière suivante :

– Recupération du tampon de profondeur : 0.05s
– Reconstruction : de 0 à 0.6s selon la distance entre les surfels dans l'espace image. :
  – 70% pour la phase de recherche des trous et de calcul des poids.
  – 30% pour l'algorithme pull-push
– Rendu des LDC Tree :
  – 10% pour le parcours de la hiérarchie incluant les tests de visibilité et l'initialisation de la projection incrémentale d'une LDI.
  – 90% pour la projection des surfels
– Calcul de l'illumination et rendu OpenGL : 0.024s

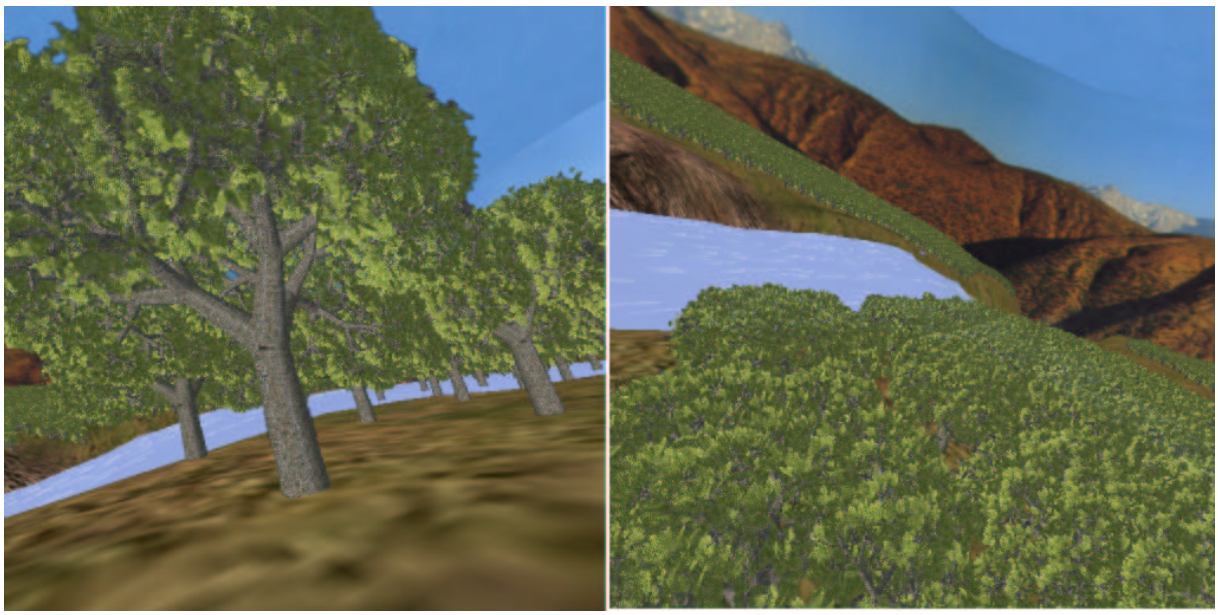


FIG. 6 – Images types

8 Conclusion

Le principal intérêt de cet article est de montrer qu'une représentation par surfels est particulierement bien adaptée à la visualisation des textures volumiques. Cependant, il reste encore de nombreux points à améliorer. Le premier concerne les tests de visibilité. Comme nous l'avons vu, il semble très difficile d'adapter de manière efficace les tests déjà proposés par Grossman. Il serait donc interressant de proposer de nouveaux tests qui soient plus adaptés aux objets subissant une déformation comme les textures volumiques ou bien les objets animés par squelette... Un second point est que l'algorithme de reconstruction utilisé ici est d'assez mauvaise qualité. Son gros avantage est qu'il n'y a aucune rastérisation des surfels projetés, ce qui permet une implémentation logicielle très efficace. Actuellement, le meilleur algorithme de reconstruction est celui proposé par Pfister et al. qui s'appuie sur le filtrage EWA [ZPvBG01] (c'est le seul à proposer un filtrage anisotrope). De plus, une implémentation utilisant le matériel graphique a déjà été proposée [RPZ02]. L'avenir passe donc sans doute par l'utilisation de cette technique de reconstruction. A priori, cela ne semble pas trop difficile puisque la déformation des texels peut facilement être implémentée au niveau des *vertex programmes* au prix d'une trentaine d'instructions supplémentaires.

Références

[GD98] J. P. Grossman and W. J. Dally. Point sample rendering. *Rendering Techniques 98*, pages 181–192, June 1998.

[GH86] N. Greene and P. Heckbert. Creating raster omnimax images from multiple perspective views using the ellipticalweighted average filter. *IEEE Computer Graphics & Applications*, 6(6) :21–27, June 1986.

[Gro98] J. P. Grossman. Point sample rendering. Master's thesis, Department of Electrical Engineering and Computer Science, MIT, August 1998.

[Her92] G. T. Herman. Discrete multidimensional jordan surfaces. *CVGIP : Graphical Modeling and Image Processing*, 54(6) :507–515, November 1992.

[KK89] James T. Kajiya and Timothy L. Kay. Rendering fur with three dimensional textures. In *Proceedings of SIGRAPH 89*, volume 23, pages 271–280, August 1989.

[KV01] Aravind Kalaih and Amitabh Varshney. Differential point rendering. In *Proceedings of 12th Eurographics Workshop on Rendering*, pages 139–150, June 2001.

[LL94] Philipe Lacroute and Marc Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. In *Proceedings of SIGRAPH'94*, pages 451–458, July 1994.

[LW85] Marc Levoy and Turner Whitted. The use of points as display primitive. Technical Report TR 85-022, 1985.

[MDG00] K. Maritaud, J.M. Dischler, and D. Ghazanferpour. Rendu réaliste d'arbres à courte distance. In *AFIG'00*, December 2000.

[MN98] Alexandre Meyer and Fabrice Neyret. Textures volumiques interactives. In *AFIG'98*, pages 261–270, Dec 1998.

[Ney95] Fabrice Neyret. A general and multiscale model for volumetric textures. In Wayne A. Davis and Przemyslaw Prusinkiewicz, editors, *Graphics Interface '95*, pages 83–91, May 1995.

[PZvBG00] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels : Surface elements as rendering primitives. In *Proceedings of SIGGRAPH 2000*, pages 335–342, July 2000.

[RL00] S. Rusinkiewicz and M. Levoy. Qsplat : A multiresolution point rendering system for large meshes. In *Proceedings of SIGGRAPH'2000*, pages 343–352, July 2000.

[RPZ02] L. Ren, H. Pfister, and M. Zwicker. Object space ewa surface splatting : A hardware accelerated approach to high quality point rendering. In *Proceedings of Eurographics 2002.*, 2002.

[SGS98] Jonathan Shade, Steven J. Gortler, and Richar Szeliski. Layered depth image. In *Proceedings of SIGRAPH 98*, pages 231–242, July 1998.

[WTRDHS87] Robert L. Cook William T. Reeves David H. Salesin. Rendering antialiased shadows with depth maps. *Computer Graphics (SIGGRAPH 87 Proceedings)*, 21(4) :283–291, July 1987.

[ZPvBG01] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. Surface splatting. In *Proceedings of SIGGRAPH 2001*, August 2001.

Écrasement de photons pour l'illumination globale

F. Lavignotte, M. Paulin

IRIT
Université Paul Sabatier
118, route de Narbonne
31062 Toulouse Cedex
lavignot,paulin@irit.fr

Résumé : *Dans cet article, une méthode basée image est présentée pour accélérer le calcul de l'éclairement global dans une scène virtuelle. L'éclairement est reconstruit après une phase de lancer de photons. Le principe est de reconstruire l'éclairement en écrasant les photons sur la surface à laquelle ils appartiennent. Cet écrasement est réalisé dans l'espace image. Par conséquent, la phase de reconstruction est indépendante de la complexité de la scène. Cette reconstruction par écrasement est accélérée par le matériel graphique ce qui permet d'avoir des temps de calcul réduit même pour un grand nombre de photons. Diverses solutions sont apportées pour que la reconstruction reste précise malgré les limites du matériel graphique. Enfin, nous présentons un lancer de rayon modifié pour prendre en compte notre méthode et qui permet de calculer efficacement une image de l'éclairement complet.*

Mots-clés : Illumination globale, Rendu basé image, Rendu accéléré

1 Introduction

Pour obtenir des images photo réalistes à partir de la description virtuelle d'une scène, il est nécessaire d'utiliser un modèle d'éclairage physiquement réaliste. De nombreux travaux ont porté sur ce domaine en commençant par les travaux de Goral et al. présentés dès 1984 [GTGB84], limités aux transferts diffus. Kajiya a posé les bases du modèle en proposant l'équation du rendu qui définit la luminance en un point [Kaj86]. Il a aussi proposé une méthode de résolution par des méthodes de type Monte Carlo qui consiste à échantillonner les chemins de transfert lumineux partant de l'oeil ou des lumières. Un échantillonnage adapté de ces chemins permet de prendre en compte de nombreux effets complexes de l'éclairage comme les caustiques, les ombres douces, les réflections sur une surface rugueuse, et l'éclairage diffus indirect [Laf96], [VG95] [VG97]. Cependant, les méthodes de type Monte Carlo, bien que très élégantes, ne sont pas encore sorties des applications de recherche du fait de leur très lente convergence et donc de la nécessité de tracer un grand nombre de chemins pour éliminer le bruit sur l'image.

Des méthodes multi-passes ont donc été proposées pour résoudre le problème d'efficacité des méthodes pures. L'équation du transfert lumineux est découpée alors en différentes parties, et chacune de ces parties est résolue avec différent algorithmes. L'éclairage direct et spéculaire est généralement pris en compte par un lancer de rayon classique, la radiosité peut être utilisée pour prendre en compte l'éclairage diffus indirect [CRMT91] ou le lancer de photons pour prendre en compte les caustiques [Hec90]. Le problème de ces travaux est de stocker leur contribution à l'éclairage sur des textures ou un maillage, et donc d'être dépendant de la complexité de la scène. D'autres travaux récents ont été proposés comme les cartes de photons [Jen96], ou les vecteurs d'éclairement [SP01] pour l'éclairage indirect. Ces travaux opèrent directement dans l'espace image et s'adapte donc mieux à des scènes complexes.

Dans cet article, nous proposons une méthode pour calculer directement une image de l'éclairement qui prend en compte l'éclairage diffus et les caustiques. Cette méthode est décomposé en deux passes, la première consiste en un lancer de photons, et la seconde en une reconstruction sur l'image de l'éclairement basé sur la technique d'écrasement de photons proposé initialement par Sturzlinger et Bastos [SB97]. Nous étendons cette technique à des scènes complexes et essayons de résoudre le problème de précision de la méthode limitée par les contraintes du matériel graphique. Dans un premier temps, nous posons le problème de manière à l'adapter au manque de précision des calculs quand on travaille directement sur le tampon de couleur. Puis, nous proposons une méthode pour réduire un artefact visuel important tel que le biais sur les bords des surfaces. Enfin, les différents détails de mise en oeuvre sur la génération actuelle de cartes graphiques sont présentés. Un des intérêts de cette méthode est de pouvoir gérer facilement un grand nombre de photons puisque l'accès aux photons est linéaire durant la reconstruction, on peut facilement le cacher sur le disque dur. L'autre intérêt est d'être de s'adapter plus facilement

aux scènes complexes. Finalement, nous présentons un algorithme basé sur un lancer de rayon adapté à l'utilisation de notre méthode de calcul de l'éclairement diffus basé image.

2 Écrasement de photons

2.1 Estimation de densité

Dans ce paragraphe, nous allons décrire rapidement le principe de reconstruction de l'éclairement à partir d'un ensemble de photons et de l'estimation de densité. La reconstruction est basée sur l'estimation de densité et cette théorie est utilisé dans de nombreux travaux : radiosité [Wal98], textures d'éclairement indirects [Mys97], textures de caustiques [Hec90] [GDW00], et cartes de photons [Jen96].

Le lancer de photons consiste à tracer des chemins lumineux aléatoires à partir des sources lumineuses. Le nom de photon est un peu trompeur puisque le but n'est pas de simuler le comportement corpusculaire de la lumière. Les photons sont en fait les sommets des chemins aléatoires générés depuis les sources lumineuses. Au final, une position ,un poids ou énergie et une direction sont associés à chaque photon.

Une fois le lancer de photons accompli, un ensemble de photons est obtenu. L'éclairement est alors reconstruit à partir de cet ensemble. Heckbert le premier a noté que c'était équivalent à un problème d'estimation de densité [Hec90]. En effet, la distribution des photons suit une densité de probabilité proportionnelle à l'éclairement :

$$p(x,\lambda) = \frac{E(x,\lambda)}{\int_S E(y,\lambda)dy} \tag{2.1}$$

L'intégration de l'éclairement peut être remplacée par $n\phi$ avec n le nombre total de photons et ϕ le poids transporté par un photon. Cette densité de probabilité est totalement inconnue, donc une estimation non paramétrique est utilisée comme l'estimateur de densité par noyau [Sil86].

La méthode d'estimation de densité par noyau construit une densité estimée $\hat{p}$ à partir d'un ensemble de n points $X_1, ..., X_n$ générés par p :

$$\hat{p}(x) = \frac{1}{nh^d}\sum_{i=1}^{n} K(\frac{x - X_i}{h}) \tag{2.2}$$

où d est la dimension de x, $K(x)$ est une fonction noyau unitaire, symétrique et généralement égale à zéro si $|x| > 1$, h est le paramètre de lissage.

Une propriété très intéressante de l'estimateur à noyau est le compromis entre la variance et le biais contrôlé par le paramètre de lissage. En effet, le biais est proportionnel à h^2 et la variance proportionnelle à nh^{-1} [Sil86].

Pour notre problème, les équations (2.1) et (2.2) sont combinés pour nous donner un estimateur pour l'éclairement à chaque point :

$$E(x,\lambda) = \frac{\phi}{h^2}\sum_{i=1}^{n} K(\frac{x - X_i}{h}) \tag{2.3}$$

2.2 Principe

Notre but est de reconstruire efficacement l'éclairement à chaque pixel en utilisant l'estimateur (2.3). Cet estimateur peut être vu comme la somme de la contribution de chaque photon X, la contribution d'un photon étant égale à :

$$c(X) = \frac{\phi}{h^2}K(\frac{x - X}{h}) \tag{2.4}$$

En fait, la contribution d'un photon est nulle si sa distance au point d'estimation est supérieure à h. La contribution non nulle d'un photon aux points d'une surface forme alors un disque de rayon h centré sur sa position X. L'estimation dans l'espace image revient donc à ajouter la contribution du photon à chaque pixel recouvert par la projection de son disque sur l'image. Cette projection de la contribution du photon est exacte si la surface est plane, pour une surface quelconque cela reste une approximation. Dans ce cas, la contribution est en quelque sorte

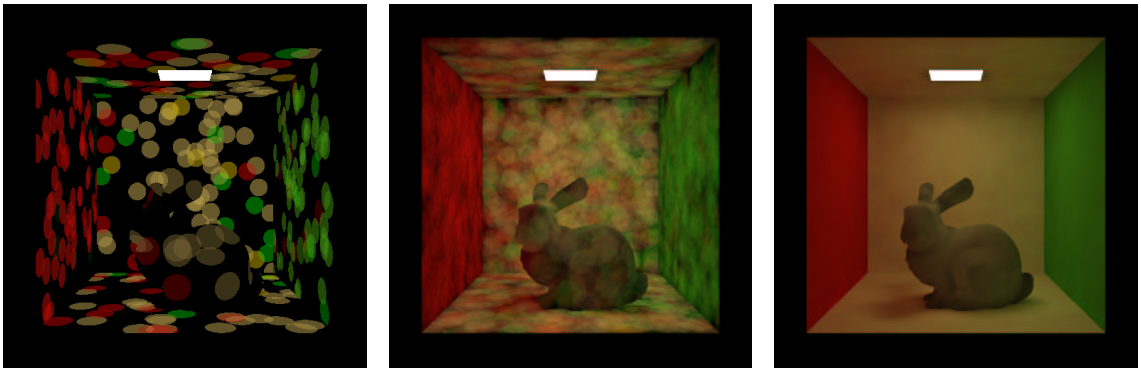


FIG. 1 – Écrasements de photons à 0.05% (Gauche), 1% (Milieu) et 100% (Droite).

écrasée sur la surface. Cette approximation est correcte si la courbure de la surface varie peu et si la taille du disque est peu importante, un exemple d'écrasement de photons, arrêté à différentes étapes, est montré sur la figure 1.

De plus, dans une scène avec plusieurs surfaces, la contribution du photon ne doit pas être ajouté à tous les pixels recouvert par le disque. En effet, la surface visible d'un pixel peut appartenir à une surface différente de la surface d'intersection du photon, donc la contribution est ajouté seulement si la surface visible du pixel correspond à la surface intersectée par le photon.

Le fait d'ajouter la contribution du photon au pixel revient donc à projeter un disque sur l'image, cela correspond à un processus de rastérisation et peut être effectué avec une libraire graphique comme OpenGL. La contribution d'un photon est alors rendue avec un quadrilatère texturé et coloré. La texture correspond à la fonction noyau discrétisée (qui a la forme d'un disque) et la couleur est égale au poids du photon divisé par h^2. Le résultat de leur modulation est alors égale à la contribution du photon. Le quadrilatère est rendu avec le mélange activé pour ajouter la contribution à chaque pixel recouvert par le quadrilatère. Il faut aussi éviter d'ajouter cette contribution aux pixels n'ayant pas la même surface visible, la mise en oeuvre de ce processus est décrite dans le paragraphe (3.2).

2.3 Précision des calculs

Le principe de l'écrasement de photons est assez simple mais une mise en oeuvre efficace sur une carte graphique accélérée pose un certain nombre de problèmes. Le tampon de couleur est utilisé pour ajouter la contribution des photons. Malheureusement, la précision du tampon de couleur est limité à huit bits par composante sur les cartes graphiques courantes. Les précédents travaux [SB97] travaillaient avec un tampon de couleur douze bits et des problèmes de quantification étaient présents sur les images. En effet, l'écrasement de photons nécessite une précision très importante, le nombre de contributions par pixel est de l'ordre de cinq cents à mille sur les scènes que nous avons testé.

Pour pouvoir utiliser des tampons avec une précision limitée un certain nombre de choix doivent être fait. Tout d'abord, nous allons choisir un noyau constant $K(x) = \frac{1}{\pi}$. Cette limitation n'est pas aussi réductrice qu'il n'y parait. Il a été démontré [Sil86] que l'efficacité de ce noyau par rapport au noyau optimal est de 0.93. C'est à dire que pour avoir une estimation de qualité comparable avec n points et le noyau optimal, il faut $n/0.93$ points avec un noyau constant.

Si on considère que le paramètre de lissage h et le poids ou l'énergie ϕ de chaque photon est constant, il nous suffit pour estimer l'éclairement du nombre de contributions N_{ij} en chaque pixel (i, j). L'éclairement est donc calculé par :

$$E(x_{ij}) = N_{ij} \frac{\phi}{\pi h^2} \tag{2.5}$$

Cette formule exige que le poids et le paramètre de lissage soit constant par photon. En fait puisque la reconstruction est effectuée par surface, le paramètre de lissage est choisi constant par surface. Il est choisi avec cette

FIG. 2 – Comparaison de deux images rendues sans réduction du biais sur les bords (Gauche) et avec (Droite) Noter les régions plus sombres sur les bords et les petites surfaces

heuristique souvent utilisé [SWH+95] [SB97] :

$$h = C\sqrt{\frac{A}{N}}$$

où A est l'aire de la surface, N le nombre de photons sur la surface et C une constante définie par l'utilisateur généralement comprise entre 20 et 30.

2.4 Biais sur les bords

L'estimateur de densité par noyau assume que le support de la probabilité de densité est infini. Dans notre cas, le support correspond à une surface qui peut être fermée. Du coup, des fuites d'énergies apparaissent du fait que l'on sous estime l'éclairement sur les bords des surfaces ouvertes. Cette sous estimation est aussi très apparente sur les surfaces dont l'aire est inférieure au support du noyau, voir Figure 2. Ce problème peut être résolu en divisant l'estimateur en x par $A_x/\pi h^2$ où A_x est l'aire de l'intersection entre le support de l'estimateur en x et la surface visible en x. Le support de l'estimateur est un disque de rayon h centré en x. La surface est projetée sur ce disque et l'aire d'intersection est alors calculée.

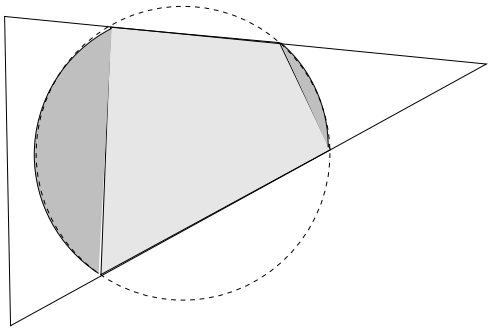


FIG. 3 – Calcul de l'aire d'intersection entre un triangle et un disque

Nous allons présenter ici le calcul de l'aire de la zone d'intersection entre en triangle et un disque. Le triangle est clippé par rapport au cercle. Le résultat est un polygone à n sommets. Si $n = 0$, soit le triangle est en dehors du cercle, soit il englobe le cercle. On vérifie alors si le centre du disque est dans le triangle, si c'est le cas alors l'aire d'intersection est égale à l'aire du disque, sinon elle est égale à zéro. Si $n = 2$, l'aire est alors l'aire d'intersection du demi plan formé par le segment. Si $n > 2$, l'aire d'intersection est alors l'aire de ce polygone (en gris clair sur la Figure 3) plus l'aire d'intersection des demi plans formés par les nouveaux segments (en gris foncé sur la

Figure 3). On appelle nouveaux segments les segments qui ne sont pas colinéaire aux trois segments initiaux du triangle. Ces nouveaux segments définissent chacun un demi plan.

A partir de l'aire d'intersection entre un triangle et un disque, on peut calculer l'aire d'intersection pour une surface représentée par un maillage triangulaire connecté. Il suffit de récupérer le triangle qui contient le point d'estimation, dans notre cas les pixels. L'aire d'intersection avec ce triangle est alors calculée, et en même temps pour chaque arête du triangle on calcule son intersection avec le disque. Il suffit après d'ajouter l'aire d'intersection des triangles qui partagent les arêtes qui intersectent le disque. Pour éviter les cycles sur le maillage, il faut marquer les triangles déjà visités.

Un autre intérêt de ce calcul est d'obtenir une meilleure approximation pour les surfaces non planes. En effet, le fait de représenter la contribution d'un photon reste une approximation pour ce genre de surface. Ainsi, l'aire du disque projeté sur la surface ne correspond pas réellement à πh^2. En calculant précisément l'aire couverte par le noyau pour réduire le biais sur les bords, l'erreur est réduite pour les surfaces non planes.

3 Mise en oeuvre

3.1 Poids constant

Pour l'instant, nous avons considéré que le poids de chaque photon est constant. Pour pouvoir réaliser cela, plusieurs conditions doivent être satisfaites. Tout d'abord, nous allons rappeler les équations qui permettent de calculer le poids ϕ_λ d'un photon généré d'une source lumineuse dans une direction ω pour une longueur d'onde donné :

$$\phi_\lambda = \frac{L_e(x,\omega,\lambda)\cos\theta}{Np_e(x,\omega)} \tag{3.1}$$

avec p_e la fonction de densité de probabilité utilisée pour sélectionner x et ω. Cette fonction doit être choisi égale à $\frac{L_e(x,\omega,\lambda)\cos\theta}{\Phi_\lambda^T}$, avec Φ_λ^T la puissance totale des sources lumineuses pour la longueur d'onde λ. Ainsi, le poids de chaque photon généré d'une source lumineuse est constant.

Puis, le photon est tracé dans la scène et à chaque intersection avec une surface, le photon est soit réfléchi, soit absorbé. S'il est réfléchi, son nouveau poids ϕ'_λ est :

$$\phi'_\lambda = \frac{f(y,\omega,\lambda)\cos\theta'}{p_r(\omega')}\phi_\lambda$$

avec f la FDRB en y, ω la direction du photon, ω' la nouvelle direction du photon sélectionnée avec la densité de probabilité p_r et θ' l'angle entre ω et ω'. Si f est réversible, p_r peut être choisi proportionnelle à $f(y,\omega,\lambda)\cos\theta'$ et le poids du photon reste constant après réflection. Il existe des modèles de FDRB réversible tel que le modèle de Phong modifié [LW94] ou le modèle de Ward [War92]. Donc en utilisant ces modèles de FDRB, un poids constant par longueur d'onde est obtenu pour chaque photon.

Il est donc possible d'obtenir un poids constant pour chaque longueur d'onde. Dans notre cas, nous travaillons en tri-chromatique RVB, il nous faudra alors générer les photons et reconstruire l'éclairement pour les trois composantes. Le fait de générer les photons de manière totalement indépendante pour chaque composante pose des problèmes. En effet, les trois composantes sont corrélées et donc les séparer artificiellement entraîne une plus grande variance.

La méthode classique est donc de choisir d'abord une composante, puis une source lumineuse par rapport à cette composante etc... La méthode que nous avons mise en oeuvre est différente, chaque photon va transporter un triplé au lieu d'un seul poids. Pour réaliser cela, une source lumineuse est choisie pour chaque composante mais à partir de la même variable aléatoire. Deux ou trois des sources lumineuses sélectionnées peuvent être identiques. Le photon sera alors lancé des sources lumineuses avec un triplé $(c0, c1, c2)$, les ci pouvant prendre comme valeur 0 ou 1. Si par exemple, le tirage des sources lumineuses nous a donné trois fois la même source lumineuse, le photon sera tracé de cette source avec le triplé (1,1,1). Évidemment, les chances d'avoir des sources lumineuses identiques est d'autant plus grande que la distribution d'émission des sources est identique. Après avoir rencontré une surface, le photon va être absorbé ou réfléchi selon la valeur de la FDRB en ce point. La probabilité d'absorption ou de réflexion est faite pour chaque composante mais toujours avec la même variable aléatoire, le triplé étant modifié

selon le résultat du tirage. Donc, au final, le poids du photon est représenté par un triplé binaire, pour avoir son poids il suffit multiplier ce masque avec le poids constant ϕ. Cette solution permet d'améliorer grandement la qualité de l'image et l'efficacité de la méthode sur les scènes testées où il est vrai les types de luminaires différents sont peu nombreux.

3.2 Rendu accéléré

Le rendu du photon doit être restreint à la partie visible de sa surface. Nous présentons ici la mise en oeuvre de ce comportement avec la librairie graphique OpenGL.

Tout d'abord, il nous faut obtenir un tampon d'identificateur, c'est à dire avoir pour chaque pixel de l'image un identificateur de la surface visible en ce pixel. Cela peut être obtenu en rendant chaque surface de la scène avec une couleur égale à son identificateur et en activant l'élimination des parties cachées avec le tampon de profondeur.

Une fois obtenu le tampon d'identificateur, il faut maintenant pouvoir réaliser une opération de comparaison entre l'identificateur de surface du photon et celui des pixels. Cette opération doit se dérouler lors des opérations sur les fragments, c'est à dire par pixel. En effet quand OpenGL rastérise une primitive géométrique, celle ci est décomposé en fragments. Un fragment correspond à un pixel recouvert par la projection de la primitive sur l'image. Une série d'opérations est réalisée sur ces fragments avant d'arriver au tampon de couleur. En particulier, le test de stencil peut être utilisé pour comparer une valeur référence avec la valeur stencil du pixel. Malheureusement, le tampon de stencil est limité à huit bits et permet donc d'effectuer une comparaison pour seulement 255 identificateurs de surfaces différents.

Une autre possibilité consiste à utiliser les opérations programmables sur les fragments. Différentes instructions sont fournies aux programmeurs pour modifier la couleur RVBA du fragment à partir de différentes entrées, telles que la couleur primaire interpolée ou la couleur correspondant à une unité de texture. En particulier, nous allons modifier la composante alpha pour utiliser le test alpha pour restreindre le rendu à la surface visible. Notre but est donc de générer un alpha égal à 1 si la surface visible et la surface du photon sont égale, et une valeur différente de 1 sinon.

Pour réaliser cela, il faut en entrée du programme sur les fragments l'identificateur de surface visible des pixels. Le tampon d'identificateur est donc chargé comme une texture. On veut pour chaque fragment l'identificateur du pixel correspondant, il faut donc que les coordonnées du texel soit les mêmes que les coordonnées du pixel. Pour cela, les coordonnées de texture pour chaque sommet du quadrilatère sont les mêmes que la position de ces sommets, et la matrice de transformation du repère scène vers l'image est chargée comme matrice de texture. En OpenGL, le repère image est défini entre -1 et 1, il faut donc appliquer une dernière transformation pour passer au repère de texture entre 0 et 1. L'identificateur de surface du photon est passé comme couleur des sommets. An niveau, du programme de fragments, on a donc en entrée l'identificateur de photon et du pixel. La comparaison de ces deux valeurs se fait en utilisant des instructions conditionnelles qui permettent de modifier la valeur d'un registre par rapport à un test sur un autre registre.

3.3 Dépassement de capacité

Le problème de reconstruction a été simplifié de manière à se ramener à un simple comptage de la projection des photons. Un problème se pose encore, ce nombre peut dépasser la précision du tampon de couleur. En effet, le tampon de couleur est limité sur les cartes graphiques courantes à huit bits par composante. Une première solution est d'ajouter périodiquement le contenu du tampon de couleur à un tampon plus précis. En OpenGL, le tampon d'accumulation permet effectivement de réaliser cela. Si le tampon d'accumulation n'est pas accéléré, il est toujours possible d'émuler son fonctionnement en lisant le tampon de couleur et en ajoutant son contenu à un tableau géré par l'application. Le problème reste à savoir quand il est nécessaire d'ajouter le contenu de tampon de couleur, et il n'est pas évident de définir une heuristique pour réaliser cela.

Une deuxième solution reste possible, utiliser le test et les opérations sur le stencil pour avoir au final 16 bits de précision. En OpenGL, le tampon de stencil est modifié par des opérations qui dépendent du résultat du test de stencil. Il existe différentes opérations et plus particulièrement une opération qui incrémente de un la valeur dans le tampon de stencil, et une autre qui met à zéro cette valeur. Ces opérations peuvent donc être configuré de manière à faire fonctionner les opérations de stencil comme un compteur des 8 bits de poids faible avec les 8 bits de poids fort dans le tampon de couleur :

– Test de stencil : la valeur du tampon de stencil doit être égal à 255
– Si le test rate : on incrémente la valeur du stencil
– Si le test réussit : on met à zéro la valeur

Si le test réussit, le fragment arrive jusqu'au tampon de couleur. Le tampon de couleur est alors incrémenté de 1, puisque le mélange est activé et que la couleur du fragment est 1. Au final après avoir rendu tous les photons, il suffit de récupérer dans deux tableaux le tampon de stencil et le tampon de couleur pour reconstruire le nombre de contributions à ce pixel. Le désavantage de cette méthode est de devoir rendre les photons séparément. En effet, il n'est plus possible de compter les trois composantes en même temps, seulement une valeur peut être compté. En utilisant la technique présentée dans le paragraphe 3.1, nous comptons d'abord les photons avec un triplé (1,1,1), puis (1,1,0), etc.. Le nombre de contributions est alors ajouté en prenant compte le triplé.

4 Lancer de rayon basé image

Une image de l'éclairement diffus est obtenue par la méthode présentée précédemment. A partir de l'éclairement, la luminance diffuse peut être obtenu facilement en combinant avec la partie diffuse de la fdrb. En utilisant la notation d'Heckbert [Hec90], l'image obtenu représente donc les chemins de type $L(D|S)^*DE$. Pour calculer tous les chemins lumineux possibles, nous devons ajouter à notre méthode un algorithme qui calcule les chemins $L(D|S)^*SE$.

En fait, nous allons calculé la luminance en trois parties :

1. L'éclairage direct $L(D|S)E$ est calculé avec un lancer de rayon stochastique [SWZ96]

2. L'éclairage indirect diffus $L(D|S)^+DE$ est calculé avec notre méthode d'écrasement de photons, en n'écrasant que les photons qui ont au moins un rebond.

3. L'éclairage indirect spéculaire $L(D|S)^+SE$ est calculé en échantillonnant les chemins de type $L(D|S)^*DS^+E$ avec un lancer de rayon basé image.

Les chemins de type $L(D|S)^*DS^+E$ sont échantillonnés en traçant un chemin spéculaire à partir de l'oeil S^+E, à chaque intersection avec une surface on récupère l'éclairement diffus $L(D|S)^*D$. L'éclairement diffus est calculé en écrasant les photons. Le problème est que nous avons besoin dans ce cas de l'éclairage diffus en divers points de la scène. Notre méthode ne peut calculer qu'une image ce qui ne couvre pas tous les points de la scène.

Nous avons décidé d'utiliser des images de profondeur multi couches communément appelées LDI [SGHS98] pour stocker l'éclairement. Un pixel d'une LDI est une liste d'échantillons de la scène correspondant aux point d'intersection des surfaces rencontrés par un rayon tracé à partir du centre du pixel. Comme dans [LR], trois LDI sont utilisées correspondant à une vue orthographique des trois directions orthogonales de la scène. Cette représentation est complète dans le sens que nous sommes assurés d'avoir un échantillon pour chaque surface de la scène quelque soit son orientation à condition que la taille de la surface soit supérieure à la résolution spatiale des LDI.

L'objectif est d'utiliser l'écrasement de photons pour reconstruire l'éclairement dans une LDI. Une première méthode est tout simplement d'écraser les photons pour chaque couche des LDI. En effet, une couche de LDI représente une image. Malheureusement, cette méthode n'est pas du tout efficace pour des scènes complexes. Il peut en effet y avoir jusqu'à une cinquantaine de couches de profondeur. De plus, ces couches consistent parfois en très peu de pixels, l'image résultante est donc pleine de trous.

Une méthode différente est donc proposée. Dans un premier temps, les LDI sont construites mais sans calculer l'éclairement, avec uniquement les informations géométriques telles que la profondeur. On marque tous les pixels du LDI comme étant non initialisé au niveau de l'éclairement. Lorsque le lancer de rayon interroge les LDI pour récupérer une valeur en un point de l'éclairement indirect, on regarde si les pixels correspondants aux points sont initialisés. Si ce n'est pas le cas, une image d'éclairement est calculé dans la direction incidente de l'interrogation. Cette image est alors reprojeté sur les LDI pour remplir la valeur d'éclairement des pixels du LDI. Ainsi, la cohérence spatiale des directions spéculaires est pris en compte et permet de diminuer grandement le coût par rapport à un calcul direct du LDI.

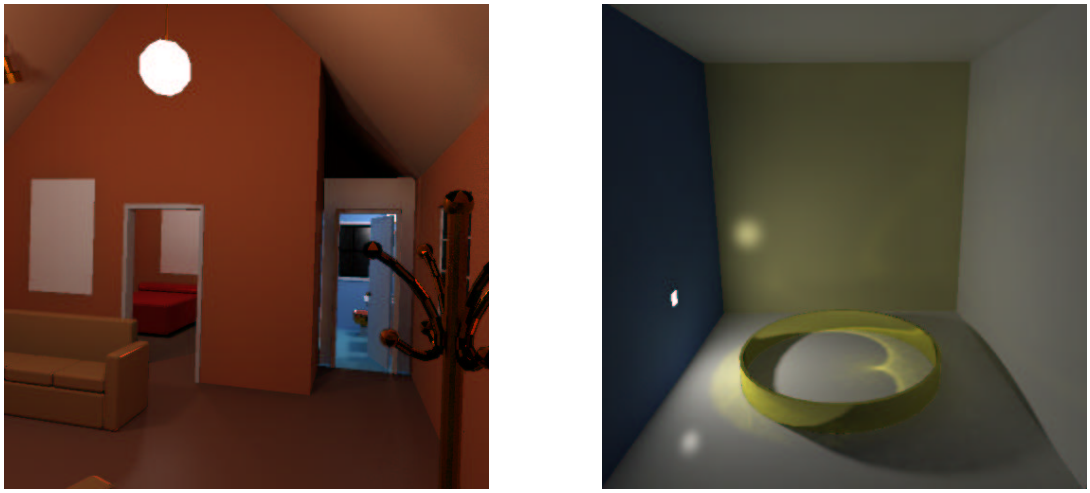


FIG. 4 – Ecrasement de photons combinés avec le lancer de rayon. Noter la réflection de la caustique sur l'anneau.

5 Résultats

Dans ce paragraphe, quelques résultats avec différentes scènes MGF sont présentées. Tous les résultats ont été obtenues sous Linux sur un AtlhonXP 1,6 GHz avec 512 Mo de RAM et une carte graphique GeForce4 Ti 4600.

- *Caustics* : une scène très simple avec des caustiques
- *Conference* : une salle de conférence, géométriquement complexe (environ 100 000 triangles) et avec un grand nombre de sources lumineuses étendues.
- *Cabin* : un chalet avec plusieurs pièces (environ 45 000 triangles)

Scène	LP	NP	EP1	EP2	EP3	RI
Caustics	7.02s	1830901	1.14s	4.44s	9.5s	0.6s
Conference	41.89s	1601516	0.96s	3.25s	6.61s	1.5s
Cabin	22.27s	1882184	0.75s	2.15s	4.52s	1.2s

TAB. 1 – Résultats sur trois scènes, LP : lancer de photons, NP : nombre total de photons, EP1, EP2, EP3 : écrasement de photons respectivement pour une image 256x256, 512x512 et 1024x1024 , RI : reconstruction de l'image 512x512

Le tableau 1 présente les temps de reconstruction de l'éclairement $L(D|S)^*DE$, pour un million de photons générés des sources lumineuses. Plusieurs observations peuvent être faites sur ces résulats. Tout d'abord, la phase de reconstruction d'image correspond à construction de l'irradiance à partir de la formule 2.5, et surtout de la correction du biais sur les bords. C'est cette correction qui prend le plus de temps puisque cela implique des calculs géométriques pour chaque pixel de l'image. En effet, la cohérence spatiale au niveau de l'image n'est actuellement pas pris en compte, c'est pour cette raison que les temps n'ont été montrés que pour une taille d'image puisque la complexité de cette phase varie linéairement avec le nombre de pixels. Enfin, la phase d'écrasement de photons est indépendante de la complexité de la scène et dépendante de la taille de l'image ce qui est un résultat tout à fait logique. Les différences au niveau des trois scènes s'expliquent par le fait que sur la scène simple l'image engloble toute la scène, donc tous les photons sont effectivement écrasés ce qui accroit le travail de la carte graphique. Globalement, la complexité de la phase d'écrasement de photons est proportionnelle aux nombres de photons fois la taille moyenne sur l'image de leur écrasement.

Ainsi sur le tableau 2, nous avons représenté les différents temps pour cent mille et un million de photons sur la même scène mais avec différentes valeurs de la constante C qui contrôle le paramêtre de lissage (voir paragraphe 2.3) donc la taille de l'écrasement des photons. En fait, la constante C est choisie généralement entre 20 et 30, en dessous l'image est trop bruitée et au dessus le résultat est trop lisse.

Enfin, dans le tableau 3, nous présentons quelques résultats pour le lancer de rayon basé image. La taille des LDIs est de 256x256 pour les trois scènes. Les LDI sont construites en utilisant un tracé de rayon modifié. La première

Constante	15	30	60
Temps (1e5)	0.77s	2.34s	8.47s
Temps (1e6)	2.04s	4.67s	10.65s

TAB. 2 – Temps pour différentes valeurs de la constante C

scène a été calculé en lancer de rayon avec 9 rayons pour les ombres et les réflections, les deux autres scènes avec 36 rayons pour les ombres et les réflections.

Scène	EP	EL	LR
Caustics	10.8s	18s	21.7s
Conference	39s	35s	95.8s
Cabin	23.76s	83s	325.73s

TAB. 3 – Résultat sur trois scènes avec le lancer de rayon basé image, EP : lancer de photons et écrasement de photons pour l'image de l'irradiance indirect, EL : écrasement de photons pour remplir le LDI, LR : lancer de rayon

6 Conclusion

Nous venons de présenter une nouvelle méthode basée image pour l'éclairement global. Le principal intérêt de notre méthode est de tirer parti de la puissance des cartes graphiques sans sacrifier la précision de la simulation. Pour l'instant, nous sommes encore partiellement dépendant de la complexité de la scène pour la résolution du biais sur les bords. Mais avec les nouvelles possibilités des prochaines cartes graphiques, une méthode qui travaille entièrement dans l'espace image est envisageable et souhaitable. Il sera aussi important de tirer parti du gain en flexibilité pour améliorer la qualité de la reconstruction en utilisant des méthodes adaptatives d'estimation de densité. Un autre point à approfondir est l'utilisation de LDI. Pour l'instant , nous nous servons de LDI uniquement pour stocker l'éclairement dans la scène. Il est aussi possible de tracer un rayon au sein d'une LDI pour récupérer un échantillon de la scène correspondant à l'intersection avec ce rayon. Cela pourrait accélérer le calcul des réflections brillantes qui ont besoin de moins de précision que les réflections de type mirroir.

Références

[CRMT91] Shenchang Eric Chen, Holly E. Rushmeier, Gavin Miller, and Douglass Turner. A Progressive Multi-Pass Method for Global Illumination. In *Computer Graphics (ACM SIGGRAPH '91 Proceedings)*, volume 25, pages 164–174, July 1991.

[GDW00] Xavier Granier, George Drettakis, and Bruce Walter. Fast global illumination including specular effects. In B. Peroche and H. Rushmeier, editors, *Rendering Techniques 2000 (Proceedings of the Eleventh Eurographics Workshop on Rendering)*, pages 47–58, New York, NY, 2000. Springer Wien.

[GTGB84] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modelling the Interaction of Light Between Diffuse Surfaces. In *Computer Graphics (ACM SIGGRAPH '84 Proceedings)*, volume 18, pages 212–222, July 1984.

[Hec90] Paul Heckbert. Adaptive Radiosity Textures for Bidirectional Ray Tracing. In *Computer Graphics (ACM SIGGRAPH '90 Proceedings)*, volume 24, pages 145–154, August 1990.

[Jen96] Henrik Wann Jensen. Global Illumination Using Photon Maps. In *Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering)*, pages 21–30, New York, NY, 1996. Springer-Verlag/Wien.

[Kaj86] James T. Kajiya. The Rendering Equation. In *Computer Graphics (ACM SIGGRAPH '86 Proceedings)*, volume 20, pages 143–150, August 1986.

[Laf96] Eric Lafortune. *Mathematical Models and Monte Carlo Algorithms for Physically Based Rendering*. Ph.D. thesis, Leuven, Belgium, February 1996.

[LR] Dani Lischinski and Ari Rappoport. Image-based rendering for non-diffuse synthetic scenes. In *Rendering Techniques '98 (Proceedings of the Ninth Eurographics Workshop on Rendering)*, pages 301–314.

[LW94] Eric P. Lafortune and Yves D. Willems. Using the Modified Phong BRDF for Physically Based Rendering. Technical Report CW197, Department of Computer Science, Katholieke Universiteit Leuven, Leuven, Belgium, November 1994.

[Mys97] Karol Myszkowski. Lighting reconstruction using fast and adaptive density estimation techniques. In Julie Dorsey and Philipp Slusallek, editors, *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)*, pages 251–262, New York, NY, 1997. Springer Wien. ISBN 3-211-83001-4.

[SB97] Wolfgang Sturzlinger and Rui Bastos. Interactive rendering of globally illuminated glossy scenes. In Julie Dorsey and Philipp Slusallek, editors, *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)*, pages 93–102, New York, NY, 1997. Springer Wien. ISBN 3-211-83001-4.

[SGHS98] Jonathan W. Shade, Steven J. Gortler, Li-Wei He, and Richard Szeliski. Layered depth images. *Computer Graphics*, 32(Annual Conference Series) :231–242, 1998.

[Sil86] B.W. Silverman. *Density estimation for statistics and data analysis*. Chapman and Hall, New York NY, 1986.

[SP01] X. Serpaggi and B. Peroche. An adaptive method for indirect illumination using light vectors. In *Computer Graphics Forum (Proceedings of Eurographics 2001)*, volume 20, pages C–278–C–287, September 2001.

[SWH+95] Peter Shirley, Bretton Wade, Philip M. Hubbard, David Zareski, Bruce Walter, and Donald P. Greenberg. Global Illumination via Density Estimation. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 219–230, New York, NY, 1995. Springer-Verlag.

[SWZ96] Peter Shirley, Changyaw Wang, and Kurt Zimmerman. Monte Carlo Techniques for Direct Lighting Calculations. *ACM Transactions on Graphics*, 15(1) :1–36, January 1996.

[VG95] Eric Veach and Leonidas J. Guibas. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Computer Graphics Proceedings, Annual Conference Series, 1995 (ACM SIGGRAPH '95 Proceedings)*, pages 419–428, 1995.

[VG97] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *Computer Graphics (ACM SIGGRAPH '97 Proceedings)*, volume 31, pages 65–76, 1997.

[Wal98] Bruce Johnathan Walter. *Density Estimation Techniques for Global Illumination*. PhD thesis, Program of Computer Graphics, Cornell University, Ithaca, NY, August 1998.

[War92] Gregory J. Ward. Measuring and Modeling Anisotropic Reflection. In *Computer Graphics (ACM SIGGRAPH '92 Proceedings)*, volume 26, pages 265–272, July 1992.

Optimisation à base de flot de graphe pour l'acquisition d'informations 3D à partir de séquences d'images

Sylvain Paris, François Sillion

*i*MAGIS - GRAVIR / IMAG - INRIA †
655, avenue de l'Europe, Montbonnot 38334 Saint Ismier CEDEX

{sylvain.paris|francois.sillion}@imag.fr

Résumé : *On se propose de montrer comment on peut appliquer la technique de "flot de graphe" pour résoudre certains problèmes complexes d'optimisation. Cette technique est utilisée pour l'acquisition d'informations 3D à partir de plusieurs points de vue. On montre notamment comment – grâce à un graphe particuliers – il est possible de minimiser une famille d'énergies qui permettent d'atteindre des résultats plus précis que les cartes de disparité obtenues jusqu'à présent sur ce type de problème. Cette nouvelle approche offre aussi la possibilité de prendre en compte les discontinuités des objets avec lesquels on souhaite travailler. Les résultats ainsi obtenus sont exposés.*

Mots-clés : Flot de graphe, coupure de graphe, optimisation, reconstruction 3D

1 Introduction

Nous nous sommes intéressés au problème de l'acquisition d'informations tridimensionnelles à partir d'une courte séquence d'images. Notre objectif est de "remplir" des environnements virtuels auxquels manquent la plupart du temps les petits détails essentiels au réalisme comme une boîte aux lettres, un banc public ou des passants. Ce sont ces objets dont nous souhaitons obtenir un modèle à l'aide de la séquence.

Au cours de notre algorithme, nous devons déterminer la position de la surface des objets de manière à ce que cette surface soit cohérente par rapport aux images de la séquence tout en présentant une certaine régularité car la plupart des objets ont une surface lisse – au moins localement. Ce type de compromis se représente classiquement sous forme d'une fonction d'énergie qui contient un terme en rapport avec la régularité et un terme en rapport avec la cohérence. Pour le terme de régularité nous avons choisi une pénalité sur les dérivées de la surface et nous obtenons le terme de cohérence des étapes précédentes de notre algorithme qui est décrit dans [PS02]. Or il se trouve que ce terme a a priori très peu de propriétés mathématiques qui peuvent servir de base aux techniques classiques d'optimisation. Les techniques à base de flot de graphe sont alors apparues comme une solution à ce problème. Nous allons donc présenter une technique d'optimisation adaptée à une famille d'énergies qui englobe notre problème.

Dans la section 2 nous allons observer les principaux travaux existants sur le problème de la reconstruction 3D pour examiner quelle technique d'optimisation est employée en nous attardant sur les techniques à base de flot de graphe. Ensuite dans la section 3 nous introduirons la notion de flot de graphe. Nous continuerons dans la section 4 en présentant comment cette notion peut être utile pour résoudre des problèmes d'optimisation. Dans la section 5 nous étudierons l'application de cette technique à notre problème d'acquisition d'informations 3D. Nous finirons par la section 6 où l'on présente quelques résultats et par la section 7 qui conclue cet article.

2 Travaux existants

Toute une famille de méthodes de reconstruction est basée sur la discrétisation de l'espace en voxels ; la plus connue est le *Space Carving* introduit par Kutulakos et Seitz [KS99] dont de nombreuses variantes ont été dérivées. Pour plus de détails, on pourra lire le tour d'horizon proposé par Slabaugh et al. [SCMS01] qui regroupe la plupart d'entre elles. Pour toutes ces méthodes, l'évaluation de la présence ou non d'un voxel se fait uniquement selon la cohérence par rapport aux images, aucune contrainte de régularité n'est prise en compte il n'y a donc aucune

† i*MAGIS* est un projet commun CNRS, INPG, INRIA, UJF.

optimisation d'effectuée. Ces techniques sont donc très sensibles à la qualité des images initiales et à l'ambiguité résultant du faible angle de vue dont nous disposons dans nos courtes séquences d'images.

On peut ensuite citer les techniques travaillant le long des lignes épipolaires pour créer des cartes de disparité [OK93, OK85, IB94, UKG98]. Ces méthodes introduisent naturellement des contraintes de régularité le long de des lignes épipolaires et obtiennent ainsi des résulats même pour un faible angle de vue. Néanmoins elles présentent deux restrictions principales : il est difficile d'introduire une régularité entre deux lignes épipolaires et les cartes de disparité obtenues ont une faible précision en profondeur, les objets reconstruits sont généralement plats. Koch et al. [KPV98] améliorent la précision en utilisant un filtre de Kalman mais n'introduisent toujours pas de régularité entre lignes épipolaires.

Faugeras et Keriven [FK98] proposent une méthode à base de courbes de niveaux qui fait évoluer une surface dans l'espace de manière à s'approcher de plus en plus de la surface de l'objet à reconstruire. Grâce aux courbes de niveaux, cette technique optimise un compromis entre la cohérence par rapport aux images et la régularité de la surface finale. Néanmoins, elle nécessite des images de très bonne qualité et des points de vue très distants pour obtenir des résultats satisfaisants.

Viennent ensuite les méthodes à base de flots de graphes. Roy et Cox [RC98] ont introduit leur technique comme une généralisation des méthodes travaillant sur les lignes épipolaires pour construire des cartes de disparités. Leur méthode montre qu'un flot de graphe permet d'étendre le processus d'optimisation à toute l'image. Ils constatent aussi que leur processus a des propriétés de régularisation similaires aux techniques à base de lignes épipolaires. Cette technique a été par la suite formalisée par Veksler [Vek99] puis Kolmogorov et Zabih [KZ01, KZ02a, KZ02b] sous la forme d'un problème de labellisation visant à construire une carte de disparité. Chaque label est une valeur de disparité possible. Un label est associé à chaque pixel de manière à minimiser une énergie qui tient compte du voisinage du pixel pour avoir un résultat final régulier. Ces méthodes ont deux limitations, la première est qu'elles construisent des cartes de disparité qui ont toujours le défaut d'aplatir les objets ; la seconde est que, comme la fonction de pénalité entre deux voisins de labels différents n'est pas nécessairement convexe, la minimisation de l'énergie est un problème NP-complet dont on obtient finalement qu'une approximation. Par contre, Ishikawa [Ish00] propose l'étude du cas où cette fonction est convexe et décrit un graphe qui permet d'obtenir le résultat exact. Toutefois la méthode de stéréovision qu'il expose ne peut pas se généraliser aisément à une séquence d'images et n'utilise qu'une pénalité linéaire. Une méthode pour utiliser trois caméras a été proposée par Buehler et al.[BGCM02] mais celle-ci ne s'étend pas non plus à une séquence d'images.

Contribution

La technique que nous allons décrire se propose de formuler le problème d'optimisation sous une forme nouvelle qui exhibe le rôle de la configuration géométrique et d'y adapter une méthode de flot de graphe qui tire partie des fonctions de pénalité convexes. Ainsi il sera possible de travailler à partir de séquences d'images et d'obtenir des modèles plus précis que les cartes de disparités produites jusqu'à présent.

3 Flot de graphe

Le problème du flot de graphe est un problème classique d'algorithmique. Initialement, il s'agit de la formulation d'un problème simple d'écoulement d'eau dans un réseau de tuyaux. On présente dans un premier temps ce problème pour donner l'intuition de ce que représente la formulation théorique qui suit.

3.1 Écoulement d'eau dans un réseau

Le problème que l'on se pose est le suivant. Étant donnés une source d'eau de débit infini, un puits de contenance infinie et un réseau de tuyaux reliant la source au puits, on cherche le flot maximum que l'on peut faire passer à travers le réseau. Comme le débit de la source et la contenance du puits sont infinis, le flot maximum est uniquement contraint par le réseau. Intuitivement, on peut voir le réseau comme un barrage entre la source et le puits qui ne laisse passer qu'un certain débit.

Si maintenant on cherche à comprendre pourquoi le réseau restreint le flot, on peut se convaincre que le réseau contient un goulot d'étranglement. Considérons un ensemble de tuyaux qui sépare la source du puits. Pour aller de la source au puits, l'eau doit nécessairement emprunter l'un de ces tuyaux. Donc dans le meilleur des cas, si tous

ces tuyaux sont pleins d'eau, le flot sera égal à la somme de leurs capacités. Le goulot d'étranglement correspond alors à un ensemble de tuyaux qui sépare la source du puits dont la somme des capacités est minimale.

Remarquons, pour finir avec cet exemple, que si le flot est maximum à travers le réseau, on est dans le cas où tous les tuyaux du goulot d'étranglement sont pleins. La valeur du flot maximum est par conséquent égale à la capacité minimale d'un ensemble séparateur. Trouver l'une ou l'autre de ces valeurs sont donc deux problèmes liés.

3.2 Formulation théorique

On considère un graphe orienté connexe $\mathcal{G}$ composé d'un ensemble de sommets $\mathcal{S}$ et d'un ensemble d'arcs orientés $\mathcal{A} \in \mathcal{S}^2$. On distingue deux sommets particuliers : la *source* s et le *puits* p. Pour un sommet x, on définit l'ensemble des arcs *entrants* $\mathcal{A}_e(x)$ et celui des arcs *sortants* $\mathcal{A}_s(x)$:

$$\mathcal{A}_e(x) = \{a \in \mathcal{A} \ / \ \exists y \in \mathcal{S}, a = (y, x)\}$$
$$\mathcal{A}_s(x) = \{a \in \mathcal{A} \ / \ \exists y \in \mathcal{S}, a = (x, y)\}$$

On définit une fonction *capacité* qui associe à un arc a un réel positif $Cap(a)$ et une fonction *flot* qui lui associe un autre réel positif $Flot(a)$. On dit que le flot est *valide* si :

$$\forall a \in \mathcal{A} \qquad Flot(a) \leq Cap(a) \tag{3.1}$$
$$\forall x \in \mathcal{S} \backslash \{s, p\} \qquad \sum_{a_e \in \mathcal{A}_e(x)} Flot(a_e) = \sum_{a_s \in \mathcal{A}_s(x)} Flot(a_s) \tag{3.2}$$

s et p étant exclus de la contrainte 3.2, pour simplifier la suite des définitions, on suppose $\mathcal{A}_e(s) = \mathcal{A}_s(p) = \emptyset$.

Dans l'exemple précédent, $\mathcal{G}$ est le réseau, $\mathcal{A}$ l'ensemble des tuyaux et $\mathcal{S}$ l'ensemble des jonctions entre tuyaux. La fonction de capacité indique le débit maximum dans tuyau, celle de flot donne le flot effectif dans un tuyau. Le flot est valide si toute l'eau qui arrive à une jonction en repart (équation (3.2)) et si le flot dans un tuyau n'est pas supérieur à son débit maximum (équation (3.1)).

À partir de maintenant, on ne considère plus que des flots valides.

À une fonction de flot donnée, on associe une valeur que l'on appellera *flot de graphe* (ou simplement *flot* si cela ne prête pas à ambiguité) :

$$Flot(\mathcal{G}) = \sum_{a \in \mathcal{A}_s(s)} Flot(a) \tag{3.3}$$

On définit une *coupure* $\mathcal{C}_\mathcal{G}$ de $\mathcal{G}$ comme la partition de $\mathcal{S}$ en deux ensembles connexes $\mathcal{S}_s$ et $\mathcal{S}_p$ tels que $s \in \mathcal{S}_s$ et $p \in \mathcal{S}_p$. On associe à cette coupure une valeur $Coup(\mathcal{C}_\mathcal{G})$ (que l'on appellera aussi *coupure* quand cela ne crée pas d'ambiguité) :

$$Coup(\mathcal{C}_\mathcal{G}) = \sum_{(x,y) \in (\mathcal{S}_s \times \mathcal{S}_p)} Cap(x, y) \tag{3.4}$$

et on dira qu'un arc (x, y) est *coupé* si $(x, y) \in (\mathcal{S}_s \times \mathcal{S}_p)$.

Par la suite, on s'intéressera à la coupure de valeur minimale. Pour cela, on a le théorème suivant qui constituera la base de la méthode proposée.

Théorème "Flot maximum - coupure minimale" : Étant donnés un graphe $\mathcal{G}$, une source s, un puits p et une fonction de capacité Cap, on a la relation suivante entre le flot maximal atteint sur l'ensemble des fonctions de flot valides et la coupure de valeur minimale :

$$\max Flot(\mathcal{G}) = \min Coup(\mathcal{C}_\mathcal{G}) \tag{3.5}$$

Par rapport à l'exemple initial, ce théorème est simplement la formalisation du fait qu'il ne peut pas passer plus d'eau à travers le réseau que le goulot d'étranglement ne le permet.

On trouvera la démonstration du théorème dans [FF62]. L'intérêt majeure de ce résultat est de montrer qu'en utilisant des algorithmes qui calculent le flot maximum [GR97, CG97] on a accès à la coupure minimale.

4 Minimisation d'énergie

L'idée à la base de la méthode que l'on propose est de créer un graphe de manière à ce qu'une coupure représente une fonction et la valeur de cette coupure représente une énergie associée à cette fonction. En trouvant la coupure minimale, on aura donc calculé la fonction qui minimise cette énergie. On commence par un exemple simple pour illustrer le concept. On étudie ensuite les cas à deux dimensions et à trois dimensions.

4.1 Exemple simple

Prenons trois points x_1, x_2 et x_3, une fonction f qui peut prendre deux valeurs y_1 et y_2. Pour définir une énergie sur f, on introduit une fonction de coût $c(x, y) > 0$ qui représente l'énergie du choix $f(x) = y$ et une fonction de pénalité $p(f(x_i), f(x_{i+1}))$ pour $i \in \{1, 2\}$ qui représente notre souhait d'avoir des valeurs de f similaire pour des points proches. $p(f(x_i), f(x_{i+1}))$ est nulle si $f(x_i) = f(x_{i+1})$ et vaut $p_0 > 0$ sinon.

On cherche f qui minimise :

$$\mathcal{E}(f) = \sum_{i=1}^{3} c(x_i, f(x_i)) + \sum_{i=1}^{2} p(f(x_i), f(x_{i+1})) \tag{4.1}$$

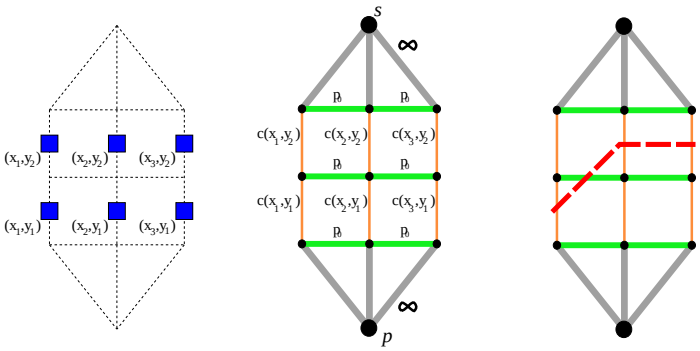


FIG. 1 – Un graphe simple. À gauche, les positions possibles pour f et la structure du graphe en pointillés (les positions sont associées à des arcs verticaux). Au milieu, le graphe avec les capacités des arcs. À droite, un exemple de coupure.

On construit un graphe tel qu'indiqué sur la figure 1. La base du graphe est une grille dont les arcs sont bidirectionnels. À chaque position (x, y) on associe un arc vertical avec la capacité $c(x, y)$, les arc horizontaux correspondent à la fonction de penalité. Les arcs qui relient la source à la grille et la grille au puits ont une capacité infinie.

Si on étudie maintenant une coupure pour ce graphe (figure 1-droite), on peut faire les remarques suivantes :
– si elle coupe un arc qui est lié à la source ou au puits, elle aura une valeur infinie donc ne sera pas minimale ;
– si on exclut ces coupures infinies, comme une coupure crée une partition des sommets, elle doit forcément passer par au moins un arc dont l'abscisse est x_i pour $i \in \{1, 2, 3\}$;
– une coupure minimale ne coupe qu'un seul arc d'abscisse x_i car une coupure coupant deux tels arcs aura une valeur plus élevée que si elle ne coupait que l'un des deux.

Une coupure qui satisfait ces trois remarques est dite *potentiellement minimale* : elle ne coupe aucun arc infini et ne coupe qu'un et un seul arc d'abscisse x_i.

Par conséquent, on peut construire une fonction f à partir d'une coupure potentiellement minimale : à partir de chaque arc (x_i, y_j) on déduit un point $f(x_i) = y_j$. Inversement, à partir d'une fonction f, on construit une coupure potentiellement minimale en coupant uniquement les arcs (x_i, y_j) tels que $f(x_i) = y_j$. Observons pour finir, la valeur d'une telle coupure : les arcs verticaux coupés forment exactement la somme $\sum_{i=1}^{3} c(x_i, f(x_i))$ et les arcs horizontaux coupés la somme $\sum_{i=1}^{2} p(f(x_i), f(x_{i+1}))$ de l'énergie (4.1).

En conclusion, en calculant la coupure minimale du graphe en figure 1, on trouve la fonction $f_0 = \text{argmin}_f \, \mathcal{E}(f)$.

Avant de passer à un cas plus complexe, observons quelques points importants.
– Nous avons un calcul direct du résultat, nous n'avons utilisé aucune technique type descente de gradient susceptible de se bloquer dans un minimum local. Nous sommes assurés d'avoir le minimum global de l'énergie.

– On peut exprimer la pénalité p_0 sous la forme $\alpha \left| \frac{\Delta f(x_i)}{\Delta x_i} \right|$ ce qui offre une interprétation géométrique de la pénalité : on pénalise proportionnellement à la pente de f.
– La fonction de pénalité p et le coefficient α peuvent aussi être fonction de x_i. Cela permet d'introduire des contraintes plus souples, on peut affecter par exemple une pénalité plus importante au cas $f(x_1) \neq f(x_2)$ qu'au cas $f(x_2) \neq f(x_3)$ ce qui se traduit par $\alpha(x_1) > \alpha(x_2)$.

4.2 Étude en deux dimensions

Après cet exemple simple, nous pouvons étendre la méthode à une fonction plus générale en deux dimensions. Considérons une fonction f réelle d'un intervalle $[a, b]$ dans un intervalle $[c, d]$, une fonction de coût $c(x, y)$ et un coefficient de pénalité $\alpha(x)$. On définit de manière similaire une énergie $\mathcal{E}$ qui est d'autant plus faible que la fonction f prend des valeurs de faible coût tout en ayant des valeurs voisines proches les unes des autres.

$$\mathcal{E}(f) = \int_a^b \left(c(x, f(x)) + \alpha(x) \left| \frac{df}{dx}(x) \right| \right) dx \tag{4.2}$$

Nous allons montrer comment trouver la fonction f_0 qui minimise cette énergie à une discrétisation près. En effet, comme nous utilisons des graphes, nous ne pouvons traiter que des cas discrets. Néanmoins, il est toujours possible de discrétiser plus finement pour obtenir des résultats plus précis. Nous découpons par conséquent l'intervalle $[a, b]$ en $n_x - 1$ sous-intervalles de même longueur Δx pour obtenir n_x points x_1, x_2,... x_{n_x}. Nous faisons de même pour $[c, d]$ pour obtenir n_y points y_1, y_2,... y_{n_y} espacés de Δy. L'énergie discrète correspondant à (4.2) est alors :

$$\mathcal{E}^d(f) = \sum_{i=1}^{n_x} c(x_i, f(x_i)) + \sum_{i=1}^{n_x - 1} \alpha(x_i) \left| \frac{\Delta f(x_i)}{\Delta x} \right| \tag{4.3}$$

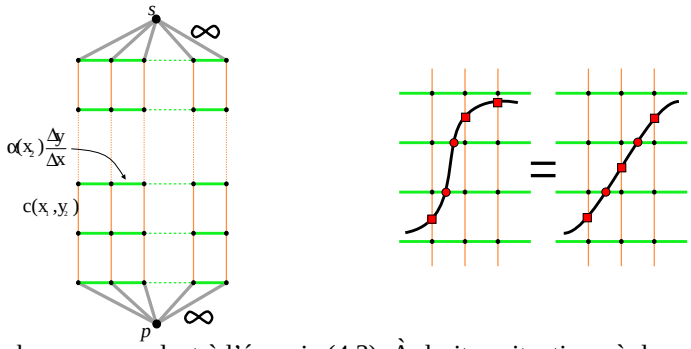


FIG. 2 – À gauche : graphe correspondant à l'énergie (4.3). À droite : situation où deux coupures ont exactement la même valeur : trois arcs de coût (les carrés) et 2 arcs de pénalité (les ronds).

On obtient une énergie très similaire à celle obtenue en (4.1). On procède par conséquent de la même manière : on utilise un graphe basée sur une grille (figure 2-gauche) où les arcs verticaux représentent la fonction de coût $c(x, y)$ et les arcs horizontaux la pénalité $\alpha(x)\frac{\Delta y}{\Delta x}$ pour les valeurs disctinctes. On introduit de manière totalement similaire les *coupures potentiellement minimales* et on leur associe à chacune une fonction f. La conclusion étant que la fonction f_0 associée à la coupure minimale réalise : $f_0 = \text{argmin}_f \mathcal{E}^d$.

Observons à nouveau quelques détails sur la méthode.
– Ce problème se résoud très bien grâce à la programmation dynamique [Bel57], il ne s'agit encore que d'un cas d'étude.
– La formulation (4.2) exhibe la géométrie du problème : l'élément d'intégration dx explicite la mesure employée. Si on change la configuration géométrique – par un changement d'échelle par exemple – il est aisé de recalculer les différentes grandeurs de (4.3) en fonction de cette nouvelle configuration.
– La seule contrainte pour la fonction de coût est de pouvoir la discrétiser donc qu'elle soit continue par morceaux.
– En regardant l'énergie (4.2), la fonction f_0 doit être continûment dérivable sauf sur les points où $\alpha(x) = 0$.
– Comme précédemment, on trouve un minimum global de manière certaine.
– On a montré la méthode sur un domaine 2D rectangulaire mais il est tout à fait possible de travailler sur un domaine plus général. Il suffit alors de discrétiser ce domaine selon une grille et de relier les nœuds de la limite supérieure à la source et ceux de la limite inférieure au puits.

– Comme la pénalité est simplement proportionnelle à la pente de la coupure, dans les zones où les fonctions de coût et de pénalité sont constantes, on peut trouver plusieurs coupures de même valeur car un palier ou une pente "douce" ont la même énergie (figure 2-droite). A priori, vu que l'on impose une pénalité aux valeurs distinctes, on souhaite obtenir la pente douce alors qu'en pratique les algorithmes donnent toujours le palier.

Une pénalité non-linéaire

La dernière remarque soulève un problème important : on obtient des paliers malgré l'introduction dans l'énergie d'une composante visant à limiter les variations de la fonction solution. La linéarité de la pénalité est à l'origine de ce phénomène car elle ne différencie pas une importante variation de plusieurs petites. Nous proposons par conséquent une structure de graphe permettant une pénalité strictement convexe : il sera alors plus pénalisant pour la fonction de faire un palier que de faire une variation régulière. Ishikawa [Ish00] propose une méthode générale pour obtenir une fonction de pénalité convexe à une constante près sur la pénalité, ce qui ne peut être génant que dans le cas d'un échantillonnage irrégulier. Nous proposons toutefois une autre méthode qui n'introduit pas de constante sur la pénalité et suffit à avoir une fonction de pénalité strictement convexe.

Le graphe est construit à partir de l'élément de base de la figure 3 : l'arc vertical de coût est remplacé par quatre arcs *mineurs* de capacité moitié et on introduit un nouveau coefficient de pénalité β dit *secondaire*. Il correspond à la capacité des arcs de pénalités reliant les milieux de deux arcs verticaux adjacents. On appellera α le coefficient *principal* pour éviter les ambiguités.

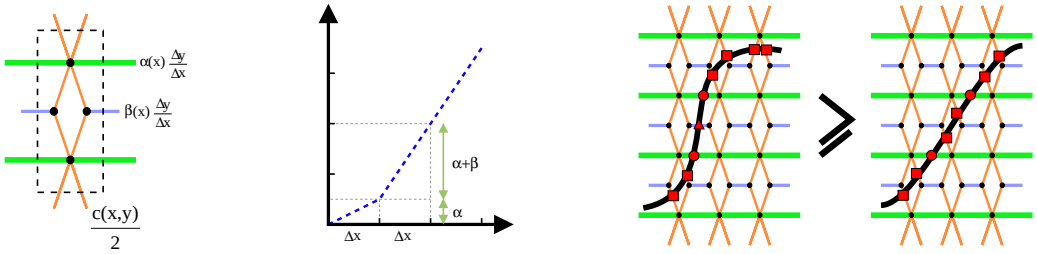


FIG. 3 – À gauche : l'élément de base utilisé. Au milieu : la fonction de pénalité p_{conv} correspondante. À droite : une variation continue est moins pénalisée qu'un palier : elle coupe une pénalité secondaire de moins (triangle).

Le processus qui permet d'obtenir une fonction de pénalité p_{conv} strictement convexe est le suivant : si lors d'un pas unitaire Δx, la fonction varie de q pas Δy, la coupure coupe seulement $q-1$ arcs principaux (ou aucun si $q \leq 1$) et q arcs secondaires car elle peut "passer au milieu des arcs de coût mineurs" (voir la figure 3-droite). On obtient donc la fonction de pénalité p_{conv} de la figure 3-milieu. On notera que l'on coupe toujours les arcs de coût mineur par paire, ce qui rétablit la fonction de coût intiale.

On a finalement construit un graphe qui permet de calculer l'énergie :

$$\mathcal{E}^d_{conv}(f) = \sum_{i=1}^{n_x} c(x_i, f(x_i)) + \sum_{i=1}^{n_x-1} p_{conv}\left(\left|\frac{\Delta f(x_i)}{\Delta x}\right|\right) \tag{4.4}$$

Grâce à $\mathcal{E}^d_{conv}$, les variations progressives sont maintenant favorisées par rapport aux paliers. Il faut néanmoins remarquer les points suivants.

– Avec le graphe tel qu'il est présenté, il peut être moins pénalisant pour une coupure de couper deux arcs de coût mineurs que de couper un arc de pénalité secondaire. Pour éviter ce problème, on peut soit ajouter une constante à la fonction de coût, ce qui présente les mêmes restrictions qu'une constante sur la fonction de pénalité, soit utiliser les *arcs de contrainte* proposés par Ishikawa[Ish00] qui permettent d'éliminer cette constante.
– L'énergie $\mathcal{E}^d_{conv}$ n'a pas directement d'équivalent en continu car p_{conv} dépend du pas de discrétisation. Toutefois, si $\beta = 0$, $\mathcal{E}^d_{conv} = \mathcal{E}^d$ et $\mathcal{E}^d_{conv}$ correspond alors à $\mathcal{E}$. On peut donc avoir une approximation discrète de $\mathcal{E}$ qui favorise les variations progressives aussi bonne que l'on souhaite en prenant β aussi petit que nécessaire.

4.3 Étude en trois dimensions

Pour le cas en trois dimensions, on cherche une fonction f définie sur un rectangle $[a_x, b_x] \times [a_y, b_y]$, qui prend ses valeurs dans $[c, d]$ et qui minimise l'énergie :

$$\mathcal{E}(f) = \int_{a_x}^{b_x}\int_{a_y}^{b_y} \left(c(x,y,f(x,y)) + \alpha_x(x,y)\left|\frac{df}{dx}(x,y)\right| + \alpha_y(x,y)\left|\frac{df}{dy}(x,y)\right| \right) dx\, dy \tag{4.5}$$

Nous proposons un procédé totalement équivalent au cas en deux dimensions, nous utilisons simplement une grille en trois dimensions comme base du graphe. Les restrictions à un plan $x = C_x$ ou $y = C_y$ (avec C_x et C_y constantes) sont simplement des graphes à deux dimensions tels que présentés précedemment. Pour une pénalité linéaire, on obtient par conséquent l'élément de base de la figure 4-gauche. Cette pénalité donne toujours des paliers dans les résultats, on préfère utiliser une approximation par une pénalité strictement convexe, ce qui conduit à l'élément décrit dans la figure 4-droite. On note simplement que maintenant les arcs de coût mineurs sont coupés par groupes de quatre, ce qui implique qu'ils aient une capacité $\frac{1}{4}c(x, y, z)$.

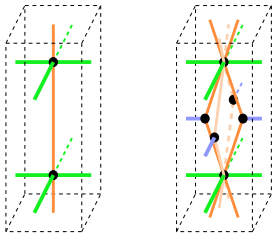


FIG. 4 – Gauche : l'élément de base utilisé pour une pénalité linéaire. Droite : pour une pénalité convexe.

FIG. 5 – Exemple de cartes de discontinuité utilisées (gauche : α_x, droite : α_y)

Il est important de noter les deux points suivants qui n'apparaissaient pas dans les études précédentes :

– le procédé est anisotrope car il dépend des axes x et y,
– le problème que l'on résout n'a plus de solution avec des techniques de programmation dynamique. La discussion n'entre pas dans le cadre de cet article, l'idée sous-jacente étant que la fonction est définie sur deux variables et que cela empêche un parcours récursif de l'espace des fonctions solutions.

4.4 Extensions

La méthode présentée permet quelques extensions utiles que l'on présente mais pour lesquelles on ne donne pas de détails techniques par soucis de concision.

***D* + 1 dimensions :** De façon totalement similaire, on peut trouver des fonctions à D variables minimisant des énergies dans un espace à $D + 1$ dimensions.

$$\mathcal{E}(f) = \int \left(c(X, f(X)) + \sum_{i=1}^{D} \alpha_{x_i}(X) \left| \frac{df}{dx_i}(X) \right| \right) dX \quad \text{avec} \quad X = (x_1, x_2, ..., x_D)$$

Fonctions périodiques : On peut traiter des fonctions périodiques (selon x par exemple) simplement ajoutant des arcs de pénalité entre les noeuds d'abscisse n_x et ceux d'abscisse 1.

Différentes mesures : On peut utiliser d'autres mesures. Pour une fonction définie en coordonnées cylindriques (r, θ, z) par exemple, l'élément d'intégration est $r\, dr\, d\theta\, dz$, on peut distribuer le r sur les fonctions de coût et de pénalité et se ramener au cas d'une fonction périodique en θ.

Approximation locale : L'algorithme de résolution de flot n'étant pas linéaire par rapport aux nombres de noeuds de la grille, il peut être utile d'avoir une approximation linéaire du résultat. Pour cela, pour chaque point X où est définie la fonction f on résout le problème d'optimisation sur un voisinage de X et on associe à $f(X)$ la valeur ainsi trouvée. On ne donnera pas ici l'étude complète de cette approximation, on notera simplement que sur des cas pratiques de taille moyenne, les temps de calcul sont similaires au calcul exact. On réservera donc cette approximation au problème de taille importante.

5 Application à la reconstruction 3D

On se propose d'appliquer la méthode d'optimisation proposée dans le cadre de l'algorithme de reconstruction 3D décrit dans [PS02]. On ne décrira pas l'ensemble de l'algorithme mais uniquement le cadre dans lequel nous utilisons le flot de graphe.

5.1 Contexte d'utilisation

La reconstruction que nous proposons travaillent à partir d'une courte séquences d'images très proches les unes des autres. Comme la séquence est courte, l'angle de vue sur les objets à reconstruire est très faible et induit une forte ambiguité sur la profondeur des points reconstruits. Comme la méthode d'optimisation à base de graphes trouve le minimum global, nous arrivons à lever cette ambiguité.

En pratique, nous travaillons dans un ensemble de voxels qui constitue la discrétisation de l'espace nécessaire à notre méthode. À chaque voxel (x, y, z), nous avons associé une valeur $V(x, y, z)$ qui est d'autant plus faible que le voxel est vu de manière similaire dans toutes les images. Dans l'hypothèse d'objets lambertiens, nous recherchons les faibles valeurs de V car alors la couleur d'un objet ne dépend pas du point de vue. Toutefois les images d'entrée n'ayant pas nécessairement une qualité parfaite et à cause de l'ambiguité sur la profondeur déjà évoquée, nous ne pouvons prendre comme surface de l'objet reconstruit la surface qui minimise uniquement V car elle risque d'être fortement discontinue. Il faut introduire un terme de régularisation et ainsi obtenir une surface qui à la fois est cohérente par rapport aux images et possède une certaine continuité. Nous sommes donc dans le cadre de l'optimisation présentée.

Pour finir, nous n'imposons pas une contrainte de continuité sur l'ensemble de la surface reconstruite car l'objet reconstruit peut très bien présenter des discontinuités. Grâce à l'éclairage, nous savons que ces discontinuités engendrent des variations d'ombrage donc des variations de couleur dans les images. Nous relâchons donc les contraintes de continuité le long des lignes de changement de couleur.

5.2 Application de la méthode

Tout d'abord, l'utilisateur fixe une pénalité de discontinuité α_{max} qui doit être appliquée sur les zones de couleur uniforme. Ensuite, à partir des images initiales, nous détectons les discontinuités de couleur horinzontales et verticales et construisons ainsi les fonctions de pénalité α_x et α_y (figure 5) qui évoluent entre 0 (sur les fortes discontinuités de couleur) et α_{max} (dans les zones de couleur uniforme). Une fois que nous avons ces deux fonctions, nous pouvons adapter l'énergie (4.2) à notre besoin, avec V comme fonction de coût et $\mathcal{D}$ le support de la fonction recherchée :

$$\mathcal{E}(f) = \iint_{\mathcal{D}} \left(V(x, y, f(x, y)) + \alpha_x(x, y) \left| \frac{df}{dx}(x, y) \right| + \alpha_y(x, y) \left| \frac{df}{dy}(x, y) \right| \right) dx\ dy \tag{5.1}$$

5.3 Améliorations obtenues

Sans cette méthode qui permet d'optimiser la position de la surface dans son ensemble, nous étions obligés de procéder à une optimisation "ligne par ligne" comme beaucoup d'autres méthodes [OK85, OK93, IB94, KPV98, UKG98]. Cela nous obligeait à avoir de forts traitements dans la suite du processus pour assurer un régularité d'une ligne à l'autre. Cela avait pour conséquence de "gommer" de nombreux détails de la surface de l'objet (le nez sur un visage par exemple).

Maintenant les traitements qui suivent sont beaucoup plus restreints : on se contente de former une surface à partir de la fonction solution et de la lisser légèrement pour supprimer l'aliassage dû à la discrétisation. Les petits détails sont ainsi conservés.

De plus, le "découpage" en ligne introduisait arbitrairement une séparation entre les directions x et y qui subissaient des traitements différents au cours du processus de reconstruction. Maintenant, ces deux directions sont traitées de manière équivalente ce qui est plus satisfaisant.

6 Résultats

La figure 6 illustre les résultats obtenus sur trois séquences d'images différentes. Pour juger de ces résultats, il est important de noter la résolution des images utilisées. Par exemple, le visage de l'homme ne couvre dans chaque image qu'une surface d'environ 40×40 pixels. Malgré cela, on remarquera les nombreux détails qui apparaissent

dans les modèles géométriques : les touches du clavier, les plis du lampion, le nez de l'homme, le contenu de sa malette, etc.

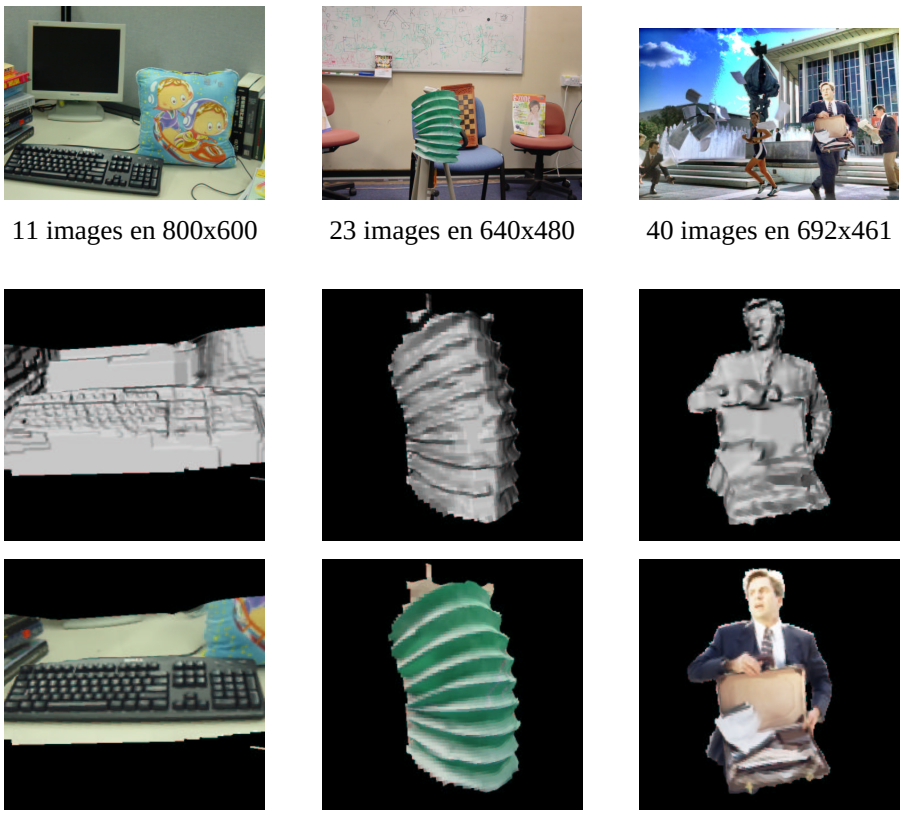


FIG. 6 – Résultats sur trois séquences d'images. La première ligne présente une image, la résolution et la longueur de la séquence, la deuxième montre le modèle obtenu sans texture et la troisième le modèle avec texture.

La phase d'optimisation pour le calcul de ces modèles a nécessité de 30 minutes à presque 2 heures selon les modèles sur un processeur MIPS R12000 à 400MHz et un espace mémoire de 300 méga-octets à 700 méga-octets. Ces chiffres qui peuvent paraître importants sont à relativiser sachant que l'optimisation se fait sur un espace contenant de 1 à 10 millions de voxels et que chaque voxel introduit dans le graphe 5 sommets et 24 arcs (voir la figure 4-gauche). De plus, en remarquant que le nombre d'arcs est proportionel à celui de sommets, le calcul de la coupure minimale est en $\mathcal{O}(n^{2,5})$ [CG97]. Bien qu'il soit souvent difficile de connaître la taille des graphes utilisés, à notre connaissance, les implémentations actuelles ne dépassent l'ordre de grandeur de 100 000 sommets et 300 000 arcs [KZ02a].

Implémentation

Pour traiter de tels graphes, nous avons apporté une attention particulière à la gestion de la mémoire :
– les relations de voisinage entre sommets et arcs ne sont pas stockées mais calculées à chaque requête,
– seules les fonctions c, α et β sont stockées ce qui évitent de stocker la capacité de chaque arc,
– un arc double (s_1, s_2) permet normalement au flot de "tourner" : il peut exister un flot non nul de s_1 vers s_2 et de s_2 vers s_1, en supprimant la partie "tournante" de ce flot, on peut stocker les flots des deux arcs qui composent (s_1, s_2) sur une seule variable, le signe indiquant la direction du flot.

Nous avons aussi implémenté les heuristiques proposées par Cherkassky et al. [CG97] ce qui nous a permis d'atteindre des temps de calcul inférieurs à 24 heures.

7 Conclusion

Nous venons de décrire une nouvelle technique d'optimisation qui s'applique à un ensemble d'énergies que nous avons caractérier. Cette technique apporte les contributions suivantes :
– une formulation continue du problème à résoudre exhibant le rôle de la géométrie de la configuration,
– la résolution exacte d'une discrétisation de ce problème,
– la mise en évidence d'ambiguités ainsi qu'une technique pour les résoudre,
– un graphe original pour représenter certaines fonctions de pénalité convexes,
– l'application de ces résultats à problème d'acquisition 3D à partir d'une courte séquence d'images grâce à une implémentation supportant des graphes de taille très importante.

Grâce à cette méthode nous avons montré que nous pouvons construire des modèles tridimensionnels avec une précision supérieure à celle offerte classiquement par les cartes de disparité et sur des modèles qui nécessitent un volume important de données.

Travaux futurs

Nous pensons que la méthode peut encore être améliorée pour encore mieux reconstruire les surfaces courbes. Nous souhaitons aussi explorer plus précisément les extensions que nous proposons dans la section 4.4.

Références

[Bel57] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[BGCM02] C. Buehler, S. Gortler, M. Cohen, and L. McMillan. Minimal surfaces for stereo. In *ECCV*, 2002.

[CG97] B. Cherkassky and A. Goldberg. On implementing the push-relabel method for the maximum flow problem. *Algorithmica*, 19(4) :390–410, 1997.

[FF62] L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[FK98] Olivier Faugeras and Renaud Keriven. Variational principles, surface evolution, PDE's, level set methods and the stereo problem. *IEEE Transactions on Image Processing*, 1998.

[GR97] A. Goldberg and S. Rao. Length functions for flow computations. Technical Report 97-055, NEC Research Institute, Inc., 1997.

[IB94] S. Intille and A. Bobick. Disparity-space images and large occlusion stereo. In *ECCV*, 1994.

[Ish00] H. Ishikawa. *Global Optimization Using Embedded Graphs*. PhD thesis, New York University, 2000.

[KPV98] R. Koch, M. Pollefeys, and L. Van Gool. Multi viewpoint stereo from uncalibrated video sequences. *Lecture Notes in Computer Science*, 1406, 1998.

[KS99] K. Kutulakos and S. Seitz. A theory of shape by space carving. In *ICCV*, 1999.

[KZ01] V. Kolmogorov and R. Zabih. Computing visual correspondence with occlusions using graph cuts. In *ICCV*, 2001.

[KZ02a] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. In *ECCV*, 2002.

[KZ02b] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts ? In *ECCV*, 2002.

[OK85] Y. Ohta and T. Kanade. Stereo by intra- and inter-scanline search using dynamic programming. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7 :139–154, 1985.

[OK93] M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4) :353–63, 1993.

[PS02] S. Paris and F. Sillion. Robust acquisition of 3d informations from short image sequences. In *Pacific Graphics*, 2002.

[RC98] S. Roy and I. Cox. A maximum-flow formulation of the n-camera stereo correspondence problem. In *ICCV*, 1998.

[SCMS01] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafer. A survey of methods for volumetric scene reconstruction from photographs. In *VolumeGraphics*, 2001.

[UKG98] M. Ulvklo, H. Knutsson, and G. Granlund. Depth segmentation and occluded scene reconstruction using ego-motion. *Proc. SPIE Vol. 3387, p. 112-123, Visual Information Processing VII*, 1998.

[Vek99] O. Veksler. *Efficient Graph-Based Energy Minimization Methods in Computer Vision*. PhD thesis, Cornell University, 1999.

Détection de collisions entre objets rigides convexes autonomes

J. Dequidt, L. Grisoni, P. Meseure, C. Chaillou

LIFL - Université de Lille 1
Bât M3, 59655 Villeneuve d'Ascq cedex
{dequidt,grisoni,meseure,chaillou}@lifl.fr

Résumé : *Dans cet article, nous proposons une méthode complète de traitement de collisions. Elle est constituée de différentes étapes qui permettent de détecter de plus en plus finement les collisions entre objets rigides convexes. La dernière étape fournit des informations nécessaires à la génération d'une force de pénalités. Ce framework est compatible avec des objets autonomes (objets capables de gèrer leur propre comportement).*

Mots-clés : Objets rigides convexes, détection de collision, distance d'interpénétration, forces de pénalités.

1 Introduction

Une application de réalité virtuelle régie par une simulation physique implique généralement un traitement systématique, chaque itération comprenant classiquement une étape de détection de collisions, une étape faisant un bilan des forces appliquées sur chaque objet, puis une intégration des équations du mouvement et/ou de déformation qui régissent le comportement de chaque objet. La détermination des collisions est donc une étape importante de ce traitement, et d'autant plus délicate qu'une version naïve impliquerait de tester la collision potentielle de toutes les paires d'objets de la scène (et donc un nombre $\mathcal{O}(n^2)$ de couples, n désignant ici le nombre d'objets de la scène), le traitement de chaque paire impliquant lui aussi un traitement de complexité quadratique en fonction du nombre de primitives géométriques utilisées pour définir les objets.

Pour les interactions entre objets, il existe deux types d'algorithmes : les algorithmes de contact ou les algorithmes d'interpénétrations. La première catégorie d'algorithmes garantit de ne pas violer la contrainte de contact mais implique de nombreux calculs et se limite aux objets rigides [RKC02]. L'autre catégorie d'algorithmes nécessite d'évaluer la *force d'interpénétration* qui permet, dans le bilan des forces, d'introduire une force de répulsion. C'est à ce deuxième cadre d'étude que nous nous intéressons.

La détection de collisions est un sujet qui a été largement étudié car il concerne un grand nombre de disciplines, comme par exemple la robotique (planification de trajectoires), l'IHM et la synthèse d'images. La majorité des travaux se sont penchés sur une détection de collisions entre objets rigides convexes (la détection de collisions entre objets concaves peut être ramenée à une détection entre objets convexes par une décomposition de l'objet en parties convexes). Cependant ces méthodes reposent sur une architecture centralisée. Le projet AlCOVe de l'équipe Graphix du LIFL a pour objectif de réaliser une plate-forme de simulation physique collaborative. La principale caractéristique de cette plate-forme est d'avoir une architecture totalement distribuée (absence de serveur). La simulation des objets doit donc être répartie sur un ensemble de machines. Pour cela, on se base sur la notion d'objets autonomes, capables de déterminer leur propre comportement dans l'environnement. Il est donc nécessaire de s'assurer que chaque étape de la simulation physique puisse être réalisée de manière autonome par les objets. Nous proposons donc une méthode de traitements de collisions qui s'affanchit de l'hypothèse d'une architecture centralisée et qui permet, pour le traitement des collisions, de rendre chaque objet autonome.

Cet article s'articule de la manière suivante : la section 2 présente les principales méthodes de détection connues et utilisées ainsi que les algorithmes donnant une distance permettant de séparer deux objets. La section 3 expose notre modèle de détection et détaille son originalité par rapport aux méthodes existantes.

2 Etat de l'art

De nombreux travaux ont été réalisés pour déterminer les collisions entre objets convexes. Un très grand nombre se sont intéressés à la détection de collisions exactes entre deux polyèdres et d'autres aux moyens d'accélérer cette

détection. Cependant peu de travaux se sont penchés sur l'élaboration d'un framework complet de détection de collisions *i.e.* comment combiner des principes d'accélérations et une détection exacte afin d'obtenir une méthode complète et performante. Zachmann [Zac01] explique que ce framework (qu'il désigne par le terme *pipeline*) doit être composé de filtres de plus en plus précis (et donc de plus en plus lourds en calcul) qui permettent de réduire le nombre de tests de collisions exactes à effectuer. Dans cette section, nous présentons les méthodes les plus utilisées pour la détection, puis celles permettant de calculer la distance d'interpénétration de deux objets en collision. Enfin nous évoquons les frameworks existants.

2.1 Détection de collisions

Détecter si deux objets sont en intersection peut se faire à différents "grains". Au niveau le plus fin de l'objet lui-même, on parle de détection exacte. Un autre choix est d'utiliser une approximation plus ou moins fine de l'objet et dans ce cas, la détection est faite entre volumes englobants. Nous présentons tout d'abord les algorithmes de détection exacte puis nous donnons un aperçu des concepts permettant d'accélérer ce traitement (avec par exemple une détection entre volumes englobants).

2.1.1 Détections exactes

Pour déterminer si deux objets convexes A et B sont interpénétrés, il existe plusieurs familles d'approches :

- déterminer s'il existe un plan partitionnant l'espace en deux demi-espaces, l'un contenant A, l'autre B. C'est la solution proposée par Chung [Chu96] et van den Bergen [vdB98] (méthodes itératives).
- calculer la distance entre ces deux objets. Si elle est inférieure ou égale à zéro, il y a collision [LC91, GJK88]

L'algorithme proposé par Lin et Canny [LC91] consiste à déterminer les éléments (*i.e.* sommet, arête, facette) de A et de B permettant d'obtenir la plus petite distance. A partir d'un couple (f_A^0,f_B^0) d'éléments de A et B, on cherche par utilisation des régions de Voronoï un nouveau couple (f_A^1,f_B^1) plus proche. Cet algorithme est répété itérativement jusqu'à ce qu'un un minimum local soit trouvé (qui est un minimum global grâce à la convexité de l'objet). Cet algorithme peut être optimisé en mémorisant le dernier couple trouvé. La cohérence temporelle [1] permet alors d'espérer une convergence en temps constant $\mathcal{O}(1)$. Une fois que ce couple est trouvé, la distance euclidienne entre les deux objets est la plus petite distance entre le couple d'éléments trouvés.

Cependant, il est possible de calculer la distance de manière moins directe en passant par la différence de Minkowski (notée $\mathcal{M}$) qui est définie de la manière suivante :

$$\mathcal{M} = A - B = \{x - y \setminus x \in A,\ y \in B\}$$

Cette différence est convexe lorsque les objets A et B sont convexes. L'intérêt de cette différence est que l'on ramène le calcul de la distance entre deux objets à la distance entre $\mathcal{M}$ et l'origine. De plus si l'origine se trouve à l'intérieur de $\mathcal{M}$ les objets sont en collision.

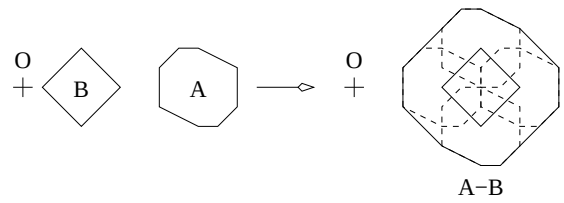


FIG. 1 – Différence de Minkowski de deux objets convexes.

Cependant la construction de cette différence est en $\mathcal{O}(nm)$ (où n et m sont les nombres de sommets de A et de B). Pour éviter une construction explicite de $\mathcal{M}$, on peut utiliser les points de supports. Le point de support d'un polyèdre A par rapport à un vecteur $\vec{v}$ donné est le point $s_A(\vec{v})$ appartenant à A tel que le produit scalaire $\vec{v} \cdot s_A(v)$ soit maximal. Une propriété sur les points de support énonce qu'un point de support de la différence de Minkowski peut facilement être obtenu à partir des points de support de A et de B suivant la formule suivante : $s_{\mathcal{M}}(\vec{v}) = s_A(\vec{v}) - s_B(-\vec{v})$. Cette propriété est un des points de départ de l'algorithme GJK [GJK88]. En effet,

[1] les objets ont des d´eplacements faibles entre chaque pas de temps

cet algorithme itératif construit à chaque étape un simplexe [2] en se basant sur cette propriété, ce nouveau simplexe étant plus proche de l'origine que le simplexe précédent. Cet algorithme converge en $\mathcal{O}(n)$ (où n est le nombre de sommets de $\mathcal{M}$).

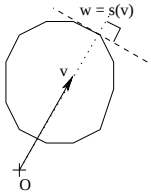


FIG. 2 – Point de support d'un objet convexe selon une direction v.

2.1.2 Accélérer la détection

Lorsque l'application contient un nombre d'objets relativement élevé, il devient impossible d'utiliser ces algorithmes sur la totalité des couples. L'idée est donc d'éliminer rapidement par des critères simples les couples d'objets ne pouvant entrer en collision. Ces techniques sont nombreuses [LG98, JTT01, Mes02] et peuvent être regroupées en 2 catégories : les techniques de *broad phase* et les techniques de *narrow phase* :

- Les techniques de *broad phase* vont considérer l'ensemble des objets de la scène et vont déterminer les collisions potentielles. Dans cette catégorie, on trouve des méthodes de partitionnement spatial, que ce soit en grilles de voxels (grille régulière), en BSP-Trees ou Octrees. Il existe aussi des méthodes qui utilisent la position des objets dans l'espace ou leur déplacement. La plus performante est le *Sweep And Prune* [CLMP95] qui consiste à projeter les objets sur les 3 axes (x,y,z). On obtient ainsi des intervalles et si les intervalles appartenant à deux objets sont disjoints alors les objets correspondant sont disjoints eux aussi.
- Les accélerations de type *narrow phase* travaillent uniquement sur des couples d'objets. Il est courant d'utiliser des volumes simples (ou des hiérarchies des volumes simples) approximant les objets dont on veut résoudre les collisions. Ces volumes peuvent être englobant et dans ce cas si les volumes englobants sont disjoints, les objets le sont aussi. Mais on peut aussi construire des volumes simples englobés par les objets, qui s'ils sont en collision implique que les objets sont aussi en collision. Généralement, la *narrow phase* s'achève par une phase de détection exacte décrite dans 2.1.1.

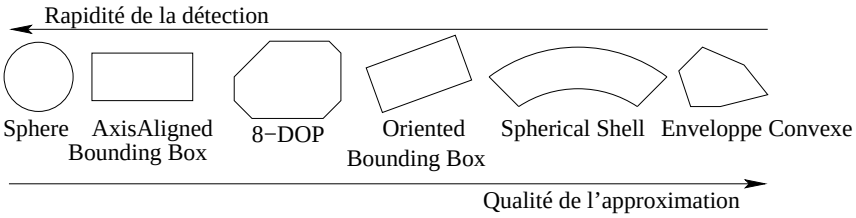


FIG. 3 – Quelques exemples de volumes englobants.

Cette sous-section a mis en évidence les principaux algorithmes permettant de déterminer si deux objets étaient en intersection et les techniques permettant d'accélérer cette détection. Cependant, nous n'avons pas assez d'informations pour séparer de manière correcte ces objets. Ce point est traité dans la section suivante.

2.2 Distance d'interpénétration

Lorsque deux objets sont en collision, il est nécessaire d'estimer le degré d'interpénétration de ces deux objets. Pour cela, on définit la distance d'interpénétration [3] : c'est la norme du plus petit vecteur (*i.e.* de plus petite norme) qui permet par translation d'avoir les objets en contact (et seulement en contact). Cette distance d'interpénétration peut être facilement obtenue avec la différence de Minkowski : c'est la plus petite distance entre $\mathcal{M}$ et l'origine (avec cette fois l'origine se trouvant à l'intérieur de $\mathcal{M}$).

[2] enveloppe convexe d'au plus $n+1$ points pour un espace de dimension n

[3] aussi not´ee MTD : Minimum Translational Distance

Cameron propose dans [Cam97] de borner cette distance à partir du dernier simplexe fourni par l'algorithme *GJK*. Cette méthode est directe, cependant la borne obtenue n'est pas assez précise dans la majorité des cas. Joukhadar [JSL99] utilise un algorithme incrémental qui applique l'algrithme *GJK* après avoir translaté un des deux objets selon un certain vecteur. Cela lui permet de déterminer la distance d'interpénétration mais aussi la direction de contact. Les inconvénients de cette méthode sont sa lenteur de convergence (même si elle peut être réduite par utilisation de la cohérence temporelle) et le fait que la résolution soit une méthode de recherche locale (le minimum trouvé ne sera pas forcément le minimum global). L'algorithme DEEP [KLM02] est aussi un algorithme de recherche locale. Kim calcule de manière implicite la différence de Minkowski en utilisant l'espace dual de l'espace objet (*i.e.* l'espace des normales). En parcourant la surface de la différence de Minkowski, il obtient le couple d'éléments donnant la distance d'interpénétration. Cet algorithme même s'il possède de bonnes performances, connaît des problèmes de convergence (liés à la recherche locale). La méthode que nous avons implémentée est celle de van den Bergen [vdB01]. Elle utilise en entrée le simplexe fourni par la dernière itération de l'algorithme *GJK*. Par la suite, elle va raffiner cette approximation de $\mathcal{M}$ jusqu'à obtenir une bonne approximation de la distance d'interpénétration. L'algorithme assure une convergence rapide (moins rapide que DEEP [KLM02]) mais possède des problèmes de précision dans certains cas.

2.3 Pipelines existants

Comme nous l'avons signalé en introduction, rares sont les travaux concernant un framework complet de détection de collision. Zachmann [Zac01] propose d'utiliser une hiérarchie de volumes englobants de type k-DOP associée à une méthode de détection probabiliste.
Lin [LMCG96] construit un pipeline beaucoup plus étoffé qui implique de calculer un certain nombre de structures : chaque objet possède un volume englobant de type AABB [4], une hiérarchie de volumes englobants de type OBB [5] et une enveloppe convexe. La *broad phase* est réduite à l'application de l'algorithme de *Sweep And Prune* sur les AABB. La *narrow phase* détermine tout d'abord si les enveloppes convexes des objets sont en intersection (à l'aide des régions de Voronoï). Dans le cas d'une collision entre enveloppes convexes, on détecte si les hiérarchie d'OBB sont en intersection. Enfin la détection exacte se fait entre triangles dont les OBB s'interpénétrent.
La méthode de Chung [Chu96] propose, pour la *broad phase* d'utiliser une décomposition spatiale régulière en grille de voxels suivie d'un *Sweep And Prune* sur des boîtes alignées par rapport aux axes. Les deux étapes suivantes dans le pipieline sont incluses dans la *narrow phase*. On recherche tout d'abord de manière itérative un axe séparateur (pour déterminer s'il y a collision ou non). S'il y a collision, l'algorithme *GJK* est appliqué sur la frame précédant la collision afin d'estimer la distance d'interpénétration.

Les pipelines [Zac01, LMCG96] ne se limitent pas aux objets convexes. Cependant le framework proposé dans [Zac01] est assez rudimentaire puisqu'il estla juxtaposition d'une détection de type hiérarchie de volumes englobants et d'une détection exacte. [LMCG96] est beaucoup plus intéressant d'une part car la *broad phase* est très efficace et d'autre part car la *narrow phase* est très complète mais elle requiert beaucoup de calculs (notamment pour la détection entre hiérarchies de OBB). L'inconvénient principal de [Zac01, LMCG96] est qu'aucune estimation de la distance d'interpénétration n'est fournie contrairement à la méthode de Chung [Chu96]. Cette méthode est très efficace mais elle peut être optimisée car on trouve dans les deux dernières étapes une redondance de certains calculs.

3 Traitement de collisions

Le framework que nous exposons doit être plus complet et plus cohérent que ceux existants en garantissant une détection rapide, exacte et qui donne la distance d'interpénétration dans le cas de collisions. Nous détaillons tout d'abord notre pipeline puis nous présentons quelques résultats.

3.1 Le framework en détail

Comme les pipelines présentés précédemment, notre framework est constitué de deux phases distinctes : une *broad phase* (qui s'applique sur tous les objets de la scène) et une *narrow phase* (qui ne fonctionne que sur des couples

[4] Axis Aligned Bounding Box
[5] Oriented Bounding Box

d'objets). Nous allons détailler ces deux phases constitutives de notre framework (voir figure 4).

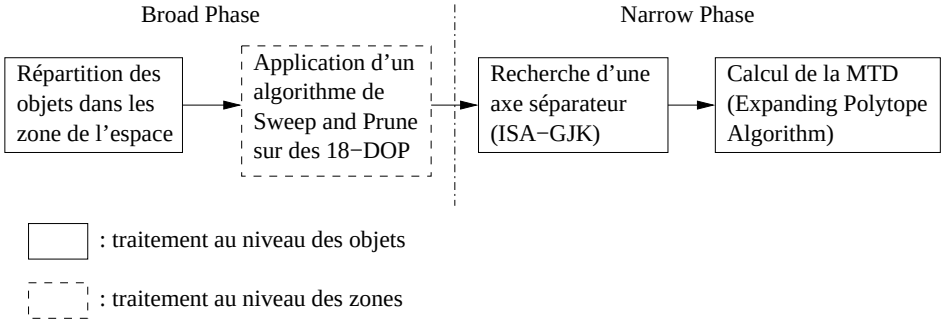


FIG. 4 – Pipeline de traitement de collisions.

3.1.1 Broad Phase

Cette première partie consiste à éliminer rapidement les couples d'objets qui ne sont pas en collision. Pour cela, nous subdivisions la totalité de l'espace en un certain nombre de cellules régulières. Dans chacune de ces zones se trouve un agent. Ces agents ont pour objectif de permettre un traitement plus rapide de la *broad phase* dans le cas d'environnements répartis : soit par un traitement parallèle (pour un environnement synchrone) soit de manière complètement autonome (pour un environnement asynchrone). La méthode la plus efficace de *broad phase* lorsque le nombre d'objets est elevé (plus d'une centaine d'objets) est le *Sweep And Prune* [CLMP95]. Cependant, il est couramment utilisé avec des AABB ce qui génère un nombre trop important de collisions potentielles (i.e. le rapport collisions potentielles sur collisions exactes est très elevé). L'idée est donc d'utiliser un volume englobant qui approxime mieux les objets, dont la détection et la construction soit rapide et qui soit compatible avec l'utilisation du *Sweep And Prune*. Nous avons donc choisi les k-DOP de [KHM+98, Zac98]. Ces volumes sont des extensions de *AABB* à $k/2$ axes (on peut considérer les *AABB* comme des 6-DOP). Plus k est grand et meilleure est l'approximation des volumes, mais plus coûteuse est la détection de ces volumes. C'est pourquoi nous nous sommes tournés vers les 18-DOP qui allie une assez bonne approximation de l'objet et dont la détection est assez rapide. La construction de ce genre de volume et leur mise à jour sont faites de manière rapide en utilisant les points de support définis en 2.1.1 : on calcule les points de supports des 9 axes et de leurs opposés ce qui nous permet d'obtenir les 18 paramètres du volume englobant. Ainsi par l'utilisation, dans chacune des zones, d'un algorithme de *Sweep And Prune* sur des 18- DOP, nous avons une *broad phase* rapide et qui génère beaucoup moins de collisions potentielles que l'algorithme classique basé sur des AABB. En sortie de cet étage du pipeline, les agents envoient à chaque objet l'identifiant des objets avec lesquels il est entré en collision. Chaque objet reçoit donc une liste d'identifiants et indique sa position courante à tous les objets désignés dans la liste d'identifiants (voir figure 5).

3.1.2 Narrow Phase

A partir des indications de position obtenues dans la phase précédente, chaque corps va résoudre ses collisions. Pour cela, il applique une variante de l'algorithme *GJK* : l'*ISA-GJK* [vdB98]. *ISA-GJK* (*ISA* pour *Incremantal Separating Axis*) comme son nom l'indique ne détermine pas la distance entre deux objets mais cherche l'existence d'un axe séparateur. Cet algorithme est très robuste, possède une convergence très rapide en temps constant (avec l'utilisation de la cohérence temporelle) et de meilleures performances que celui de Lin-Canny [LC91].
Si aucun axe séparateur n'a été trouvé, la dernière étape du framework est activée. La distance d'interpénétration est calculée en utilisant l'algorithme de van den Bergen [vdB01] qui par raffinement progressif d'une approximation de la différence de Minkowski peut déterminer une estimation de la *MTD*. A partir de cette distance d'interpénétration, il est possible de calculer une composante vectorielle permettant de déterminer la force de réaction (i.e. force de pénalité) en assimilant ce vecteur à l'allongement d'un ressort possédant une certaine raideur k (soit une expression de la force $\vec{F} = -k \cdot \vec{x}$ où $\vec{x}$ est une estimation de l'interpénétration).
Comme on peut le constater, si on considère deux objets en collision, la détection et le traitement associé sont calculés séparément par les deux objets concernés (i.e. A va traiter sa collision avec B et B sa collision avec A). Si la simulation est synchrone les forces de pénalités sont identiques (mais de sens opposé). Par contre, pour une simulation asynchrone où chaque corps fonctionne à une fréquence qui lui est propre, il est possible que la position

de A reçue par B ne soit pas la dernière calculée et donc les forces de pénalités générées ne sont pas nécessairement symétriques. Par conséquent, le principe d'action / réaction n'est pas nécessairement garanti.

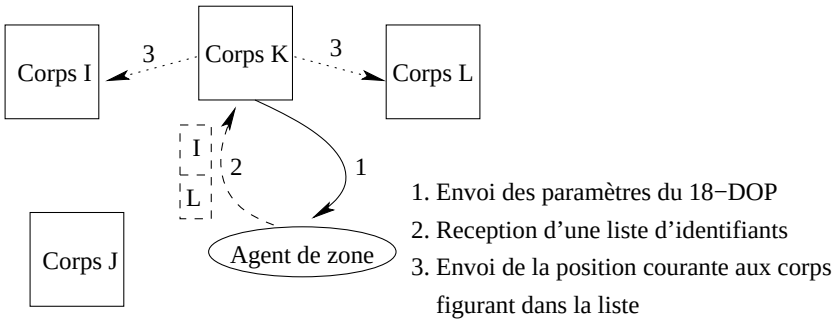


FIG. 5 – Informations échangées lors de la détection de collisions.

3.2 Résultats

La méthode proposée est naturelle et cohérente dans l'enchaînement des différentes étapes. Notre *Broad Phase* permet de réduire sensiblement les couples d'objets à tester par l'utilisation de volumes englobants approximant de manière assez fine les objets. De même l'utilisation des points de support n'ajoute qu'un faible surcoût à la construction de k-DOP par rapport à des volumes englobants plus simples. Les deux étapes de la phase de détection exacte s'imbriquent de manière logique et permettent de réutiliser certains calculs effectués par l'étage précédent (ex : l'obtention de la première approximation de $\mathcal{M}$ se fait lors de la recherche d'un axe séparateur).

Pour illustrer notre framework, nous avons choisi de simuler de manière très simple des corps rigides. A chaque pas de temps, les collisions sont résolues (*i.e.* calcul des forces de pénalités), un bilan de forces est effectué et les équations de mouvements sont intégrées (à l'aide de la méthode numérique d'Euler), afin de déterminer la position et la vitesse des objets. Cette résolution mécanique est faite elle aussi de manière autonome par chaque objet. La figure 6 montre des sphères tombant en chute libre sur quelques briques fixes. Les tests montrent que le temps de calcul moyen pour la résolution mécanique d'un système de 200 objets est de 16 ms sur un pentium IV 2Ghz (41 ms pour 600 objets).

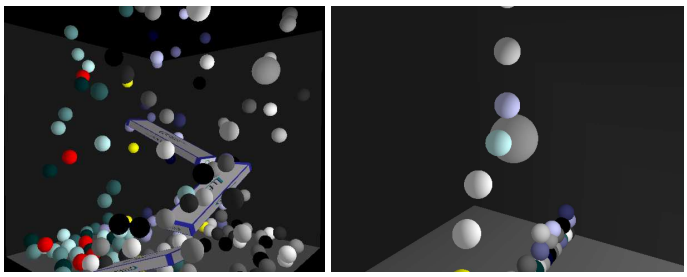


FIG. 6 – Simulation de corps rigides convexes.

Une autre particularité de notre framework est que nous n'avons pas opté pour une approche centralisée. Le fait que chaque objet détecte les collisions exactes avec d'autres objets sans passer par un serveur permet un portage dans un environnement virtuel distribué. Dans le cas où une telle implémentation était réalisée, les choix effectués pour notre framework ne sont pas en contradiction avec les contraintes d'un environnement réparti. Notamment, la stratégie d'avoir des volumes englobants assez complexes permet de réduire les nombres de messages échangés entre objets et aussi de diminuer le nombre de tests de détection exacte. De plus la résolution des collisions se faisant au niveau de chaque objet composant la scène, il est possible de rendre chaque objet autonome (l'objet résoud lui même ses collisions, ses équations de mouvement . . .) et donc d'engendrer un simulateur complétement distribué.

4 Conclusion

Dans cet article, nous avons présenté un modèle complet de traitements de collision. Ce modèle détermine les collisions et permet de générer des forces de pénalités en réponse à une collision. Il se limite à l'heure actuelle aux objets convexes et rigides mais est extensible aux objets déformables. Pour cela, il est nécessaire de complexifier le pipeline soit en utilisant la méthode de Fisher [FL01] qui permet de déterminer la distance d'interpénétration d'objets déformables en se basant sur les champs de distance et sur l'approche Level-Set [OS88]. Une autre possibilité est de sélectionner les facettes potentiellement en collision (voir [JSL99]).

Ce framework est le premier pas vers une simulation physique répartie. De nombreux problèmes existent encore pour avoir une simulation répartie effective : par exemple la gestion des interactions, la gestion de contraintes entre objets (*i.e.* objets articulés). De même la simulation devant se faire à une fréquence élevée (proche du kHz), de nombreux choix doivent être effectués pour garantir une simulation réaliste avec les limitations matérielles actuelles (temps de latence et bande-passante des réseaux).

Références

[Cam97] S. Cameron. Enhancing GJK : Computing minimum and pentration distances between convex polyhedra. In *International Conference on Robotics and Automation*, pages 3112–3117, 1997.

[Chu96] K. Chung. *An Efficient Collision Detection Algorithm for Polytopes in Virtual Environment*. PhD thesis, Department of Computer Science, University of Hong Kong, 1996.

[CLMP95] J. Cohen, M.C. Lin, D. Manocha, and M.K. Ponamgi. I-collide : An interactive and exact collision detection system for large-scale environments. In *ACM interactive 3D Graphics Conference*, pages 189–196, 1995.

[FL01] S. Fisher and M.C. Lin. Fast penetration depth estimation for elastic bodies using deformed distance field. In *Intelligent Robots and Systems*, 2001.

[GJK88] E.G. Gilbert, D.W. Johnson, and S.S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. In *IEEE Journal of Robotics and Automation*, volume RA-4, pages 193–203, 1988.

[JSL99] A. Joukhadar, A. Scheuer, and C. Laugier. Fast contact detection between moving deformable polyhedra. In *IEEE-RSJ Intelligent Robots and Systems*, 1999.

[JTT01] P. Jiménez, F. Thomas, and C. Torras. 3D collision detection : A survey. In *Computer Graphics*, volume 25, pages 269–285, 2001.

[KHM+98] J.T. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral, and K. Zikan. Efficient collision detection using bounding volume hierarchies of k-DOPs. In *T-VCG(4)*, pages 21–36, 1998.

[KLM02] Y.J. Kim, M.C. Lin, and D. Manocha. Deep : Dual-space expansion for estimating penetration depth between convex polytopes. In *IEEE International Conference on Robotics and Automation*, Mai 2002.

[LC91] M.C. Lin and J.F. Canny. A fast algorithm for incremental distance calculation. In *IEEE International Conference on Robotics and Automation*, 1991.

[LG98] M.C. Lin and S. Gottschalk. Collision detection between geometric models : A survey. In *IMA Conference on Mathematics of Surfaces*, 1998.

[LMCG96] M.C. Lin, D. Manocha, J. Cohen, and S. Gottschalk. Collision Detection : Algorithms and Applications. In *Algorithms for robotics motion and manipulation*, pages 129–142, 1996.

[Mes02] P. Meseure. *Animation basée sur la physique pour les environnements intéractifs temps réel : habilitation à diriger des recherches*. Université des Sciences et Technologies de Lille, 2002.

[OS88] S. Osher and J.A. Sethian. Fronts propagating with curvature-dependent speed : Algorithms based on Hamilton-Jacobi formulations. In *Journal of Computational Physics*, volume 79, pages 12–49, 1988.

[RKC02] S. Redon, A. Kheddar, and S. Coquillart. Fast continuous collision detection between rigid bodies. In *Eurographics*, 2002.

[vdB98] G. van den Bergen. A fast and robust GJK implementation for collision detection of convex objects. In *Journal of Graphic Tools*, volume 4(2), pages 7–25, 1998.

[vdB01] G. van den Bergen. Proximity queries and penetration depth computation on 3D game object. In *Game Developers Conference*, 2001.

[Zac98] G. Zachmann. Rapid collision detection by dynamically aligned DOP-trees. In *Proceedings of IEEE, VRAIS*, 1998.

[Zac01] G. Zachmann. Optimizing the collision detection pipeline. In *First International Game Technology Conference (GTEC)*, 2001.

Modélisation de sable 3D – Visualisation par structuration du flux

C. Guilbaud, A. Luciani

Laboratoire ICA/INPG
38031 Grenoble cédex

Claire.Guilbaud@imag.fr, Annie.Luciani@imag.fr

Résumé : *Un phénomène naturel se compose d'un ensemble de dynamiques complexes, de formes variables ainsi que d'une topologie en perpétuel mouvement. Tous ces composants le caractérisent. Pour modéliser un ensemble de phénomènes naturels, nous utilisons CORDIS-ANIMA un modeleur-simulateur générique basé sur les interactions entre particules. A l'aide de peu de paramètres, il est alors possible de créer une large gamme de phénomènes, sans connaître la physique sous-jacente. Nous modélisons des modèles allant du sable aux fluides turbulents. Les résultats des simulations sont constitués de l'ensemble des coordonnées des points mobiles du modèle physique. Ces points sont dispersés dans l'espace ; ils décrivent le flux de matière – ils ne sont pas sur la surface du phénomène modélisé. La visualisation des modèles n'est pas simple. Nous avons mis au point une méthode de construction par structuration du flux à partir d'informations partielles. Cette méthode nous a permis de calculer de nouvelles informations. Ainsi nous avons pu caractériser le flux de matière.*

Mots-clés : Animation par ordinateur, modèle physique particulaire, modélisation de phénomènes naturels, visualisation d'un nuage de points, surfaces implicites

1. Introduction

Modéliser un phénomène naturel, c'est reproduire ses dynamiques, sa forme qui permettent à un observateur de l'identifier sans peine. La méthode de modélisation choisie doit être robuste et générique pour être à même de simuler les dynamiques caractéristiques d'un phénomène. Plusieurs études se sont intéressées au développement de telles méthodes. Peu d'entre elles sont parvenues à réaliser un modeleur unique pour l'ensemble des phénomènes naturels. Au sein du laboratoire ICA, nous développons depuis de nombreuses années un modeleur-simulateur physique particulaire appelé CORDIS-ANIMA.

Ce modeleur est une méthode de modélisation physique, qui construit un objet physique, sans obligatoirement d'équivalent réel, en termes de réseau composé de masses ponctuelles et d'interactions. Le résultat d'une telle modélisation est un ensemble de points mobiles (appelées masses ponctuelles dans l'espace de modélisation, puis particules dans l'espace de visualisation) interagissant les uns avec les autres.

Notre savoir acquis au cours des années de développement du modeleur-simulateur, nous a permis de constater qu'un observateur est apte à reconnaître un phénomène simulé complexe par la simple visualisation des points du modèle physique, et cela même si ces points ne sont pas sur la surface de l'objet modélisé mais le constituent. Visualiser de telles données n'est pas chose simple, car nous devons élaborer un volume présentant les dynamiques et formes caractéristiques, si complexes soit-elles, du phénomène modélisé.

L'article se décompose ainsi : dans un premier temps nous expliquerons le formalisme CORDIS-ANIMA ainsi que la façon dont nous avons modélisé du sable. Ensuite, nous décrirons la méthode de structuration du flux qui permet d'obtenir des plus amples informations sur le nuage de points décrivant la simulation.

2. Modélisation par système physique particulaire

2.1. Travaux antérieurs

Logan et al. [LWA94] distinguent deux grandes classes de méthodes de modélisation : les méthodes contraintes et les méthodes physiques. Les méthodes physiques se scindent en deux catégories :

- Les méthodes qui résolvent les lois de la dynamique du phénomène (comme les CFD « Computational Fluid Dynamics), que l'on appellera modélisation de la physique. Elles sont en règle générale associée à un type de phénomène ;
- Les méthodes qui simulent le comportement physique des phénomènes, que l'on appellera modélisation physique. Elles peuvent modéliser un ensemble de phénomènes en gardant le même paradigme.

La modélisation physique est plus générique que les autres méthodes de modélisation, dans le sens où à partir d'un même formalisme, il est possible en faisant varier un ensemble de paramètres d'obtenir une large gamme de phénomènes. Dès 1973, Greenspan [Gre73] s'est penché sur la mise au point d'un modeleur générique pour simuler l'ensemble des états possibles de la matière. Un objet, un phénomène est alors représenté en termes de masses et d'interactions (interactions représentant la loi de potentielle non dissipative, la loi de Lennard-Jones). Les études postérieures aux travaux de Greenspan se distinguent par la manière dont est exprimée l'interaction (analytique ou discrète), et l'existence ou non d'un terme dissipatif.

Pour simuler un comportement thermo-conducteur, Terzopoulos et al. [TPF89] se servent d'une liaison élastique non-linéaire. Tonnesen [Ton91] a mis au point un système particulaire où l'évolution de l'interaction dépend d'une énergie thermale. Miller et Pearce [MP89] ont préféré adopter la loi de Lennard-Jones en introduisant dans la méthode un terme dissipatif. Dans le même temps, Luciani et al. [LJFCR91, LHM95, LHVD95, Luc00] ont montré qu'il est possible d'implémenter un modeleur générique unique à partir d'une expression discrète de la loi d'interaction et de la présence d'un terme dissipatif. Le modeleur-simulateur, CORDIS-ANIMA, utilise une interaction non-linéaire pour simuler des matériaux, des objets physiques, des phénomènes naturels.

2.2. Le formalisme CORDIS-ANIMA

CORDIS-ANIMA est un modeleur-simulateur physique particulaire qui emploie la loi de Newton pour faire évoluer un objet physique. L'objet est matérialisé par un réseau de masses ponctuelles (MAT) et d'interactions viscoélastiques non-linéaires (LIA). Un LIA comporte 4 paramètres pour exprimer le comportement recherché : une élasticité K, et son seuil S_K, une viscosité Z et son seuil S_Z. Le seuil élastique ou visqueux s'exprime comme une distance entre les masses ponctuelles reliées par une interaction. Si la distance entre deux masses ponctuelles est supérieure ou inférieure au seuil de l'interaction non-linéaire, alors l'élasticité ou la viscosité n'aura pas la même valeur.

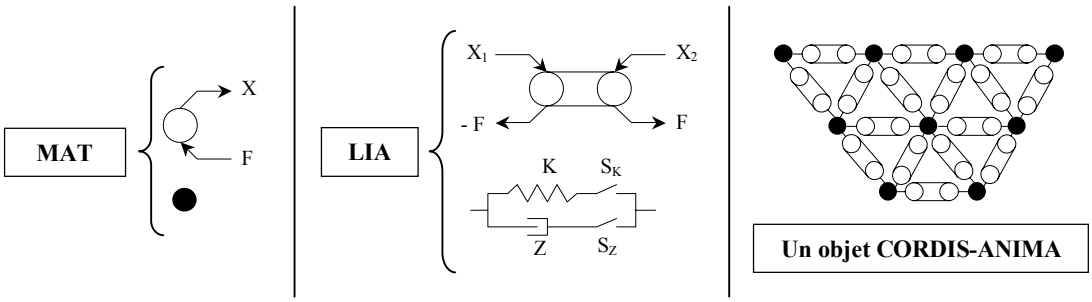


Fig. 1 : le formalisme CORDIS-ANIMA

Modéliser un phénomène naturel, c'est définir un modèle topologique du flux (spécification du nombre de MAT, de la manière dont ils sont reliés par des LIA, caractérisation des paramètres des LIA) ainsi que de son environnement physique. Un modèle topologique est un réseau de masses ponctuelles interconnectées par des liaisons viscoélastiques non-linéaires. Ce sont les paramètres des LIA qui vont régir le comportement du réseau. Ainsi il est possible d'obtenir une large gamme de phénomènes naturels sans avoir à connaître la physique sous-jacente des phénomènes.
Le résultat d'une simulation est, pour chaque instant simulé, l'ensemble des coordonnées des masses ponctuelles. Ces informations sont assimilables à un ensemble de points mobiles, dispersés dans l'espace sans information de topologie. Nous disposons uniquement de ces informations car nous avons voulu séparer la phase de modélisation de la phase de visualisation du processus d'animation par ordinateur.

2.3. Les modèles de sable

Nous avons réalisé un ensemble de modèles tridimensionnels de phénomènes naturels tels que les pâtes, le gel, mais nous ne présenterons ici que les modèles de sable. D'autres modèles ont été réalisés en 2D (pâte, fluide turbulent, fumée), et sont expliqués dans [LHVD95] et [Luc00]. Les modèles présentés ici sont basés sur des études préalablement faites par d'autres chercheurs en deux dimensions [LHM95].

Un tas de sable est une structure triangulaire, qui à mesure qu'elle augmente de taille se divise en sous-tas séparés par des lignes de force de cisaillement. Le tas s'accroît sous l'action de deux comportements chaotiques qui caractérisent une situation instable. Le premier est une avalanche de surface qui se produit lorsque les pentes

du tas de sable ont un angle supérieur à un angle caractéristique ; la seconde, une avalanche interne, advient lorsque les sous-tas sont instables et nécessitent une réorganisation de l'ensemble du tas.
Pour obtenir la formation d'un tas de sable le sol doit être rugueux ou l'environnement physique doit comporter des murs qui stoppe l'étalement du sable sur le sol lisse. Nous présentons ici deux modèles de sable, l'un à sol lisse, l'autre à sol rugueux. Ces deux modèles ont des caractéristiques, du point de vue de la modélisation, communes : la matière est constituée d'un ensemble de masses ponctuelles reliées entre elles par une interaction élastique. Seule la nature du sol que le flux va atteindre change.

Les résultats que nous présentons ici sont tridimensionnels. Les mêmes modèles ont été fait en deux dimensions ; ils nous ont permis de mettre au point les modèles 3D. Les modèles physiques 2D de sable s'écoulant sur un sol lisse ou sur un sol rugueux sont satisfaisants.

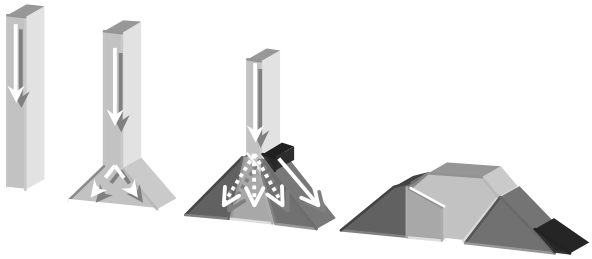


Fig. 2 : Caractéristiques dynamiques du sable

2.3.1. Modèle de sable à sol rugueux

L'environnement physique du modèle est composé d'un entonnoir et d'un sol rugueux. L'entonnoir est constitué d'un ensemble de MAT dégénérés (la position qu'ils renvoient est toujours la même quelles que soient les forces qui leur sont appliquées) positionnés de manière à former le haut d'un sablier (forme triangulaire ; 4 fois 4 MAT). Le sol rugueux est représenté par un ensemble de MAT dégénérés de positions fixes placés de manière aléatoire en ordonnée pour se situer de part et d'un plan horizontal (576 MAT). La schématisation du modèle physique se trouve à la Fig. 3.
Le flux de matière est composé d'un ensemble de MAT (500 MAT) reliés les uns aux autres par des interactions élastiques. Ici les masses ponctuelles sont reliées deux à deux, c'est à dire que si il y a n masses ponctuelles, l'une d'entre elle est reliée aux *n-1* autres, on parle alors d'agglomérat. Les masses ponctuelles sont initialement placées entre les MAT dégénérés symbolisant l'entonnoir : elles sont agencées de manière triangulaire.

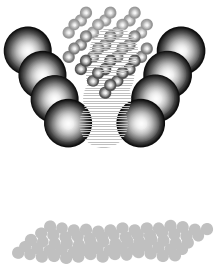


Fig. 3 : Modèle physique à sol rugueux

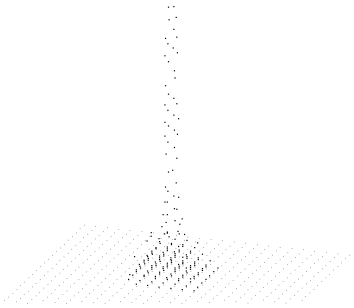


Fig. 4 : Image à t=300

Fig. 5 : Image à t=530

2.3.2. Modèle de sable à sol lisse

L'environnement physique du modèle est composé d'un entonnoir, d'un sol lisse ainsi que de quatre parois. L'entonnoir est constitué de 8 MAT dégénérés, le sol d'un seul MAT dégénéré, une paroi d'un MAT dégénéré.
Le flux de matière est composé d'un ensemble de MAT (900 MAT) interconnectés en agglomérat par des liaisons élastiques.

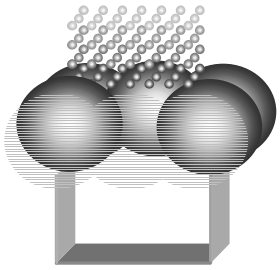


Fig. 6 : Modèle physique à sol lisse

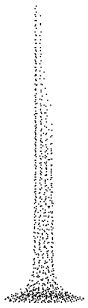


Fig. 7 : Modèle 1 - Image à t=450

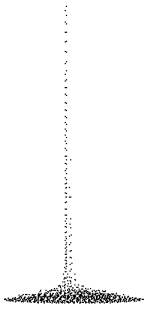


Fig. 8 : Modèle 1 - Image à t=700

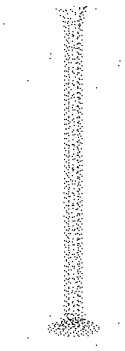


Fig. 9 : Modèle 2 - Image à t=466

Fig. 10 : Modèle 2 - Image à t=800

2.3.3. Conclusion

L'analyse des modèles 3D est complexe lorsqu'il s'agit d'apprécier des dynamiques internes et les types de réarrangements à stabilité limite. Le type d'interaction entre la 2D et la 3D est identique, nous espérions alors que les comportements dynamiques se ressemblent. Cela n'a pas été le cas dans le modèle à sol lisse : le nombre de masses ponctuelles constituant le flux de matière a été diminué, le tas résultant a donc couvert une surface moins importante, or les parois n'ont pas été rapprochées pour contenir les masses ponctuelles du flux. Le modèle ne ressemble plus alors à du sable, mais plutôt à de l'eau. Nous avons donc réalisé un second modèle où les parois sont plus proches. Le flux est alors correctement stoppé par les parois, le comportement de sable est ici intact. Néanmoins la forme du tas à sa base ne correspond pas à la réalité. Le modèle 3D du sable sur sol rugueux fonctionne comme nous nous y attendions malgré l'immobilisation d'une partie des masses ponctuelles dans l'entonnoir (formations de voûtes). Cependant nous ne pouvons distinguer d'effondrements internes, mais le tas est correctement formé.

Les études ont montré que pour obtenir un modèle de sable avec une formation de tas satisfaisante, surtout en trois dimensions, il faut que le sol soit rugueux. Car c'est l'interaction des masses ponctuelles avec le sol rugueux qui provoque le développement de l'amoncellement. Néanmoins, dans le but d'optimiser le temps de calculs des simulations, nous avons élaboré deux modèles de sable 3D à sol lisse. Le premier comporte des parois trop éloignées de la région d'évolution du flux de matière. On s'aperçoit alors que le modèle ne semble plus être un modèle de sable mais plutôt un modèle d'eau. Ce soucis de modélisation va nous permettre de mettre au point des modèles d'état intermédiaire entre le sable et l'eau. De plus, cela nous a montré que l'environnement physique joue un rôle important sur le comportement dynamique de l' objet physique.

3. Visualisation

Comme on peut le voir sur les images Fig. 4, Fig. 5, Fig. 7, Fig. 8, Fig. 9, Fig. 10, le résultat des simulations est un ensemble de points mobiles dispersés dans l'espace sans aucune information de structure.

3.1. Travaux antérieurs

Pour savoir comment nous allions habiller le nuage de points issu de la simulation, nous nous sommes intéressées à différentes méthodes de simulation de phénomène naturel et à des méthodes de construction de volume à partir d'un ensemble de points.

La seconde catégorie est composée de méthodes de reconstruction. Une méthode de reconstruction élabore une surface à partir d'un ensemble de points représentant la surface de l'objet à visualiser. Hoppe [Hop94] propose une méthode automatique de reconstruction de surface lisse à partir de données non bruitées et non organisées. Dans un premier temps, il estime la surface initiale puis optimise et lisse le maillage. Avec le même type de données initiales, Guo [GMW97] propose une méthode de reconstruction de surface en construisant un premier maillage à l'aide d'un graphe de voisinage (« alpha-shapes »), puis en le simplifiant. Ce type de méthode ne peut nous servir pour visualiser nos modèles car nos points n'étant pas répartis à la surface du flux, peu d'entre eux peuvent servir comme base à la construction d'un maillage, ce qui aboutirait à une surface pas assez détaillée. Les méthodes de reconstruction ne s'appliquent pas uniquement à des données éparpillées en surface de l'objet. Nullans [Nul98] reconstruit des structures géologiques à partir de données hétérogènes et incomplètes. Il assemble ses données selon leur diagramme de Voronoï. Le diagramme de Delaunay, dual du diagramme de

Voronoï, initialement bi-dimensionnel, a été entendu à la 3D par Joe [Joe91]. Les diagrammes de Voronoï donnent à chaque germe un espace propre. L'approche de Nullans serait intéressante pour nous, mais contrairement à lui nous ne disposons pas de suffisamment d'informations (nature géologique des germes qui caractérise les germes aux bords et à l'intérieur de la portion de sol en étude) pour organiser de manière adéquate nos points.

Les méthodes de simulation de phénomènes naturels effectuent pour la plupart la phase de modélisation et la phase de visualisation en parallèle. Tonnesen [Ton91] utilise des surfaces implicites pour habiller ses modèles de liquides simulés sur un système de particules sensibles à la chaleur. Gareau [Gar97], Stam [Sta97], Stora et al. [SACNG99] utilisent aussi une combinaison système particulaire-surfaces implicites pour visualiser leurs modèles. Foster et al. [FF01] utilise une méthode de génération de contour à partir de fonctions implicites pour représenter des liquides.

Toutes ces méthodes obtiennent un volume ou une surface pour l'objet qu'elles ont à représenter à partir de points. Or dans tous les cas, elles disposent d'informations supplémentaires permettant de caractériser au mieux l'objet à visualiser. Nous ne disposons pas de suffisamment d'informations morphologiques pour faire de même. Nous devons chercher davantage d'informations à partir d'un nuage de points pour être à même de l'habiller. La relation spatiale entre les particules de la matière évolue de manière complexe et change continuellement dans le temps.

3.2. Structuration du flux

Etant donné que lors des observations des résultats des simulations nous réalisons implicitement une structuration du flux, nous souhaitons faire de même pour visualiser le nuage de points représentant la matière. Donner une structure à un ensemble de particules sans information de topologie permet de caractériser le positionnement des particules les unes par rapport aux autres, par rapport à une surface implicitement définie.

Nous construisons dans un premier temps, un graphe qui s'apparente à un graphe de voisinage. Il connecte deux particules dont l'éloignement est inférieur à une distance donnée (Section 3.2.1). De ce graphe, nous dégageons un ensemble d'informations pertinentes (Section 3.2.2) qui nous permettront d'analyser un nuage de points (Section 3.2.3).

3.2.1. Graphe de voisinage

Lors de la conception d'un modèle physique, chaque interaction entre particules est définie par 4 paramètres, dont deux sont des indications des distances. Ces deux variables déterminent, suite à un ensemble de calculs, les positions des masses ponctuelles les unes par rapport aux autres. La distance entre particules varie au cours de la simulation. Nous extrayons une structure pour un nuage de points à partir uniquement de l'évolution des particules du flux, sans avoir d'information complémentaire sur le modèle physique.

Un graphe de voisinage attribue à chaque particule un voisinage. Il met en avant les singularités dynamiques du phénomène. Il est constitué d'un ensemble de connexions qui relient deux particules lorsque la distance qui les sépare est inférieure à un certain seuil. Ce seuil correspond à la distance moyenne entre couples de particules sur toute la longueur de la simulation divisée par un scalaire. Ce scalaire correspond à la finesse des dynamiques du phénomène (une sorte de caractérisation du niveau de détails).

Un graphe sera satisfaisant si le nombre de connexions est nécessaire et suffisant pour décrire le nuage de particules, sans masquer de dynamiques, de formes. Nous obtenons les résultats présentés à la Fig. 11 sur nos différents modèles de sable. En deux dimensions, on y détecte bien la formation d'un tas triangulaire caractéristique de l'écoulement de sable. Les singularités sont correctement représentées et visibles. Le graphe de voisinage représente d'une manière satisfaisante l'évolution des différentes dynamiques du phénomène. Les graphes de sable se caractérisent par un maillage régulier quasi-triangulaire. Les graphes de voisinage tridimensionnels sont difficiles à analyser. Cependant, on retrouve la structure habituelle d'un tas de sable (sauf pour le modèle de sable à sol lisse dont les parois sont trop éloignées).

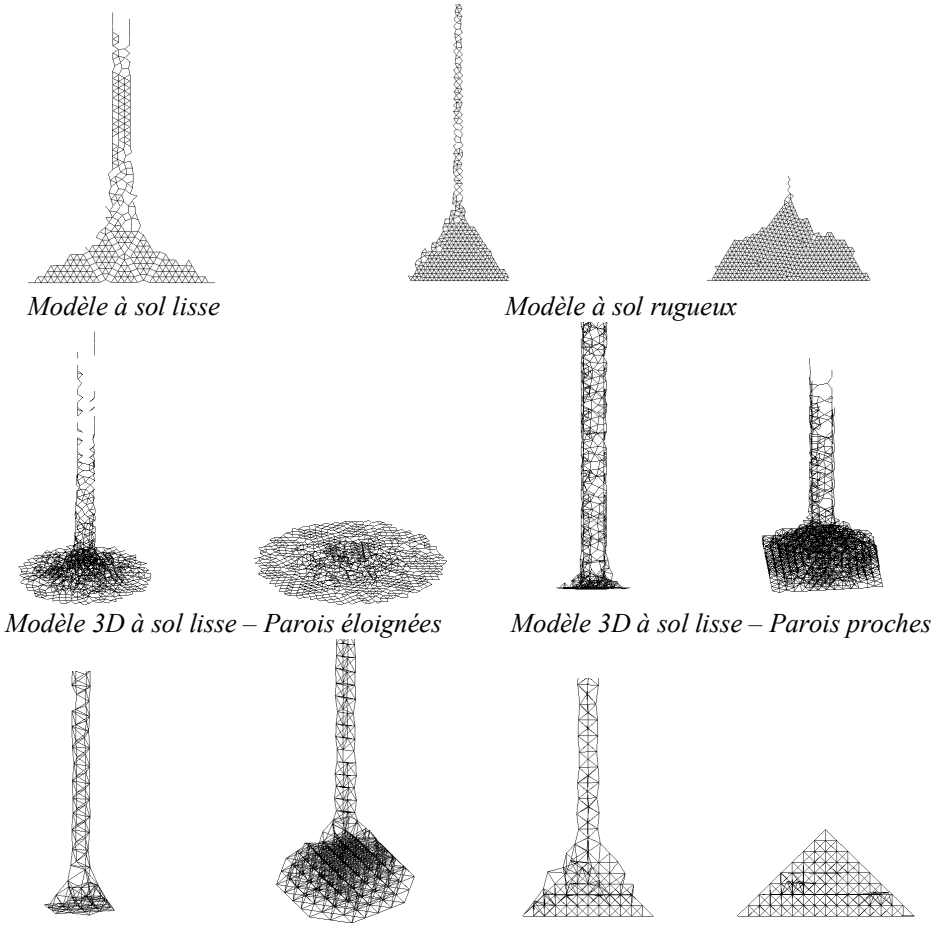


Fig. 11 : Quelques exemples de graphes de voisinage

3.2.2. Vecteur voisin moyen

Le graphe de voisinage est une manière de décrire la structure d'un nuage de points. Néanmoins il ne produit pas de surface ou de volume. Il nous faut donc calculer des informations permettant de révéler la distribution des particules dans l'espace. Les graphes de voisinages font apparaître des zones dépeuplées de particules, des régions sans connexions alors que mentalement nous en avions construites. La structure d'un flux de matière est constitué de sous-structures. Cette organisation traduit des caractéristiques pertinentes pour des dynamiques données.

Un vecteur voisin moyen est, pour une particule, la moyenne des différentes connexions - utilisées comme des vecteurs - de son voisinage. Il définit le gradient de la densité, dont l'opposé pointe vers l'extérieur de la structure principale (le flux) et des sous-structures. Cela est visible sur les images de la Fig. 13. Toutefois, comme pour l'analyse du graphe de voisinage en trois dimensions, une exploration approfondie du vecteur voisin moyen pour les modèles 3D est délicate. Toutefois nous remarquons que certaines particules ont un vecteur voisin moyen non nul tandis que pour d'autres, il est quasiment inexistant. Cela donne des informations sur la répartition des particules dans le nuage de points.

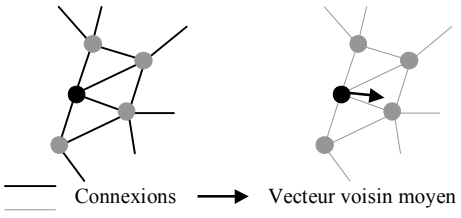


Fig. 12 : Définition d'un vecteur voisin moyen

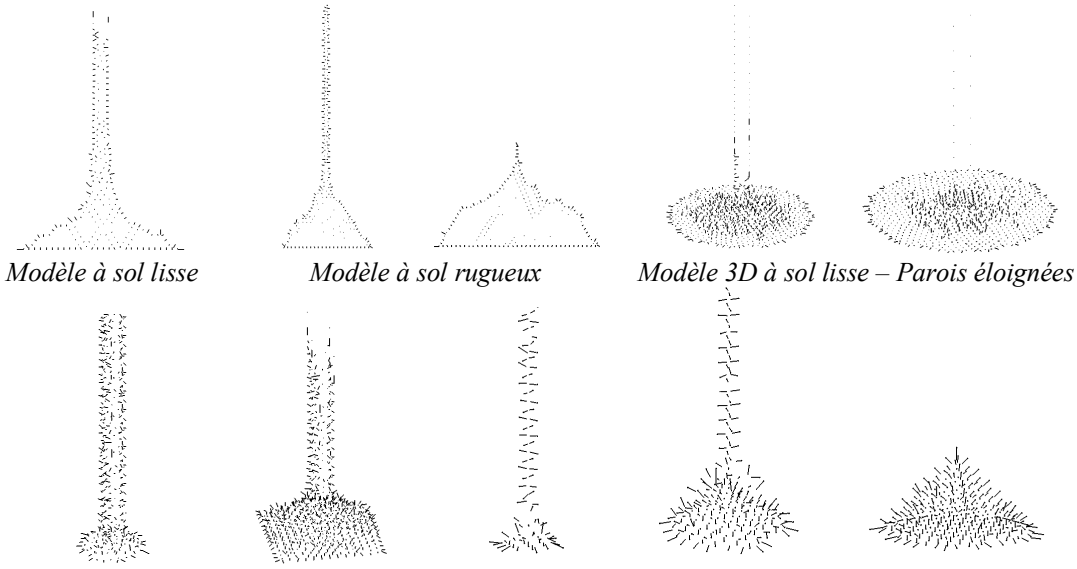


Fig. 13 : Quelques exemples de vecteurs voisins moyens

3.2.3. Répartition des particules dans le flux

Le vecteur voisin moyen est une classification initiale des particules (internes ou en bordure de flux). A partir de cette notion et du graphe de voisinage, nous avons établi un ensemble de règles pour déterminer la position d'une particule par rapport à la distribution des points dans l'espace. Nous avons fait un ensemble d'essais sur différents types de phénomènes (pas uniquement des phénomènes « élastiques »). Il en est ressorti que les particules que nous qualifions visuellement comme faisant partie de l'intérieur de la matière comportent plus de voisines et ont un vecteur voisin moyen quasiment nul. Pourtant certaines particules répondant à ces critères ne sont pas à l'intérieur du flux mais sur sa surface. L'observation des résultats et la modification des règles de base s'est effectué sur des indices subjectifs.

Comme on peut le voir (Fig. 14), les résultats sont satisfaisants : les particules sont correctement « étiquetées » suivant leur position dans le nuage de points (en blanc les particules internes, en noir les particules de surface). La structure régulière des tas de sable est manifeste. Si l'on ne visualise que les particules désignées comme étant en surface du nuage, le phénomène est reconnaissable sans peine.

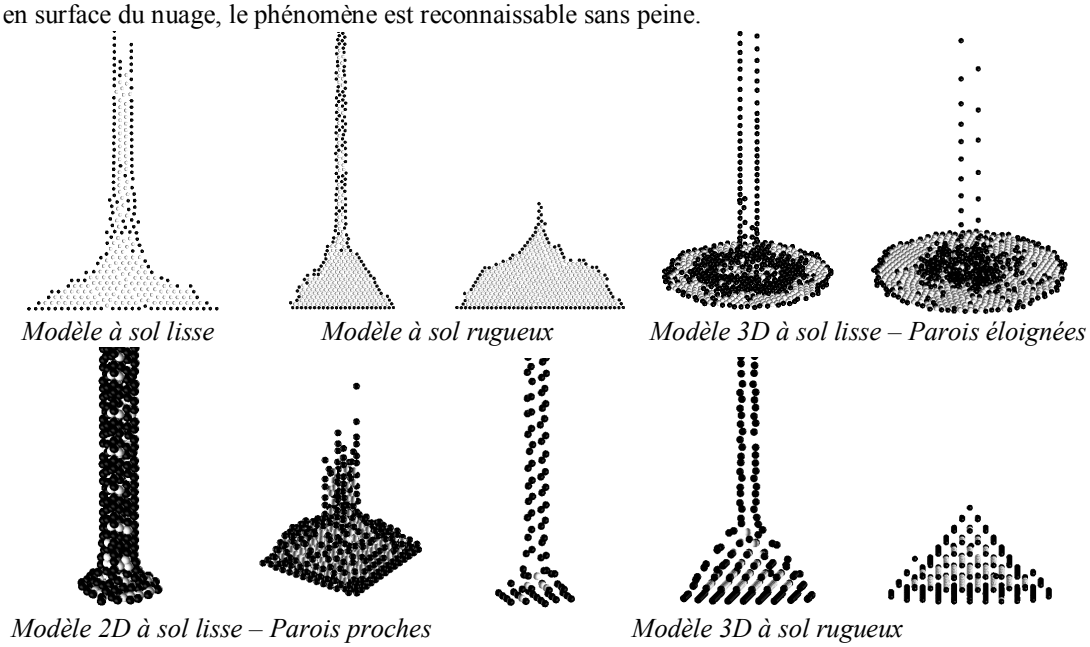


Fig. 14 : Quelques exemples de caractérisation de particules

4. Conclusion

Nous proposons des modèles de sable réalisés à l'aide de CORDIS-ANIMA, un modeleur-simulateur physique particulaire. Une fois l'environnement physique du modèle déterminé, il faut caractériser le comportement du flux de matière à l'aide de quatre paramètres (élasticité, viscosité, seuil élasticité, seuil viscosité). Le sable est un modèle dont les dynamiques pertinentes apparaissent avec des interactions entre masses ponctuelles purement élastiques. Ici nous ne proposons qu'un seul type de phénomène, mais nous avons réalisé d'autres modèles représentant des mouvements visqueux (pâte), et des modèles de morphogenèse.

Le résultat des simulations des modèles de sable à sol lisse ou à sol rugueux 2D et 3D sont les coordonnées des masses ponctuelles formant le modèle physique (masses de l'environnement ou de la matière) à chaque instant simulé. Nous ne disposons alors pas d'informations suffisantes permettant de construire un volume ou une surface pour ce nuage de particules dispersées dans l'espace. C'est pourquoi, nous avons analysé l'évolution des particules les unes par rapport aux autres pour caractériser la distribution des particules dans l'espace et pouvoir ainsi définir un volume.

Nous construisons un graphe de voisinage qui structure le flux en définissant pour chaque particule de la matière un voisinage. Ce graphe permet de mettre en avant les singularités des phénomènes représentés. Il présente une structure quasi-régulière de forme triangulaire. Cette information est insuffisante pour étudier la répartition d'une particule dans le nuage de points. Nous avons cherché à caractériser le positionnement des particules dans le nuage de points par le calcul de vecteurs voisins moyens. L'examen des images résultats montre que les particules que nous situons mentalement en bordure de surface ont un vecteur voisin moyen non nul, dirigé vers le centre de la matière. Avec cette nouvelle information pertinente, nous avons établi un ensemble de règles qui nous a permis de différencier les particules en bordure de surface, de celles à l'intérieur de la matière.

La dernière étape à réaliser serait la mise en place d'une méthode de rendu basée sur ces informations nouvellement acquise. Le volume, ou surface, construit ne devra en aucun cas masqué les différentes caractéristiques dynamiques et de forme du phénomène modélisé.

Références

[FF01] N. Foster, R. Fedkiw. Pratical animation of liquids. *Computer Graphics, Proceedings of SIGGRAPH'2001*. pp15--22. August 2001.

[Gre73] D. Greenspan. Discrete Models. Editor Addison-Wesley. reading in applied mathematics. 1973.

[Gar97] A. Gareau. Utilisation des systèmes de particules pour la simulation de phénomènes naturels – Présentation d'une architecture permettant l'intégration de systèmes animés hétérogènes. *Thèse de l'université de Lyon* 1. 1997.

[GMW97] B. Guo, J.P. Menon, B. Wilette. Surface reconstruction using alpha-shapes. *RC 20689, IBM Research Center*. Yorktown Heights. January 1997.

[Hop94] Hugues Hoppe. Surface reconstruction from unorganized points. *PhD thesis*. University of Washington (USA). 1994.

[Joe91] B. Joe. Construction of three-dimensional Delaunay triangulations using local transformations. *Computer Aided Design*, vol. 8, n°2, pp 123-142. May 1991.

[LWA94] L.P. Logan, D.P.M. Wills, N.J. Avis. Deformable objects in virtual environnements. *2nd UK VR-SIG Conference*. Silicon Graphics Reality Center, Theale (UK). December 1994.

[LJFCR91] A. Luciani., S. Jimenez., J.L. Florens., C. Cadoz., and O. Raoult. Computational physics: a modeler simulator for animated physical objects. *Proceedings of the European Computer Graphics Conference and Exhibition*. Vienne, Austria. September 1991. Ed. Elsevier.

[LHM95] A. Luciani, A. Habibi, and E. Manzotti. A multi-scale physical model of granular materials. *Proceedings of Graphics Interface'95.*

[LHVD95] A. Luciani, A. Habibi, A. Vapillon, and Y. Duroc. A physical model of turbulent fluids. *Computer Animation and Simulation, Eurographics'95*, September 1995.

[Luc00] A. Luciani. From granular avalanches to fluid turbulences through oozing pastes - A mesoscopic physically-based particle model. *Proceedings of GraphiCon*. (10):282—289. August 2000. Moscow, Russia.

[Nul98] S. Nullans. Reconstruction géométrique de formes – Application à la géologie. *Thèse de l'université de Sophia-Antipolis*. Décembre 1998.

[MP89] G. Miller A. Pearce. Globular Dynamics: a Connected Particle System for Animating Viscous Fluids. *Computers and Graphics*, 13(3):305—309. 1989. also in SIGGRAPH'89 Course notes number 30.

[Sta97] J. Stam. A general animation framework for gaseous phenomena. *ERCIM Research Reports ERCIM-01/97-R047*, ERCIM, VTT. Jaunary 1997.

[SACNG99] D. Stora, P.O.Agliati, M.P. Cani, F. Neyret, and J.D. Gascuel. Animating Lava Flows. *Graphics Interface* (GI'99) Proceedings. Pages 203-210. June 1999.

[TPF89] D. Terzopoulos, J. Platt, K. Fleisher. Heating and melting deformable models (from goop to glop). *Graphics Interface'89*. pp219—226. June 1989. London, Ontario.

[Ton91] D. Tonnesen. Modelling liquids and solids using thermal particles. *Graphics Interface'91*. pp255—262. June 1991. Calgary, AL.

Animation efficace de solides en contact par modèle physique

Galizzi Olivier & Faure François

iMAGIS-GRAVIR/IMAG 655 av. de l'Europe, 38330 Montbonnot

Olivier.Galizzi@imag.fr

Résumé : *Dans cet article, nous proposons une nouvelle approche au problème du calcul de réponses aux collisions permettant de manipuler interactivement plusieurs centaines de solides en contact. La méthode proposée est de type corrective, en ce sens ou elle intervient après le calcul des nouveaux états des solides selon les lois de Newton et sans tenir compte des collisions. Des contraintes sont ensuite appliquées, permettant de supprimer les interpénétrations et de corriger vitesses et accélérations des solides. Les trois corrections se font de façon itérative à l'aide d'un nouvel algorithme dérivé des algorithmes d'optimisation de la famille des gradients conjugués. Un tel algorithme de résolution nous permet de plus de régler un compromis entre précision et rapidité des calculs et ainsi de réaliser des animations peu précises mais rapides ou au contraire plus exactes mais également plus lentes.*

Mots-clés : simulation solides contacts

1 Introduction

La simulation de solides par modèle physique couvre un vaste champ d'application allant des effets spéciaux cinématographiques, aux programmes de jeux vidéos en passant par toutes sortes de simulations de type éboulement rocheux ou destruction d'empilements. Les scènes de la vie quotidienne sont également composées, pour la plupart, d'un grand nombre d'objets solides, articulés ou non et soumis aux lois de la gravité. Les méthodes d'animation par modèle physique permettent la conception d'animation réalistes de ce type de scènes, difficiles à générer par la main d'un animateur. C'est pourquoi ce domaine a été déjà largement exploré durant les quinze dernières années. Cependant les méthodes actuelles de simulation de solides par modèle physique se heurtent à la complexité des algorithmes utilisés (en $O(n^3)$ voire $O(n^4)$ où n est le nombre de solides en jeu dans la simulation). Leurs performances deviennent donc catastrophiques dès lors que ce nombre s'accroît de trop, et elles deviennent alors inutilisables pour des applications temps réel. Une nouvelle approche est donc nécessaire afin de palier à ce problème qu'est la trop grande complexité des scènes, avec pour objectif, de réaliser des simulations perceptuellement convaincantes, c'est à dire où les trajectoires des solides sont réalistes à l'œil même si elles ne sont pas physiquement parlant exactes.

Les problèmes majeurs auquels se sont heurtés toute une génération de chercheurs sont principalement la stabilité des simulations, notamment aux empilements de solides, mais également la lenteur des algorithmes utilisés étant donné leur complexité. Cependant de grands progrès ont déjà été faits depuis les premières méthodes dites de pénalité présentées en 1988 par Moore et Wilhelms dans [MW88] où des ressorts de longueur à vide nulle étaient placés ente deux solides en collision permettant ainsi de les faire ressortir de cet état incohérent. Les plus grandes avancées restent pourtant récentes. Ainsi c'est en 2000 seulement dans [Mir00] que Mirtich fut capable de gérer un grand nombre de solides en un temps raisonnable mais il supposait qu'ils étaient relativement bien espacés et répartis dans l'espace. En 2001, dans [MS01] Milenkovic proposa, lui, de synchroniser toutes les collisions ainsi que leur traitement à la fin de chaque pas de temps. Tous les états des solides en collisions étaient alors corrigés en même temps à l'aide d'algorithmes d'optimisation minimisant une énergie cinétique. Pour la première fois quasiment, une solution très stable au problème de l'animation de solides par modèle physique fut proposée. Ainsi Milenkovic fut capable d'empiler dix cubes les uns sur les autres. Cependant la méthode utilisée restait encore relativement lente et non interactive (1 image par seconde).

Stabilité ou rapidité personne n'a encore été capable de concilier ces deux aspects importants de la simulation de solides par modèle physique, d'où notre travail sur ces deux points essentiels.

2 Prérequis

2.1 Notations

Voici les quelques conventions et notations adoptées pour la rédaction de cet article :

– les lettres en caractères gras majuscules représentent des matrices
– les lettres en caractères gras minuscules représentent des vecteurs
– les lettres en caractères standard minuscules représentent des scalaires

De plus on pose $\mathbf{0}$ comme étant la matrice ou le vecteur nul de la taille appropriée.

Voici également les conventions utilisées pour manipuler les différents composants des solides notés S_i :

– le centre de gravité est noté o_i
– les points situés sur la surface du solide sont notés $p_1, p_2 \cdots, p_n$
– les coordonnées, la vitesse et l'accélération d'un point p_i sont notées respectivement $\mathbf{p_i}$, $\dot{\mathbf{p}}_\mathbf{i}$, $\ddot{\mathbf{p}}_\mathbf{i}$

2.2 La mécanique du solide

L'animation par modèle physique de solides peut être mathématiquement formulée comme étant l'intégration sur le temps des équations différentielles suivantes :

$$\dot{\mathbf{o}}_\mathbf{i} = \frac{do_i}{dt} \quad , \quad \ddot{\mathbf{o}}_\mathbf{i} = \frac{d\dot{o}_i}{dt} \tag{2.1}$$

$$m\ddot{\mathbf{x}} = \sum \mathbf{f_i} \quad , \quad \mathbf{I_M}\dot{\omega} = \sum \mathbf{c_i} \tag{2.2}$$

Les équations 2.1 traduisent la relation entre positions et vitesses ainsi qu'entre vitesses et accélération des solides. Les relations 2.2 traduisent les lois de Newton-Euler, où m est la masse du solide considéré, $\sum \mathbf{f_i}$ la somme des forces exercées sur le solide (gravité, forces de contraintes, forces élastiques ou visqueuses ...), $\mathbf{I_M}$ la matrice d'inertie du solide et $\sum \mathbf{c_i}$ la somme des couples appliqués au solide.

Nous modélisons la vitesse d'un solide au sein de son repère local par le vecteur de dimension six noté $\dot{\mathbf{x}}_\mathbf{i} = \left[\begin{array}{cc} \mathbf{o_i}^\mathbf{T} & \omega_\mathbf{i}{}^\mathbf{T} \end{array} \right]^T$ ou $\mathbf{o_i}$ dénote la vitesse à l'origine et $\omega_\mathbf{i}$ la vitesse angulaire du solide S_i. De même l'accélération peut être exprimée comme le vecteur $\ddot{\mathbf{x}}_\mathbf{i} = \left[\begin{array}{cc} \dot{\mathbf{o}}_\mathbf{i}^\mathbf{T} & \dot{\omega}_\mathbf{i}^\mathbf{T} \end{array} \right]$. La vitesse et l'accélération d'un point p_i lié au solide S_i s'expriment comme le montrent les relations 2.3 et 2.4.

$$\dot{\mathbf{p}}_\mathbf{1} = \dot{\mathbf{o}}_\mathbf{1} + \omega_\mathbf{1} \times \mathbf{o_1p_1} \tag{2.3}$$

$$\ddot{\mathbf{p}}_\mathbf{1} = \ddot{\mathbf{o}}_\mathbf{1} + \dot{\omega}_\mathbf{1} \times \mathbf{o_1p_1} + \omega_\mathbf{1} \times (\omega_\mathbf{1} \times \mathbf{o_1p_1}) \tag{2.4}$$

$$\mathbf{n}.\ddot{\mathbf{p}}_\mathbf{1} = \mathbf{n}.\ddot{\mathbf{o}}_\mathbf{1} + (\mathbf{o_1p_1} \times \mathbf{n}).\dot{\omega}_\mathbf{1} + \mathbf{n}.\omega_\mathbf{1} \times (\omega_\mathbf{1} \times \mathbf{o_1p_1}) \tag{2.5}$$

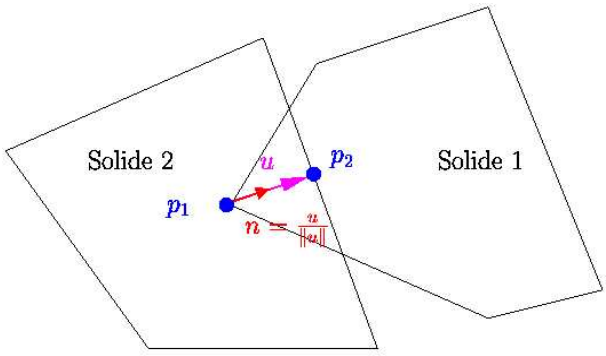
$$\mathbf{n}.(\ddot{\mathbf{p}}_\mathbf{1} - \ddot{\mathbf{p}}_\mathbf{2}) = \mathbf{j_1}\ddot{\mathbf{x}}_\mathbf{1} - \mathbf{j_2}\ddot{\mathbf{x}}_\mathbf{2} + \mathbf{c_1} - \mathbf{c_2} \tag{2.6}$$

L'accélération du point p_i projetée sur une direction de contrainte $\mathbf{n}$ (voir équation 2.5) peut également s'écrire $\mathbf{n}\ddot{\mathbf{p}}_\mathbf{i} = \mathbf{j_i}\ddot{\mathbf{x}}_\mathbf{i} + \mathbf{c_i}$ ou $\mathbf{j_i} = \left[\begin{array}{cc} \mathbf{n}^\mathbf{T} & (\mathbf{o_1p_1} \times \mathbf{n})^\mathbf{T} \end{array} \right]$. Dans le cas de solides en collision, cette direction $\mathbf{n}$ et celle portée par le vecteur d'extraction modélisé lors de la détection de collisions (voir figure 1). À partir de cette formulation, il est facile de trouver l'accélération de pénétration des deux solides S_1 et S_2 aux points p_1 et p_2 comme le montre la relation 2.6.

3 Formalisation des contraintes

3.1 Ecriture des contraintes

Comme mentionné dans l'introduction, nous utilisons des contraintes afin de corriger les états de solides et remettre le système dans un état cohérent. Dans cette section, nous montrons comment poser les contraintes pour la



Deux solides en collision

FIG. 1 – Modélisation d'une collision. Dû à l'intégration discrète du temps, les solides ne sont pas en contact mais en interpénétration. Le vecteur $\mathbf{u}$ tel qu'il est représenté est appelé vecteur d'extraction.

correction des accélérations des solides. L'objectif est d'annuler l'accélération de pénétration entre deux solides S_1 et S_2 en collision. Pour cela on va donc chercher, en partant des valeurs de $\ddot{\mathbf{p}}_\mathbf{1}$ et $\ddot{\mathbf{p}}_\mathbf{2}$, les valeurs corrigées $\ddot{\mathbf{p}}_{\mathbf{1_{corr}}}$ et $\ddot{\mathbf{p}}_{\mathbf{2_{corr}}}$ satisfaisant la contrainte 3.1.

$$\mathbf{n}(\ddot{\mathbf{p}}_{\mathbf{1_{corr}}} - \ddot{\mathbf{p}}_{\mathbf{2_{corr}}}) \quad = \quad \mathbf{0} \tag{3.1}$$

On va donc chercher les $\mathbf{\Delta\ddot{p}_1}$ et $\mathbf{\Delta\ddot{p}_2}$ tels que en posant :

$$\ddot{\mathbf{p}}_{\mathbf{1_{corr}}} = \ddot{\mathbf{p}}_\mathbf{1} - \mathbf{\Delta\ddot{p}_1} \quad \text{et} \quad \ddot{\mathbf{p}}_{\mathbf{2_{corr}}} = \ddot{\mathbf{p}}_\mathbf{2} - \mathbf{\Delta\ddot{p}_2}$$

on ait la condition 3.1 à vrai. Pour cela, on veut donc que :

$$\mathbf{n}(\ddot{\mathbf{p}}_{\mathbf{1_{corr}}} - \ddot{\mathbf{p}}_{\mathbf{2_{corr}}}) = \mathbf{n}(\ddot{\mathbf{p}}_\mathbf{1} - \mathbf{\Delta\ddot{p}_1} - \ddot{\mathbf{p}}_\mathbf{2} + \mathbf{\Delta\ddot{p}_2}) = \mathbf{0}$$

C'est à dire que :

$$\mathbf{n}(\mathbf{\Delta\ddot{p}_1} - \mathbf{\Delta\ddot{p}_2}) = \mathbf{n}(\ddot{\mathbf{p}}_\mathbf{1} - \ddot{\mathbf{p}}_\mathbf{2})$$

Soit en utilisant la relation 2.6 :

$$\mathbf{j_1}\ddot{\mathbf{x}}_\mathbf{1} + \mathbf{c_1} - \mathbf{j_2}\ddot{\mathbf{x}}_\mathbf{2} - \mathbf{c_2} = \mathbf{j_1}\mathbf{\Delta\ddot{x}_1} - \mathbf{j_2}\mathbf{\Delta\ddot{x}_2}$$

En remarquant que :

$$\ddot{\mathbf{x}}_\mathbf{1} = \mathbf{M_1^{-1}f_{ext}} \quad \text{et} \quad \ddot{\mathbf{x}}_\mathbf{2} = \mathbf{M_2^{-1}f_{ext}} \tag{3.2}$$

Et en notant :

$$M = \begin{bmatrix} \mathbf{M_1} & 0 \\ 0 & \mathbf{M_2} \end{bmatrix} \quad , \quad J = \begin{bmatrix} \mathbf{j_1} & -\mathbf{j_2} \end{bmatrix}$$
$$\mathbf{\Delta\ddot{x}} = \begin{bmatrix} \mathbf{\Delta\ddot{x}_1} \\ \mathbf{\Delta\ddot{x}_2} \end{bmatrix} \quad , \quad c = c_1 - c_2$$

On peut écrire :

$$\mathbf{JM^{-1}f_{ext}} + \mathbf{c} = \mathbf{J\Delta\ddot{x}} \tag{3.3}$$

Or on sait que :

$$\mathbf{\Delta\ddot{x}_1} = \underbrace{\mathbf{M_i^{-1}} \quad \underbrace{\mathbf{J_i^T}\, f_{12}}_{\text{force } f_{12} \text{ exprimée en } o_1}}_{\text{variation d'accélération induite en } o_1 \text{ par } f_{12}} \tag{3.4}$$

Et de même pour $\mathbf{\Delta\ddot{x}_2}$, où f_{12} (respectivement $-f_{12}$) est une force appliquée au point p_1 (respectivement p_2) selon une direction qui est la direction n de pénétration.

Le problème n'est plus alors de trouver $\mathbf{\Delta\ddot{p}_1}$ et $\mathbf{\Delta\ddot{p}_2}$ mais le scalaire f_{12} vérifiant le système d'équation suivant :

$$\begin{aligned} \mathbf{JM^{-1}J^T f_{12}} &= \mathbf{-JM^{-1}F_{ext} + c} \\ &= -\text{accélération de pénétration de } p_1 \text{ et } p_2 \\ &= -\text{erreur à corriger} \end{aligned} \tag{3.5}$$

On peut noter que la force f_{12} ne travaille pas, c'est à dire qu'elle ne génère pas de mouvement. Elle est là, uniquement pour garantir la contrainte posée (i.e. annuler l'accélération de pénétration) : elle correspond, en fait, au multiplicateur de lagrange de cette même contrainte.

3.2 Cas général

Dans le cas général de n solides en jeu dans la simulation, les matrices $\mathbf{J}$ et $\mathbf{M}$ sont de la forme 3.6 et 3.7.

$$\mathbf{J} = \begin{bmatrix} \bullet & \bullet & \mathbf{j_i} & \bullet & \mathbf{-j_k} & \bullet \\ \bullet & \mathbf{j_l} & \bullet & \bullet & \mathbf{-j_m} & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \mathbf{j_n} & \bullet & \mathbf{-j_o} \\ \mathbf{j_0} & \bullet & \bullet & \mathbf{-j_p} & \bullet & \bullet \end{bmatrix} \tag{3.6}$$

$$\mathbf{M} = \text{diag}(M_1, M_2, \cdots, M_n) \tag{3.7}$$

La matrice $\mathbf{M}$ est de taille $6s \times 6s$ ou s est le nombre de solides, et $\mathbf{J}$ a pour taille $c \times 6s$ ou c est le nombre de contraintes. Le blocs nuls sont représentés par des $\bullet$. Sur chaque lignes de $\mathbf{J}$ seulement $\mathbf{j_i}$ et $\mathbf{j_k}$ sont non nuls lorsque les solides S_i et S_k ont un point de collision détecté. Chaque ligne correspond donc à une contrainte et les blocs non nuls aux solides contraints.

Dans le cas général la correction des accélérations revient à résoudre un système linéaire de la forme $\mathbf{Af = b}$ ou $\mathbf{A = JM^{-1}J^T}$ et $\mathbf{b}$ égal à $\mathbf{JM^{-1}F_{ext} + c}$. Le vecteur $\mathbf{f}$ est composé de forces exprimées en $kg.m.s^{-2}$ et correspond au vecteur des multiplicateurs de lagrange associés aux contraintes (de façon similaire à la méthode de Baraff exposée dans [Bar96] mais appliqué ici au cas des solides en contact et non pas seulement au cas des solides articulés).

4 Résolution de $\mathbf{JM^{-1}J^T f \geq b}$

4.1 Gradient bi-conjugué modifié

La matrice $\mathbf{A = JM^{-1}J^T}$ de notre système est relativement large. Elle se compose en effet, d'autant de colonnes qu'ils y a de solides en jeu dans la simulation, et, d'autant de ligne qu'il y a de contraintes, donc de collisions. Ses dimensions exèdent donc souvent plusieurs centaines de lignes et colonnes. Pour être à même de résoudre un tel système plusieurs fois par seconde afin d'atteindre notre objectif de temps réel, une approche classique est impossible. Nous avons donc développé un nouvel algorithme basé sur l'algorithme d'optimisation du gradient bi-conjugué (voir [PTVF93] chapitre *Sparse Linear Systems* pages 83 à 89) qui résoud un système d'équations linéaires $\mathbf{Af + b = 0}$ par minimisation itérative de $(\mathbf{Af + b})^2$. Cet algorithme itératif effectue comme seul calcul coûteux, deux produits matrices vecteurs par itérations, ce qui nous permet d'exploiter au mieux le fait que $\mathbf{J}$ et $\mathbf{M^{-1}}$ sont creuses alors que $\mathbf{A}$ ne l'est pas. Nous effectuons pour cela, le produit $\mathbf{A}f$ en trois étapes $O(n)$ en ne calculant jamais explicitement $\mathbf{A}$.

Cependant des conditions supplémentaires doivent être ajoutées au système afin de prendre en compte le sens physique des grandeurs manipulées (d'où l'inégalité à la place du signe égal dans le titre de la section). Considérons un ensemble fini de collisions entre deux solides S_1 et S_2, modélisées chacunes par deux points $\mathbf{p_{i1}}$ et $\mathbf{p_{i2}}$ et un vecteur d'extraction $\mathbf{n_i = p_{i1} - p_{i2}}$ donnant une direction normale au contact et selon lequel le mouvement est contraint. Soit $a_i = \mathbf{n_i}.(\mathbf{\ddot{p}_{i_1} - \ddot{p}_{i_2}})$ l'accélération de pénétration de S_1 et S_2 et f_i la force de contact agissant entre ces mêmes corps selon l'axe $\mathbf{n_i}$ (voir figure 1).

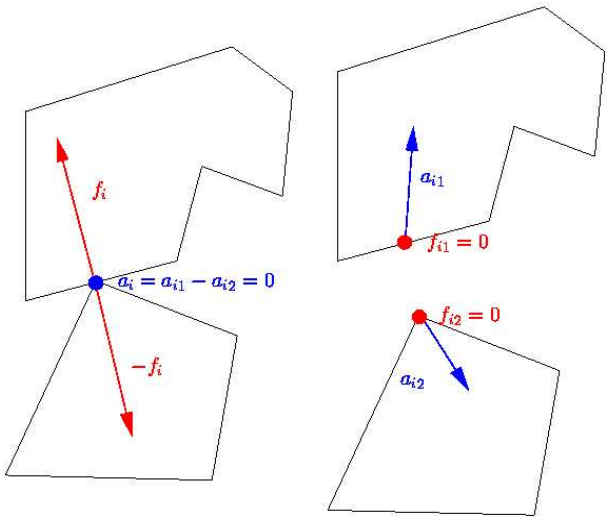


FIG. 2 – Dans le cas sans frottement deux cas de figures exclusifs sont possible. S_1 et S_2 ne peuvent pas se rapprocher ($a_i < 0$) mais sont libres de s'éloigner l'un de l'autre ($a_i \geq 0$). D'un autre coté, les forces de contacts f_i sont toujours répulsives (par convention on dira $f_i \geq 0$). Bien évidemment, soit les deux corps sont dans un état de contact actif et leur accélération de pénétration est nulle mais il existe une force répulsive donc positive qui maintient le contact et empêche une plus profonde pénétration (à gauche), soit les deux corps ne sont plus en contact et aucune force n'agit entre eux mais leur accélération relative est positive et non nulle (à droite).

Pour chaque collision i, les trois contraintes expliquées figure 2 se résument respectivement par les trois conditions suivantes :

$$a_i \geq 0 \qquad f_i \geq 0 \qquad f_i a_i = 0$$

Or, puisque a_i et f_i sont contraintes positives pour tout i, on a :

$$f_i a_i = 0 \Leftrightarrow \mathbf{f}^T \mathbf{a} = \mathbf{0}$$

Le problème du calcul des forces de contact peut donc se ramener à la formulation suivante, connue sous le nom de problème linéaire complémentaire (LCP) :

$$\begin{cases} \mathbf{Af + b} & \geq & \mathbf{0} \\ \mathbf{f} & \geq & \mathbf{0} \\ \mathbf{f^T(Af + b)} & = & \mathbf{0} \end{cases} \tag{4.1}$$

C'est lors de la résolution de ce LCP qu'entre en jeu notre algorithme, basé sur celui du gradient bi-conjugé, et qui peut prendre en compte ces nouvelles conditions. Nous utilisons pour cela la notion d'ensemble actif. Considérons un ensemble de n collisions. Nous dirons pour chaque contact qu'il est soit *actif* et $f_i \geq 0,\ a_i = 0$ soit *inactif* et $a_i \geq 0,\ f_i = 0$ selon les conditions qu'il vérifie. Nous construisons donc deux partitions, nommées A pour *actif* et IA pour *inactif*. Au début de chaque résolution, nous avons choisi pour heuristique de mettre tous les contacts dans la classe A. Les itérations du gradient, ne sont maintenant effectuées que sur les équations appartenant à la classe A. Ces deux classes peuvent cependant évoluer lors de la résolution du système, donc au fil des itérations. En effet, comme le montre la figure 3, l'algorithme de gradient cherche à faire le meilleur compromis, afin de satisfaire au mieux toutes les équations. Cependant, il n'est pas toujours possible de les satisfaire toutes en même temps. A chaque itération, et pour chaque contrainte, nous appliquons donc l'automate représenté figure 4. Ainsi, pour chaque contact de la classe A, si la force associée est devenue attractive ($f_i < 0$), nous passons ce contact dans la classe IA et nous n'en tenons plus compte. Au contraire, si pour un contact de la classe IA, nous détectons que l'accélération de pénétration devient négative ($a_i < 0$), alors nous repassons ce contact dans la classe A. Bien évidemment, pour chaque itération où il y a eu un changement de classe, l'algorithme du gradient doit être redémarré, car la dimension du système a changé par l'ajout ou la suppression d'une ou plusieurs contraintes (i.e. d'équations). Comme solution initiale du nouveau système, nous utilisons alors simplement la valeur calculée lors de la dernière itération.

L'avantage de notre méthode est qu'elle intervient au coeur même de l'algorithme du gradient. Nous construisons en effet, les partitions A et IA à la volée et n'avons ainsi pas besoin de résoudre un sytème d'équations linéaires entier à chaque itération pour les mettre à jour, comme cela a été le cas jusqu'a présent (voir algorithmes ci-dessous).

Jusqu'à présent : résolution d'un sytème entier avant de pouvoir mettre à jour les partitions A et IA. chaque résolution de $\mathbf{Af} + \mathbf{b} = \mathbf{0}$ est en $O(n^2)$ et l'on fait au minimum $O(n)$ itérations d'où un algorithme au mieux en $O(n^3)$.

Procedure resoudreSysteme1
initialisation
$\mathbf{f_{old}} = \mathbf{0}$
Tantque pas resolu **faire**
    $\mathbf{f_{new}}$ = resoudre $\mathbf{Af} + \mathbf{b} = \mathbf{0}$ sur A $\quad (\geq O(n^3))$
    $\mathbf{f_{new}} = \mathbf{f_{old}} + \alpha(\mathbf{f_{new}} - \mathbf{f_{old}})$
    (A,IA) = mise à jour de (A,IA)

Où α est le plus grand pas qu'il est possible d'effectuer tout en respectant les inégalités.

Notre algorithme : Mise à jour de A et IA à chaque itération du gradient bi-conjugué. À chaque itération (1 itération = chercher direction + longueur pas) on met à jour les partitions. On effectue donc $O(n)$ itérations en $O(n)$ d'ou un algorithme en $O(n^2)$ au maximum.

Procedure resoudreSysteme2
initialisation
$\mathbf{f_{old}} = \mathbf{0}$
Tantque pas resolu **faire**
    d = nouvelle direction de recherche $\quad (O(n))$
    β = longueur du pas à effectuer
    $f_{new} = f_{old} + \beta\mathbf{d}$
    (A_{new},IA_{new}) = mise à jour de (A,IA)
    Si $A_{new} \neq A$ **alors**
        réinitialisation
    (A,IA) = (A_{new},IA_{new})

Où β est la longueur du pas standard du gradient bi-conjugué.

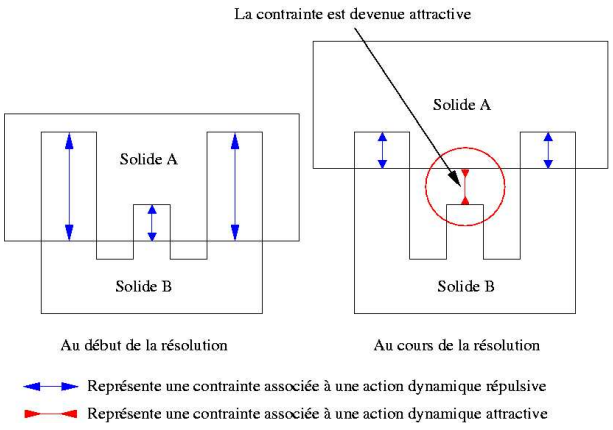


FIG. 3 – L'algorithme de gradient conjugué cherche un meilleur compromis satisfaisant au mieux toutes les conditions. Dans cet exemple au cours de la résolution une force deviendra inévitablement attractive : notre algorithme va donc la transférer dans la classe IA.

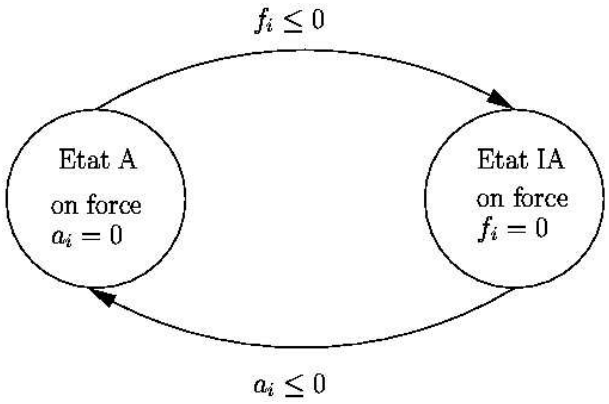


FIG. 4 – Automate de transition entre les classes A et IA

5 Performance et convergence

5.1 Convergence

Etant donné que les forces solutions du système $\mathbf{Af} = \mathbf{b}$ correspondent aux multiplicateurs de lagrange associés aux contraintes, l'algorithme converge vers une solution unique si et seulement si la matrice $\mathbf{A} = \mathbf{JM^{-1}J^T}$ est définie positive. Ce n'est pas toujours le cas, notamment lorsque se créent des cycles (par exemple une chaine fermée de solides articulés), pourtant nous n'avons pas remarqué de problèmes liés à cette non unicité de la solution dans les scènes que nous avons manipulées.

Quoi qu'il en soit, et c'est le point principal, notre algorithme nécéssite très peu d'itérations pour converger vers un résultat visuellement correct. En effet, un algorithme de type gradient conjugué trouve la solution exacte d'un système en autant d'itérations qu'il y a d'inconnues. Or dans notre cas, pour un système contenant près de 300 inconnues (i.e. 300 collisions), une dizaine d'itérations peuvent suffir pour obtenir un résultat très correct. Chaque itération s'effectuant en temps linéaire, notre algorithme qui a une complexité au pire en $O(n^2)$ voit donc cette complexité descendre très près de $O(n)$.

5.2 Performance

Voici quelques résultats de performance que nous avons obtenus pour différents types de simulation (des captures d'écran des simulations correspondantes sont fournies en annexe A) : on notera que les solides sont constitués d'assemblages de sphères pour simplifier la détection de collisions, ce qui n'affecte en rien la validité de notre méthode étant donné que la dynamique d'un solide ne dépend pas de sa géométrie.

	# solides - sphères	# collisions	fps
Pendule	11 - 61	10	300
125 sphères	126 - 126	250	45
216 sphères	217 - 217	400	20
343 sphères	344 - 344	700	10
Chaine	13 - 361	30	50
Tourniquet	126 - 602	700	10
Pile	14 - 405	120	30

TAB. 1 – Complexité de quelques scènes et efficacité de la méthode (captures d'écran sont disponibles annexe A).

6 Application à la correction des positions et vitesses

Afin de calculer la correction à effectuer sur les accélérations des solides nous avons procédé comme suit :

$$\begin{aligned}
\mathbf{f} &= \text{forces } f_i \text{ à appliquer au niveau de chaque collision selon l'axe de contrainte} \\
\mathbf{J^T f} &= \text{forces } f_i \text{ exprimées au centre de gravité } o_i \text{ des solides} \\
\mathbf{M^{-1} J^T f} &= \text{variations d'accélérations linéaires et angulaires induites en } o_i \text{ par les } f_i \\
\mathbf{J M^{-1} J^T f} &= \text{variations d'accélérations de pénétration induites aux points de collisions pas les } f_i
\end{aligned}$$

Et l'on résoud le système suivant :

$$\mathbf{J M^{-1} J^T f + c = \text{- erreur sur les accélérations}}$$

Pour la correction des vitesses, on a montré que l'on peut procèder de même, en résolvant le système suivant :

$$\begin{aligned}
\mathbf{J M^{-1} J^T} \pi + \mathbf{c} &= \mathbf{J \dot{x}} \\
&= \mathbf{-\text{erreur sur les vitesses}}
\end{aligned} \tag{6.1}$$

où π est un vecteur d'impulsions ($kg.m.s^{-1}$) et va permettre d'appliquer une variation instantanée de vitesse sur les solides selon le même schéma que précédemment :

$$\begin{aligned}
\pi &= \text{impulsions } \pi_i \text{ à appliquer au niveau de chaque collision selon l'axe de contrainte} \\
\mathbf{J^T} \pi &= \text{impulsions } \pi_i \text{ exprimées au centre de gravité } o_i \text{ des solides} \\
\mathbf{M^{-1} J^T} \pi &= \text{variations de vitesses linéaires et angulaires induites en } o_i \text{ par les } \pi_i \\
\mathbf{J M^{-1} J^T} \pi &= \text{variations de vitesse de pénétration induites aux points de collisions pas les } \pi_i
\end{aligned}$$

Pour ce qui est de la correction des positions, les contraintes sont maintenant géométriques et l'on va travailler sur des approximations. En effet, afin de pouvoir corriger l'orientation et la position des solides il est nécéssaire de linéariser les rotations. Nous manipulons donc, dans ce dernier cas, un vecteur δ d'action de déplacement ($kg.m$) qui vont induire des déplacements instantanés sur les solides afin de les faire ressortir de leur état d'interpénétration. Ceci peut se formuler comme suit :

$$\begin{aligned}
\mathbf{J M^{-1} J^T} \delta &= \mathbf{n(p_1 - p_2)} \\
&= \mathbf{-\text{profondeurs pénétrations}}
\end{aligned} \tag{6.2}$$

On remarquera que la correction que nous apportons sur les positions, vitesses et accélérations, se calcule de manière analogue et résolvant un système d'équations linéaires du type

$$\mathbf{J M^{-1} J^T} \lambda = \mathbf{-erreur}$$

où λ est respectivement un vecteur d'actions de déplacement, un vecteur d'impulsions et un vecteur d'accélérations. Nous avons donc un seul et même algorithme pour les trois corrections.

La gestion du rebond se fait via le modèle de Poisson, simplement en remplaçant le vecteur π calculé, par le vecteur $(1+\epsilon)\pi$, où $0 \leq \epsilon \leq 1$ est le coefficient de rebond.

7 Extension aux solides articulés

L'extension du simulateur à la gestion des solides articulés ne pose pas de problème particulier avec notre méthode. Par exemple, une liaison de type point-point se modélise avec trois contraintes scalaires dont chaque direction est alignée sur un des trois axes principaux $\mathbf{n}$, $\mathbf{t}$ et $\mathbf{s}$ d'un repère orthogonal (voir figure 5). Pour simplifier les calculs le repère choisi est tout simplement le repère du monde.

La résolution de ces contraintes se fait en même temps que celles associées aux collisions et notre algorithme reste globalement inchangé, à la différence près que ces contraintes doivent toujours rester dans la classe *ACTIF*. En effet, une articulation n'a pas lieu de ce désolidariser et donc les deux points p_1 et p_2 de se décoller.

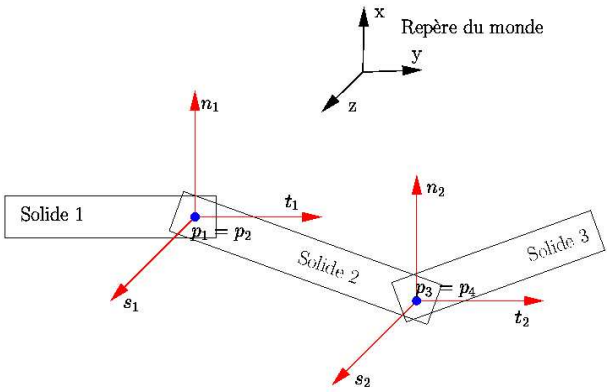


FIG. 5 – Les repères locaux des contraintes point sur point sont alignés avec les axes du monde. Les points p_1 et p_2 appartenant respectivement aux solides 1 et 2 doivent être maintenus confondus et leur vitesse ainsi que leur accélération relative doivent être nulles. Idem pour les point p_3 et p_4.

8 Extension au frottement adhérent

Un exemple simple de frottement adhérent est celui d'une boule roulant sans glisser sur un plan : c'est une simplification du frottement de Coulomb où les objets peuvent glisser les uns sur les autres. Il suffit pour le prendre en compte dans notre méthode, de construire non plus une contrainte normale par collision mais trois : une normale et deux dans le plan tangent au contact. On procède ensuite de la même façon qu'auparavant mais avec trois fois plus de contraintes. On garantit ainsi, en plus du cas sans frottement que la vitesse et accélération de pénétration des deux solides en jeu est nulle non seulement dans la direction $\mathbf{n}$, mais aussi dans les deux directions tangentielles orthogonales $\mathbf{t}$ et $\mathbf{s}$.

9 Répartition de la charge de calcul

Nous nous sommes posé la question de savoir quelle est la répartition optimale en nombre d'itérations sur les positions, vitesses et accélérations, à distribuer, selon un budget de calcul exprimé lui aussi en nombre d'itérations. En partant du principe que notre critère de qualité définissant une bonne simulation est la distance de pénétration moyenne des solides après correction des positions, on va parcourir l'espace constitué des différentes répartitions et trouver le minimum de la fonction ainsi parcourue. L'espace des paramètres de réglage a trois dimensions : nombres d'itérations sur les positions, sur les vitesses et sur les accélérations mais leur somme étant constante, le domaine de définition de notre fonction est sur deux dimensions et a une forme triangulaire comme le montre la figure 6. La figure 7 montre les résultats obtenus pour la répartition optimale de 30 itérations pour la scène appelée

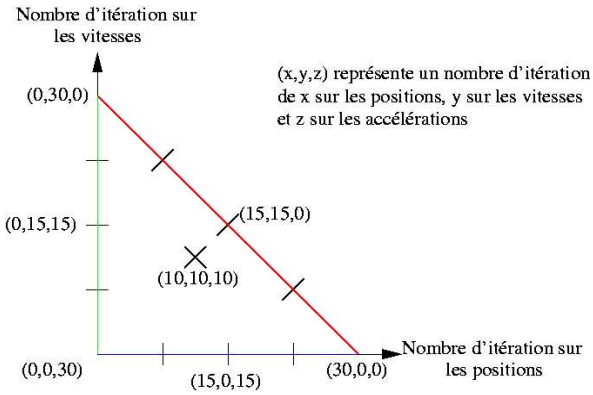


FIG. 6 – Domaine de définition de la fonction représentant l'erreur sur les positions en fonction du nombre d'itérations attribuées pour la correction des positions, des vitesses et des accélérations

tourniquet (voir annexe A). On remarque que si on ne corrige pas du tout les positions ou les vitesses, l'erreur diverge. Par contre on remarque également que trop corriger les positions (voir zone 1) ne garantit pas forcément un bon résultat car les erreurs engendrées sur les vitesses deviennent tellement grandes qu'elles se répercutent irrémédiablement sur les positions. Tout miser sur la correction des accélérations (voir zone 2) engendre le même problème. La vallée traversant le centre du triangle contient des répartitions beaucoup plus satisfaisantes avec un point optimal (en magenta) correspondant à la répartition optimale pour cet exemple.

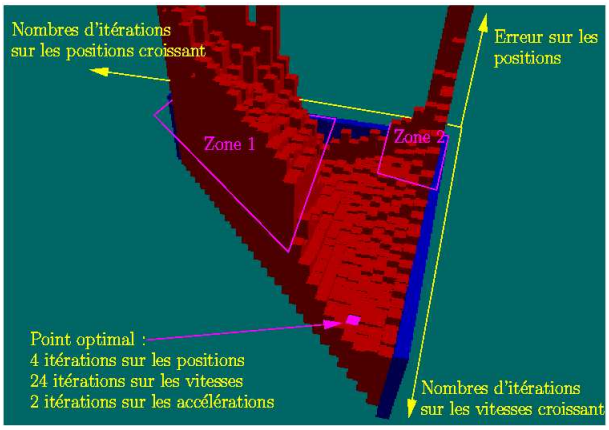


FIG. 7 – Graphique montrant l'erreur commise sur la correction de position en fonction de la répartition de 30 itérations. En bleu on peut voir l'ensemble des paramètres faisant diverger le programme et en magenta le point optimal

Cette rapide étude de la répartition montre qu'il est nécessaire de corriger un minimum les positions afin essentiellement de supprimer les dérives dues aux erreurs numériques et à l'intégration discrète du temps. Cependant, l'essentiel des calculs doit être concentré sur la correction des vitesses.

10 Bilan et perspective

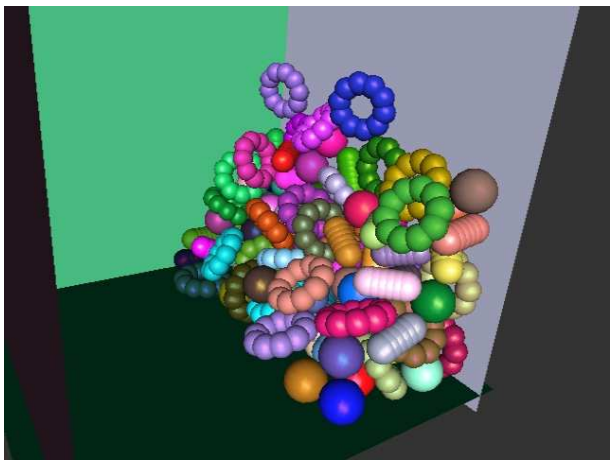
L'algorithme présenté ici fonctionne très bien pour de nombreux types de simulations et peut être utilisé pour des applications diverses et variées, comme la simulation d'éboulis ou de chute de pierre, pour les jeux de constructions ou encore les scènes "alimentaires" (objets en contact dans un bol...). La rapidité du calcul de réponse aux collisions en fait un outil parfait pour la simulation temps réel d'un grand nombre de corps (plusieurs centaines) avec un grand nombre de collisions (plusieurs centaines également). Le système de compromis entre efficacité et rapidité de calcul accroît cette possibilité et l'étend même à la simulation non temps réel d'un très grand nombres de corps (plusieurs milliers), possibilité facilitée par une complexité en mémoire linéaire (car les matrices sont creuses et sont donc stockées sous forme de graphe). L'étude de la répartition optimale va de plus permettre de dégager une loi qui nous servira à construire un unique paramètre de réglage de ce compromis. Ainsi un non spécialiste pourra facilement manipuler notre algorithme qui pour l'instant nécessite le réglage de nombreuses valeurs.

Les tests montrent que la répartition du temps de calcul optimal est relativement non conventionnelle et nécessite des corrections en position ce qui est très rarement fait dans les méthodes de simulation de solides par modèle physique. La correction des vitesses est quant à elle prépondérante, ce qui confirme la tendance dans ce domaine, d'autant plus que c'est elle qui donne son réalisme à la simulation.

Cependant, et vous l'aurez remarqué, nous ne gérons pas encore le modèle de frottement de Coulomb qui est plus réaliste que le modèle statique déjà implémenté. Nous avons tout de même déjà obtenu quelques résultats, sans pour autant avoir reproduit tous les cas de figures possibles, en discrétisant le cône de Coulomb en une pyramide à base polygonale pour pouvoir appliquer des contraintes linéaires sur les composantes normales et tangentielles des forces.

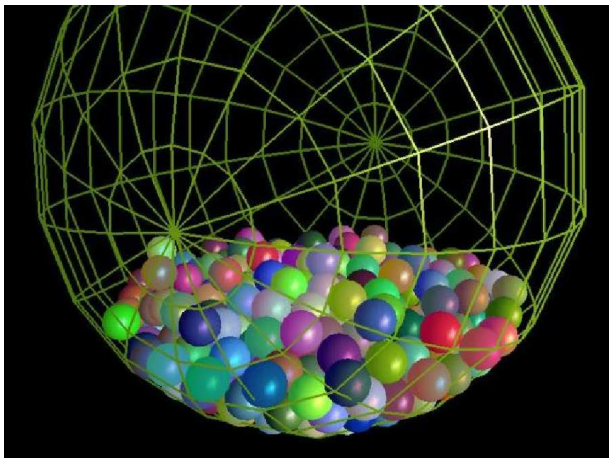
Dans un dernier temps, nous intégrerons à notre méthode un module performant de détection de collisions entre polyèdres. Pour l'instant, avec une détection tout à fait naïve c'est en entre 50 et 80 pourcents du temps de calcul total qui est perdu dans cette étape.

A Quelques images

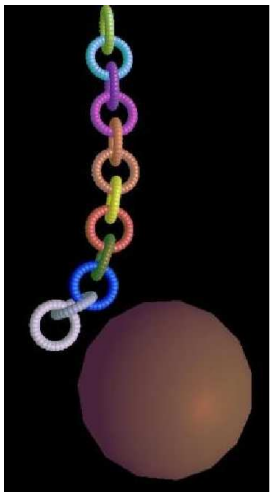


Tourniquet : 36 barres de 6 sphères,10 tores de 10 sphères et 36 sphères dans un cube

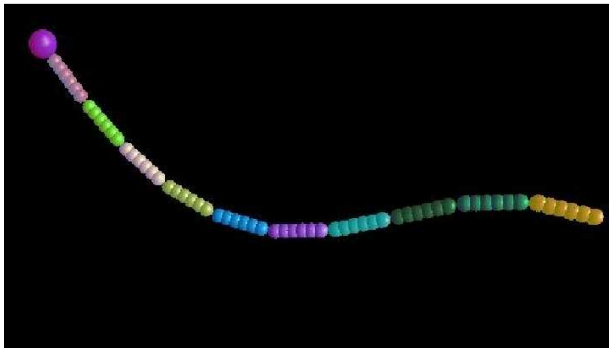
Pile : 13 tores de 30 sphères



316 spheres : 316 spheres dans une sphère creuse



Chaine : 12 tores de 30 sphères



Pendule : pendule articulé composé de 12 barres rigides de 6 sphères chacunes

Références

[Bar89] David Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *Computer Graphics*, 23(3) :223–232, 1989.

[Bar91] David Baraff. Coping with friction for non-penetrating rigid body simulation. *Computer Graphics*, 25(4) :31–40, 1991.

[Bar94] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. *Computer Graphics*, 28(Annual Conference Series) :23–34, 1994.

[Bar96] David Baraff. Linear-time dynamics using Lagrange multipliers. *Computer Graphics*, 30(Annual Conference Series) :137–146, 1996.

[BtDP+02] B. Brogliato, AA ten Dam, L Paoli, F Genot, and M Abadie. Numerical simulation of finite dimensional multibody nonsmooth mechanical systems, march 2002.

[Fau99] Francois Faure. Fast physically-based simulation of nonpenetrating rigid bodies. unpublished, january 1999.

[Fle00] R. Fletcher. *Practical Methods of Optimization*. Wiley, may 2000.

[Hah88] James K. Hahn. Realistic animation of rigid bodies. *Computer Graphics*, 22(4) :299–308, 1988.

[MC94] Brian Mirtich and John F. Canny. Impulse-based dynamic simulation, 1994.

[MC95] Brian Mirtich and John F. Canny. Impulse-based simulation of rigid bodies. In *Symposium on Interactive 3D Graphics*, pages 181–188, 1995.

[Mir96a] B. Mirtich. Hybrid simulation : combining constraints and impulses, 1996. Technical Report, Department of Computer Science, University of California, Berkeley.

[Mir96b] Brian Mirtich. Fast and accurate computation of polyhedral mass properties. *Journal of Graphics Tools : JGT*, 1(2) :31–50, 1996.

[Mir98] B. Mirtich. Rigid body contact : Collision detection to force computation, 1998. Technical Report TR-98-01, Mitsubishi Electrical Research Laboratory.

[Mir00] Brian Mirtich. Timewarp rigid body simulation. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 193–200. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[MS01] Victor J. Milenkovic and Harald Schmidl. Optimization-based animation. *Computer Graphics Proceedings*, (Annual Conference Series) :37–46, 2001.

[MW88] Matthew Moore and James Wilhelms. Collision detection and response for computer animation. *Computer Graphics*, 22(4) :289–298, 1988.

[PTVF93] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipies in C : The Art Of Scientific Computing*. Cambride University Press, january 1993.

[RB94] E. Rimon and J. Burdick. Mobility of bodies in contact - I : A new 2nd order mobility index for multiple-finger grasps, 1994. Submitted to IEEE Trans. on Robotics and Automation.

[SS98] J. Sauer and E. Sch. A constraint based approach to rigid body dynamics for virtual reality applications, 1998. Proceedings of the ACM Symposium on Virtual Reality Software and Technology.

[WB97] Andrew Witkin and David Baraff. Physically based modeling : Principles and practice, 1997. Siggraph '97 Course notes.

OpenMASK: {Multi-threaded | Modular} Animation and Simulation {Kernel | Kit}: un bref survol

David Margery, Bruno Arnaldi, Alain Chauffaut, Stéphane Donikian et Thierry Duval

Irisa, Campus de Beaulieu, 35042 Rennes Cedex, France
David.Margery@irisa.fr

Résumé : *Dans cet article, nous présentons OpenMASK, un noyau d'animation et de simulation multi activités qui est aussi une plate-forme pour le développement et l'exécution d'applications modulaires dans le domaine de l'animation, de la simulation et de la réalité virtuelle. OpenMASK est le résultat d'un compromis original entre la performance, l'abstraction et la modularité permettant de répondre aux besoins d'une grande variété d'applications de réalité virtuelle, y compris dans des contextes d'exécution distribuée. Cet article présente les objectifs, les concepts, les notions de base et la performance de la version publique* [1] *d'OpenMASK.*

Mots-clés : Simulation interactive, réalité virtuelle distribuée, boite à outil pour la réalité virtuelle

1 Introduction

Les applications des technologies de la réalité virtuelle sont nombreuses et prometteuses. Néanmoins, leur usage reste à ce jour encore relativement limité à cause de la complexité d'intégration dans un même logiciel de tous les composants logiciels et matériels nécessaires. En effet, pour une application donnée, le savoir faire, et donc le code, provenant de différente spécialités doit être assemblé sans nuire à la performance globale. En effet, les utilisateurs des technologies de la réalité virtuelle demandent toujours plus de réalisme et d'interactivité parce que les deux promesses de la réalité virtuelle sont celles d'une grande richesse de retour sensoriel et d'outils d'interaction puissants.

Cependant, afin d'être utiles, ce retour sensoriel et cette interaction doivent être calculés dans un intervalle de temps contraint, ce qui n'est pas toujours compatible avec le temps de calcul des simulations sous-jacentes. En effet, la réalité virtuelle est souvent utilisée pour interagir avec des simulations qui sont elles-mêmes coûteuses en temps de calcul, ce qui conduit à l'utilisation du parallélisme ou de la distribution pour réduire celui-ci. Ce facteur doit être pris en compte dès la conception d'un environnement pour la programmation et l'exécution d'applications de réalité virtuelle.

1.1 Problématique et résultats présentés

Un problème fondamental en réalité virtuelle est donc de concilier performance, multi-résolution des temps de calcul et abstraction. La performance est centrale à la richesse d'interaction et du retour sensoriel. La multi-résolution est importante parce que les applications de réalité virtuelle doivent coordonner des cycles de calcul de granularité différente. Mais on ne peut négliger l'abstraction, parce qu'intégrer dans la même application des composants logiciels d'origines différentes implique une abstraction commune à ces logiciels. Dans cet article nous présentons une telle abstraction : OpenMASK. OpenMASK est une plate-forme pour le développement et l'exécution d'applications modulaires dans le domaine de l'animation, de la simulation et de la réalité virtuelle. Elle est déjà utilisée au sein du projet Siames de l'Irisa, pour des applications de travail collaboratif (fig.2), de simulation mécanique et d'animation comportementale (fig.1). L'utilisation d'une même plate-forme pour ces programmes aux contraintes différentes est à notre sens un résultat significatif, puisqu'il démontre la possibilité de réutilisation de composants logiciels et donc qu'il abaisse le coût logiciel lié à l'utilisation de la réalité virtuelle. De plus, en utilisant la distribution non seulement à des fins de collaboration mais aussi pour améliorer la performance, la richesse des environnements virtuels produits s'en trouve augmentée.

1. voir http://www.openmask.org pour les détails

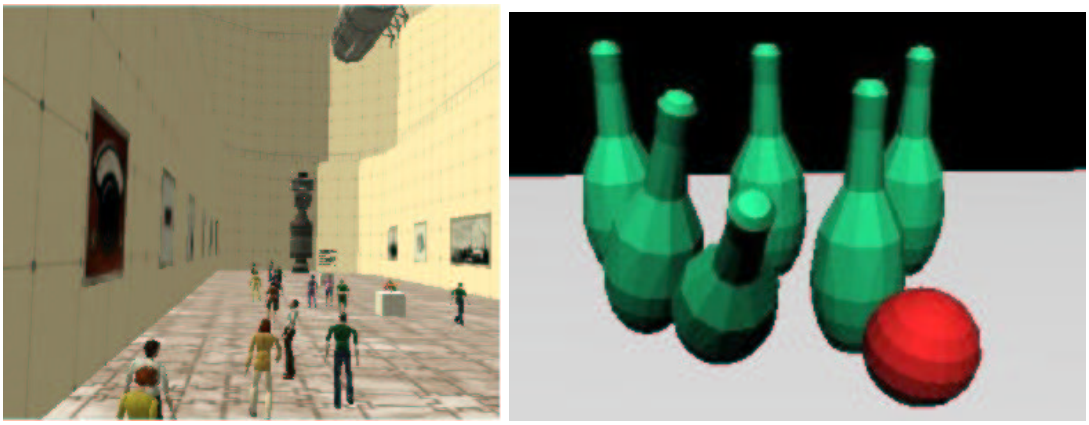


FIG. 1 – *Une simulation comportementale et une simulation mécanique utilisant OpenMASK*

1.2 Travaux connexes

Il y a de nombreuses boîtes à outils pour la réalité virtuelle. Que ce soit les premières boîtes à outils telles que MR-Toolkit[SG93] ou des plus récentes telles que VR-Juggler[JBBCN98], CAVERNSoft[KYN+00], Avango[Tra99], ou Maverick[RJM+99], l'objectif principal semble toujours d'abstraire le matériel pour le programmeur, permettant ainsi le changement de matériel, soit pendant l'exécution soit entre deux exécutions du programme. Ce travail est nécessaire, mais en général il ne fournit pas au programmeur un cadre conceptuel, forçant soit une reconception de la structure de l'application à chaque nouvelle application, soit l'utilisation du cadre conceptuel très basique fourni avec la boite à outils. Par conséquent, ces boîtes à outils, bien qu'utiles pour un développement plus rapide, n'aident pas à structurer l'application en composants qui pourraient facilement être réutilisés.

Dans le domaine de la réalité virtuelle distribuée, des cadres conceptuels plus complexes ont été proposés, parce qu'ils deviennent nécessaires pour pouvoir assurer la cohérence à travers le réseau avec de bonnes performances. Ceux-ci, qu'ils soient indépendants (Aviary[WHH+92], VEOS[BC94] ou Spline[BWA96]) ou basés sur une des boîtes à outils présentées ci-dessus (Environment Manager au-dessus de MR-Toolkit[WGS95] ou Deva[SJJA00]), se préoccupent davantage de la structuration de l'environnement virtuel que de la structuration logicielle de l'application. C'est pourquoi les cadres conceptuels proposés ne sont adaptés qu'à une classe limitée d'environnements virtuels. De plus, ces projets favorisent l'abstraction à la performance. Néanmoins, ces abstractions ont grandement influencées notre travail parce qu'elles cherchent toutes à définir les abstractions de base nécessaires dans une application de réalité virtuelle, et dont certaines sont déjà présentes dans GASP[SATR98, TD00b, TD00a], le prédécesseur d'OpenMASK.

Dans le même contexte de réalité virtuelle distribuée, des recherches visant la performance ont concentré leur attention sur l'infrastructure logicielle sous-jacente nécessaire à une distribution performante d'un environnement virtuel. La performance de ces systèmes distribués (la série MASSIVE [GB95, CJD00], la série NPSNET [MDM+95, CMBZ00] ou Community Place[LHMM97]) est basée sur un relâchement de la cohérence des différentes copies de l'environnement virtuel et sur la limitation du nombre d'objets impliqués dans les algorithmes de maintient de la cohérence. Là encore, les résultats obtenus ne sont applicables qu'à condition d'avoir prévu les objets explicitement pour la distribution. Dans OpenMASK, nous utilisons la cohérence relâché contrainte (décrite dans [Mar01]), mais en utilisant des informations systèmes pour limiter le nombre d'objets, une stratégie ne limitant pas le nombre possible d'applications.

Un dernier domaine connexe est le vaste domaine de la réutilisation de code, et plus spécifiquement de la construction d'une infrastructure logicielle capable de charger, décharger, connecter et découvrir des composants logiciels. Cette approche est relativement récente dans le cadre de la réalité virtuelle avec des prototypes tels que JADE[MJM00], Bamboo[Wat00] ou NPSNET-V[CMBZ00]. Une telle dynamicité n'est pas possible avec OpenMASK, qui doit avoir la connaissance a priori de tous les composants logiciels utilisés par une application donnée. Cependant, la notion de composants logiciels d'OpenMASK est beaucoup plus structurée que l'approche très fondamentale des projets cités précédemment, et ce parce qu'ils se placent à un niveau plus abstrait.

OpenMASK est donc notre contribution à la recherche du meilleur compromis possible lors de la conception

d'un environnement de développement et d'exécution pour la réalité virtuelle qui prenne en compte les problématiques de performances, d'abstraction et de distribution. Cet article est structuré de la façon suivante. Dans la prochaine section, les concepts de base d'OpenMASK sont présentés. Dans la section 3, nous présentons les politiques d'ordonnancement des objets de simulation d'OpenMASK (l'unité de modularité, composant logiciel d'OpenMASK) puis dans la section 4 les outils pour permettre la communication et donc l'interaction entre ces objets. La gestion de ces objets est ensuite présentée dans la section 5 avant que soient expliqués les principes et les performances du noyau d'exécution distribué.

2 Présentation générale d'OpenMASK

2.1 Objectifs de conception

L'objectif principal d'OpenMASK est de fournir un noyau d'animation et de simulation :

1. indépendant du niveau d'animation (descriptive, générative or comportementale) utilisé ;
2. indépendant du style de programmation de l'animation (réactive, orienté agent, objets actifs ...) utilisé ;
3. indépendant de la bibliothèque de rendu utilisée ;
4. capable d'utiliser plusieurs activités en parallèle pour le calcul de la simulation ;
5. indépendant du type d'exécution multi-activité utilisé (calcul distribué ou calcul parallèle).

Ces deux derniers points ont particulièrement influencés la conception du noyau d'OpenMASK, puisque notre objectif a été de permettre un parallélisme et une distribution performante. De plus, définir les objets manipulés par le noyau revient à privilégier une certaine granularité de l'animation et donc certains niveaux et styles d'animation.

Cependant, il convient de noter que les choix de conception sont davantage biaisés en faveur de l'exécution multi-activité qu'en faveur d'un niveau ou d'un style d'animation particulier. De plus, les outils fournis avec OpenMASK ont pour objectif de permettre à un programmeur l'expression de ses algorithmes d'animation dans le style naturel à ceux-ci, afin de permettre au noyau de manipuler des objets ayant une sémantique forte, et donc de faire des optimisations pertinentes.

Dans cet article, les termes simulation et animation sont utilisés avec un sens légèrement différent. Le terme simulation est utilisé lorsque le résultat produit (qui peut ne pas être visuel) est plus important que le temps mis à le produire, alors que le terme animation est utilisé pour des simulations interactives pour lesquelles l'interactivité est un aspect clé de l'application. Nous croyons que les mêmes composants logiciels devraient pouvoir être utilisés pour ces deux types d'applications, avec des optimisations différentes effectuées par le noyau en fonction du contexte d'utilisation. Cependant, dans cet article, nous ne nous intéresserons que très peu au contexte d'exécution et c'est pourquoi nous y discutons de la simulation d'objets de simulation sans présumer du contexte de leur utilisation.

FIG. 2 – *Une application de réalité virtuelle coopérative*

2.2 Concepts de base d'OpenMASK

Dans OpenMASK, la brique de base pour construire une application est l'objet de simulation. C'est au sein de l'objet de simulation que tout le code décrivant l'évolution de l'objet et son interaction avec les autres objets est

localisé. Les deux questions qui se posent sont donc les suivantes:

1. Quand le code de l'objet est-il exécuté? C'est la question de l'activation de l'objet.
2. Comment les objets communiquent-ils entre eux? C'est la question de la communication.

Une troisième question se pose : quelle est la granularité de l'objet de simulation? Ou encore, qu'anime un objet de simulation? Il s'agit d'une question à laquelle OpenMASK ne répond pas, puisque que l'objectif est de réaliser un noyau d'animation et de simulation qui soit le plus général possible. L'expérience d'utilisation d'OpenMASK montre que la granularité d'un objet de simulation varie d'un humanoïde virtuel complet avec ses comportements à une sphère inerte et ce dans la même application. Le meilleur exemple de la flexibilité et de la généralité du concept d'objet de simulation est que le rendu des environnements virtuels conçus au sein du projet Siames est réalisé à travers un objet de simulation appelé OpenMASK-3DVis [2] et qui gère le rendu sur dispositif immersif (workbench ou reality center) aussi bien que sur une station de travail, avec ou sans stéréo-vision.

3 Activation de l'évolution des objets

Chaque objet de simulation peut avoir le calcul de son évolution déclenché par deux méthodes. La première, la méthode `compute` est appelée à une certaine fréquence pour tous les objets dans l'état actif (voir fig 3 pour un diagramme des états possibles) ayant une fréquence non-nulle. La seconde méthode est liée au traitement des événements. Pour les objets dans l'état suspendu, le traitement des événements se fait au rythme de réception des événements, soit au maximum à la fréquence du contrôleur. Pour les objets dans l'état actif, le traitement des événements a lieu à la fréquence d'activation de l'objet par défaut, mais il est possible de le faire à la fréquence du contrôleur.

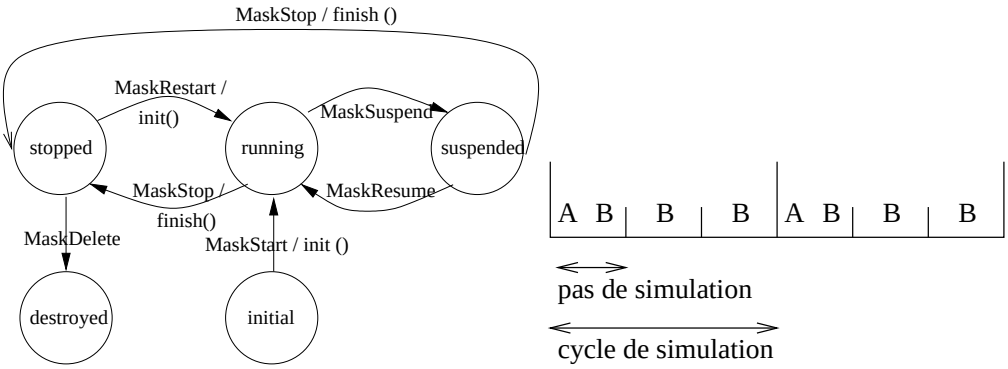


FIG. 3 – *Diagramme des états possibles d'un objet de simulation et cycles d'activation dans OpenMASK*

3.1 Politique d'ordonnancement

L'unité minimal d'ordonnancement d'OpenMASK est le pas de simulation (voir fig 3, où la fréquence de A est de 10Hz et celle de B 30Hz). Toute tâche ordonnancée est exécutée dans les limites d'un pas de simulation. Du point de vue de l'ordonnancement, OpenMASK est un système synchrone : aucun objet ne peut être ordonnancé 2 fois pendant le calcul d'un autre objet. Ce dernier point est souvent mal compris, parce que les objets peuvent avoir des fréquences différentes.

Pour calculer les structures d'ordonnancement, le contrôleur calcule le pgcd et le ppcm des fréquences non nulles de tous les objets de simulation. Le pgcd est alors la fréquence d'un cycle de simulation et le ppcm la fréquence d'un pas de simulation. On obtient alors, du point de vue de l'ordonnanceur un certain nombre de pas de simulation dans un cycle de simulation. Pour un objet ayant la fréquence d'un cycle de simulation, il sera ordonnancé sur le premier pas de simulation [3] de chaque cycle (la cas de l'objet A de la figure 3). Cependant, un

2. Aussi disponible à l'adresse www.openmask.org
3. Un choix arbitraire d'implémentation

pas de simulation ne peut commencer que lorsque le calcul de tous les objets ordonnancés sur le pas de simulation précédant ont terminé leur calcul.

La conséquence la plus gênante de cette stratégie d'ordonnancement est de faire varier le temps de calcul pour chaque pas de simulation en fonction du nombre et de la nature des objets ordonnancés à chaque pas. Même si du point de vue des objets simulés le temps s'écoule régulièrement, du point de vue d'un observateur extérieur à la simulation le temps peut s'écouler par à coups. C'est pourquoi il faut écrire les applications utilisant des objets fonctionnant à différentes fréquences de manière prudente, sinon le résultat obtenu pourrait être différent de celui attendu.

Dans un avenir proche, ce problème doit être réduit à cause du besoin de faire coopérer dans une même application des objets ayant des fréquences d'ordre de grandeur différent (retour haptique et visuel par exemple). Cependant, à cause des racines synchrones d'OpenMASK, la résolution de ce problème n'est pas triviale.

4 Gestion des objets

4.1 Les classes de base pour la gestion des objets

Plusieurs classes sont utilisées par OpenMASK pour la gestion des objets. Les objets de simulation sont connus entre eux grâce au descripteur d'objet (*objectDescriptor*), qui est une donnée qui n'évolue pas et qui contient le nom de l'objet, sa classe, et ses paramètres d'ordonnancement et de configuration.

Le contrôleur *PsController* est l'objet responsable pour toutes les fonctions globales à la simulation, en particulier pour la création, l'ordonnancement et la destruction des objets. Par ailleurs, le contrôleur est l'objet qui maintient une date de simulation globale à la simulation.

Le contrôleur ne manipule pas directement les objets de simulation, sauf lors de leur création et de leur destruction. Pour toutes les autres opérations, le contrôleur manipule un *PsObjectHandle* dans lequel sont stockés toutes les données d'un objet liées à l'exécution. Lors de l'utilisation d'un noyau distribué, un référentiel est utilisé pour manipuler la version de référence de l'objet de simulation (celle qui effectue les calculs), et un miroir est utilisé lorsqu'un contrôleur manipule une copie de l'objet pour permettre la communication entre objets simulé sur des nœuds différents.

4.2 L'arbre de simulation

Pour une application donnée, les objets de simulation sont structurés dans un arbre de simulation. L'objet racine de cet arbre est le contrôleur, mais le reste de la sémantique de l'arbre de simulation est laissé au concepteur d'application. Il y a deux fonctionnalités d'OpenMASK dont le comportement change en fonction de la structuration de l'arbre de simulation.

1. la création d'objet, puisque l'interprétation de la chaîne de caractères représentant la classe de l'objet à créer est faite récursivement à partir de l'objet père de l'objet à créer. Les prochaines versions d'OpenMASK pourrait faire la distinction entre le créateur d'un objet et le père de celui ci pour l'interprétation de cette chaîne de caractère, influant sur le type exact de l'objet créé.
2. les fonctions de recherche d'un objet. L'arbre de simulation permet de faire des recherches globales ou limités à un sous-arbre, aux frères ou encore au fils dans l'arbre de simulation.

Il convient pour finir de noter que l'arbre de simulation est une notion totalement différente de l'arbre d'héritage ou du graphe de scène.

4.3 La création d'objets

Les objets de simulation d'OpenMASK peuvent être créés soit :

1. statiquement : leur description est faite dans l'arbre de simulation paramètre du contrôleur à la création ;
2. dynamiquement, à l'initiative du programmeur ;

3. dynamiquement, en fonction des besoins des autres objets. Ceci est vrai en particulier dans le cas de l'exécution distribuée, puisque des objets miroirs sont créés au besoin pour communiquer avec les objets de simulation de référence.

Le C++ ne fournit pas de méthodes pour créer un objet à partir de la chaîne de caractère décrivant sa classe. C'est pourquoi nous utilisons un mécanisme spécifique : tous les objets, y compris le contrôleur ont une liste des classes d'objets qu'ils sont capables de créer. Pour chacune de ces classes, l'objet possède un pointeur sur un objet de type *PsObjectCreator* dont l'appel à une méthode spécifique provoque l'instanciation d'un nouvel objet.

5 La communication entre les objets de simulation

Il y a de nombreux outils sous OpenMASK pour faire communiquer les objets entre eux. Seulement, il faut se limiter aux outils fournis par le noyau d'OpenMASK afin de profiter des propriétés de calcul parallèle d'OpenMASK. En particulier, ceci implique que l'appel de méthodes entre objets de simulation est à proscrire, sauf dans certains contextes particuliers.

La raison de ce choix est que l'exécution multi-activité introduit des problèmes d'intégrité des données. Un noyau tel que celui d'OpenMASK peut gérer ces problèmes en protégeant les données de façon à éviter au programmeur d'objet de simulation de devoir comprendre le paradigme d'exécution multi-activité utilisé. Par contre, le noyau ne peut intercepter les appels de méthodes fait entre les objets de simulation (à moins d'introduire une indirection par précompilation du programme) et c'est pourquoi ceux-ci sont interdits.

Ce choix constitue le compromis de conception fondamental d'OpenMASK. De ce choix dérive les bonnes propriétés pour le calcul parallèle d'une simulation, mais aussi les quelques contraintes de programmation. D'un point de vue théorique, ce compromis s'exprime de la façon suivante : un objet ne peut avoir son exécution interrompue par l'attente du résultat d'une requête faite auprès d'un autre objet. De plus, il ne peut y avoir 2 flots d'exécution actifs simultanément au sein d'un même objet sans être programmés explicitement.

OpenMASK distingue plusieurs styles de communication entre les objets de simulation. La communication standard permet à un objet de lire les attributs d'un autre objet de manière régulière. Un objet de simulation rend ses attributs lisibles par les autres objets de simulation en les plaçant dans des sorties ou des paramètres de contrôle, et il sont généralement lus à travers des entrées. L'autre moyen de communication se fait à travers des signaux émis par un objet, et reçus soit directement, soit à travers des auditeurs d'événements et sont conçus pour une communication sporadique. De ces notions est dérivée celle d'événement qui permet la communication entre deux objets spécifiés à l'avance et permet l'échange de requêtes entre les objets de simulation.

5.1 Les sorties

Une sortie est utilisée pour rendre public un attribut (une position par exemple) d'un objet de simulation dont la valeur a toujours un sens : quel que soit le moment ou cet attribut sera lu, la valeur lue doit avoir un sens. C'est pourquoi l'interpolation et l'extrapolation des valeurs d'une sortie sont légitimes, même si la sémantique de la valeur rendue publique dans la sortie ne se prête pas bien aux méthodes numériques classiques d'interpolation ou d'extrapolation fournit par le noyau d'OpenMASK. Par conséquent, il est possible d'associer un *polateur* (un objet réalisant l'extrapolation et l'interpolation) dédié à chaque sortie. Il convient de voir la déclaration d'une sortie comme une déclaration d'interface. Alors que les méthodologies de programmation objet recommandent de protéger les données d'un objet et de ne rendre public que ses attributs, avec OpenMASK c'est l'opposé qui est recommandée. Un objet rend publiques certaines de ces données, et protège les méthodes qui sont utilisées pour calculer son évolution. Pour cette raison, la création de sortie ne doit être faite que dans le constructeur de l'objet de simulation, le noyau supposant que l'interface d'un objet est constante après sa création. De plus, cette contrainte garantit que cette interface est totalement préservée par héritage. La construction d'une sortie nécessite 3 paramètres :

1. le type de valeurs de la sortie (c'est un paramètre template)
2. le nom de la sortie
3. (optionnel) un polateur. Si aucun polateur n'est spécifié, le polateur par défaut associé au type est utilisé.

Le polateur le plus simple est appelé polateur naïf, et est pertinent pour tout les types de données, puisqu'il ne peut calculer que des valeurs de la file de valeur conservant l'historique de la sortie. Les autres polateurs se servent de cet historique pour calculer les valeurs demandées.

La liste de toutes les sorties d'un objet de simulation est stockée dans une table des sorties qui est accessible par tous les autres objets afin de permettre la découverte dynamique des propriétés d'un objet. Cette forme simple de réflexivité est importante pour intégrer dans une même application des objets conçus pour des applications différentes et donc avec des graphes d'héritage indépendants.

5.2 Les paramètres de contrôle

Un paramètre de contrôle est une sortie spéciale pour 2 raisons. La première, c'est que tout objet de simulation peut tenter de changer la valeur d'un paramètre de contrôle, et la seconde est qu'un paramètre de contrôle n'est pas référencé dans la table des sorties mais dans une table annexe, la table de paramètre de contrôle. Cependant, il est tout a fait possible de brancher une entrée à un paramètre de contrôle.

Lorsqu'un objet autre que le propriétaire du paramètre de contrôle tente de changer la valeur de celui-ci, un événement valué est envoyé au propriétaire. Cette événement est interprété par défaut par un auditeur d'événement qui va remplacer la valeur du paramètre de contrôle par la nouvelle valeur. Mais il est bien-sûr possible de changer ce comportement par défaut en surchargeant l'auditeur d'événement (voir le paragraphe 5.5).

5.3 Les entrées

Une entrée est utilisée pour établir un chemin de données entre une sortie d'un autre objet et l'objet de simulation propriétaire de l'entrée. Une fois établi, ce chemin de donnée donne accès à la valeur de la sortie à laquelle l'entrée est branchée. Il y a deux type d'entrées :

1. les entrées privées : le branchement de ces entrées aux sorties d'un autre objet peut seulement être fait à la demande du propriétaire de l'entrée.
2. les entrées publiques : elles ont les même propriétés que les entrées privées, mais acceptent aussi des branchement fait à l'initiative du propriétaire de la sortie sur laquelle elles sont branchés. Par le même mécanisme que celui utilisé pour changer la valeur d'un paramètre de contrôle, lorsqu'une sortie prend l'initiative d'un branchement sur une entrée publique, cela provoque l'envoi d'un événement demandant le branchement qui est par défaut accepté par un auditeur d'événement associé à l'entrée, mais qui peut être surchargé.

5.3.1 Lecture d'une entrée

La méthode par défaut pour lire une entrée est d'utiliser la méthode `get()`. Cette méthode accepte un argument optionnel qui correspond à un retard entre la date de simulation courante et la date de la valeur renvoyée par la méthode `get`. Par exemple, `get(0)` renverra une valeur calculée par la sortie à l'aide de son polateur et correspondant à une valeur pour la date courante. Il est possible que cette valeur soit une valeur exacte ou une extrapolation. De même, `get(20)` renverra une valeur calculée pour correspondre à la valeur de la sortie 20 ms secondes avant la date courante. S'il est important que la valeur lue par l'entrée corresponde à une valeur produite par le propriétaire de la sortie, il faut utiliser la méthode `getLastExactValue()`.

En utilisant une entrée sensible, un service de plus haut niveau est fourni : la détection de nouvelles valeurs produites sur la sortie à laquelle l'entrée est branchée. Il devient ainsi possible de savoir si une nouvelle valeur a été produite sans la lire et la manipuler (pour la comparer à l'ancienne valeur lue par exemple). En utilisant une entrée sensible signalante, une évolution de l'entrée sensible, un événement est envoyé au propriétaire de l'entrée lorsque la valeur de la sortie change.

5.4 Les signaux

Un signal est une information émise par un objet (ou par le noyau pour les signaux systèmes) dans l'environnement. Les signaux sont différenciés entre eux par des identifiants (la signature du signal) et il est possible de leur

associer une valeur.

Les objets voulant être informé de l'émission d'un signal donné doivent s'enregistrer pour recevoir le signal. Si l'objet ne s'intéresse qu'aux signaux émis par un objet particulier, l'enregistrement à lieu auprès de cet objet, sinon il faut s'enregistrer auprès du contrôleur de la simulation. Lorsqu'un signal est émis par un objet, celui est transformé en événement envoyé à tous ceux qui ont manifesté de l'intérêt dans un signal avec la même signature. Par défaut la signature de l'événement envoyé est celle du signal émis, mais il est possible de donner un prototype d'événement à envoyer lors de l'enregistrement afin de spécifier la signature de l'événement reçu en réaction au signal émis.

Ainsi, en utilisant des signaux et des entrées sensibles signalante, il est possible d'utiliser un mode de programmation réactif pour faire communiquer les objets d'OpenMASK.

5.5 Événements, événements valués et auditeurs d'événement

Un événement est une structure de donnée composée de l'émetteur, le destinataire, la signature et la date d'émission de l'événement. Un événement valué est un événement auquel un champ de donnée supplémentaire (de n'importe quel type compatible avec OpenMASK) a été ajouté permettant à un événement de porter une valeur.

Le traitement des événements est fait au niveau de l'*object handle* (see 4.1) qui est la structure de données permettant au contrôleur de la simulation de manipuler des objets de simulation. Les événements reçus sont triés à leur arrivée selon leur date d'émission puis leur ordre d'arrivée.

Le fait pour un objet de simulation de réagir d'une façon prédéterminée à un certain nombre d'événements peut faire partie de son interface, et doit donc être préservé à travers l'héritage et être visible par réflection. Les auditeurs d'événements sont des objets remplissant ce rôle. Ils encapsulent des fragments de code qui sont associés à certains événements et qui sont automatiquement appelés lorsque ces événements sont reçus par l'objet. En tant qu'éléments de l'interface d'un objet ils doivent être construits dans le constructeur de l'objet de simulation.

6 Distribution et Parallélisme

Tout dans la conception d'OpenMASK a été fait pour permettre une distribution aisé des calculs, puisque toutes les interaction entre les objets on lieu à travers le noyau ou en utilisant des objets construit par le noyau. En particulier, les objets de simulation n'ont pas à être retouchés pour pouvoir être utiliser par un contrôleur distribué, puisque tous les problèmes d'intégrité des données sont à gérer au niveau des outils fournit par le contrôleur.

En supposant que le nombre d'objets de simulation à distribuer soit au moins d'une ordre de grandeur supérieur au nombre de processeurs disponibles, l'algorithme d'équilibrage de charge le plus simple (round robin) produit des résultats tout à fait acceptables.

C'est pourquoi les problèmes à résoudre lors de la distribution et de la parallélisation sont la cohérence de l'environnement virtuel et l'intégrité des données. L'intégrité des données est un problème devant être résolu par le contrôleur parallèle et la cohérence par le contrôleur distribué.

6.1 Le contrôleur parallèle

Le contrôleur parallèle (utile pour les machines multiprocesseur) est une adaptation directe du contrôleur classique. Ce contrôleur ordonnance les objets de simulation sur un des processeurs [4], en utilisant une stratégie d'allocation équilibrant le nombre d'objets ordonnancé sur chaque processeur. Puisque les sortie gèrent une file d'historique des valeurs, l'exclusion mutuelle entre l'écriture d'une nouvelle donnée et la lecture des valeurs de la sortie pour garantir l'intégrité des données est réglée de manière triviale.

Les résultats obtenus avec le contrôleur parallèle sont très dépendant de l'utilisation de la mémoire dynamique utilisée par l'application et ces différents modules, puisque l'allocation est un point de contention au niveau du système. Les résultats présenté dans le tableau 1, un cinquième des objets utilisent fortement l'allocation dynamique

4. En utilisant un thread par processeur

pour leur structures de données internes. L'application test est ici une simulation urbaine simulant 30 véhicules, avec leur simulation mécanique et comportementale associée. 5 objets de simulation sont utilisés pour chaque véhicule, et la simulation complète utilise 187 objets de simulation une fois l'animation du décor (en particulier des feux de circulation) prise en compte.

nombre d'activités	temps d'exécution en s	accélération
1	23,378	1
2	13,387	1,75
3	10,605	2,20
4	8,941	2,61

TAB. 1 – *Accélération obtenue sur une machine SMP à 4 processeurs*

6.2 Le contrôleur distribué

Les principes utilisés pour le contrôleur distribué ont déjà été présenté dans [SATR98, TD00b, Mar01] et peuvent être résumé en 2 points :

1. Chaque fois qu'un objet de simulation a besoin d'accéder aux données publiques d'un objet simulé sur un autre nœud, une copie locale appelée miroir est crée. Elle est alors synchronisée à chaque pas de temps avec la version originale appelé référentiel.
2. Le cohérence et la synchronisation de tous les nœuds est réalisée à l'aide d'un algorithme original qui permet la parallélisation du calcul d'un pas de simulation et le transfert sur le réseau des informations de mises à jour. Un paramètre de cet algorithme mettant en œuvre une cohérence relâchée contrainte, la latence, permet de borner le nombre de pas de simulation qu'un nœud peut faire sans avoir reçu des informations de mise à jour suffisamment récentes.

nombre de nœuds	temps d'exécution (ms)	accélération
1	91 882	1
2	64 500	1.42
3	51 800	1.77
4	46 000	1.99
5	40 000	2.29

TAB. 2 – *Accélération obtenu en utilisant le contrôleur distribué au-dessus de PVM*

7 Conclusion

Dans cet article, nous avons présenté un survol d'OpenMASK, une plate-forme pour la conception modulaire d'applications d'animation et de simulation pouvant utiliser des noyaux d'exécution parallèle ou distribué, disponible en téléchargement. Cette plate-forme pour les applications de réalité virtuelle cherche à combiner abstraction, performance, distribution et réutilisation de composants logiciels dans le domaine de la réalité virtuelle, sans limiter le domaine d'application.

Nous croyons qu'OpenMASK représente un compromis intéressant, puisqu'il est déjà utilisé comme plate-forme logicielle pour différents programme de recherche. Ces programmes incluent des applications de réalité virtuelle coopérative, d'interaction utilisateur en environnement virtuel, de retour haptique, de simulation comportementale, de capture de mouvement et de contrôle de mouvement. OpenMASK est en particulier utilisé au sein de Perf-RV.

Références

[BC94] W. Bricken and G. Coco. The VEOS project. *Presence*, 3(2):111–129, 1994.

[BWA96] J. Barrus, R. Waters, and D. Anderson. Locales and beacons: Precise and efficient support for large multi-user virtual environments. *Proceedings of VRAIS'96, Santa Clara CA*, pages 204–213, 1996.

[CJD00] C. Greenhalg, J. Purbrick, and D. Snowdon. Inside MASSIVE-3: Flexible support for data consistency and world structuring. In *Collaborative Virtual Environments*, number 1-58113-303-0, pages 139–146. ACM, september 2000.

[CMBZ00] Michael Capps, Don McGregor, Don Brutzman, and Michael Zyda. Projects in VR: NPSNET-V: A new beginning for dynamically extensible virtual environments. *IEEE Computer Graphics and Applications*, 20(5):12–15, September/October 2000.

[GB95] C. Greenhalgh and S. Benford. MASSIVE: A distributed virtual reality system incorporating spatial trading. In *Proceedings of the 15th International Conference on Distributed Computing Systems (ICDCS'95)*, pages 27–35, Los Alamitos, CA, USA, May30 June–2 1995. IEEE Computer Society Press.

[JBBCN98] C. Just, A. Bierbaum, A. Baker, and C. Cruz-Neira. Vr juggler: A framework for virtual reality development. In *Proceedings of the 2nd International Immersive Projection Technology Workshop*, 1998.

[KYN+00] K. Park, Y. Cho, N. Krishnaprasad, C. Scharver, M. Lewis, J. Leigh, and A. Johnson. Cavernsoft g2: A toolkit for high performance tele-immersive collaboration. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology 2000*, pages pp. 8–15, Oct 22-25 2000.

[LHMM97] Rodger Lea, Yasuaki Honda, Kouichi Matsuda, and Satoru Matsuda. Community place: Architecture and performance. In Rikk Carey and Paul Strauss, editors, *VRML 97: Second Symposium on the Virtual Reality Modeling Language*, New York City, NY, February 1997. ACM SIGGRAPH / ACM SIGCOMM, ACM Press. ISBN 0-89791-886-x.

[Mar01] David Margery. *Environnement logiciel temps-réel distribué pour la simulation sur réseau de PC*. PhD thesis, Université de Rennes 1, 2001.

[MDM+95] M. R. Macedonia, D. P. Brutzmann, M. J. Zyda, D. R. Pratt, P. T. Barham, J. Falby, and J. Locke. NPSNET: A multi-player 3D virtual environment over the internet. In Pat Hanrahan and Jim Winget, editors, *1995 Symposium on Interactive 3D Graphics*, pages 93–94. ACM SIGGRAPH, apr 1995. ISBN 0-89791-736-7.

[MJM00] M. Oliveira, J. Crowcroft, and M. Slater. Componant framework infrstructure for virtual environments. In *Collaborative Virtual Environments*, number 1-58113-303-0, pages 139–146. ACM, september 2000.

[RJM+99] R. Hubbold, J. Cook, M. Keates, S. Gibson, T. Howard, A. Murta, A. West, and S. Pettifer. Gnu/maverik a micro-kernel for large-scale virtual environments. In *VRST99*, December 1999.

[SATR98] S. Donikian, A. Chauffaut, T. Duval, and R. Kulpa. Gasp: from modular programming to distributed execution. In *Computer Animation'98, IEEE, Philadelphia, USA*, pages 79–87, june 1998.

[SG93] C. Shaw and M. Green. The MR toolkit peers package and experiment. *Proceedings of VRAIS'93*, pages 463–469, 1993.

[SJJA00] S. Pettifer, J. Cook, J. Marsh, and A. West. Deva3: Architecture for a large-scale distributed virtual reality system. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology 2000*, Oct 22-25 2000.

[TD00a] T. Duval and D. Margery. Building objects and interactors for collaborative interactions with gasp. In *Proceedings of the Third International Conference on Collaborative Virtual Environments (CVE'2000)*, pages 129–138. ACM, September 2000.

[TD00b] T. Duval and D. Margery. Using gasp for collaborative interactions within 3d virtual worlds. In *Proceedings of the Second International Conference on Virtual Worlds (VW'2000)*, Paris, France, july 2000. Springer LNCS/AI.

[Tra99] Henrik Tramberend. Avango: A distributed virtual reality framework. In *Proc. Of the IEEE Virtual Reality 1999*. IEEE, march 1999.

[Wat00] Kent Watsen. Bamboo: A platform and language independant mechanism enabling dynamically reconfigurable applications. présenté au 4th Annual Workshop on Distributed System Aspects of Sharing a Virtual Reality, September 2000.

[WGS95] Q. Wang, M. Green, and C. Shaw. EM - an environment manager for building networked virtual environments. *Proceedings of VRAIS'95*, 1995.

[WHH+92] A. West, T. Howard, R. Hubbold, A. Murta, D. Snowdon, and D. Butler. AVIARY - A generic virtual reality interface for real applications. *Proceedings of Virtual Reality Systems, London*, 1992.

Arrondi d'arêtes : de la topologie à la G^1-continuité

Franck Ledoux [1], Laurent Fuchs [2]

[1] LaMI, UMR 8042, 523 Place des Terrasses, 91000 Évry Cedex

[2] IRCOM-SIC, UMR 6615, SP2MI, 86962 Futuroscope Cedex
fledoux@lami.univ-evry.fr, fuchs@sic.sp2mi.univ-poitiers.fr

Résumé : *L'opération d'arrondi d'arêtes dans des objets surfaciques consiste à remplacer des arêtes vives par des surfaces aux formes arrondies. Nous en proposons une version générale qui permet de traiter un nombre quelconque d'arêtes dans un objet avec des arrondis différents pour chaque arête et l'éclatement ou non des sommets incidents. Cette approche est rendue possible par l'utilisation d'une structure topologique sous-jacente qui permet d'isoler et d'identifier le traitement à effectuer au niveau des sommets incidents aux arêtes à arrondir. Une fois la topologie du nouvel objet déterminé, nous lui associons une géométrie à l'aide de surfaces de Bézier triangulaires et quadrangulaires. Pour toutes les configurations, la G^1-continuité est assurée entre ces différentes surfaces en prêtant une attention toute particulière aux régions qui remplacent les sommets incidents aux arêtes à arrondir.*

Mots-clés : Arrondi d'arêtes, topologie, plongement linéaire et surfacique, G^1-continuité, surfaces de Bézier.

1 Introduction

En modélisation géométrique, L'arrondi d'arêtes consiste à remplacer les arêtes vives d'un objet par des surfaces planes ou arrondies pour obtenir un objet plus lisse. Cette opération est fréquemment utilisée car la plupart des objets de la vie courante possèdent des formes arrondies pour des raisons d'esthétisme, de sécurité ou de procédé de fabrication. Bien que cette opération soit largement étudiée [VMV94], il n'existe à notre connaissance aucune solution réellement générale au sens que l'on puisse arrondir n'importe quelle arête d'un objet avec des paramètres propres à chaque arête. Par exemple, considérons l'objet de la figure $1-A$ qui est typique des difficultés rencontrées lors de l'arrondi d'arêtes. Nous n'arrondissons que les arêtes en traits gras avec des arrondis plus ou moins importants. En remplaçant ces arêtes par des faces planes, nous obtenons l'objet de la figure $1-B$. Si maintenant, nous introduisons des surfaces arrondies, nous obtenons l'objet de la figure $1-C$. Dans les deux cas, le problème majeur est le traitement à effectuer sur les sommets incidents aux arêtes arrondies. Ce traitement est conditionné par la topologie de l'objet en ces sommets.

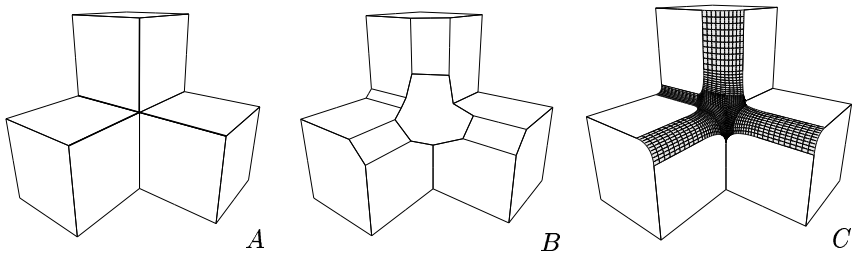


FIG. 1 – *Arrondi de quatre arêtes d'un objet géométrique.*

L'importance de la topologie aux extrémités des arêtes justifie l'utilisation d'un modèle géométrique tenant compte de la topologie de l'objet. Dans ce cadre, un objet géométrique est représenté en mettant en évidence sa topologie, c'est-à-dire sa subdivision en sommets, arêtes, faces, etc. La géométrie (courbes et surfaces) d'un objet est alors obtenue en associant une géométrie à chaque élément de la subdivision. On associe par exemple une courbe à une arête et une surface à une face. Cette association est appelée un *plongement*. Cette représentation des objets permet de découper les opérations en deux parties, l'une topologique qui modifie la structure des objets, et l'autre géométrique qui modifie leurs plongements. Ce découpage assure entre autres d'obtenir des définitions plus générales des opérations de transformation [Elt94] et de réduire les temps de calcul et les risques d'erreurs. En

effet, la présence d'une structure topologique permet de construire des objets complexes constitués de nombreuses primitives simples tout en conservant un accès efficace à chacune de ces primitives.

Dans le cas des figures $1-A$ et $1-B$, nous parlons d'objets filaires. Les travaux de H. ELTER et L. FUCHS [EF94, Elt94] définissent une opération d'arrondi d'arêtes sur ce type d'objets, en tenant compte de la topologie. En nous basant sur les algorithmes développés dans leurs travaux, nous avons étudié les principes généraux de l'opération d'arrondi d'arêtes que nous avons ensuite généralisée via une définition plus abstraite [Led02]. Nous obtenons une plus grande souplesse dans le choix des paramètres, une extension aux objets volumiques et la déclinaison de plusieurs versions d'arrondi. Le travail d'abstraction que nous avons effectué permet de ne se soucier d'aucun détail d'implantation, ce qui facilite la généralisation des algorithmes. Nous présentons le résultat de ce travail dans la première section de cet article en introduisant l'opération d'arrondi d'arêtes d'objets filaires.

Dans la seconde partie de cet article, nous plongeons les faces introduites lors de l'opération d'arrondi par des surfaces de Bézier, ce qui nous permet d'obtenir des résultats tels que celui de la figure $1-C$. Nos travaux s'appuient sur ceux de T. VÁRADY et A. L. ROCKWOOD [VR95, VR97] qui fournissent une opération d'arrondi d'arêtes avec éclatement des sommets. L'introduction de surfaces paramétriques amène à se poser la question de la gestion de la continuité, tout en conservant la généralité de l'approche avec plongement linéaire. En l'occurrence, nous proposons une méthode générale d'arrondi d'arêtes avec plongement surfacique assurant la G^1-continuité [Far02]. La partie topologique de cette opération est similaire à celle de l'opération d'arrondi d'arêtes sur les objets filaires. Quant à la partie géométrique, elle s'inscrit parmi les méthodes de réseaux [Fjä86, Chi88, SN91, EF94] qui consistent à définir des surfaces en assurant le degré de continuité nécessaire entre elles. L'approche de T. VÁRADY et A. L. ROCKWOOD [VR95, VR97] fournit la G^1-continuité mais ne traite qu'un seul type d'arrondi en introduisant une gestion intéressante des surfaces qui remplacent les sommets incidents aux arêtes à arrondir. Nous étendons cette approche pour obtenir une opération générale.

L'implantation de l'arrondi d'arêtes que nous proposons est effectuée en considérant le modèle topologique des *n-G-cartes* [Lie91]. Le modèle de plongement est un simple plongement linéaire qui donne une représentation "filaire" des objets, ou un plongement surfacique utilisant les surfaces de Bézier. Cette implantation est réalisée au sein du modeleur *Moka* [Vid01].

Dans cet article, nous ne présentons qu'intuitivement notre méthode d'arrondi d'arêtes. Le lecteur intéressé pourra toutefois se référer à [Led02] où l'ensemble des détails techniques sont développés.

2 Arrondi avec plongement linéaire

Nous proposons une méthode locale d'arrondi d'arêtes permettant à l'utilisateur de sélectionner les arêtes qu'il veut arrondir dans un objet, la façon dont il veut les arrondir et s'il veut aussi arrondir les sommets incidents. La difficulté inhérente à ce type d'approches se situe dans le traitement des sommets incidents aux arêtes à arrondir. Il faut pouvoir définir de façon systématique, homogène et pertinente le traitement à effectuer en tout sommet, et ce quelque soit le nombre d'arêtes incidentes et les paramètres d'arrondis choisis. La prise en compte de la topologie de l'objet géométrique au sein du modèle de représentation est donc primordiale. Pourtant, à notre connaissance, seuls les travaux de H. ELTER et L. FUCHS s'appuient réellement sur la présence d'un modèle topologique [EL94, Elt94]. C'est sur ces travaux que sont basés les résultats présentés dans cette section.

Comme H. ELTER et L. FUCHS, nous séparons au maximum les traitements géométriques et topologiques, ce qui favorise la généralité de notre approche. Nous commençons cette section en présentant intuitivement le résultat attendu. Le point important de la construction effectuée est le traitement topologique aux sommets incidents aux arêtes à arrondir. Nous introduisons ensuite les traitements topologiques et géométriques réellement effectués.

2.1 Construction intuitive

Arrondir une arête isolée dans un objet polyédrique est une tâche relativement aisée à condition de traiter avec attention ses sommets incidents. Par contre, les choses se compliquent dès que l'on arrondit plusieurs arêtes. De manière générale, le problème posé revient à déterminer la topologie d'une face qui remplacera un sommet incident à n arêtes parmi lesquelles m arêtes sont arrondies ($m \leq n$). Il faut ajouter à cela que les arêtes ne sont pas toutes arrondies de la même façon et que le sommet lui-même peut être arrondi. Ce problème très général n'est habituellement que partiellement résolu, des hypothèses étant faites pour ne considérer que certaines confi-

gurations [Fjä86, Chi88, SN91, VR97]. L'approche de H. ELTER et L. FUCHS fournit une solution plus générale car ils permettent d'arrondir autant d'arêtes que l'on souhaite, mais sous certaines conditions tout de même. En l'occurrence, sur la figure $2-A$, les *arêtes de coupe* ar_1 et al_1 qui définissent l'influence de l'arrondi de l'*arête à arrondir* a_1 sur ses faces incidentes doivent être parallèles à l'arête a_1. Pour notre part, nous levons cette contrainte en permettant par exemple à l'arête al_3 de ne pas être parallèle à l'arête a_3. De plus, nous ajoutons des paramètres permettant d'arrondir les sommets incidents aux arêtes à arrondir. On parle d'*éclatement de sommet*. Pour un sommet incident à une arête à arrondir, on définit pour chacune de ses arêtes incidentes, un *point d'éclatement* qui caractérise l'éclatement du sommet initial relativement à cette arête. Dans le cas du sommet S, si l'on n'arrondissait pas ses arêtes incidentes et que l'on disposait d'un point d'éclatement pour chacune d'elles, le sommet S serait remplacé par une face triangulaire dont les sommets seraient les points d'éclatement. Sur la figure $2-A$, un point d'éclatement est associé au sommet S relativement à l'arête a_3. Le sommet S est appelé un *sommet à arrondir*.

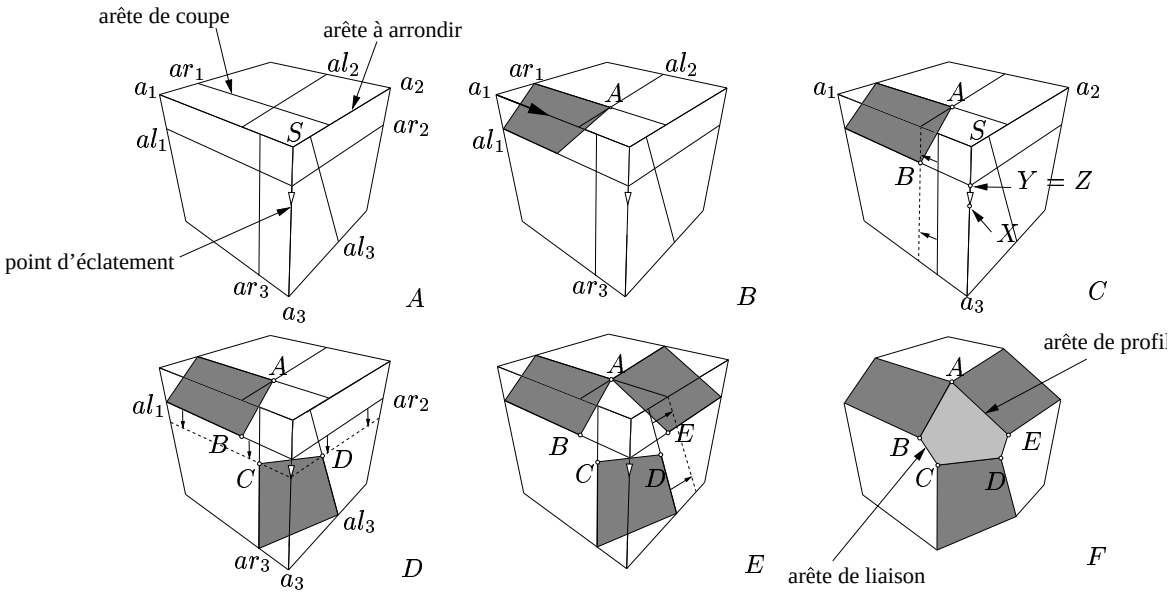


FIG. 2 – *Détermination de la zone d'influence en un sommet à arrondir.*

En un sommet à arrondir, plusieurs arrondis d'arêtes peuvent se rencontrer et la topologie de la face qui remplacera ce sommet est alors sous l'influence conjuguée de plusieurs arrondis. La démarche préconisée est de déterminer pour chaque arête la plus grande zone pour laquelle nous maîtrisons complètement son comportement. Considérons la figure $2-A$ qui représente un cube dont nous voulons arrondir les arêtes a_1, a_2 et a_3. L'arête a_3 est arrondie de façon non homogène et le sommet S est éclaté dans la direction de a_3.

Commençons par l'arête a_1 et déterminons la plus grande zone pour laquelle nous savons définir son arrondi (voir figure $2-B$). Si l'arête a_1 avait été la seule à arrondir, nous aurions coupé le cube par une surface reposant sur les arêtes de coupe al_1 et ar_1. De façon analogue, commençons la coupe de l'arête a_1 par l'extrémité opposée à S, comme l'indique la flèche. Nous maîtrisons cette "coupe" tant que l'on ne rencontre pas un autre arrondi agissant sur une même face que l'arrondi de a_1. En l'occurrence, l'arrondi de a_1 rencontre celui de a_2 puis celui de a_3. L'influence de la zone d'arrondi de a_1 sur la face contenant l'arête ar_1 s'arrête donc au point A qui est l'intersection des arêtes ar_1 et al_2 (voir figure $2-B$). Nous fermons la surface d'arrondi associée à l'arête a_1 en reliant le point A au point B qui est sur l'arête al_1 (voir figure $2-C$). Ce point est obtenu en effectuant un report de l'arête ar_3. Il tient ainsi compte de la prédominance de l'arrondi de a_2.

Intéressons-nous maintenant à l'arrondi de l'arête a_3. Au voisinage de S, il est sous l'influence des arrondis de a_1 et a_2 et de l'éclatement de S dans la direction de a_3. Afin de déterminer la zone d'influence au sommet, on compare les positions des points X, Y et Z. Dans le cas présent, le point le plus éloigné de S est le point X qui caractérise l'influence de l'éclatement de S (voir figure $2-C$). Pour construire la surface d'arrondi associée à l'arête a_3, on pose donc les points C et D comme l'intersection de ar_3 et de la translatée de al_1 d'une part, et de al_3 et de la translatée de ar_2 d'autre part. Enfin, on procède de même pour l'arête a_2 pour obtenir le résultat de la figure $2-E$. On obtient au final une surface à cinq côtés sur la figure $2-F$. Cette surface est appelée la *zone d'influence maximale* au sommet S. Les arêtes $[A, B]$, $[C, D]$ et $[A, E]$ sont appelées des *arêtes de profil* et les

arêtes $[B, C]$ et $[D, E]$ sont appelés des *arêtes de liaison*.

La construction géométrique que nous venons d'effectuer se systématise à toutes les configurations possibles. Il suffit pour cela de se placer dans un cadre topologique et d'effectuer quelques calculs simples d'intersections et de reports d'arêtes.

2.2 Construction pratique

Basé sur la notion de zone d'influence maximale, notre algorithme se décompose en deux étapes : la première, purement topologique, consiste à créer une *topologie générique* ; la seconde étape, à la fois topologique et géométrique, supprime certaines arêtes de la topologie générique tout en calculant le plongement de sommets. Ce découpage privilégie la topologie et permet d'obtenir une opération extensible au traitement d'objets de dimensions supérieures (i.e. des volumes).

2.2.1 Topologie générique

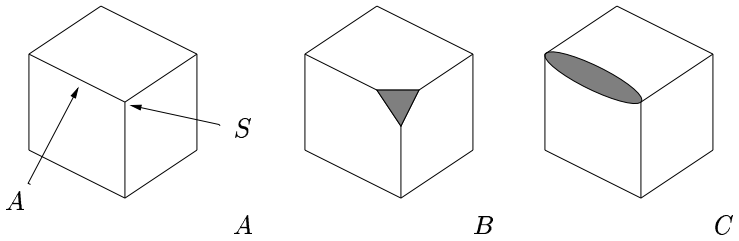


FIG. 3 – *Chanfreinage d'un sommet et d'une arête.*

Considérons un sommet à arrondir S incident à n arêtes dont m sont arrondies ($m \leq n$). Selon la position des arêtes de coupe et l'amplitude de l'éclatement de S, la face à introduire à la place de S peut avoir entre m et $m + n$ côtés. Dans le cas minimal, seules les arêtes de profil sont introduites. Dans le cas maximal, les arêtes de profil et toutes les arêtes de liaison sont présentes. La démarche préconisée par H. ELTER et L. FUCHS et que nous adoptons ici, est de créer toutes les arêtes de liaison possibles puis de supprimer celles qui sont inutiles. La topologie contenant toutes les arêtes de profil et de liaison est appelée *topologie générique*. L'obtention de cette topologie est aisée dès lors que l'on dispose de l'opération de chanfreinage qui est une opération purement topologique [Elt94, Led02]. Étant donnée une subdivision S de dimension n, chanfreiner une de ses cellules, notée C, consiste à remplacer cette cellule par une cellule de dimension n dont le nombre de cellules de dimension $n - 1$ la bordant est égal au nombre de cellules de dimension n incidentes à C. Par exemple, sur la figure 3, le sommet S incident à trois faces est remplacé par une face à trois côtés en B, et l'arête A incidente à deux faces est remplacée par une face à deux côtés en C.

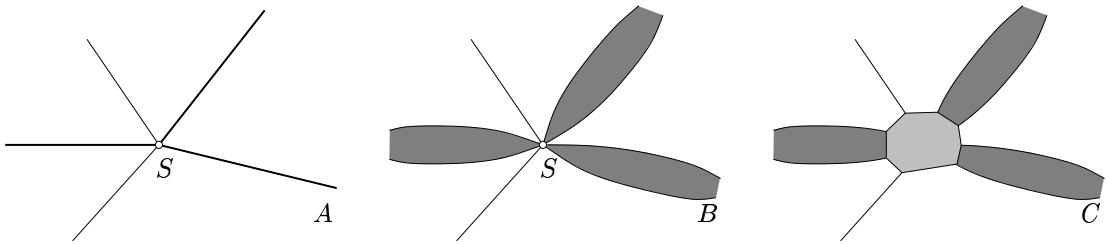


FIG. 4 – *Obtention de la topologie générique.*

À l'aide du chanfreinage, la topologie générique s'obtient simplement en chanfreinant les arêtes à arrondir puis les sommet incidents à ces arêtes. Ainsi, sur la figure 4, nous arrondissons trois arêtes parmi cinq. Partant de la configuration A, nous chanfreinons les trois arêtes désignés pour obtenir la configuration B, à partir de laquelle nous chanfreinons le sommet S pour obtenir la configuration C qui correspond à la topologie générique. Cette dernière comporte les arêtes de profil nécessaires ainsi que toutes les arêtes de liaison possibles.

2.2.2 Topologie effective et plongement

Une fois la topologie générique créée, il faut encore déterminer quelles sont les arêtes de liaison à supprimer pour obtenir la *topologie effective* après l'arrondi d'une ou plusieurs arêtes.

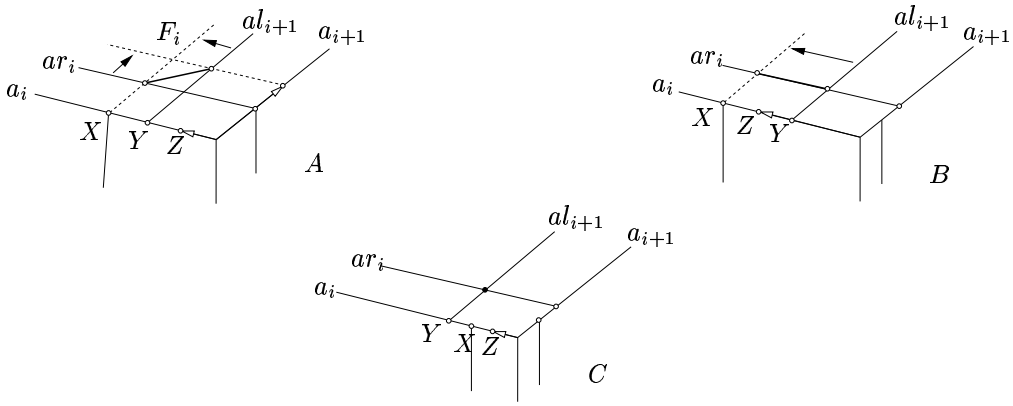


FIG. 5 – *Présence ou non d'une arête de liaison.*

Déterminer si une arête de liaison doit être conservée ou non dépend des arrondis d'arêtes ayant une influence sur la face F_i contenant l'arête en question. On distingue alors trois cas possibles représentés sur la figure 5. L'arête de liaison peut :
– être distincte des arêtes de coupe (cas A),
– être portée par une arête de coupe (cas B),
– ne pas exister (cas C).
Prenons le cas de l'arête a_i. On récupère le point le plus éloigné de S entre l'éclatement du sommet relativement à a_i (Z) et les points d'intersection de a_i avec les deux arêtes de coupe susceptibles de l'intersecter (X et Y). C'est ce point, noté X_i pour l'arête a_i, qui délimite la face qui remplace le sommet le long de a_i. Une fois le point X_i défini, deux cas peuvent se présenter :
– le point X_i est déterminé par l'arête de coupe contenue sur la face F_i. L'extrémité de l'arête de profil relative à a_i et appartenant à F_i est alors l'intersection de ar_i et de al_{i+1}. C'est le cas de l'arête a_i dans le cas C.
– le point X_i est déterminé par l'éclatement de sommet relatif à a_i ou par l'autre arête de coupe intersectant a_i. L'extrémité de l'arête de profil relative à a_i et appartenant à F_i est alors l'intersection de ar_i et du report de l'arête al_{i+1} au point X_i. C'est le cas de l'arête a_i dans les cas A et B.
L'existence d'une arête de profil sur la face F_i dépend donc seulement de la comparaison des positions de trois points sur les arêtes a_i et a_{i+1}. Si jamais les deux points les plus éloignés sont produits par les arêtes de coupe contenues dans F_i, l'arête de liaison de la face F_i n'existe pas (voir figure $5{-}C$). Dans le cas contraire, elle existe (voir figures $5{-}A$ et $5{-}B$). En traitant de cette façon tous les sommets incidents aux arêtes à arrondir, tous les nouveaux sommets sont directement calculés comme étant des intersections entre arêtes de coupe (reportées ou non).

2.3 Exemples

Nous achevons cette section en présentant quelques exemples d'arrondi d'arêtes sur une configuration usuelle formée de quatre cubes (voir figure 6). La particularité de cette configuration est l'alternance entre arêtes concaves et convexes au sommet commun aux quatre cubes. Ce sommet est incident à six arêtes que l'on va arrondir de plusieurs façons. Dans les deux premières lignes, nous arrondissons une arête puis deux, puis trois, ... , pour finalement arrondir les six arêtes incidentes au sommet commun aux quatre cubes. la seconde ligne s'achève avec l'arrondi de toutes les arêtes de l'objet surfacique deux et trois fois. La dernière ligne présente des objets légèrement différents. À gauche, le cube du dessus a été déformé, ce qui donne après trois arrondis successifs l'objet juste à sa droite. Les deux derniers objets illustrent l'arrondi d'une configuration différente des quatre cubes.

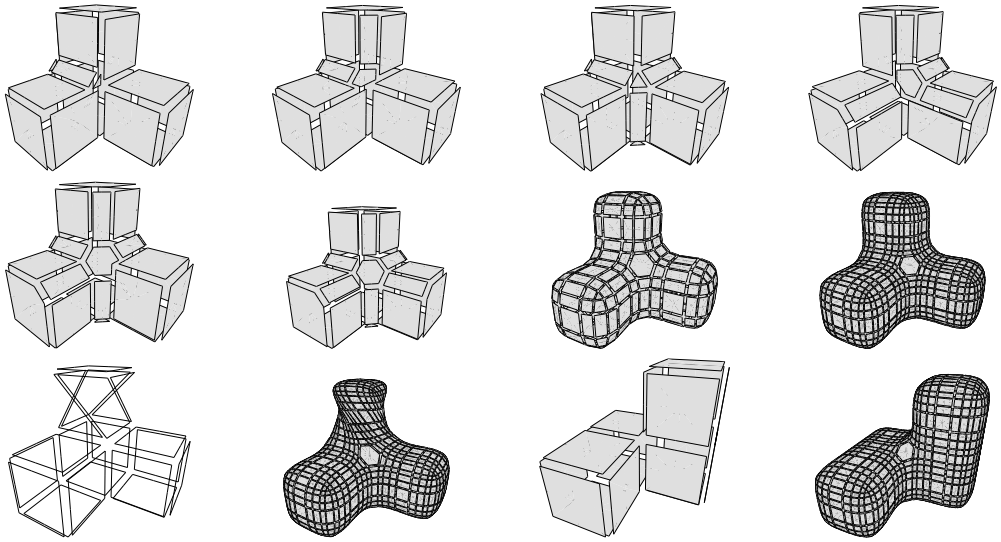


FIG. 6 – *Différents arrondis possibles d'une configuration classique de quatre cubes.*

3 Arrondi avec plongement surfacique

Dans cette section, nous présentons intuitivement notre méthode d'arrondi d'arêtes avec des carreaux de surfaces, en mettant en avant ses caractéristiques et les difficultés à prendre en compte. De ce fait, nous ne rentrons pas dans des détails techniques qui sortent du cadre de cet article.

3.1 Définition d'un arrondi général

L'aspect topologique de l'arrondi avec plongement surfacique est identique à celui de l'arrondi avec plongement linéaire. Seul le traitement de la géométrie change en plongeant désormais une arête par une courbe de Bézier et une face par un carreau surfacique de Bézier. La difficulté est maintenant de gérer la continuité entre les surfaces. Nous nous appuyons sur les travaux de T. VÁRADY et A. L. ROCKWOOD [VR95, VR97] qui proposent une méthode d'arrondi d'arêtes avec éclatement des sommets. Cette méthode introduit des surfaces de Bézier quadrangulaires tout en assurant la G^1-continuité interne[1]. Cependant, elle ne fonctionne que si toutes les arêtes incidentes en un sommet sont arrondies et que toutes les arêtes de liaison de la topologie générique sont conservées. Nous généralisons cette approche à toutes les configurations possibles en un sommet.

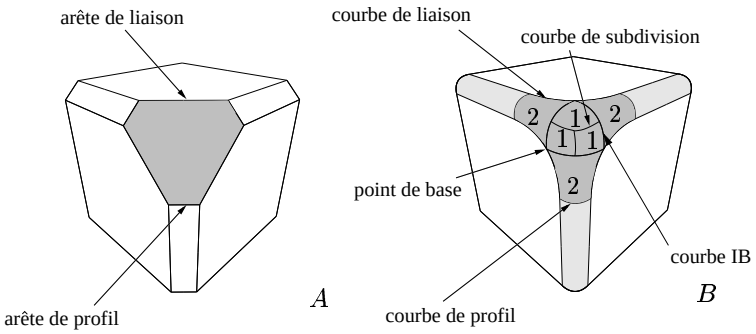


FIG. 7 – *Association d'une géométrie à une topologie générique.*

Tout d'abord, regardons le cas traité par T. VÁRADY et A. L. ROCKWOOD et faisons la correspondance avec notre approche topologique. Considérons la figure 7 où les trois arêtes incidentes au sommet d'un cube sont arrondies avec prédominance de l'éclatement de ce sommet. En A, la face remplaçant ce sommet est de topologie générique,

[1] C'est-à-dire la G^1-continuité entre toutes les surfaces introduites lors de l'arrondi.

c'est-à-dire que toutes les arêtes de liaison sont conservées. En B, est représentée la géométrie associée en suivant l'approche de T. VÁRADY et A. L. ROCKWOOD. Une arête de profil est plongée par une *courbe de profil*, et une arête de liaison est plongée par une *courbe de liaison*. Les milieux géométrique[2] de ces courbes sont appelés les *points de base* et ce sont les extrémités des *courbes de bord*, ou *courbes IB*. Ces courbes découpent la région au sommet en deux régions : la *région de retrait* composée de *carreaux de retrait* (notés 2 sur la figure 7), et la *région intérieure* qui est subdivisée en *carreaux intérieurs* (notés 1 sur la figure 7). La région intérieure est elle-même découpée en carreaux intérieurs suivant les *courbes de subdivision*. Finalement, on plonge la face grisée de la figure $7-A$ par le pavage des surfaces grisées de la figure $7-B$.

Pour obtenir une opération générale, il faut pouvoir n'arrondir que certaines arêtes avec des paramètres différents. Ceci nous oblige à envisager deux cas :
– des arêtes de profil sont supprimées ;
– des arêtes de liaison sont supprimées.
Le premier cas implique l'introduction de surfaces de retrait triangulaires. En effet, lorsqu'une courbe de profil est supprimée, la région de retrait incidente à cette arête n'a plus que trois côtés. Le second cas nécessite de prêter une attention particulière à la topologie aux points de base : supprimer une arête de liaison implique que le point de base qui en était le milieu va être à l'intersection de deux courbes de profil. En fait, dans pareil cas, ce point est incident à six courbes au plus : deux arêtes de coupe, deux courbes IB et deux courbes de profil. Une étude détaillée montre qu'il existe quatre topologies possibles en un point de base (voir figure 8) : le cas A où le point B_1 est incident à 4 courbes (aucune courbe de liaison n'a été conservée et il n'y a aucune région de retrait de sommets) ; le cas B qui est le seul cas où l'arête de liaison est conservée, le point B_1 est alors incident à 4 courbes ; le cas C où le point B_1 est incident à 6 courbes ; le cas D où le point B_1 est incident à 5 courbes.

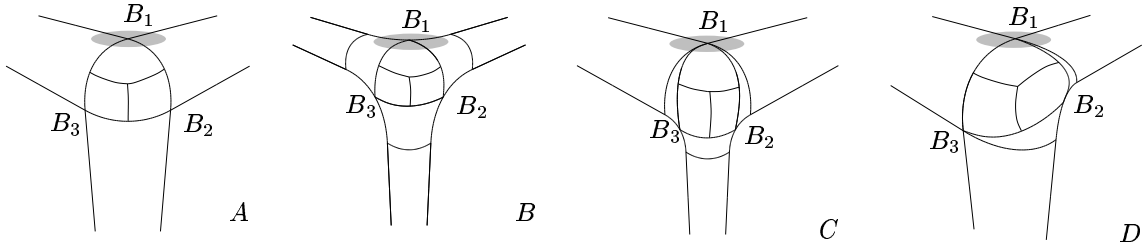


FIG. 8 – *Les différentes topologies aux points de base.*

Quelque soit la configuration considérée, la détermination des différentes courbes et surfaces nécessite de tenir compte de contraintes de continuité. En l'occurrence, il faut assurer la continuité entre les carreaux de surfaces adjacents le long d'une courbe, et entre plusieurs surfaces incidentes à un même sommet. La seconde condition est la plus difficile à vérifier, en particulier au voisinage des points de base. C'est pourquoi, notre algorithme, comme celui de T. VÁRADY et A. L. ROCKWOOD, traite prioritairement ce point. Même si l'on effectue le même traitement que T. VÁRADY et A. L. ROCKWOOD dans le cas B, le découpage de notre algorithme est différent du leur pour des raisons de généralité. Intuitivement, les différentes étapes sont (voir figure 9) :

1. construire les courbes de profil et de liaison ;
2. construire les surfaces d'arrondi d'arêtes en assurant la continuité avec les faces initiales de l'objet ;
3. créer les courbes IB en tenant compte de la continuité aux points de base qui sont ses extrémités ;
4. déterminer la position du point central et son plan tangent associé selon un critère de moindres carrés. La position du point central dépend d'un coefficient de profondeur permettant des arrondis plus ou moins pointus ;
5. assurer la continuité au voisinage des points de base B_i ;
6. assurer la continuité entre la région intérieure et les régions de retrait par l'introduction d'une fonction commune de dérivée transversale [VR95] le long de chaque courbe les séparant ;
7. construire les régions de retrait en se préoccupant de la continuité avec les faces initiales de l'objet ;
8. construire la région intérieure, ce qui revient à poser les courbes de subdivision en tenant compte de la continuité au point central puis à assurer la G^1-continuité le long des courbes de subdivision.

Afin d'illustrer les difficultés rencontrés, intéressons-nous à la gestion de la continuité au voisinage d'un point de base.

[2] C'est-à-dire le point de la courbe la séparant en deux arcs de même longueur.

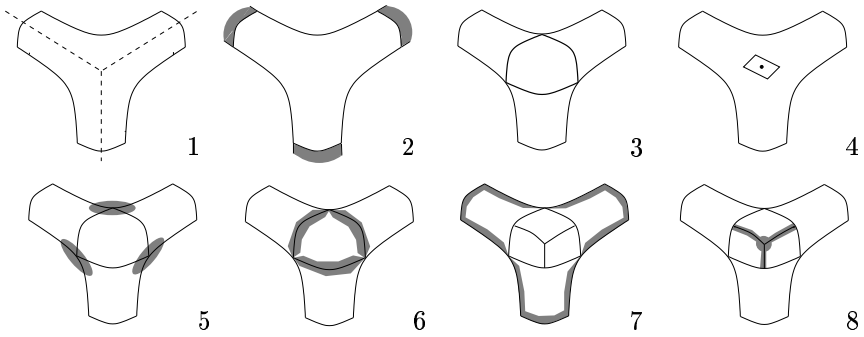


FIG. 9 – *Déroulement de l'arrondi d'arêtes.*

3.2 Continuité en un point de base

Pour assurer la G^1-continuité interne en un point de base B_i dans tous les cas, il est nécessaire de satisfaire deux critères [Pet91, YN95] : les tangentes de toutes les courbes incidentes à B_i doivent être coplanaires en B_i ; leurs courbures en B_i doivent être "compatibles". En d'autres termes, il existe une surface de courbure continue au voisinage de B_i sur laquelle reposent les courbes incidentes à B_i. Si le premier critère est facilement vérifiable, le second demande un traitement plus évolué. T. VÁRADY et A. L. ROCKWOOD proposent une construction simple permettant de respecter ce critère dans le cas B de la figure 8 où quatre courbes sont incidentes à B_i. Ils construisent la surface grisée de la figure $10-A$, appelée *surface de recouvrement*, sur laquelle on fait reposer localement toutes les courbes incidentes au point B_i. Cette surface est construite à l'aide d'un *carreau de Coons* [Far02].

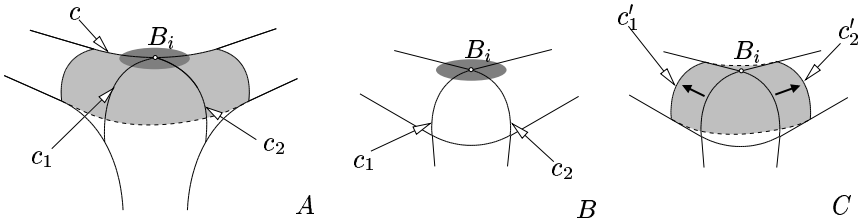


FIG. 10 – *Surface de recouvrement.*

L'adaptation directe de la construction de T. VÁRADY et A. L. ROCKWOOD aux cas A, C et D de la figure 8 n'est pas possible. En effet, la surface posée sur la figure $10-A$ repose sur la courbe de liaison c qui n'existe plus dans les cas A, C et D. Par exemple, dans le cas A (voir figure $10-B$), les courbes de profil c_1 et c_2 s'intersectent directement en B_i. Nous adoptons la méthode de T. VÁRADY et A. L. ROCKWOOD, nous proposons une construction simple qui approxime la région au point B_i. En fait, nous posons une surface de recouvrement qui approxime la position du point B_i et celles des arêtes de coupe qui lui sont incidentes. On construit ainsi une surface telle que celle de la figure $10-C$ où les arêtes c'_1 et c'_2 sont des translatés des arêtes de profil c_1 et c_2.

Quelque soit la topologie au point B_i, nous définissons donc une surface de recouvrement sur laquelle nous récupérons la courbure normale de chacune des courbes à construire. Par exemple, considérons une courbe IB définie comme appartenant à un triangle formé des extrémités de la courbe IB considérée, c'est-à-dire des points de base, et d'un point de l'arête à arrondir initiale. Sur la figure $11-A$, la courbe IB c est contenue dans le triangle formé par les points B_1, B_2 et I_1. Il faut que la courbure de c en B_1 soit compatible avec celles des autres courbes incidentes à B_1. Dans le cas de la figure 11, le point B_1 est de configuration B (voir figure 8), nous construisons donc une surface de recouvrement dont nous récupérons la courbure normale relativement à la section normale P (voir figure $11-C$). Nous déduisons la courbure de la courbe c par application du *théorème de Meusnier* [Car76], qui exprime la courbure de la courbe c en fonction d'une courbure normale de la surface de recouvrement. Dans la pratique, tous ces développements s'accompagnent de calculs importants sur les courbes et les surfaces traitées.

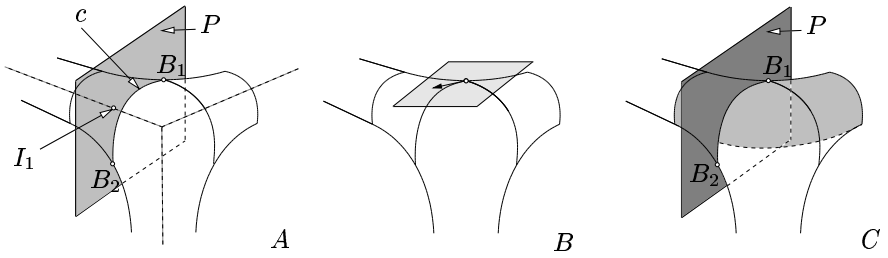


FIG. 11 – *Contraintes sur les courbes IB.*

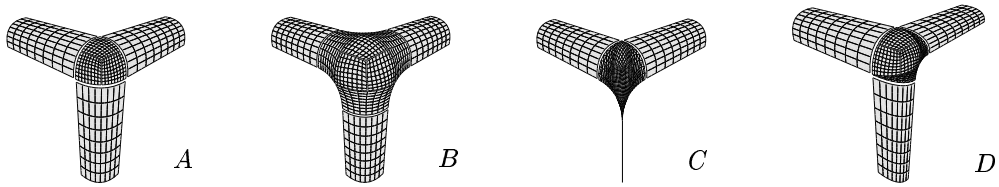


FIG. 12 – Sommet d'un cube dans les cas A, B, C et D.

3.3 Exemples

Nous commençons ces exemples par la figure 12 qui illustre le résultat obtenu pour chacun des cas particuliers de la figure 8. Ensuite, sur la figure 13 nous reprenons la configuration des quatre cubes traitée dans la section précédente. Nous arrondissons une arête puis deux, puis trois, . . . , pour finalement arrondir les six arêtes incidentes au sommet commun aux quatre cubes.

4 Conclusion

Nous avons développé une opération générale d'arrondi d'arêtes avec un plongement filaire ou surfacique. Pour atteindre un degré élevé de généralité, nous avons utilisé une approche topologique, mais nous avons aussi mené un travail important d'abstraction qui nous fournit une description précise, et indépendante de toute implantation dans le cas du plongement filaire. Ce dernier point présente l'avantage d'autoriser une généralité impossible à atteindre avec une approche programmatoire pragmatique. De plus, notre opération sur les objets filaires se généralise aux dimensions supérieures [Led02]. Cette généralisation permet par exemple de creuser des galeries dans des volumes en surmontant les problèmes de pincement qui peuvent survenir lors de l'arrondi d'arêtes en des sommets à la topologie complexe.

L'étude de l'arrondi avec plongement surfacique fournit un résultat en accord avec l'arrondi avec plongement linéaire et assure la G^1-continuité interne. L'opération obtenue fournit un résultat satisfaisant dès lors que l'objet et les paramètres considérés sont "raisonnables". Cependant quelques points méritent d'être améliorés. Par exemple, les courbes IB telles que nous les posons actuellement sont bien construites à leurs extrémités (les points de base) mais nous ne maîtrisons par contre pas suffisamment le comportement en leurs milieux. À la suite de ce travail, nous avons acquis une expertise suffisante qui va nous permettre de remodeler plus en profondeur notre algorithme afin d'en améliorer quelques points et de gérer la G^1-continuité externe.

Références

[Car76] M. P. Do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, Inc., 1976.

[Chi88] H. Chiyokura. Solid Modeling with DESIGNBASE, Theory and Implementation, 1988.

[EF94] H. Elter and L. Fuchs. Topologie de l'opération d'arrondi d'arêtes. *revue internationale de CFAO et d'infographie*, 9(6) :807–829, 1994.

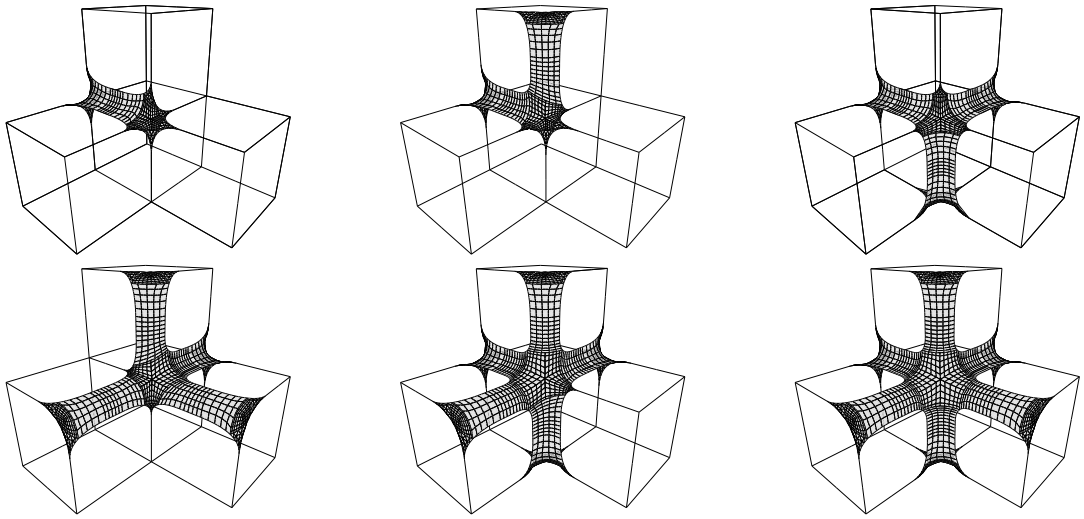


FIG. 13 – Arrondis d'arêtes en un sommet incident à six arêtes avec éclatement des sommets.

[EL94] H. Elter and P. Lienhardt. Cellular complexes as structured semi-simplicial sets. In *International Journal of Shape Modeling*, volume 1, pages 191–217. 1994.

[Elt94] H. Elter. *Étude de structures combinatoires pour la représentation de complexes cellulaires*. PhD thesis, Université Louis-Pasteur de Strasbourg, 1994.

[Far02] G. Farin. *Curves and Surfaces for CAGD, a practical guide, fifth edition*. Morgan Kaufmann, 2002.

[Fjä86] P. O. Fjällström. Smoothing of polyhedral models. In Alok Aggarwal, editor, *Proceedings of the 2nd Annual ACM Symposium on Computational Geometry*, pages 226–235, Yorktown Heights, NY, June 1986. ACM Press.

[Led02] F. Ledoux. *Étude et spécifications formelles de l'arrondi d'objets géometriques*. PhD thesis, Université d'Évry Val d'Essonne, 2002.

[Lie91] P. Lienhardt. Topological models for boundary representations : a comparison with n-dimensional generalized maps. *Computer-Aided Design*, 23(1) :59–82, 1991.

[Pet91] J. Peters. Smooth interpolation of a mesh of curves. *Constructive Approximation*, 7 :221–247, 1991. Winner of SIAM Student Paper Competition 1989.

[SN91] M. Szilvasy-Nagy. Flexible rounding operation for polyedra. *Computer-Aided Design*, 23(9) :629–633, 1991.

[Vid01] F. Vidil. Développement d'un modeleur à base topologique. Technical report, stage de deuxième année, enseirb, 2001.

[VMV94] T. Várady, R. R. Martin, and J. Vida. A survey of blending methods that use parametric surfaces. *Computer-aided Design*, 26(5) :341–365, 1994.

[VR95] T. Várady and A. P. Rockwood. Vertex Blending Based on the Setback Split. *Mathematical Methods for Curves and Surfaces*, pages 527–542, 1995.

[VR97] T. Várady and A. P. Rockwood. Geometric construction for setback vertex blending. *Computer-aided Design*, 29(6) :413–425, 1997.

[YN95] X. Ye and H. Nowacki. Optimal tangent-plane and curvature continuous modification of curves at common nodepoint. In *Design Engineering Technical Conferences*, pages 49–56, 1995.

Partition de l'espace et hiérarchie de cartes généralisées : application aux complexes architecturaux

David FRADIN, Daniel MENEVEAUX, Pascal LIENHARDT

Laboratoire IRCOM- SIC,Université de Poitiers, BP 30179, 86962 Futuroscope- Chasseneuil Cedex, France

{fradin,meneveaux,lienhardt}@sic.sp2mi.univ-poitiers.fr

Résumé : *La modélisation d'environnements complexes nécessite de traiter un nombre important d'informations géométriques, topologiques et photométriques. Le traitement de ces informations implique une gestion rigoureuse des diverses structures de données et de la mémoire. Le travail présenté dans cet article concerne la modélisation géométrique à base topologique de bâtiments composés d'un grand nombre de pièces meublées. Nous décrivons une structure topologique correspondant à une extension des cartes généralisées, permettant de représenter à la fois une hiérarchie de niveaux de détails et des partitions multiples pour un même bâtiment. Cette structure est une optimisation d'un modèle général reposant sur l'étiquetage de cellules. Elle définit le noyau d'un modeleur géométrique à base topologique spécialisé dans la construction de bâtiments complexes. Mais elle est essentiellement destinée à la visualisation et à la simulation d'éclairage de bâtiments structurés. En effet, à plus long terme, notre objectif est d'exploiter les propriétés de cette structure topologique pour la visualisation de bâtiments et les calculs de visibilité.*

Mots-clés : Complexes architecturaux, cartes généralisées, niveaux de détails, partitions multiples, hiérarchie.

1 Introduction

Le domaine de la modélisation géométrique propose des outils permettant de définir des objets à l'aide d'opérations élémentaires. De nombreuses structures dites à base topologique ont été proposées en modélisation par les bords[1] (cf. par exemple [BAU72], [WEI85], [BRI89], [LIE89]). Elles permettent de représenter des subdivisions de l'espace en cellules (sommets, arêtes, faces, volumes) liées entre elles par des relations d'adjacence et d'incidence. Chaque cellule possède également des informations géométriques pour compléter leur description. Les modèles topologiques ont pour atout majeur d'être basé sur des définitions algébriques formelles assurant une fiabilité de la structure et une cohérence de la représentation obtenue.

L'objectif de cet article est de présenter un modeleur dédié aux complexes architecturaux d'intérieur. Il est développé à partir d'un noyau géométrique à base topologique développé au sein du laboratoire IRCOM-SIC, permettant de manipuler des subdivisions de $\mathbb{R}^3$. A plus long terme, nous souhaitons tirer profit de ce travail pour réaliser des calculs de rendu réaliste et de simulation d'éclairage par la méthode de radiosité. Traditionnellement, ces algorithmes manipulent des structures moins riches, à base de listes de faces, pour des raisons d'espace mémoire et de simplicité d'utilisation. À l'évidence, le modèle proposé doit donc permettre un accès rapide aux données afin de ne pas ralentir les calculs de visualisation.

Nous intégrons la notion de *niveau de détail*, souvent employée pour réduire le nombre de primitives géométriques à traiter en fonction de la distance entre les objets et le point de vue. Cette notion est également utile pour faciliter la construction d'un environnement complexe. Par ailleurs, pour des raisons de convivialité nous avons également souhaité permettre à l'utilisateur de créer des groupes d'objets à l'intérieur des bâtiments. Par exemple, un bâtiment peut être décomposé à la fois en ailes ou en étages et les pièces sont regroupées suivant leur utilité (bureaux, salles de cours, etc). Pour permettre cette décomposition, notre structure intègre la notion de *partition multiple*.

La prochaine section décrit la problématique traitée. Nous proposons ensuite une structure topologique intégrant les notions de multi-partition et de niveau de détail par étiquetage. Enfin, nous présentons la structure finale ainsi que le modeleur dédié aux environnements architecturaux complexes avant de conclure.

[1] En anglais, boundary representation (B-Rep)

2 Problématique

La description d'objets complexes nécessite de manipuler une grande diversité de données : forme (attributs géométriques), apparence (matériaux, caractéristiques photométriques), etc. Pour les visualiser de manière réaliste et/ou interactive, deux points fondamentaux doivent être respectés : une gestion rigoureuse de la mémoire et un parcours rapide des différentes cellules de l'objet. Par ailleurs, un travail de construction progressive est fréquemment effectué de manière manuelle, par exemple par les architectes durant la phase de conception des bâtiments. Cela permet d'accroître la lisibilité des informations lors de la construction ; mais souvent les modèles mis à disposition pour la visualisation n'intègrent aucune structure relative à cette organisation.

Notre objectif dans cet article est de proposer une structure permettant de représenter conjointement les notions de partition et de niveau de details, afin qu'elles puissent être réutilisées au cours des étapes de visualisation ou de simulation d'éclairage. Une partition décrit des groupes de sous-objets pour un objet donné. Dans de nombreuses applications, plusieurs partitions peuvent être utiles pour représenter l'ensemble de la structure des objets. Par exemple, un bâtiment est vu comme un ensemble de pièces regroupées soit par ailes soit par étages. Dans la suite de ce document, nous appelons ce concept *multi-partition* ou *partition multiple*. La figure 1.a montre un exemple de multi-partition d'un objet. Parallèlement, la structure de niveaux de détail[2] permet de représenter les étapes de raffinement successif des objets. Les niveaux de détail sont fréquemment utilisés en visualisation pour accélérer les calculs. La figure 1.b montre un exemple de décomposition progressive d'un étage de bâtiment.

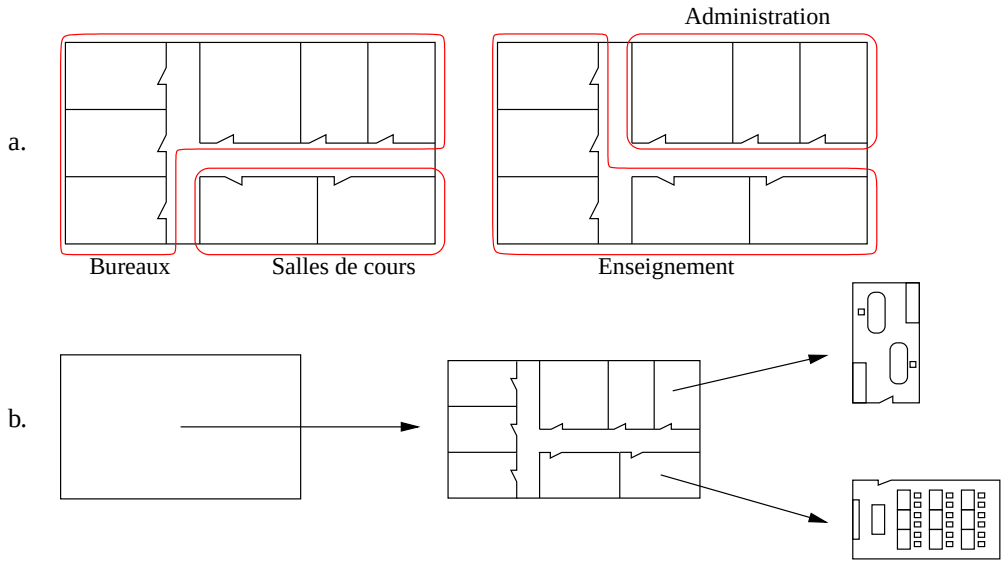


FIG. 1 – a. Partition multiple d'un même objet : un étage est subdivisé de deux manières différentes. Chaque subdivision a sa propre sémantique. La première subdivision représente la séparation entre les bureaux et les salles de cours. La seconde met en évidence les parties réservées à l'enseignement et à l'administration. - b. Décomposition d'un objet par niveaux de détail : un étage rectangulaire est détaillé par l'ensemble des pièces qui le constitue, dont deux sont elles-mêmes précisées (un bureau et une salle de cours).

La structure par niveaux de détail d'un objet peut être interprêtée soit comme une décomposition progressive de l'objet, soit comme un regroupement de plusieurs objets détaillés. Une solution évidente est d'assigner un numéro unique à chaque détail de l'objet. Ceci permet d'identifier à la fois les niveaux de détail et les partitions de l'objet. Plus précisément, chaque niveau de détail est décrit à partir de plusieurs objets plus détaillés et chaque identifiant a une sémantique particulière : nom, utilité, etc. Cette structure définie dans la section 3 reste inefficace dans le cadre de notre problématique pour des raisons d'occupation mémoire et d'accès aux données, en particulier pour la visualisation réaliste.

Dans la littérature, la plupart des modèles reposent sur des hiérarchies d'objets et permettent de répartir les traitements selon les différents niveaux de détail. Ces niveaux de détail sont mis en correspondance à l'aide de structures hiérarchiques ([KRO95],[FF88],[CDM+94],[PFP95]). Floriani *et al.* proposent un modèle à base topologique

[2] En anglais : Level Of Details (LOD).

appelé *Multiresolution Simplicial Model* [DPM97] présentant une synthèse de la plupart de ces structures. La décomposition d'un objet est représentée par un graphe orienté acyclique de complexes simpliciaux. Chaque état contient une *modification* relative à une partie de l'objet. Cette structure a aussi été généralisée à des complexes cellulaires : *Multiresolution Meshes* [DM01]. D'autres structures hiérarchiques ont été proposées dont certaines reposent sur une structure cellulaire appelée les cartes généralisées [LIE89]. Un travail mis en oeuvre par LEVY [LEV99] concerne la modélisation de couches géologiques à partir d'une hiérarchie à deux niveaux de cartes généralisées de dimension 2 et 3. GUILBERT [GUI00] a élaboré une structure de hiérarchie de cartes généralisées en incluant une technique de *clonage* des objets pour économiser l'espace mémoire.

Notre travail repose également sur les cartes généralisées et permet de représenter des partitions d'objets de $\mathbb{R}^3$. Nos contributions concernent les points suivants :
- la **multi-partition** permet de présenter plusieurs décompositions possibles d'un même objet. Par exemple, un bâtiment peut être divisé en ailes ou en étage, mais la description géométrique du bâtiment reste la même ;
- la **multi-hiérarchie**, complémentaire de la structure précédente, met en évidence une représentation du bâtiment par niveaux de détail et permet un chargement partiel des sous-arbres pour un travail localisé ;
- l'**accès aux données** pour la visualisation est simplifié et accéléré par la structure proposée quelle que soit la complexité de l'objet.

Autour de cette structure, nous avons développé un modeleur dédié aux complexes architecturaux d'intérieur. Mais ce modèle se veut suffisamment générique pour pouvoir plus tard être adapté à d'autres contextes (représentation de couches géologiques, industrie automobile, etc).

3 Étiquetage et partitions

La multi-partition correspond à une organisation de l'objet en ensemble de cellules, et les niveaux de détail définissent des ensembles d'ensembles. Une manière classique pour représenter cette organisation est l'étiquetage de cellules, servant de base théorique pour la définition de notre structure. Cette section rappelle la notion de cartes généralisées et décrit l'utilisation de l'étiquetage pour la représentation de multi-partitions et de niveaux de détail.

3.1 Rappel sur les cartes généralisées

Les cartes généralisées permettent de modéliser des objets géométriques subdivisés en cellules (sommets, arêtes, faces, etc.) reliées entre elles par des relations d'adjacence et d'incidence. Cette structure fait partie des modèles de répresentation par bords[3]. Les cartes généralisées permettent de représenter les objets sur lesquels nous travaillons : des subdivisions de $\mathbb{R}^3$. Les définitions qui suivent sont tirées de [LIE89].

Définition 1 *Soit n $\geq$ 0, une carte généralisée de dimension n (ou n-G-Carte) est le (n+2)-uplet G=(B,α_0,α_1,...,α_n) où :*
- *B est un ensemble fini de brins,*
- α_0, α_1, ..., α_n *sont des applications telles que :*
  - $\forall 0 \leq i \leq n$, α_i *est une involution*[4],
  - *pour* $\forall 0 \leq i < i+2 \leq j \leq n$, $\alpha_i\alpha_j$ *est une involution.*

À partir des éléments de base appelés brins et des applications α définies sur ces brins, les cartes généralisées représentent les cellules composant l'objet et leurs relations de bords. La i-cellule associée à un brin b donné, est formée de l'ensemble des brins obtenus par composition des involutions $\alpha_j, j \neq i$ (voir figure 2). De manière générale, toute composition d'involution est appelée *orbite* et est notée $< \alpha_k, ..., \alpha_p >$. Pour des informations complètes, le lecteur peut se référer à [LIE94].

3.2 Etiquetage

En modélisation, une structuration précise est nécessaire, en particulier pour regrouper des objets de même sémantique (nom, utilité, matériau, etc.). Une manière classique est d'étiqueter les cellules par l'identifiant du groupe

[3] En anglais, Boundary Representation (B-rep).

[4] Une application f est une involution signifie que f∘f est la fonction identité.

auquel elles appartiennent. Pour grouper des cellules de dimension n, (des ailes, des étages ou des pièces en dimension 3), nous définissons une fonction de partition ϕ_n associant à chaque cellule son identifiant. Dans le cas des cartes généralisées, ϕ_n est définie sur les brins. La figure 2 montre un exemple d'étiquetage de 2-cellules (faces) sur une 2-G-Carte.

Définition 2 *Soient $G_n = (B, \alpha_0, ..., \alpha_n)$ une carte généralisée de dimension n et $\phi_n : B \longrightarrow \mathbb{N}$ une fonction de numérotation des brins. On dit que ϕ_n est une fonction de partition de G en sous-groupes de n-cellules si et seulement si ϕ_n est la même sur tous les brins d'une n-cellule, i.e. $\forall b \in B$, tous les brins de l'orbite $< \alpha_0, ..., \alpha_{n-1} >(b)$ ont la même image par ϕ_n que b.*

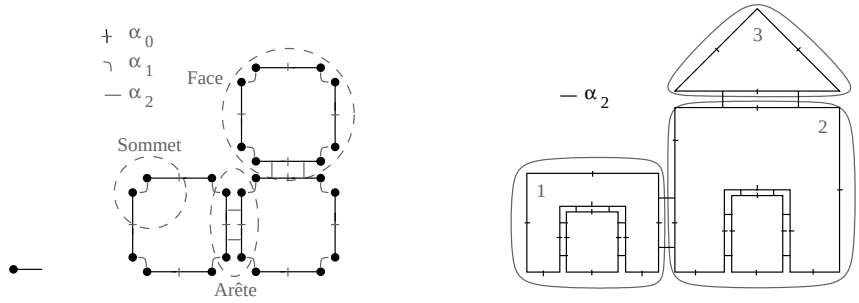


FIG. 2 – a. Schéma représentant un brin. - b. Sur une carte généralisée simple, trois types de cellules sont mise en évidence : une face (orbite $< \alpha_0, \alpha_1 >$), une arête (orbite $< \alpha_0, \alpha_2 >$) et un sommet (orbite $< \alpha_1, \alpha_2 >$). - c. Exemple de partition de l'espace en carte généralisée munie d'un étiquetage. Nous représentons ici une maison subdivisée en trois groupes principaux : l'habitat, le toit et le garage.

3.3 Multi-partition et niveaux de détail d'un objet

Dans cette partie, nous décrivons l'utilisation de l'étiquetage pour représenter les mécanismes de multi-partition et de niveau de détail. La multi-partition (voir figure 1) peut être représentée à l'aide de plusieurs étiquetages. La définition suivante ne spécifie aucune contrainte pour la multi-partition, les décompositions de l'objet étant *a priori* indépendantes les unes des autres.

Définition 3 *Soient $G_n = (B, \alpha_0, ..., \alpha_n)$ une carte généralisée de dimension n et P fonctions de partition $(\phi_n^p)_{p\in\{1..P\}}$. On dit que l'ensemble de ces fonctions de partitions $\Phi_n = \{\phi_n^p\}_{p\in\{1..P\}}$ est une multi-partition.*

Un niveau de détail est une suite de partitions de l'objet qui, à la différence de la multi-partition, sont liées entre elles par une relation d'inclusion. Une multi-partition est un ensemble de cellules (un groupe) et les niveaux de détail correspondent à des ensembles d'ensembles (groupes de groupes). Le premier niveau de détail correspond à un regroupement des cellules de l'objet et tous les autres niveaux peuvent être déduits du précédent en réunissant les groupes du niveau de détail plus précis. Un niveau de détail donné peut être considéré comme la réunion de n-cellules de l'objet initial. Nous pouvons ainsi définir, à partir d'une suite ordonnée d'étiquetages, les niveaux de détail d'un objet, comme le montre la figure 4.a. La définition suivante décrit la relation d'ordre associée à la suite d'étiquetages : les groupes de l'étiquetage d'un niveau de détail donné sont définis à partir des groupes du niveau de détail précédent.

Définition 4 *Soit une carte généralisée de dimension n représentant un objet. Une famille de D étiquetages $(\phi_n^d)_{d\in\{1..D\}}$ représente des niveaux de détail de cet objet si et seulement si chaque groupe étiqueté par $\phi_n^d, d \in 1..D$ correspond à une union des groupes de ϕ_n^{d-1} :*
– *$\forall d > 1$, $\phi_n^d \neq \phi_n^{d-1}$;*
– *et $\forall id \in \mathbb{N}$ un numéro de groupe de la fonction ϕ_n^{d-1}, tous les brins numérotés par id appartiennent à un même groupe de ϕ_n^d, i.e. $\exists id' \in \mathbb{N}$ tel que $\phi_n^d(\{b \in B/\phi_n^{d-1}(b) = id\}) = \{id'\}$.*
$\forall d > 1$, ϕ_n^{d-1} est un détail *de ϕ_n^d.*

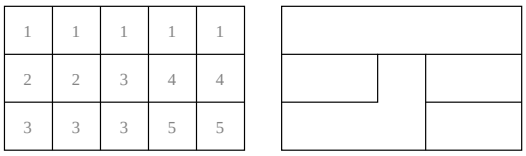


FIG. 3 – a. Un objet détaillé par des 2-cellules (faces) et une fonction d'étiquetage associée. - b. Niveau de simplification déduit de cet étiquetage.

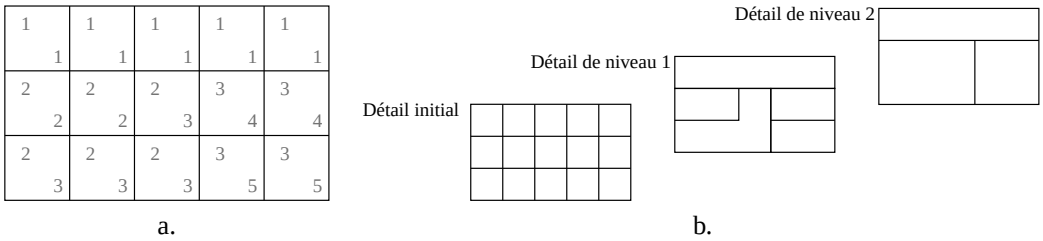


FIG. 4 – a. Un objet détaillé par des 2-cellules (faces) et deux niveaux d'étiquetage donnés par la suite (ϕ_2^1, ϕ_2^2). L'étiquetage ϕ_2^1 (dans le coin inférieur droit) correspond à un niveau de détail intermédiaire entre le détail maximal et le niveau de détail le moins précis représenté par ϕ_2^2 (affiché dans le coin supérieur gauche des 2-cellules). - b. Niveaux de simplifications correspondant à ces étiquetages.

Étant donnée une G-Carte et un niveau de détail, nous pouvons en déduire une G-Carte simplifiée, correspondant à une fusion des cellules de même étiquette (cf. figures 3.b et 4.b). Cela implique non seulement la fusion de cellules de dimension n, mais aussi des cellules de dimension inférieure. Pour définir formellement cette opération, nous utilisons l'étiquetage pour des i-cellules (i de 1 à n) et des opérations de fusion de ces cellules. Pour un modeleur de bâtiments, nous pouvons utiliser au choix un processus automatique (à partir de tests de coplanarité des faces et d'alignement des segments) ou une description manuelle des regroupements de cellules. Sur l'exemple de la figure 5, nous illustrons la simplification d'un escalier. De manière automatique, il est uniquement possible de déduire la simplification (c) à partir de l'objet initial (a). Alors qu'avec la numérotation des i-cellules (faces et arêtes), l'utilisateur peut pousser la simplification jusqu'à remplacer l'escalier par le plan incliné (d).

Pour un niveau de détail d donné, l'étiquetage des i-cellules (des $n-1$ hyperplans jusqu'aux arêtes) est donné par la fonction ϕ_i^d (avec i respectivement de $n-1$ à 1). La figure 5 illustre la notion de simplification définie par un étiquetage.

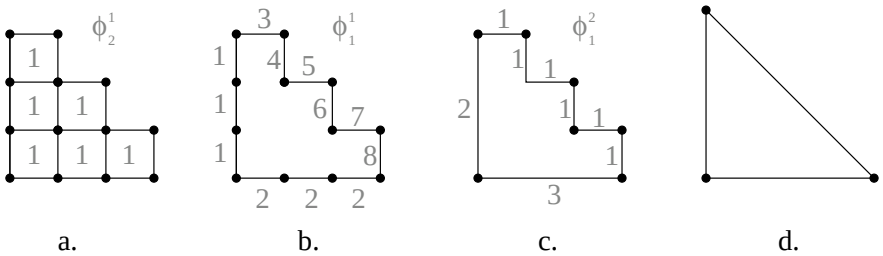


FIG. 5 – a. Détail initial d'un escalier (de précision maximale) et fonction d'étiquetage ϕ_2^1 (des faces) indiquant le niveau de détail suivant. - b. Simplification intermédiaire mettant en évidence l'étiquetage ϕ_1^1 (des arêtes). - c. Niveau de détail déduit des étiquetages précédents (ϕ_2^1,ϕ_1^1), et nouvel étiquetage des arêtes pour la simplification suivante ϕ_1^2 - d. Détail le plus simple de l'escalier déduit de ϕ_1^2.

4 Structure finale : hiérarchie de multi-partitions

L'étiquetage est un mécanisme simple pour définir une structure théorique intégrant les notions de multi-partition et de niveau de détail. Néanmoins, notre objectif est de définir une structure optimisée pour la modélisation et la visualisation d'environnements architecturaux complexes. Les contraintes imposées par la gestion de la mémoire et les temps de calcul doivent être prises en compte. En effet, le modèle d'un bâtiment meublé nécessite le stockage d'un nombre très important de de partitions, de niveaux de détail et de brins.

Pour parvenir à une occupation mémoire raisonnable et conserver l'efficacité des parcours, nous proposons deux optimisations. D'une part, un marquage des liaisons α_n est utilisé pour représenter les multi-partitions (détails dans la sous-section 4.1). D'autre part, les niveaux de détails peuvent être représentés l'aide d'une hiérarchie de cartes généralisées (sous-section 4.2).

4.1 Optimisation pour les multi-partitions : liaisons groupantes

Pour les complexes architecturaux, les objets modélisés sont composés de groupes de n-cellules *connexes*. Par exemple, les murs et les sols délimitent des pièces et celles-ci sont connectées par des ouvertures. Nous avons donc opté pour une solution médiane entre coût mémoire et accès aux données : le marquage des liaisons α. En effet, les liaisons α_n relient les n-cellules entre elles. Ainsi, une marque booléenne sur chaque liaison α_n nous permet d'indiquer qu'elle est groupante, c'est à dire qu'elle relie deux n-cellules du même groupe (voir le premier schéma de la figure 6).

Définition 5 *Soit une carte généralisée $G_n = (B, \alpha_0, ..., \alpha_n)$ et une fonction de partition ϕ_n. ϕ_n représente une partition de G_n en groupes connexes si et seulement si $\forall b_1 et b_2 \in B$ deux brins tels que $\phi_n(b_1) = \phi_n(b_2)$, alors il existe un chemin allant de b_1 à b_2 en ne passant que par des brins ayant pour numéro de partition $\phi_n(b_1)$. Dans ce cas, nous définissons l'involution groupante α_n^g définit par α_n restreinte aux partitions de la fonction ϕ_n, i.e. soient deux brins b_1 et b_2 cousus par α_n :*
– *si $\phi_n(b_1) = \phi_n(b_2)$, alors $\alpha_n^g(b_1) = b_2$ et $\alpha_n^g(b_2) = b_1$.*
– *sinon, $\alpha_n^g(b_1) = b_1$ et $\alpha_n^g(b_2) = b_2$.*

La définition des cellules groupantes permet de créer de nouvelles orbites, que nous appelons *orbites de groupes*, sur lesquelles nous pouvons stocker une sémantique, par exemple des caractéristiques de matériau ou des attributs de nomenclature. Dans les rares cas où l'étiquetage ne définit pas des groupes connexes de cellules, une étiquette peut être ajoutée parmi ces caractéristiques pour définir un numéro de groupe. Cette définition est suffisamment générale pour permettre une extension à plusieurs partitions dans une même carte généralisée. A chaque partition est associée une liaison groupante.

Définition 6 *Soient une carte généralisée $G_n = (B, \alpha_0, ..., \alpha_n)$ et p fonctions de partition ϕ_n^1, ..., ϕ_n^p. Pour chaque fonction ϕ_n^k avec $1 \leq k \leq p$, nous posons l'involution $\alpha_n^{g_k}$ telle que :*
– *$\alpha_n^{g_k}(b) = \alpha_n(b)$ si $\phi_n^k(b) = \phi_n^k(\alpha_n(b))$*
– *et $\alpha_n^{g_k}(b) = b$ sinon.*

La figure 6 montre un exemple simple d'utilisation des multi-partitions sur une carte généralisée de dimension 2. Notons que notre représentation prend en compte le cas (non illustré sur la figure) où deux liaisons $\alpha_n^{g_i}$ sont superposées. En effet, le marquage d'une involution groupante est réalisé à l'aide d'un booléen. Nous pouvons ainsi avoir plusieurs marquages sur une même liaison, tout en conservant un coût mémoire acceptable (un octet pour huit partitions différentes).

4.2 Optimisation pour les niveaux de détails : hiérarchie

Pour la représentation des niveaux de détail d'un objet par une hiérarchie, l'objet le plus simple est situé au niveau de la racine, et chaque sous-arbre correspond à un détail d'une partie de l'objet. Nous utilisons ce principe avec les cartes généralisées pour éviter un étiquetage explicite. La structure que nous proposons permet de travailler localement sur une partie de l'objet (sous-arbre) afin de réduire l'occupation mémoire par une technique de chargement

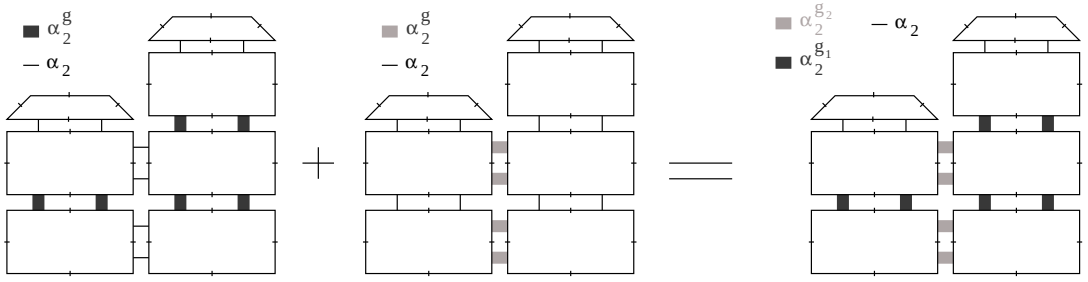


FIG. 6 – Liaisons groupantes : à gauche sont représentées deux partitions différentes d'un même bâtiment. - a. Une première décomposition en deux ailes - b. Une seconde en trois étages. - c. La G-Carte bi-partitionnée correspondante.

partiel des données en mémoire (*swap*). De la même manière, elle facilite l'utilisation d'algorithmes de visualisation multi-échelle. Notons toutefois que cette structure impose une définition précise de la relation de filiation et des critères de cohérence. Dans cet article, nous présentons les contraintes uniquement de manière intuitive, à partir de la définition de niveau de détail par étiquetage.

Définition informelle 1 *Soient une carte généralisée détaillée G^0 et un niveau d'étiquetage $(\phi_i^1)_{i\in(1..n)}$ exprimant un niveau de simplification. Nous appelons G^1 la carte simplifié de G^0 issu de l'étiquetage $(\phi_i^1)_{i\in(1..n)}$, la carte généralisée obtenue après une succession d'opérations de fusion des i-cellules suivant un ordre décroissant des dimensions (de n à 1).*
Dans ce cas, nous posons l'application injective[5] *η, appelé liaison hiérarchique, qui relie chaque brin de G^1 avec le brin de G^0 correspondant.*

L'opération de fusion, classique en topologie, a récemment été redéfinie de manière précise pour les cartes généralisées en dimension quelconque par Damiand et Lienhardt [DL02] (voir également [ELT94]). La figure 7.a montre un exemple de succession de fusions pour obtenir une simplification d'objet.

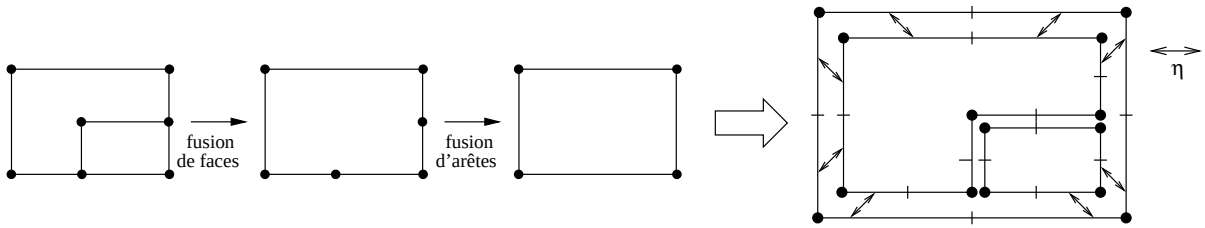


FIG. 7 – a. Simplification d'un objet par une série de fusions de cellules : un couple de faces, puis deux couples d'arêtes. - b. Mise en évidence de la liaison hiérarchique η qui relie les brins de la simplification à son détail.

La notion de simplification est directement issue de la relation d'ordre qui existe entre ϕ_n^i et ϕ_n^{i-1}. Cette relation nous aide à définir les contraintes de filiation entre deux cartes généralisées. Afin d'éviter les redondances, lorsqu'une G-Carte G^1 est une simplification de la carte G^0, nous supprimons dans G^0 les i-cellules ($i > 1$) communes à G^0 et G^1. Dans G^0, les composantes connexes résultantes de cette opération forment chacune une nouvelle G-Carte appelée *détail* de G^1. Toutes ces G-Cartes représentent les fils de G^1. Certaines i-cellules redondantes ne sont néanmoins pas supprimées lorsqu'elles permettent à un niveau de détail de conserver toutes les informations nécessaires à sa représentation. Cela permet d'éviter des parcours coûteux de la hiérarchie. Notons que dans le cadre particulier de la modélisation de bâtiments, les calculs de visualisation sont réalisés en dimension 3 pour des

[5] L'application est une injection, car l'ensemble des brins de G^0 a plus d'éléments que celui de G^1 et chaque brin de G^0 possède au plus un antécédent dans G^1.

volumes. Par conséquent, la structure hiérarchique que nous avons mise en place concerne des niveaux de détail de cellules de dimension 3.

Définition 7 *Un arbre de cartes généralisées H est soit vide soit un triplet (G, Γ, η), appelé noeud, où :*
- *G est une carte généralisée de dimension n, appelée racine ;*
- *Γ est un ensemble de sous-arbres disjoints, tel que $\forall \gamma$ une carte généralisée racine d'un sous-arbre de Γ, γ est un* détail *de G ;*
- *η est la fonction de correspondance décrivant l'ensemble des liaisons hiérarchiques entre les i-cellules de la carte parent G et les G-Cartes de détails, racines des sous-arbres de Γ.*

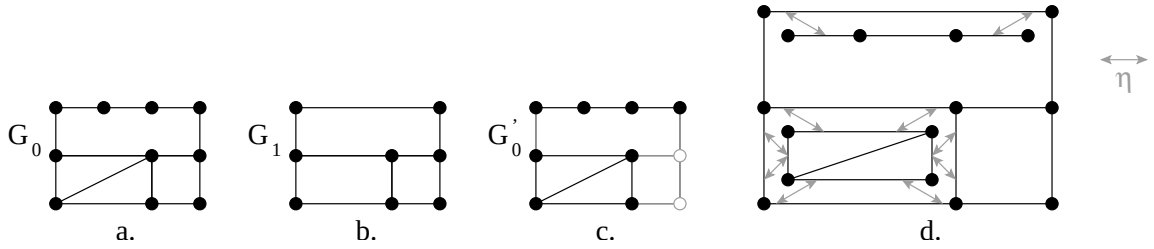


FIG. 8 – a. Un objet représenté à un niveau détaillé par une carte G_0. - b. Une simplification G_1 de cet objet. - c. Pour construire la hiérarchie, nous supprimons dans G_0 les cellules en commun avec G_1 pour obtenir une carte G'_0. - d. Les composantes connexes de G'_0 sont ensuite mis en correspondance avec la simplification G_0 pour former la hiérarchie de détail.

Finalement, la structure proposée intègre à la fois les niveaux de détail et la multi-partition d'un objet, tout en respectant les impératifs de faible occupation mémoire et de rapidité d'accès. À ce stade, notre structure se rapproche de la notion de graphe orienté acyclique[6] de simplexes proposée par De Floriani *et al* [FM01].

5 Résultats

Autour de la structure présentée dans la section précédente, nous avons développé un ensemble d'opérations dédiées à la construction de complexes architecturaux ainsi qu'une interface graphique pour un premier prototype de modeleur de bâtiment. Cette section est dédiée à la présentation de ces opérations et du modeleur. À cette fin, nous illustrons les étapes de modélisation pour un bâtiment très simple, puis montrons avec un exemple les capacités du modeleur à créer un complexe architectural plus réaliste.

La première étape de construction correspond à la définition d'un contour servant de base à l'édifice (voir figure 9.a). Une première G-Carte généralisée, racine de la hiérarchie, reçoit l'enveloppe extérieure du bâtiment 3D obtenue par une première extrusion (figure 9.b). Puis le premier détail, défini dans une seconde G-carte, correspond à une copie du contour auquel une seconde opération d'extrusion est appliquée pour obtenir l'épaisseur des murs (figure 9.b). Cette figure forme les fondations du bâtiment. Un premier étage est obtenu par extrusion des fondations, puis recopié autant de fois que le souhaite l'utilisateur. Les copies sont superposées pour finaliser le premier détail du bâtiment (figure 9.d).

L'interface graphique permet de détailler indépendamment chacun des étages. Pour cela, l'utilisateur sélectionne un étage, automatiquement recopié dans une carte généralisée fille, pour créer un nouveau niveau de détail (figure 9.e). L'étage est alors visualisé comme sur un plan en deux dimensions à l'aide d'une projection orthogonale. L'utilisateur peut alors dessiner des murs (figure 9.f) et des ouvertures telles que des portes ou des fenêtres (figure 9.g). Chaque pièce déduite de la création des murs peut à son tour être précisée dans une nouvelle carte généralisée fille. Une fonction d'importation d'objets depuis des fichiers décrivant des listes de faces (comme le format VRML 1.0 / Inventor) permet d'insérer des meubles dans les pièces. À tout moment, l'utilisateur peut se déplacer dans la hiérarchie à l'aide de l'arborescence du bâtiment (figure 10. a), ou afficher la totalité du bâtiment (figures 9.h et 10.b). Les affichages sont réalisés à l'aide de parcours en profondeurs de la hiérarchie. Par ailleurs, il est également possible pour chaque G-Carte de la hiérarchie de définir des groupes de volumes (pièces, murs, etc.)

[6] En anglais, Directed Acyclique Graph (DAG).

pour créer des partitions du bâtiment. Les multi- partitions peuvent être mises en évidence soit lors de la sélection explicite d'un groupe, soit lors de l'affichage des liaisons groupantes.

Pour finir, la figure 10 illustre un exemple complet de complexe architectural créé à l'aide de notre modeleur. Il est inspiré du bâtiment hébergeant notre laboratoire : le SP2MI sur le site du Futuroscope.

6 Conclusion

La difficulté principale de la gestion de complexes architecturaux en synthèse d'image concerne la quantité de données à prendre en compte et à traiter aussi bien lors de la modélisation que pour la visualisation. Dans cet article, nous présentons une structure de données à base topologique pour la modélisation et la visualisation de grands bâtiments. La structure proposée étend le modèle des cartes généralisées et intègre deux notions qui nous paraissent essentielles : (1) la multi-partition permettant de représenter plusieurs décompositions d'un objet donné, par exemple le regroupement des pièces d'un bâtiment selon leurs sémantiques, et (2) la notion de niveau de détail souvent utilisée pour la visualisation et permettant également la construction d'un bâtiment par décomposition progressive.

Les perspectives de ce travail peuvent être réalisées selon deux axes principaux : l'utilisation de la structure pour la visualisation et l'intégration du modèle à d'autres domaines de la modélisation. Le premier axe, en cours d'étude actuellement, consiste à intégrer notre modèle à un outil de simulation d'éclairage et de visualisation interactive. Nous allons étudier de manière approfondie l'apport des informations topologiques et des niveaux de détail aux phases de pré-calcul de visibilité dans une scène complexe. Le second point concerne les opérations de construction pour l'instant dédiées aux bâtiments. En l'état, le modèle est intrinsèquement lié aux environnements architecturaux. Il nous semble cependant que la structure est suffisamment générale pour des utilisations dans d'autres domaines (géologie, conception automobile) ou pour des dimensions supérieures (animation), à condition de compléter les opérations associées. Pour cela, certains points importants doivent être étudiés en priorité. Par exemple, pour le moment le déplacement des objets ou des murs dans une pièce peut engendrer des modifications topologiques importantes non prises en compte au niveau de la hiérarchie. Le traitement de ces modifications nécessite de définir des contrôles de cohérence sur la structure, impliquant l'utilisation d'opérations telles que le co-raffinement.

Pour affiner la structure, quelques compléments sont envisagés. Par exemple, le clonage (très souvent utilisé en modélisation) peut participer à la réduction du nombre d'objets mémorisés dans la structure. Cependant, l'intégration de cette notion implique une une modification de la définition et des traitements de la structure car un noeud de la hiérarchie peut posséder plusieurs parents. Un autre complément concerne la gestion de plongements non linéaires, permettant d'obtenir une compression des informations géométriques de notre structure (une suite de segments formant une courbe peut être remplacée par une seule arête topologique dotée d'une équation).

Références

[BAU72] Bruce BAUMGART. Winged-edge polyhedron representation. In *Technical Report CS-320, Stanford University,CA*, 1972.

[BRI89] Erik BRISSON. Representing geometric structures in d dimensions : topology and order. In *Proceedings of 5th ACM Symposium of Computational Geometry, Saarbrücken, Germany*, pages 218–227, 1989.

[CDM+94] Paolo CIGNONI, Leila De FLORIANI, Claudio MONTANI, Enrico PUPPO, and Roberto SCOPIGNO. Multiresolution modeling and visualization of volume data based on simplicial complexes. *Symposium on Volume Visualization*, pages 19–26, 1994.

[DL02] Guillaume DAMIAND and Pascal LIENHARDT. Removal and contraction for n-dimensional generalized maps. In *Computer Vision Winter Workshop*, pages 208–221, Bad Aussee, Austria, february 2002.

[DM01] Leila De FLORIANI and Paola MAGILLO. Multiresolution modeling of three-dimensional shapes. Chapter 2 in 3D Synthetic Environment Reconstruction, M. Abdelguerfi (Ed.), Kluwer Academic Publishers, Boston, 2001. pp. 35-59.

[DPM97] Leila De FLORIANI, Enrico PUPPO, and Paola MAGILLO. A formal approach to multiresolution hypersurface modeling. In W. Straßer, R. Klein, and R. Rau, editors. Geometric Modeling : Theory and Practice. Springer Verlag., 1997.

[ELT94] Hervé ELTER. *Etude de structures combinatoires pour la représentation de complexes cellulaires*. PhD thesis, Université Louis Pasteur de Strasbourg, 1994.

[FF88] Leila De FLORIANI and Bianca FALCIDIENO. A hierarchical boundary model for solid object representation. *ACM Transactions on Graphics*, 7(1) :42–60, 1988.

[FM01] Leila De FLORIANI and Paola MAGILLO. Multiresolution meshes, principles of multiresolution in geometric modeling. Primus01 summer school Munich, August 2001. pp. 193-234.

[GUI00] Oskar GUILBERT. *Un Modèle Hiérarchique pour la Modélisation Géométrique à Base Topologique*. PhD thesis, Université Louis Pasteur de Strasbourg, Janvier 2000.

[KRO95] Walter KROPATSCH. Building irregulars pyramids by dual graph contraction. In *IEEE Proceedings. Vision, Image and Signal Processing*, volume 142, pages 366–374, 1995.

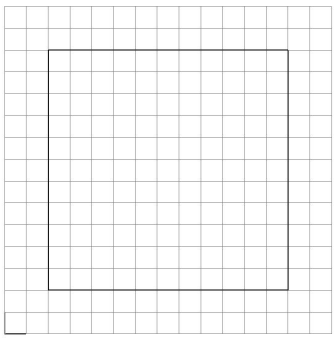
[LEV99] Bruno LEVY. *Topologie Algorithmique : Combinatoire et Plongement*. PhD thesis, Institut National Polytechnique de Lorraine, Octobre 1999.

[LIE89] Pascal LIENHARDT. Subdivisions of n-dimensional spaces and n-dimensional generalized maps. In *Symposium on Computational Geometry*, pages 228–236, 1989.

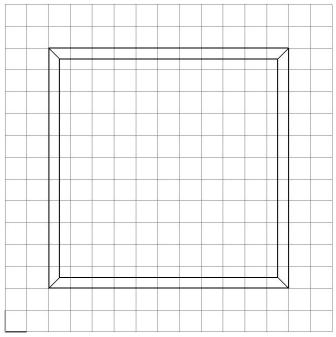
[LIE94] Pascal LIENHARDT. N-dimensional generalized combinatorial maps and cellular quasi-manifolds. *International Journal of Computational Geometry and Applications*, 4(3) :275–324, 1994.

[PFP95] Valerio PASCUCCI, Vincenzo FERRUCCI, and Alberto PAOLUZZI. Dimension-independent convex-cell based hierarchical polyhedral complex : Representation scheme and implementation issues. In *SMA '95 : Proceedings of the Third Symposium on Solid Modeling and Applications*, pages 163–174, 1995.

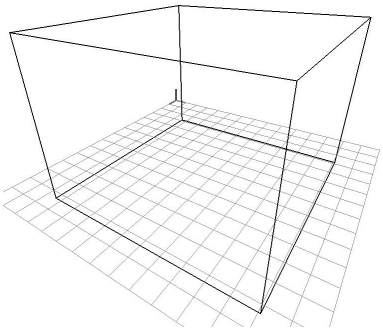
[WEI85] Kevin WEILER. Edge-based data structures for solid modeling in curved-surface environments. *IEEE Computer Graphics and Applications*, 5(1) :21–40, Janvier 1985.



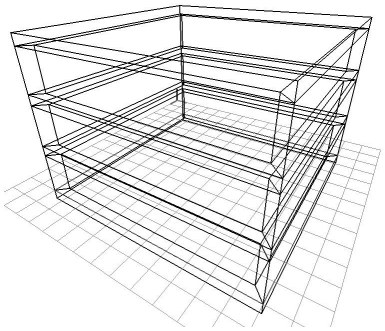
a. Définition du contours du bâtiment.



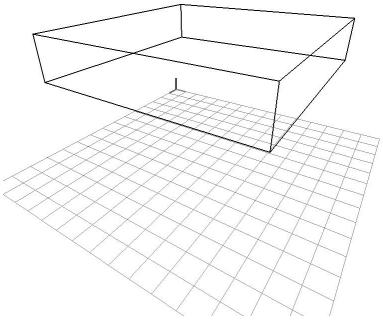
b. Positionnement des murs extérieurs.



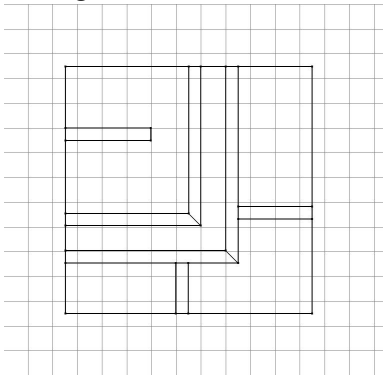
c. Extrusion du contours *(a)* pour obtenir le profil de bâtiment.



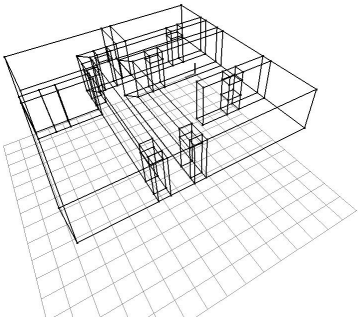
d. Extrusion des murs *(b)* pour obtenir chacun de ses étages.



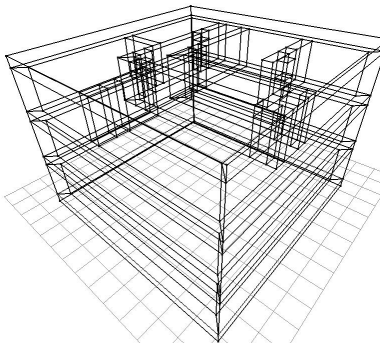
e. Description indépendante de chaque étage.



f. Mise en place des murs intérieurs d'un étage.

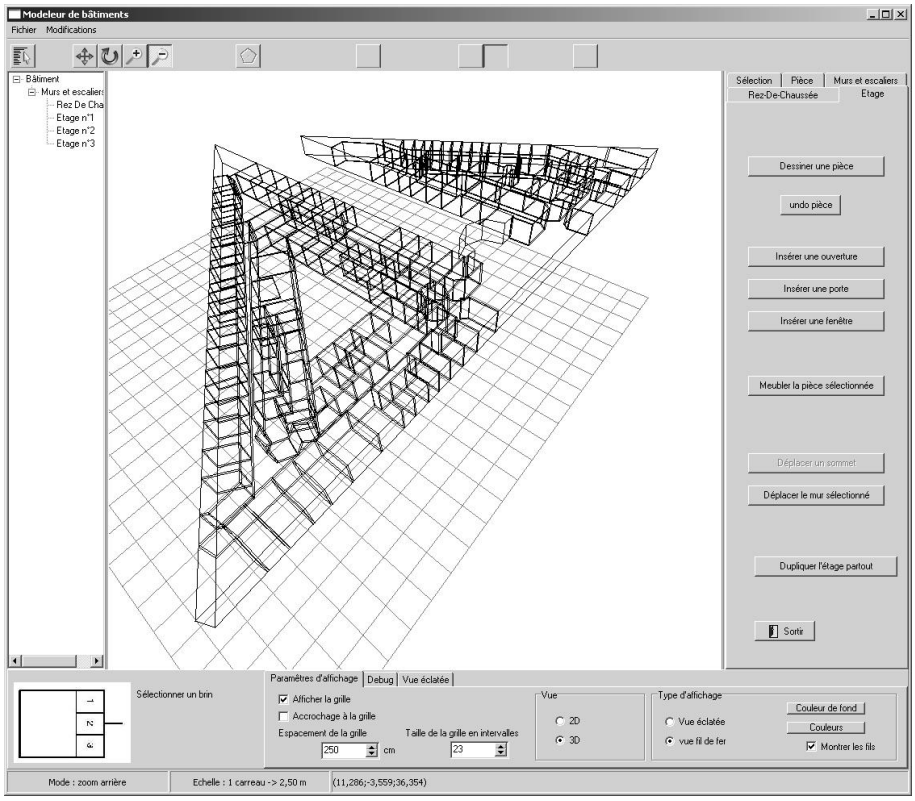


g. Mise en place des ouvertures (portes).

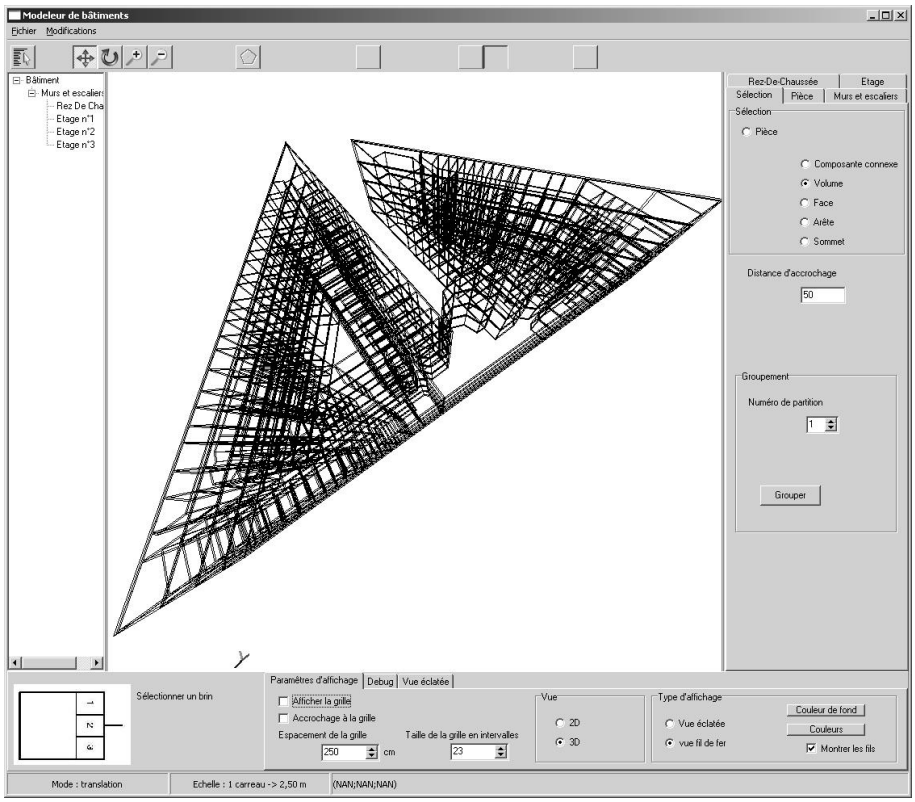


h. Affichage complet du bâtiment.

FIG. 9 – Étapes de construction d'un bâtiment.



a.



b.

FIG. 10 – Bâtiment inspiré par les plans du SP2MI : a. vue d'un étage. - b. vue du bâtiment complet.

La chambre photographique

S. Michelin, C. Pichard[1]

Université de Marne- la- Vallée, Institut Gaspard Monge, Equipe SISAR
6 cours du Danube, 77700 Serris, France
michelin@univ-mlv.fr,cyril@duboi.com

Résumé : *Les modèles de caméra existants ne sont pas adaptés à la simulation des caractéristiques optiques particulières de la chambre photographique. Nous proposons donc dans cet article deux nouvelles méthodes, fondées sur le fonctionnement de la chambre photographique, dans le but de recréer deux de ses effets optiques. La première méthode permet de réaliser des anamorphoses en temps réel en utilisant simplement deux paramètres et peut remplacer la caméra OpenGL classique. La deuxième méthode repose sur une application du lancer de rayon distribué qui permet de générer du flou de profondeur de champ dont la répartition spatiale ne se fait plus frontalement.*

Mots-clés : Photographie ; modèle de caméra ; projection perspective ; warping ; lancer de rayon distribué ; OpenGL.

1 Introduction

La chambre photographique est un dispositif photographique rudimentaire destiné aux photographes professionnels. Elle permet néanmoins un contrôle plus grand de l'image que les appareils photographiques classiques, comme par exemple la possibilité de redresser les lignes fuyantes, ce qui est très utile en photographie d'architecture. Nous nous sommes donc fixé comme objectif de réaliser un modèle de caméra simulant deux des caractéristiques optiques les plus intéressantes de la chambre photographique : l'anamorphose et le changement d'orientation de la profondeur de champ. Pour cela, après avoir présenté et comparé la projection perspective au modèle de la lentille mince, nous donnons un état de l'art non exhaustif des modèles de caméra existants en synthèse d'image. Dans un second temps nous décrivons les caractéristiques mécaniques et optiques de la chambre photographique. Ensuite nous proposons une méthode de déformation perspective rapide suivant le principe mécanique de la chambre photographique. Enfin, pour simuler une orientation du plan de netteté, nous proposons une modification de l'implémentation du lancer de rayon distribué qui prend en compte les orientations du plan film et du plan de netteté.

2 Les modèles de caméras

2.1 La caméra perspective

En synthèse d'image, le rôle de la caméra est de retranscrire une scène tridimensionnelle sur un plan image. Les opérations effectuées par les caméras sont d'ordre géométrique, et font appel en grande partie à la géométrie projective. La caméra la plus utilisée est la caméra perspective. Elle consiste en l'application du principe de réduction perspective qui était utilisé autrefois par les peintres. Ce principe permet de rendre de manière rationnelle la diminution ou l'agrandissement des choses qui résulte pour la vision humaine de leur éloignement ou de leur proximité. Ainsi, les premiers appareils photographiques étaient utilisés pour réaliser des perspectives dessinées car les supports sensibles à la lumière capables d'enregistrer l'image n'existaient pas encore. La caméra perspective permettait donc de "dessiner" une image proche de celle perçue par la vision humaine. Maintenant ces images sont générées de manière informatique selon le même principe.

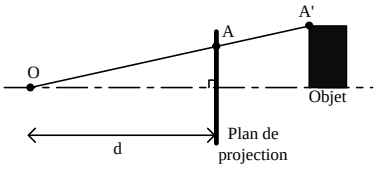


FIG. 1 – Projection perspective

[1] DUBOI, 221 bis bld Jean Jaurès, 92100 BOULOGNE

On retrouve la manière de calculer la projection perspective dans tous les livres de synthèse d'image [FD95, Gla89, WW92]. Si l'on considère une projection perspective simple (pixel carrés, etc...), l'image $P(X, Y, Z, W)$ d'un point $p(x, y, z, w)$, est donnée par la formule :

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} d.x/z \\ d.y/z \\ d \\ 1 \end{bmatrix} \tag{2.1}$$

On remarque que tous les points sont positionnés dans le plan $z = d$ perpendiculaire à l'axe de visée. Ce plan est le plan de projection et la valeur d donne sa position sur l'axe Z (cf Figure 1).

2.2 Lentille mince

En optique, le modèle le plus simple associé à un objectif de prise de vue est une lentille mince. La caméra perspective et la lentille mince n'ont pas les mêmes propriétés. L'image $P(X, Y, Z, W)$ d'un point $p(x, y, z, w)$ par une lentille mince est donnée par la formule :

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f' & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} (f'.x)/(z+f') \\ (f'.y)/(z+f') \\ (f'.z)/(z+f') \\ 1 \end{bmatrix} \tag{2.2}$$

La valeur f' est une caractéristique physique de la lentille appelée distance focale. On voit donc avec cette formule que l'image de tous les objets de la scène n'est pas une image plane mais une image tridimensionnelle. En général, la lentille mince peut-être assimilée à une caméra perspective lorsque l'objet photographié est à une distance telle que la distance focale devienne négligeable : $z + f' \approx z$. Dans ce cas le plan de projection est positionné sur la distance focale $d = f'$, d'où la confusion généralement faite dans les livres de synthèse d'image entre la distance focale et la distance du plan de projection.

La prise de vue est réalisée en plaçant un plan dans la zone image de la lentille. L'image d'un point sur un plan est alors un point ou une tache (cf Figure 2). De cette manière, les optiques font apparaître des zones de netteté et des zones de flou. La taille de la tache de confusion dépend de l'ouverture de la lentille.

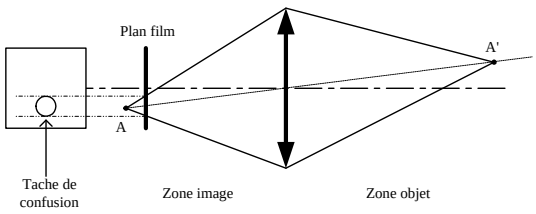


FIG. 2 – Principe du flou de profondeur de champ.

2.3 Modèles de caméra évolués

Il est donc clair que la caméra perspective ne permet pas de simuler les caractéristiques visuelles des objectifs de prise de vue cinéma ou photo, que ce soit au niveau géométrique (distorsion optique) ou radiométrique (absorption des lentilles). Plusieurs travaux traitent de la simulation des effets visuels obtenus par les optiques. Ces recherches se divisent en deux grands axes : la simulation par un traitement ajouté à l'image de synthèse (post-traitement) et la simulation du parcours de la lumière dans l'objectif.

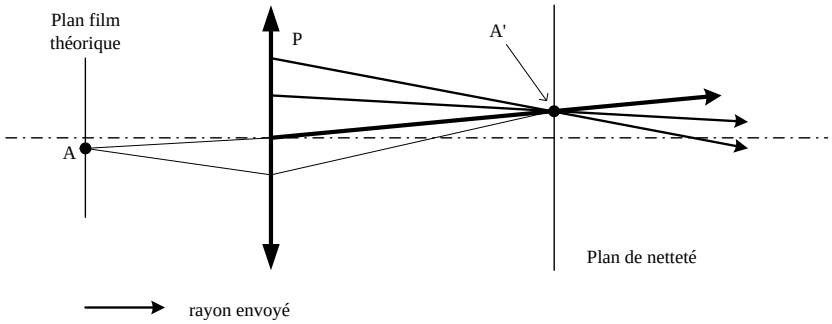


FIG. 3 – Lancer de rayon distribué

2.3.1 Simulation par post-traitement de l'image

Potmesil et al, dans leur article publié en 1981 [PC81], nous présentent une méthode pour générer du flou de profondeur de champ en appliquant une convolution dont les paramètres varient en fonction de la profondeur du point projeté. Les matrices de convolution utilisées dans ces travaux sont déduites du calcul des caractéristiques des taches de confusion créées au travers d'une lentille mince par des points situés à des intervalles donnés. Shinia [Shi94] perfectionne la méthode en détournant l'algorithme du lancer de rayon pour connaître les pixels qui vont être masqués par d'autres.

2.3.2 Simulation du parcours de la lumière

Acquisto et Groller [AG93] proposent des caméras alternatives (hémisphérique, cylindriques) en utilisant le lancer de rayon et en modifiant la forme des plans de projections. Ces caméras alternatives permettent par exemple de simuler des caméras hémisphériques, mais aussi les défauts d'aplanétisme des systèmes optiques.

Cook, Porter et Carpenter [CPC84] présentent en 1984 une méthode issue de l'anti-aliasing en lancer de rayon pour simuler notamment du flou de profondeur de champ et du flou de mouvement. Cette méthode porte le nom de lancer de rayon distribué. Leur idée est d'utiliser les rayons servant à l'anti-aliasing pour simuler le faisceau de lumière qui arrive sur un point image. L'algorithme est décrit dans l'encadré si dessous. En pratique, on utilise le plan de projection comme plan de netteté et la notion de distance focale disparaît complètement ou est assimilée à la distance de mise au point. Cela provient du fait que les plans (projection, netteté) sont tous perpendiculaires à l'axe optique et que l'algorithme permet de placer le plan de netteté à n'importe quelle position sur l'axe.

Algorithme 1 Lancé de rayon distribué

pour tous les points A du plan film associés aux pixels de l'image **faire**
  trouver l'image A' du point A par la lentille dans le plan de netteté
  pour tous les points P' choisis aléatoirement sur la surface de la lentille **faire**
    envoyer un rayon PA'
  fin pour
  la couleur du pixel associé au point A est la moyenne des couleurs trouvées par tous les rayons envoyés
fin pour

Kolb, Mitchell et Hanrahan [KMH95] nous présentent un modèle de caméra qui est censé simuler les caractéristiques physiques de l'objectif. Leur modèle repose sur la simulation physique du parcours réel de la lumière entre différentes lentilles constituant l'objectif. Leur modèle permet de simuler avec précision la géométrie et la radiométrie de l'image formée, même s'il comporte des simplifications importantes comme la réflexion entre lentille jugée nulle. De plus elle nécessite un temps de calcul relativement important.

Enfin, Haeberli et Akelay [HA90] présentent un moyen de réaliser du flou de profondeur de champ avec OpenGL [NDW99]. Cette méthode repose sur l'accumulation d'images prises de différents points de vue proches. Grace à

l'accélération matérielle, on peut obtenir une simulation de flou de profondeur de champ en temps réel. Néanmoins, leur méthode ne s'appuie sur aucun modèle physique. Heidrich, Slusallek, Seidel [HSS97] étendent ce principe à la simulation de n'importe quel type d'optique en calculant précisément les images accumulées.

3 La chambre photographique

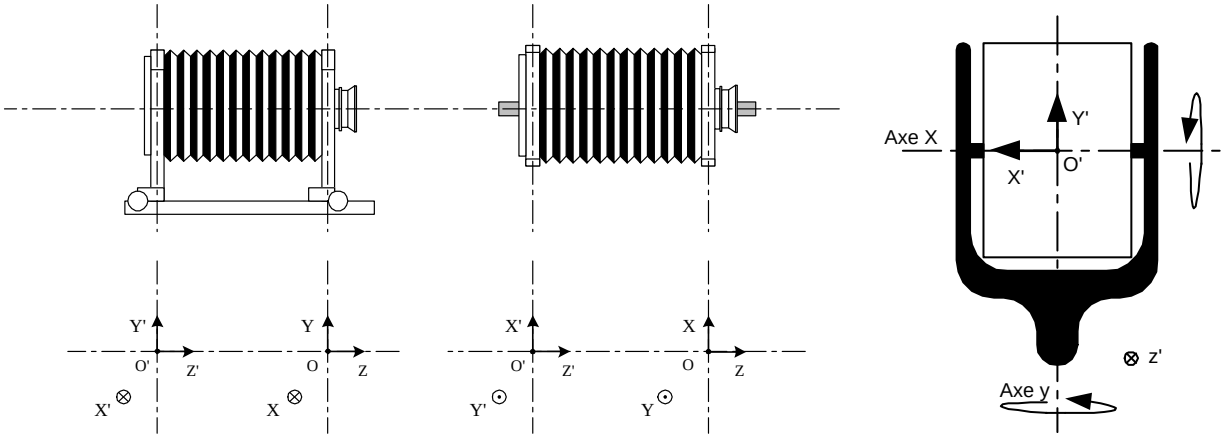


FIG. 4 – La chambre photographique vue de coté, de haut et de derrière

3.1 Présentation

La chambre photographique est l'appareil le plus rudimentaire en photographie après le sténopé. Elle est simplement constituée de deux corps : le corps avant, où repose l'objectif et le corps arrière où va être placée la surface sensible à la lumière, qui va enregistrer l'image (généralement de la pellicule photographique ou un capteur électronique). Les deux corps sont montés mécaniquement sur un support monorail sur lequel ils peuvent se déplacer l'un par rapport à l'autre. Un soufflet étanche à la lumière relie le corps avant et le corps arrière. Le corps arrière comporte généralement un verre dépoli où apparaît l'image inversée du sujet générée par l'objectif. Le verre dépoli va servir à régler l'image et à faire la mise au point, avant de recevoir la surface sensible à la lumière. La chambre photographique permet une meilleure flexibilité du contrôle sur l'image finale.

3.2 Caractéristiques mécaniques

Les chambres photographiques professionnelles possèdent un certain nombre de caractéristiques mécaniques qui les différencient des appareils photographiques classiques.

Les corps avant et arrière de la chambre ont la capacité de pivoter autour de deux axes, l'un vertical, l'autre horizontal. Il peuvent aussi être translatés le long de ces deux axes. La chambre photographique permet donc de décentrer et de basculer le plan film ainsi que l'optique utilisée en combinant ces mouvements. Dans cet article, nous allons exclusivement nous intéresser aux bascules du plan film de la chambre. L'angle de rotation autour de l'axe Y est ϕ et l'angle de rotation autour de X est θ, comme présenté dans la figure 5. Il faut aussi noter que l'axe X va suivre les rotations imposées par l'axe Y, ce qui donne l'ordre des rotations dans la transformation finale du plan $T = R_y R_x$.

3.3 Caractéristiques optiques

Les mouvements des deux corps influencent l'image résultante. Ils jouent sur l'orientation de la profondeur de champ, l'impression de perspective, le recadrage, l'anamorphose et la mise au point. Le tableau suivant résume les effets visualisés lors du déplacement des corps de la chambre en fonction du mouvement appliqué :

	Corps avant	Corps arrière
rotation X/Y	profondeur de champ	profondeur de champ+anamorphose
translation X/Y	perspective (point de vue)	recadrage
translation Z	perspective (point de vue)	mise au point

Le déplacement en translation du corps avant implique un changement de point de vue et donc un changement de perspective, ce qui revient à déplacer l'appareil. Les bascules du corps avant vont avoir un effet sur l'orientation du plan de netteté et donc sur la profondeur de champ mais le point de vue ne change pas. La bascule du corps arrière permet de changer l'impression de perspective en anamorphosant l'image, et de cette manière de corriger les lignes fuyantes, par exemple dans des scènes architecturales (cf Figure 8). De plus, la bascule du corps arrière, comme celle du corps avant va avoir une influence sur l'orientation du plan de netteté. Le décentrement du corps arrière (translation X/Y) permet de recadrer dans l'image et par exemple de prendre des photographies en face d'un miroir sans avoir le reflet de l'appareil[Til92]. Avec les appareils photographiques classiques, le plan de netteté est frontal. La chambre permet de changer l'orientation du plan de netteté et par exemple d'avoir une profondeur de champ verticale (cf Figure 11). Le déplacement sur l'axe des Z du corps arrière permet de faire la mise au point. Lors d'une prise de vue les photographes utilisent la règle de Schempflug [Str86] pour maîtriser l'orientation du plan de netteté. Cette règle dit que le plan de netteté et le plan film s'intersectent en une droite qui appartient au plan de la lentille.

4 Notre méthode de déformation perspective rapide

Dans cette partie, nous proposons une méthode simple qui permet de simuler les anamorphoses dues aux rotations du corps arrière de la chambre en temps réel avec OpenGL. Nous montrons comment une rotation du plan peut être décomposée en une translation et une homothétie du plan de projection et un changement d'axe de visée en nous fondant sur le fonctionnement de la chambre photographique. Il serait possible d'utiliser une transformation perspective en post-traitement, néanmoins, notre méthode apporte une meilleure qualité de rendu sans traitement supplémentaire et une utilisation plus pratique nécessitant seulement le réglage de deux paramètres.

4.1 Hypothèse

Avant tout, on fait l'hypothèse que la chambre photographique peut-être modélisée par une caméra perspective et que le plan de projection est le symétrique du plan film par le point nodal de l'optique qui devient le centre de projection. Les mouvements appliqués au plan film sont donc appliqués au plan de projection, comme on peut l'observer dans la figure 6.

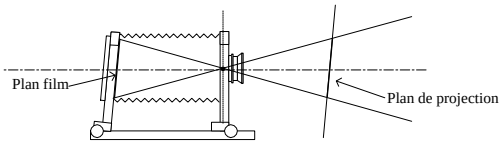


FIG. 6 – Plan film et plan de projection

4.2 Principe proposé

Notre méthode part d'une constatation : basculer le plan de projection revient à faire une nouvelle projection perspective simple en changeant l'axe de visée et avec une fenêtre de projection décentrée par rapport à ce nouvel axe, comme l'illustre la figure 7. En effet, lorsque l'on fait pivoter le plan de projection, on voit apparaître une nouvelle projection perspective avec un axe perpendiculaire au plan de projection basculé et une distance de projection plus courte sur ce nouvel axe. Pour inclure cette méthode dans OpenGL, il faut modifier l'étape de visualisation qui se décompose en deux parties, la transformation de visée, et la projection perspective.

4.2.1 Correction de la transformation de visée

La transformation de visée consiste à placer et orienter la caméra dans la scène. Le nouvel axe de visée est perpendiculaire au plan basculé. On doit donc ajouter à la rotation de visée initiale deux autres rotations correspondant aux rotations du plan de projection.

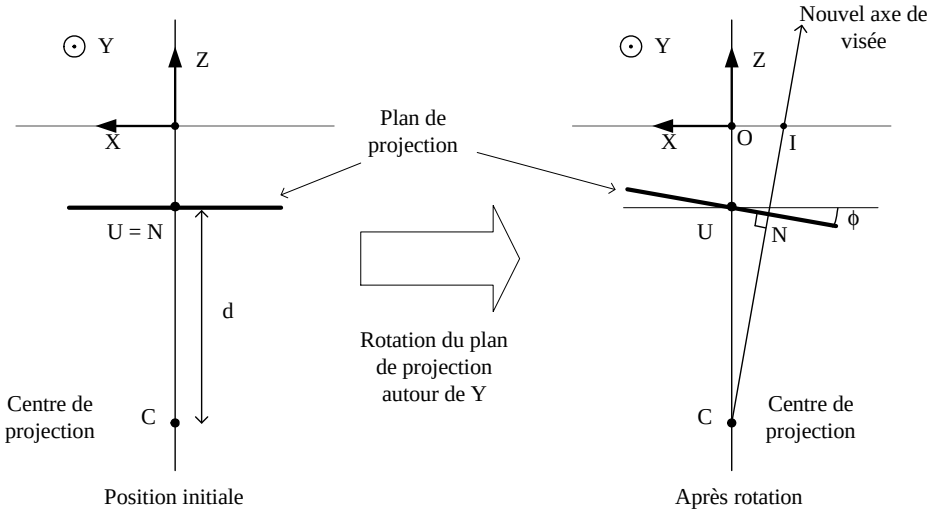


FIG. 7 – Nouvelle projection perspective

La transformation finale est alors $T = R_y R_x$ avec $R_y = \begin{bmatrix} cos\phi & 0 & sin\phi \\ 0 & 1 & 0 \\ -sin\phi & 0 & cos\phi \end{bmatrix}$, $R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{bmatrix}$

4.2.2 Nouvelle transformation perspective

Le plan de la nouvelle projection perspective n'est plus à la même position sur l'axe. Sa nouvelle position $\overline{CN}$ sur l'axe se déduit facilement par le calcul vectoriel et on trouve $\overline{CN} = d.cos\phi.cos\theta$. Le centre du plan de projection a lui aussi changé de position. Le centre de la fenêtre de projection a pour coordonnées dans le nouveau repère :

$$\overrightarrow{UN} = \begin{bmatrix} -d.sin\phi \\ d.cos\phi.sin\theta \\ 0 \end{bmatrix}$$

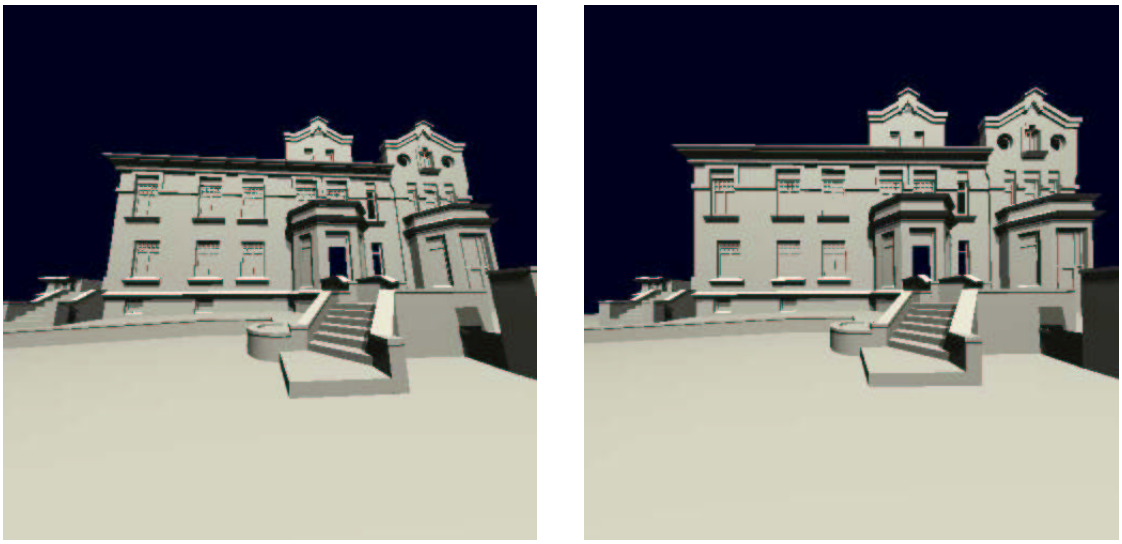
4.2.3 Implémentation

Avec les valeurs trouvées précédemment, on peut modifier n'importe quel code OpenGL pour intégrer les bascules du corps arrière. Un exemple de simulation des bascules est donné ci-dessous. Les décentrements additionnels du plan film arrière sont triviaux à implémenter, il suffit de rajouter un décalage dans la fonction `glFrustrum()`.

```
float teta; // angle de bascule autour de X
float phi;  // angle de bascule autour de Y
float d;    // position du plan sur l'axe

main()
{
....
    /* Modification de la transformation perspective */
    glMatrixMode(GL_PROJECTION);
    glFrustum(  -0.5-d*sin(phi), 0.5-d*sin(phi),
                -0.5+d*cos(phi)*sin(teta), 0.5+d*cos(phi)*sin(teta),
                d*cos(phi)*cos(teta),
                d*1000 )

    /* Modification de la direction de visée */
    glMatrixMode(GL_MODELVIEW);
```

```
    glRotated(teta,0.0,1.0,0.0);
    glRotated(phi,1.0,0.0,0.0);

    /* La scene */
    glutSolidTeapot(0.5);
.....
}
```

4.3 Résultats



(a) plan de projection non basculé *(b) plan de projection basculé*

FIG. *8 – Exemple de redressement des lignes fuyantes avec OpenGL.*

La première image correspond à une prise de vue classique en contre-plongé. Le bâtiment présente des lignes fuyantes que l'on peut corriger en réglant l'orientation du plan de projection. Dans la deuxième image, le plan de projection est parallèle à la façade du manoir. Les lignes fuyantes ne s'intersectent plus. Le manoir donne alors une impression de stabilité, impression utilisée en photographie d'architecture pour mettre en valeur les bâtiments.

5 Orientation du plan de netteté

Nous avons vu précédemment que les rotations des corps de la chambre ont aussi une influence sur l'orientation du plan de netteté ce qui permet d'avoir une profondeur de champ qui n'est plus frontale. Notre but est donc de simuler un flou de profondeur de champ non perpendiculaire à l'axe optique, comme le ferait la chambre photographique.

5.1 Principe

Avec le modèle simple de caméra perspective, on ne peut pas générer du flou de profondeur de champ. Pour prendre en compte les effets de flou dus au basculement, nous avons utilisé le principe du lancer de rayon distribué. Malheureusement l'algorithme du lancer de rayon distribué présenté dans [CPC84] donne la distance du plan de mise au point sur l'axe en fonction du plan film en supposant que ces deux plans sont perpendiculaires à l'axe optique. Les formules ne sont donc pas adaptées au basculement des plans.

L'idée qui consiste à utiliser les rayons de l'antialiasing pour simuler le faisceau reste néanmoins valable, et il suffit alors de calculer pour chaque point A du plan film son image A'' par la lentille à partir de la formule 2.2 et

intersecter tous les rayons secondaires vers le point image A''. Comme utilisation d'un plan film n'est pas pratique en synthèse d'image, et comme il est plus simple d'utiliser un plan de projection, nous avons donc réalisé une modification simple de l'implémentation du lancé de rayon distribué permettant l'utilisation du plan de projection basculé. Nous faisons à nouveau l'hypothèse que le plan de projection est le symétrique du plan film ce qui revient à dire que basculer le plan film est équivalent à basculer le plan de projection. Il est alors important de noter que le plan de projection et le plan de netteté ne se correspondent pas du tout. Ce sont deux plans différents, comme l'illustre la figure 9.

Dans l'implémentation classique du lancer de rayon distribué [Gla89], le vecteur $\overrightarrow{OA'}$ est connu et le point A' est le point de netteté où vont converger tous les rayons secondaires. Dans notre implémentation modifiée, les rayons vont s'intersecter en A'' (cf Figure 10). On remarque alors que $\overrightarrow{OA''} = \alpha\overrightarrow{OA'}$ et $\overrightarrow{PH} = \alpha\overrightarrow{OA'} = \overrightarrow{OG} + \beta\overrightarrow{GF'}$. Ces deux relations nous permettent de trouver α :

$\alpha = \left|\frac{z_{f'}}{z_{a'} - z_{f'}}\right|$ si $z_{a'} \neq z_{f'}$ sinon les rayons secondaires envoyés suivent la direction $\overrightarrow{OA'}$.

On trouve alors chaque vecteur secondaire $\overrightarrow{PA''}$ avec la relation $\overrightarrow{PA''} = \overrightarrow{PH} - \overrightarrow{OP}$ avec $\overrightarrow{PH} = \alpha\overrightarrow{OA'}$.

Les rayons secondaires sont envoyés d'une surface de diamètre $D = \frac{f'}{n}$ (n étant l'indice de diaphragme) et leurs points de départ sur la surface sont choisis aléatoirement en suivant une distribution de Poisson. Le choix de la distribution de Poisson est discuté dans [Gla89].

La méthode du lancer de rayon distribué modifié a plusieurs avantages : l'anamorphose due à la bascule arrière est simulée, lorsque la mise au point est proche, on observe un agrandissement du champ de vision, comme avec une optique classique, et enfin la profondeur de champ est bien respectée. Ces propriétés sont principalement dues à l'utilisation de la notion de distance focale. Malheureusement le basculement des plans implique une divergence plus grande des rayons ce qui oblige à en envoyer beaucoup plus et de ce fait augmente le temps de calcul déjà important. (Environ trois minutes par images de 800x600 sur un athlon 1,2 Ghz)

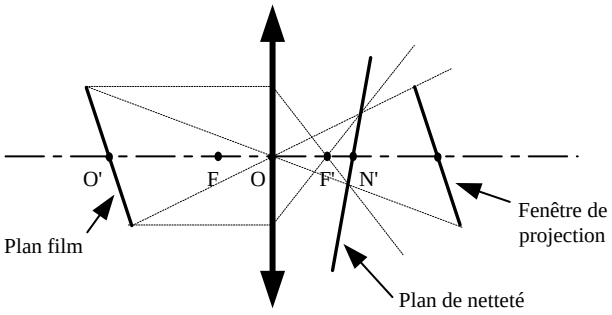


FIG. 9 – Différence entre plan de projection et plan de netteté

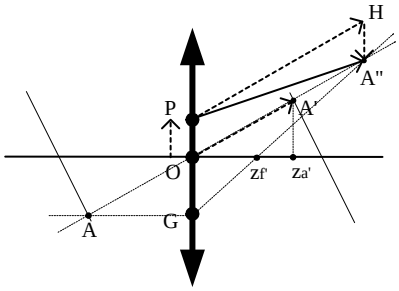


FIG. 10 – Construction géométrique

5.2 Résultats

Pour la scène de la figure 11, nous avons basculé le plan film de quelques degrés autour de l'axe des X. On observe alors un plan de netteté qui est complètement basculé et une profondeur de champ qui s'étale plus ou moins verticalement.

6 Conclusions et travail futur

Nous avons présenté deux méthodes pour simuler deux des caractéristiques visuelles de la chambre photographique qui sont l'anamorphose et le changement d'orientation du plan de netteté. L'anamorphose peut être réglée en temps réel en réalisant une modification simple de n'importe quel code source OpenGL, l'avantage étant de pouvoir utiliser seulement deux paramètres pour corriger la perspective. De cette manière, on peut corriger les perspectives relativement facilement. On pourrait aussi utiliser cette méthode pour corriger manuellement les perspectives des images projetées par les projecteurs vidéo. La deuxième méthode présentée s'appuie sur le modèle de la lentille

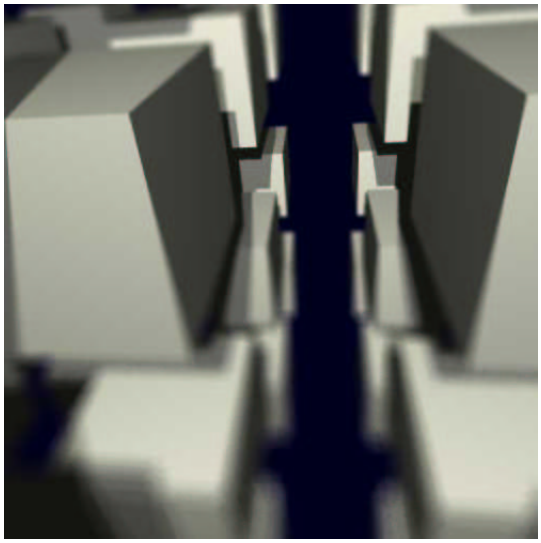


FIG. 11 – Flou de profondeur de champ avec plan de projection basculé

mince et le lancer de rayon distribué. Elle permet de simuler réellement le comportement d'une optique simple avec n'importe quelle orientation du plan film et a l'avantage de mettre en avant les différences existant entre le plan de projection et le plan de netteté qui sont généralement confondus dans beaucoup de livres portant sur la synthèse d'image. Cette méthode peut-être aussi utilisée pour simuler les objectifs cinéma à bascule et décentrement qu'utilisent souvent les réalisateurs de publicité et de clips musicaux. On pourrait alors incruster des images de synthèse dans les séquences comportant des effets de flous "alternatifs". Néanmoins, au vu des temps de calcul importants liés au lancer de rayon distribué, il serait intéressant d'adapter le principe de Heidrich et al [HSS97] à des plans de projections quelconques afin de simuler quasiment en temps réel un flou non perpendiculaire à l'axe optique.

Références

[AG93] P. Acquisto and E. Gröller. A distorsion camera for ray-tracing. In *Visualization and intelligent design in engineering and Architecture*, pages 105–118. Computational Mechanics Publications, Elsevier, 1993.

[CPC84] R. Cook, T. Porter, and L. Carpenter. Distributed ray tracing. In *Proceeding of SIGGRAPH 84*, Computer Graphics, pages 137–144, Boston, 1984. ACM Press.

[FD95] J. Foley and A. Van Dam. *Introduction à l'infographie*. Addison-Wesley France, Paris, 1995. Traduction de Introduction to computer graphics, Addison-Wesley, 1994.

[Gla89] A. Glassner. *An introduction to ray-tracing*. Academic Press, London, 1989.

[HA90] P. Haeberli and K. Akeley. The accumulation buffer : Hardware support for high-quality rendering. In *Proceeding of SIGGRAPH 90*, volume 24 of *Computer Graphics*, pages 309–318, Boston, August 1990. ACM Press.

[HSS97] W. Heidrich, P. Slusallek, and H. Seidel. An image-based model for realistic lens systems in interactive computer graphics. In *Graphic Interface 97*, pages 68–75. Wayne A. Davis and Marilyn Mantei and R. Victor Klassen, 1997.

[KMH95] C. Kolb, D. Mitchell, and P. Hanrahan. A realistic camera model for computer graphics. In *Proceedings of SIGGRAPH 95*, volume 29 of *Computer Graphics*, pages 317–324, Boston, 1995. ACM Press.

[NDW99] J. Neider, T. Davis, and M. Woo. *OpenGL Programming Guide : the Official Guide to Learning OpenGL, version 1.2*. Addisson-Wesley Publishing Company, New York, 1999.

[PC81] M. Potmesnil and I. Chakravarty. A lens aperture camera model for synthetic image generation. In *Proceedings of SIGGRAPH 81*, volume 15 of *Computer Graphics*, pages 297–305, Boston, 1981. ACM Press.

[Shi94] M. Shinia. Post-filtering for depth of field simulation with ray distribution buffer. In *Proceedings of Graphics Interface 94*, pages 39–66. Canadian Computer Communications Society, 1994.

[Str86] L. Stroebel. *View camera technique, 5th Edition*. Focal Press, Boston, 1986.

[Til92] U. Tillmanns. *Bases et applications Grand format créatif*. Sinar Edition, Feuerthalen, Allemagne, 1992. Traduction de l'allemand Kreatives Grossformat : Grundlagen und Anwendung, Sinar Edition, Feuerthalen.

[WW92] A. Watt and M. Watt. *Advanced Rendering and Animation Techniques : theory and pratice*. Addison-Wesley publishing company, ACM Press, 1992.

Respect des niveaux de visibilité dans la restitution d'images de synthèse en unités physiques

R. Brémond

Laboratoire Central des Ponts et Chaussées
58 boulevard Lefebvre 75015 Paris

Roland.bremond@lcpc.fr

Résumé : *La visualisation sur écran des images de synthèse calculées en unités physiques (luminance, coordonnées XYZ) pose un problème spécifique si on souhaite respecter les performances visuelles des observateurs, et notamment les niveaux de visibilité des objets présents dans la scène. Nous présentons une méthode d'évaluation de la qualité visuelle du système de visualisation, qui comprend à la fois le système d'affichage et l'algorithme de transposition de luminance qui lui est associé. Un exemple de mise en œuvre de cette méthodologie est présenté, dans le cas des performances visuelles des automobilistes dans la simulation de conduite de jour. Les principales questions non résolues qui se posent pour la visualisation des images calculées en unités physiques sont identifiées dans le cas des images de synthèse destinées à la simulation de conduite.*

Mots-clés : Visibilité, perception, *tone mapping*, visualisation, transposition, luminance, couleur.

1. Introduction

Le LCPC[1] développe des techniques de synthèse d'images permettant de réaliser des études de visibilité et de lisibilité routière, notamment sur simulateur de conduite. Au stade actuel de nos recherches, le principal obstacle identifié à l'utilisation de telles images porte sur les procédés de visualisation sur écran des images calculées. Nous présentons, dans ce papier, la perspective selon laquelle nous envisageons ce problème, qui se pose d'une manière plus large en synthèse d'image, et nous indiquons selon quelle approche nous essayons de le résoudre.

Nous cherchons à évaluer la capacité des procédures de visualisation à respecter les performances visuelles des observateurs. L'objectif est de quantifier l'écart entre les images présentées et les scènes modélisées, et donc d'optimiser la chaîne globale « mesure / modélisation / calcul / visualisation », ce qui doit permettre d'élargir le domaine de validité des images de synthèse en unités physiques pour les études de visibilité et de lisibilité routière.

2. Les images en unités physiques

[1] Le *Laboratoire Central des Ponts et Chaussées* est un établissement public à caractère scientifique et technologique sous la double tutelle du ministère de la recherche et du ministère de l'équipement.

Fig. 1 : image de luminances calculée à partir d'un modèle des effets visuels du brouillard [Dum02].

Certaines images numériques sont définies en unités physiques. Elles proviennent de calculs simulant la propagation de la lumière (exemple Fig. 1), ou de mesures photométriques de l'environnement lumineux par des systèmes d'acquisition adéquats (exemple Fig. 2). Elles permettent de décrire les informations visuelles dans des unités physiques (luminance[2], coordonnées XYZ[3] de la CIE[4]) qui représentent le signal visuel perçu par un observateur. La restitution de telles images sur un moniteur, sur l'écran d'un simulateur de conduite ou sur papier pose des problèmes spécifiques.

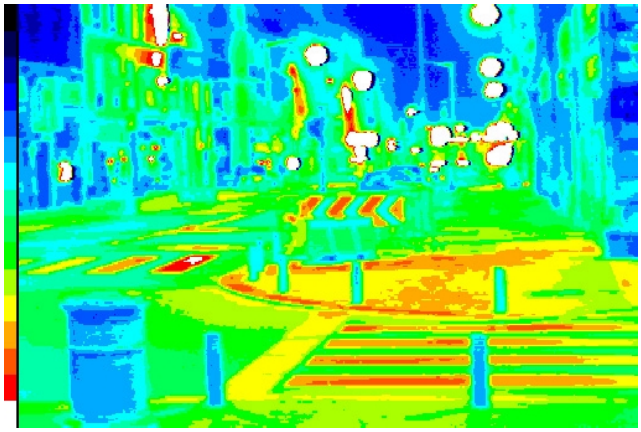


Fig. 2 : image de luminances (en fausses couleurs) obtenue par le système de mesure Mélusine [BH96]

3. Le système de restitution visuelle

3.1. Calibrage des écrans

Le premier problème à résoudre lors de l'affichage sur écran d'une image en unités physiques est de s'assurer que les valeurs physiques des couleurs et des luminances affichées correspondent aux valeurs prévues (Fig. 3). Pour cela, il est nécessaire de caractériser précisément le système de restitution. Pour un écran CRT, par exemple, l'affichage des images se base sur des valeurs d'adressage (des triplets d'entiers) correspondant aux photophores *Rouge*, *Vert* et *Bleu* de l'écran : c'est le triplet [RVB].

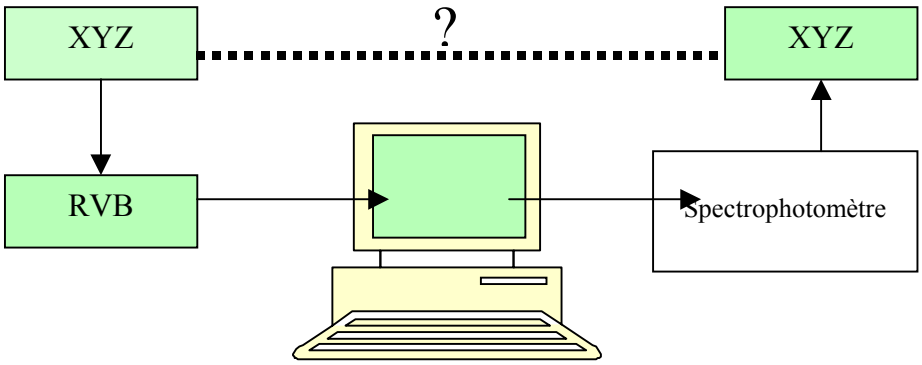


Fig. 3 : problématique du calibrage des écrans pour la restitution des images de synthèse en unités physiques.

A partir d'une image calculée décrite en unités tri-chromatiques [XYZ], il est donc nécessaire de connaître la correspondance entre un triplet quelconque [RVB] et le triplet [XYZ] qui sera observé sur cet l'écran. En pratique, cette étape repose sur une caractérisation photométrique et colorimétrique de l'écran utilisé. Pour un écran CRT, on utilise classiquement le modèle GOG (gain-offset-gamma) préconisé par la CIE [CIE122]. Il

2 La luminance représente l'intensité du signal visuel perçu par un observateur, en tenant compte des spécificités du système visuel humain.
3 Le triplet XYZ représente une quantité mesurable associée à un signal visuel. Il comporte trois composantes, correspondant aux trois variables nécessaires pour décrire la manière dont un observateur perçoit la lumière en conditions photopiques.
4 Commission Internationale de l'Eclairage.

consiste à modéliser la luminance affichée par un moniteur CRT comme la somme des luminances des trois canaux. Dans ce modèle, on estime que la luminance d'un canal peut être décrite par une fonction puissance de la valeur d'adressage (modulée par un gain) et par un bruit de fond (offset). Par exemple, pour le canal rouge, on écrira :

$$\boldsymbol{R} = \left\{ \boldsymbol{g}_r \left(\frac{\boldsymbol{I}}{\boldsymbol{I}_{\max}} \right) + (1 - \boldsymbol{g}_r) \right\}^{\boldsymbol{\gamma}}$$

$\boldsymbol{R}$ représentant la luminance du canal rouge, $\boldsymbol{I}$ la valeur d'adressage en rouge ($\boldsymbol{I}_{max}$ = 255), les paramètres $\boldsymbol{\gamma}$ et $\boldsymbol{g}_r$ étant déterminés par des mesures photométriques sur l'écran utilisé.

A partir d'une telle formulation, on peut inverser les équations et calculer la valeur d'adressage nécessaire pour afficher à l'écran une couleur XYZ donnée. Cette procédure permet de soumettre l'observateur au même stimulus visuel que ce qui était prévu dans l'image exprimée en unités physiques.

3.2. Limites techniques des écrans

Les systèmes de restitution présentent des limites techniques incontournables, qui font que l'affichage des valeurs en unités physiques (calculées ou mesurées) n'est pas toujours possible. Ces limites sont de plusieurs ordres:

- La stabilité des propriétés dans le temps (dérive) et dans l'espace (hétérogénéité), qui font que les modèles d'écran ne sont que des approximations, et qu'il est également nécessaire de caractériser les écarts par rapport au modèle.
- Le domaine de couleur (*gamut*) d'un écran est limité, certaines couleurs ne peuvent donc pas être affichées (Fig. 4).
- La quantification liée à la numérisation du signal conduit à des approximations. Il en résulte des écarts relatifs qui peuvent être significatifs, en particulier pour les faibles niveaux de luminance.
- La dynamique de luminance disponible est souvent beaucoup plus faible que celle qui est présente dans la scène à afficher. De plus, la luminance maximale disponible sur un écran donné dépend de la couleur affichée.

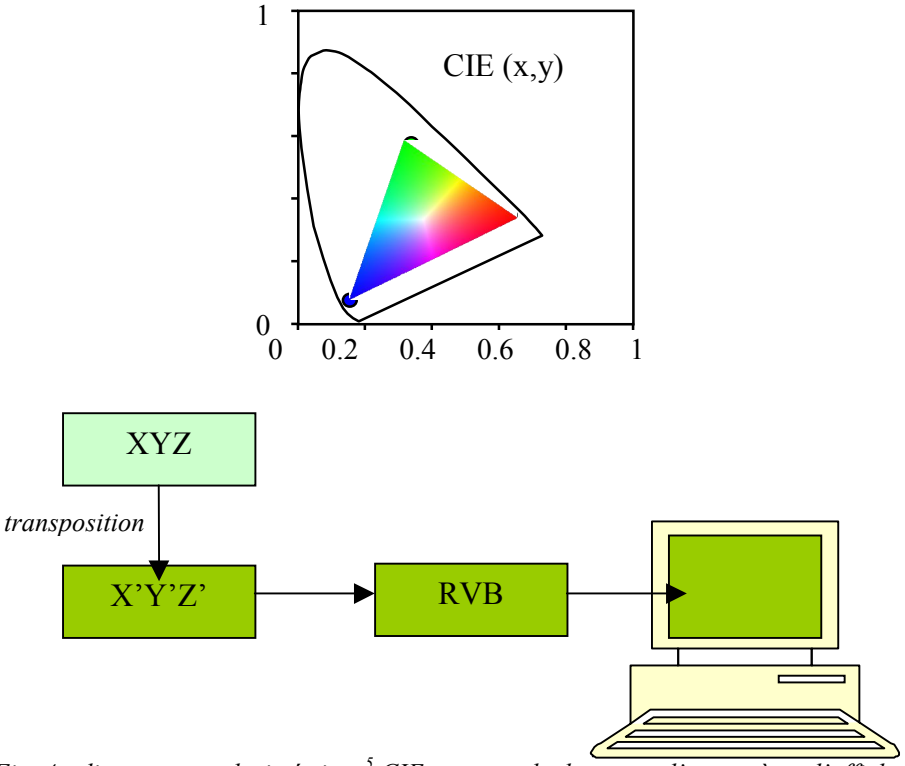


Fig. 4 : diagramme colorimétrique[5] CIE, et exemple de gamut d'un système d'affichage.

[5] Les coordonnées colorimétriques (x,y) sont des quantités mesurables caractérisant la couleur d'un objet indépendamment de sa luminance, définies par la CIE à partir de la distribution spectrale de la lumière.

Fig. 5 : principe de fonctionnement d'un algorithme de transposition

Ces limites sont contournées, en pratique, par des algorithmes de transposition permettant de convertir un signal visuel à afficher (de couleur XYZ) en un signal visuel différent (noté X'Y'Z' sur la Fig. 5) affichable par le système de restitution utilisé. Les algorithmes de transposition les plus étudiés sont les algorithmes de transposition de luminance (*tone mapping*), qui transposent les luminances à afficher dans un domaine de luminance accessible sur l'écran considéré.

Il résulte globalement de l'utilisation d'un algorithme de transposition une dégradation du signal, et des stratégies sont proposées dans la littérature pour minimiser cette dégradation [Tr93,War94,LRP97].

4. Evaluation de la qualité d'une transposition

4.1. Problématique

Le LCPC produit des images de synthèse en unités physiques, et contribue à la production de bases de données en unités physiques pour la simulation de conduite [BG02]. Ces images sont produites dans l'intention de procéder à des expérimentations avec des observateurs, afin d'étudier la visibilité et la lisibilité de l'environnement routier (Cf. par exemple [Pau01]).

Pour ces expérimentations, la dégradation des images calculées due aux limites des systèmes de restitution est susceptible d'avoir pour conséquence une modification des performances visuelles des observateurs, par comparaison avec les performances qu'ils auraient avec un système de restitution « idéal ». L'enjeu est donc pour nous de déterminer une stratégie de dégradation du signal qui minimise la modification du comportement des observateurs, notamment le niveau de visibilité des éléments visuels présentés aux observateurs [Pou99].

Dans ce but, un programme de travail a été défini, à partir des principaux enjeux identifiés, et une collaboration avec l'équipe « Vision » du MNHN[6] a permis de proposer une méthode d'évaluation spécifique des algorithmes de transposition de luminance. Cette méthode a un champ d'application plus large, mais elle doit être adaptée en fonction des critères de « qualité » retenus pour toute utilisation particulière des images affichées. Elle est présentée ci-dessous, ainsi que son application dans le cas des scènes routières diurnes. Le problème particulier que posent les scènes routières nocturnes est détaillé à la section 4.

4.2. Méthodologie d'évaluation

On distingue deux grands domaines d'évaluation de la perception visuelle : l'apparence visuelle et la performance visuelle, cette dernière étant attachée non seulement à une image mais à une tâche visuelle associée à cette image. La méthode que nous présentons n'est pas spécifique à l'un ou l'autre de ces domaines, mais présente deux variantes, adaptées à l'un ou à l'autre.

L'apparence visuelle désigne les jugements subjectifs des observateurs relatifs à un stimulus visuel. Elle est souvent liée à des catégories d'appréhension implicites, comme le brillant, l'uniformité, l'apparence colorée, etc. Dans ce domaine, il n'y a pas de référence objective, pas de « bonne » réponse, mais chaque observateur conserve une certaine consistance dans ses jugements. C'est sur cette cohérence interne que l'on s'appuie pour évaluer la proximité entre deux stimuli visuels.

La performance visuelle est une performance associée à une tâche liée à un stimulus visuel. Il s'agit en général de tâches de détection ou d'identification d'objets dans une scène, pour lesquelles les réponses des observateurs ont une valeur de vérité (vrai ou faux). A partir du taux de bonnes réponses et éventuellement du délai de réponse, il est possible de définir un indicateur objectif de la performance visuelle de l'observateur pour une tâche spécifique [CIE145].

L'élément fondamental de notre approche consiste à utiliser un cadre de référence extérieur au domaine des images de synthèse, comme cela peut être fait, par exemple, avec une « véritable » *Cornell box*. Une scène matérielle est construite, dans laquelle des observateurs vont être soumis à une série de tests, permettant

[6] *Muséum National d'Histoire Naturelle*

d'apprécier, soit leurs performances visuelles dans le domaine souhaité, soit les patterns caractéristiques de leurs jugement d'apparence visuelle.

Cette scène de référence est ensuite caractérisée photométriquement et numérisée (à partir de mesures photométriques, ou d'une modélisation géométrique et d'une simulation de la propagation de la lumière). C'est cette image numérique, considérée comme une représentation « fidèle » de la scène réelle, qui peut être visualisée sur écran. Pour cela, on doit utiliser un algorithme de transposition, qui doit être adapté au système physique de restitution utilisé (moniteur CRT ou LCD, projecteur sur écran CRT, LCD ou DSP, casque de réalité virtuelle, etc.)

L'indicateur visuel retenu sur la scène de référence (apparence ou performance visuelle) peut alors être évalué sur les images de synthèse affichées. C'est la plus ou moins grande correspondance entre la valeur de référence de l'indicateur et la valeur obtenue à partir des images de synthèse qui permettra de juger de la qualité de la restitution, *pour l'objectif poursuivi*, de l'ensemble « algorithme de transposition / système de restitution ».

4.3. Exemple d'application

Cette démarche suppose, avant toute chose, de spécifier l'usage des images de synthèse utilisées sur un support matériel donné. Nous nous sommes intéressés, avec cette méthode, au problème de la visibilité routière, et à la pertinence des images de synthèse (en présentation fixe ou sur simulateur de conduite) pour étudier ce problème crucial dans le domaine de la sécurité routière.

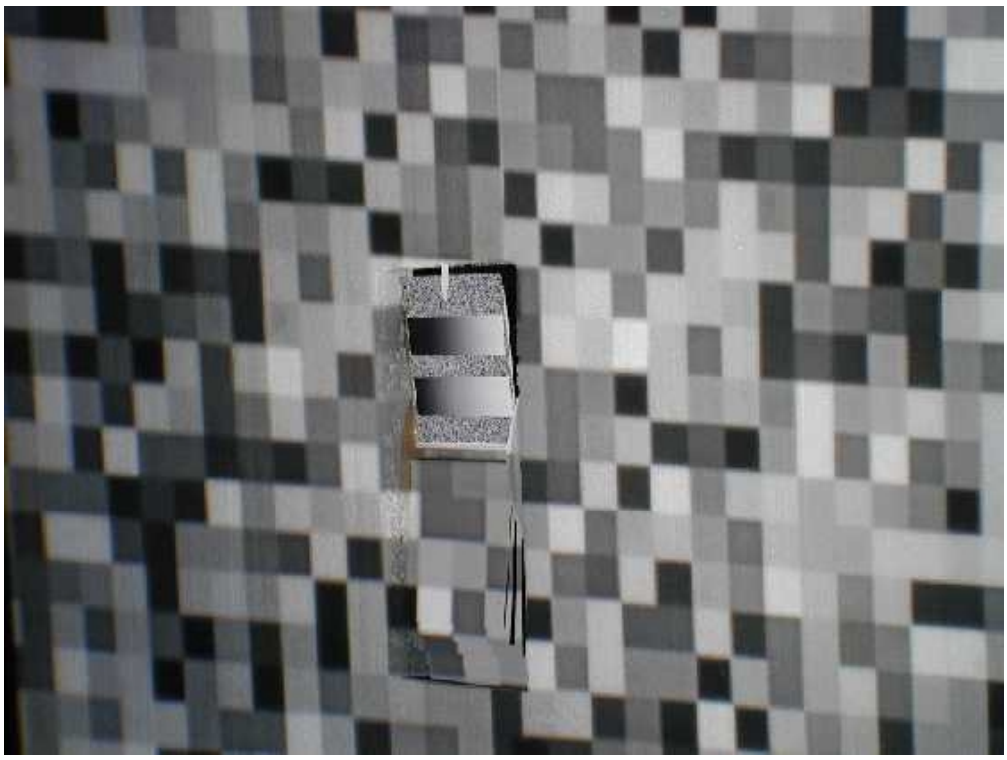


Fig. 6 : photographie de la scène de référence utilisée pour évaluer les indicateurs visuels pour des automobilistes en conduite de jour : panneau extérieur en polystyrène, tambour de présentation des images imprimées (ci-contre, pour une évaluation de l'apparence visuelle [Vie02a]).

A partir de cette problématique, les principaux éléments d'une expérimentation ont pu être fixés :

- L'enjeu porte principalement sur des performances visuelles liées à la tâche de conduite.
- Dans le domaine de la conduite de jour, les niveaux de luminance sont trop élevés pour être affichés sur un moniteur CRT classique, à plus forte raison sur l'écran d'un simulateur de conduite.

Une tâche visuelle a pu être définie (l'identification d'une ligne pointillée parmi 4 lignes parallèles), un moyen de restitution choisi (un moniteur CRT du MNHN) ainsi qu'un domaine de luminance, à partir des connaissances que nous avions sur les niveaux lumineux usuels en conduite automobile de jour.

Une expérimentation a eu lieu en deux temps : une scène de référence a été construite, et un système de présentation d'images imprimées a été mis au point dans ce cadre (Fig. 6). Des images correspondant à la tâche visuelle ont été imprimées avec différents niveaux de contraste entre les lignes et le fond. Des observateurs ont été confrontés à ces séries d'images imprimées, dans un environnement lumineux compris entre 0 et 1000 candélas par mètres carrés.

Fig. 7 :modélisation de la scène de référence utilisée pour évaluer les performances visuelles des automobilistes en conduite de jour.

Dans un deuxième temps, cette scène de référence a été caractérisée par des mesures photométriques, d'une part en ce qui concerne l'environnement (qui reste le même d'une présentation d'image à l'autre), d'autre part en ce qui concerne les tests présentés. A partir de ces mesures de luminance, des images de synthèse en luminance ont été produites. L'écran CRT utilisé a également été caractérisé par des mesures photométriques, et quatre algorithmes de transposition de luminance ont été sélectionnés dans la littérature [Vie02b]. Des séries d'images de synthèse RVB ont été calculées en vue de la visualisation sur cet écran (exemple Fig. 7). Nous avons ainsi pu comparer les performances visuelles des observateurs dans la situation de référence avec leurs performances devant les images résultant de 4 différents algorithmes de *tone mapping*. Cette démarche a ainsi permis d'évaluer la « qualité » relative de ces algorithmes, relativement à une tâche visuelle de détection en situation de conduite de jour.

5. Questions ouvertes

La méthodologie présentée a un domaine d'application relativement large dans le domaine de la synthèse d'image, pour tout ce qui touche au « réalisme ». Elle permet en particulier de lever l'ambiguïté sur ce mot, en quantifiant la notion de réalisme par rapport à un objectif bien défini en termes de perception visuelle.

Cependant, dans certains cas, la littérature en synthèse d'images ne propose pas de solution opérationnelle à des problèmes de « réalisme » visuel. On présente ci-dessous des exemples de problèmes non résolus, porteurs d'enjeux en matière de visibilité routière.

5.1. Conduite de nuit

L'exemple qui a été cité consistait à comparer entre eux des algorithmes de la littérature, et à quantifier à chaque fois l'écart par rapport à une performance visuelle de référence. Nous avons observé que dans le cas de la conduite de nuit, les algorithmes classiques de *tone mapping* ne sont pas adaptés, du fait d'une hypothèse forte qui pose un problème dans ce cas précis.

Un algorithme de *tone mapping*, comme son nom l'indique, est un algorithme de transposition de luminances. A partir d'une image en luminance dont la dynamique n'est pas affichable sur le système de restitution dont on dispose, il construit une image « compressée », selon une stratégie spécifique, dont la dynamique est compatible avec le système de restitution.

Dans un deuxième temps, l'image issue du *tone mapping* est transformée en image RVB grâce au modèle de calibrage du système (Cf. partie 2.2). Implicitement, cette procédure repose sur l'hypothèse que la dynamique de

luminance disponible sur le système de restitution est la même dans toute l'image. Malheureusement, ça n'est pas le cas : la dynamique disponible dépend de la couleur que l'on veut afficher, elle en dépend même fortement.

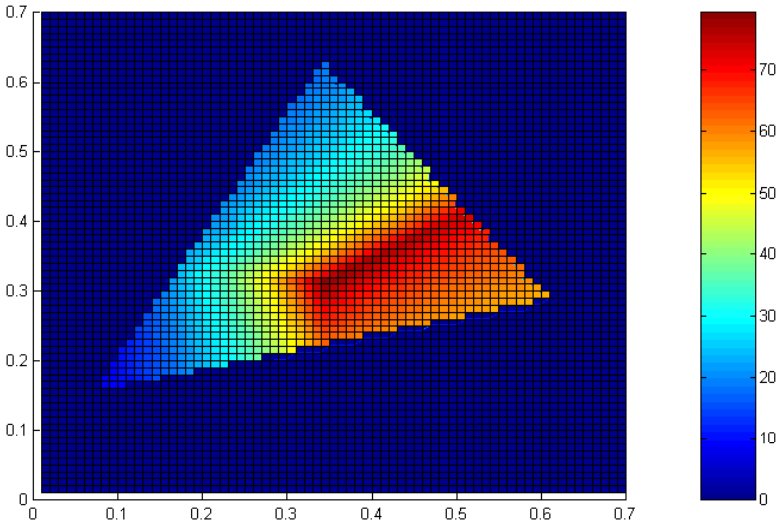


Fig. 8 : luminance maximale d'un écran CRT du MNHN en fonction de la couleur affichée, à l'intérieur du gamut de cet écran (en coordonnées colorimétriques xy).

En conduite de nuit, on se heurte à ce problème, car les feux arrière des véhicules doivent être rouges et très lumineux (par rapport au reste de la scène), alors que dans le rouge, la luminance maximale disponible est souvent significativement plus faible que dans le blanc.

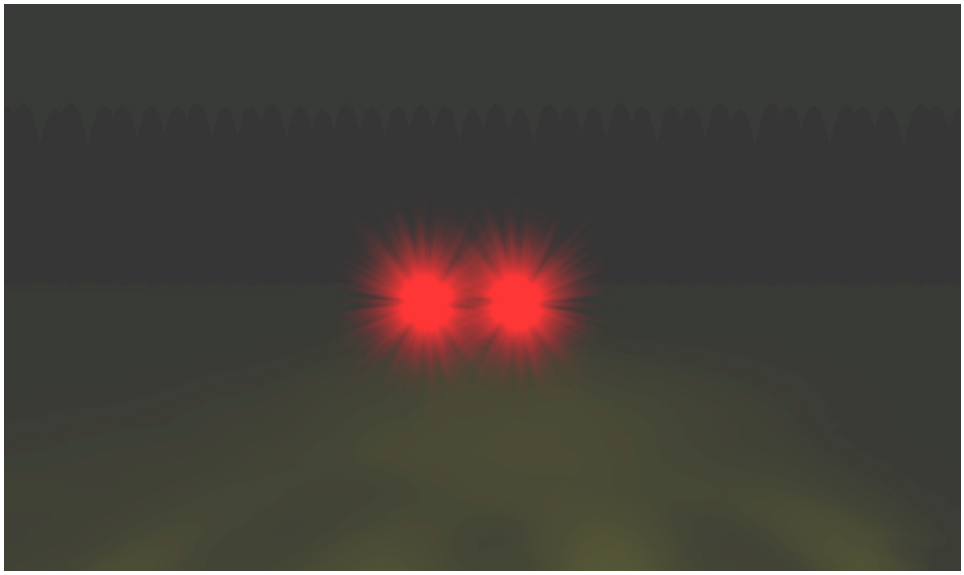


Fig. 9 :exemple d'image de synthèse comportant les feux arrières d'un véhicule [Voi02].

On se heurte donc à un problème très général, qui est de reconsidérer la notion de *tone mapping* en l'intégrant dans un processus complet portant sur une image physique colorée (décrite en unités XYZ) et non plus seulement sur une image de luminances. Les rares travaux abordant cette question [Voi02] doivent pouvoir être évalués à travers la méthodologie proposée ci-dessus.

5.2. Simulation de conduite

Si on considère le problème général de la tâche de conduite sur simulateur, on peut se fixer pour objectif de respecter le niveau de visibilité des objets présents dans la scène, entre une référence (le comportement sur la route) et un simulateur de conduite. La qualité des images doit donc être étudiée relativement à cette tâche de conduite, ce qui doit permettre de hiérarchiser les principales difficultés rencontrées dans cette étape de

visualisation des images de synthèse en unités physiques. On distingue ici trois cas de figure, pour lesquels la hiérarchie des enjeux est différente.

Conduite de jour :

- Le principal enjeu est le choix d'un algorithme de *tone mapping* respectant les niveaux de visibilité des objets, y compris en conditions de visibilité dégradée (chaussée humide, pluie, brouillard).

Conduite nocturne [BD02] :

- Le principal enjeu consiste à définir une procédure satisfaisante, du point de vue des performances visuelles, de « *color mapping* », qui transpose des couleurs XYZ et non pas seulement des luminances.
- Le rendu de l'éblouissement [Spe95], qui a un impact significatif sur le comportement de conduite la nuit, et qui ne peut pas être restitué directement sur écran, doit être évalué.
- La qualité de la restitution des bas niveaux lumineux est particulièrement importante.

Conduite en tunnel :

- Les problèmes d'adaptation visuelle lors des entrées et sorties de tunnel doivent être évalués, en fonction des solutions algorithmiques choisies pour le rendu de ces situations dans lesquels les niveaux lumineux changent brusquement.

Dans tous les cas de figure envisagés, les qualités et les défauts des différents modes de restitution (CRT, LCD et DSP) doivent être évalués en fonction des enjeux prioritaires.

Références

[BD02] R. Brémond, E. Dumont, « *simulation de conduite nocturne* », rapport à la Direction des Routes du ministère de l'Equipement (diffusion restreinte).

[BG02] R. Brémond, G. Gallée, « *Image quality for driving simulation experiments* », actes du congrès Virtual Reality International Conference 2002, Laval (France), juin 2002.

[BH97] C. Brusque, R. Hubert, « *La métrologie de la luminance par caméra CCD. Etalonnage et qualification du système Mélusine* », Bulletin des Laboratoires des Ponts et Chaussées, n° 205, pp. 39-47.

[CIE122] CIE, "*The relationship between digital and colorimetric data for controlled CRT displays*", CIE publication 122-1996.

[CIE145] CIE, "*The correlation of models for vision and visual performance*", CIE Publication 145-2002.

[Dum02] E. Dumont, « *caractérisation, modélisation et simulation des effets visuels du brouillard pour l'usager de la route* » thèse de doctorat, université Paris V, novembre 2002.

[Dum03] E. Dumont, R. Brémond, C. Boust, E. da Costa, F. Viénot, « assessment of the visual quality of images for visibility experiments : psychometric evaluation of tone mapping algorithms », poster retenu par le congrès de la Commission Internationale de l'Eclairage, San Diego, juin 2003.

[LeG72] Y. Le Grand, "*Optique physiologique – Tome 2 : Lumière et couleurs*", Masson et Cie, Paris, France, 1972 (2ème édition).

[LRP97] G. W. Larson, H. Rushmeier and C. Piatko, "*A Visibility Matching Tone Reproduction Operator for High Dymanic Range Scenes*", IEEE Transactions on Visualization and Computer Graphics, 3(4), October-December 1997, pp. 291–306.

[Pou99] G. Pouliquen, "*Respect des niveaux de visibilité dans la restitution d'images de synthèse*". Rapport de DEA, ESME/LCPC, septembre 1999.

[Pau01] G. Paulmier, C. Brusque, V. Carta, V. Nguyen, "*Laboratory study of the influence of the visual complexity of the environment on the detection of the various types of target*", Lighting Research and technology, 2001, Editor : RH Simons, Publisher : Arnold.

[TR93] J. Tumblin and H. Rushmeier, "*Tone reproduction for Realistic Images*", IEEE Computer Graphics & Applications, 13(6), November 1993, pp. 42–48.

[Vie02a] F. Viénot, C. Boust, R. Brémond, E. Dumont, "*rating gradations for tone-mapping algorithms*", IS&T, CGIV 2002, Poitiers, 2-5 April 2002, pp. 221-225.

[Vie02b] F. Viénot, C. Boust, E. da Costa, R. Brémond, E. Dumont, « *psychometric assessment of the look and feel of digital images* », actes du congrès Driving Simulator Conference 2002, Paris, septembre 2002.

[Voi02] Rapport final du projet Prédit VOIR, octobre 2002 (diffusion restreinte)

[War94] G. Ward, "*A Contrast-based Scale Factor for Luminance Display*", in Graphics Gems IV, ed. P. S. Heckbert, 1994, pp. 391–397.

Conversion de cyclides de Dupin en carreaux de Bézier Rationnels biquadriques

L. GARNIER, S. FOUFOU, M. NEVEU

LE2I, FRE CNRS 2309
UFR Sciences, Université de Bourgogne, BP 47870,
21078 Dijon Cedex, France
<lgarnier,sfoufou,mneveu>@u-bourgogne.fr

Résumé : *D* ans cet article, nous allons convertir les cyclides de Dupin en carreaux de Bézier Rationnels biquadriques. Les Cyclides de Dupin ont été inventées en 1822 par le mathématicien français Charles Dupin. Ce sont des surfaces algébriques de degré inférieur à 4 dont les lignes de courbures sont des cercles ayant une équation paramétrique et deux équations implicites. Elles permettent d'effectuer des jointures en n'utilisant que des concepts géométriques sans se soucier de problèmes de paramétrisation. M. Pratt a développé un algorithme de conversion de cyclides de Dupin en carreaux de Bézier Rationnels biquadriques à partir de concepts d'analyse. Celui-ci ne permet pas de convertir toute une cyclide. Nous allons améliorer cet algorithme afin de résoudre ce problème. Cependant, certaines valeurs des variables de la cyclide seront interdites ce qui peut gêner lors de jointure. Nous allons développer un nouvel algorithme de conversion basé sur des concepts géométriques (calculs barycentriques, symétrie de cercles) et les courbes de Bézier Rationnelles quadriques.

Mots clés : cyclide de Dupin, courbes et surfaces de Bézier Rationnelles de degré 2, cercles, calculs barycentriques.

1 Introduction

L'informatique a permis à l'industrie de réaliser des économies en utilisant des scènes 3D réalistes en remplacement de moules physiques. Il a fallu concevoir des modèles mathématiques permettant ces modélisations. Plusieurs modèles de courbes et de surfaces ont été développées : le modèle de P. Bézier [3, 13], le modèle B-Splines [12], les quadriques et superquadriques [31, 17]. Nous pouvons modéliser une scène à l'aide de primitives simples combinées à l'aide d'un arbre CSG [20, 28]. Il se pose alors des problèmes de jointures entre ces surfaces dus en particulier à la paramétrisation de celles-ci. Il est possible de remédier à ce problème de paramétrisation en introduisant une paramétrisation canonique [19].
Depuis une dizaine d'années, il est possible d'effectuer ces jointures $\mathcal{G}^1$-continues de façon géométrique en utilisant les cyclides de Dupin, inventée en 1822 [10, 15, 7, 6]. Ce sont des surfaces non sphériques ayant des lignes de courbures circulaires, pouvant être représentées à la fois par des équations paramétriques ou implicites. Beaucoup d'auteurs ont travaillé sur les problèmes de jointures de surfaces quadriques à l'aide de cyclides de Dupin [22, 5, 11, 32, 27, 26, 29, 1].

Dans la deuxième section, nous faisons un rappel sur les courbes et les surfaces de Bézier Rationnelles de degré deux, les cyclides et les cyclides de Dupin. Dans la troisième section, nous modélisons des cercles par des courbes de Bézier Rationnelles quadriques. Dans la quatrième section, nous montrons la conversion de cyclides de Dupin en carreaux de Bézier Rationnels biquadriques réalisée par Pratt [24] et en faisons une amélioration. Les lignes de courbures des cyclides de Dupin étant des cercles, il est possible sous certaines conditions de les "traiter" comme des surfaces de révolution et nous développons un algorithme de conversion des cyclides de Dupin en carreaux de Bézier Rationnels biquadriques, section suivante. Dans la dernière section, nous comparons les différents algorithmes.

2 Etat de l'art

L'espace affine euclidien à trois dimensions est muni d'un repère orthonormé $\left(O, \vec{\imath}, \vec{\jmath}, \vec{k}\right)$. Le produit scalaire usuel entre les vecteurs $\vec{u}$ et $\vec{v}$ est noté $\vec{u} \bullet \vec{v}$. Sauf mention contraire, les équations des cyclides seront données dans ce repère.

2.1 Courbes et surfaces d'approximation ou d'interpolation en synthèse d'images

Les surfaces utilisées en mathématiques sont définies sur des ouverts afin de ne pas se soucier des problèmes de différentiabilité aux bornes. Par contre, en synthèse d'images, nous souhaitons effectuer des jointures et donc nous sommes obligé d'utiliser des fermés. Les courbes sont généralement définies sur l'intervalle $[0;1]$ et les surfaces sur le pavé $[0;1] \times [0;1]$.

2.1.1 Courbes de Bézier Rationnelles quadriques

Les courbes de Bézier Rationnelles quadriques sont des courbes paramétriques définies à partir des polynômes de Bernstein de degré 2, Formule (2.1). La propriété de la symétrie des polynômes de Bernstein, Formule (2.2), est essentielle pour la construction d'arc de cercle puisque elle assure la symétrie des constructions géométriques.

$$B_0(t) = (1-t)^2 \quad B_1(t) = 2t(1-t) \quad B_2(t) = t^2 \tag{2.1}$$

$$\forall i | 0 \leq i \leq 2,\ \forall t \in [0;1],\ B_i(t) = B_{2-i}(1-t) \tag{2.2}$$

Une courbe de Bézier Rationnelle quadrique est définie par la Formule (2.3), [9], où la droite (P_0P_1) (resp. (P_2P_1)) est la tangente à la courbe au point P_0 (resp. P_2). Ce type de courbes permet de modéliser une conique à l'aide de trois points de contrôles $(P_i)_{0\leq i\leq 2}$ et de trois nombres $(w_i)_{0\leq i\leq 2}$ appelés poids.

$$\overrightarrow{OM(t)} = \frac{1}{w_0B_0(t) + w_1B_1(t) + w_0B_2(t)} \left(w_0B_0(t)\overrightarrow{OP_0} + w_1B_1(t)\overrightarrow{OP_1} + w_2B_2(t)\overrightarrow{OP_2} \right),\ t \in [0;1] \tag{2.3}$$

Il est possible de simplifier cette définition pour modéliser une conique en prenant $w_0 = w_2 = 1$, cette courbe de Bézier Rationnelle quadrique est dite standard, Formule (2.4). Dans ce cas, on pose $w_1 = w$ et ce nombre détermine la nature de la conique : la courbe est un arc d'ellipse si et seulement si $0 < w < 1$; la courbe est un arc de parabole si et seulement si $w = 1$; la courbe est un arc d'hyperbole si et seulement si $w > 1$.

$$\overrightarrow{OM(t)} = \frac{1}{B_0(t) + wB_1(t) + B_2(t)} \left(B_0(t)\overrightarrow{OP_0} + wB_1(t)\overrightarrow{OP_1} + B_2(t)\overrightarrow{OP_2} \right),\ t \in [0;1] \tag{2.4}$$

2.1.2 Surfaces de Bézier Rationnelles biquadriques

Les surfaces de Bézier Rationnelles biquadriques sont des surfaces définies comme produit tensoriel de courbes de Bézier Rationnelles quadriques, Formule (2.5), où $(P_{ij})_{0\leq i,j\leq 2}$ sont les points de contrôles, $(w_{ij})_{0\leq i,j\leq 2}$ les poids et $(u,v) \in [0;1]^2$, [24].

$$\overrightarrow{OM(u,v)} = \frac{1}{\sum_{i=0}^{2}\sum_{j=0}^{2} w_{ij}B_i(u)B_j(v)} \sum_{i=0}^{2}\sum_{j=0}^{2} w_{ij}B_i(u)B_j(v)\overrightarrow{OP_{ij}} \tag{2.5}$$

Une telle surface peut être vue comme l'ensemble des barycentres des points pondérées, Formule (2.6), [16, 18]. Le fait que la somme des coefficients fasse 1 assure l'indépendance de la définition par rapport au point O choisi. On peut simplifier cette définition pour obtenir une surface de Bézier Rationnelle biquadrique quasi-standard. Pour cela, on prend $w_{00} = w_{02} = w_{20} = w_{22} = 1$. Les courbes et les surfaces de Bézier Rationnelles sont invariantes par applications affines et projectives [9]. Les courbes tracées sur une surface de Bézier Rationnelle biquadrique obtenues avec une des variables u ou v constante sont des coniques.

$$\left(P_{ij}, \frac{w_{ij}B_i(u)B_j(v)}{\sum_{i=0}^{2}\sum_{j=0}^{2} w_{ij}B_i(u)B_j(v)} \right)_{0\leq i,j\leq 2} \tag{2.6}$$

2.2 Les cyclides

Dans son étude des surfaces anallagmatiques (on peut trouver une inversion de telle façon que cette surface soit invariante par cette transformation), M. Moutard a rencontré en 1804 les cyclides, [21]. Il a ainsi montré que les cyclides possèdent cette propriété par rapport à cinq pôles différents, ces pôles sont les centres des inversions. Les surfaces anallagmatiques de $\mathcal{E}$ sont les surfaces enveloppes d'une sphère variable, orthogonale à une sphère fixe S, appelé *sphère directrice*, et dont le centre décrit une surface quelconque Γ appelée *déferente* [7, 8]. Les cyclides de Dupin forment une sous-famille de cyclides [7, 8].

2.3 Les cyclides de Dupin

Il est possible de définir ce type particulier de cyclides de différentes façons : une cyclide de Dupin est l'image d'un cône de révolution par une inversion ; une cyclide de Dupin est l'image d'un tore par une inversion ; les cyclides de Dupin sont les enveloppes de sphères centrées sur une conique, appelée *déférente*, et orthogonale à une sphère fixe, appelé *sphère d'inversion*, centrée sur l'axe focal de la cyclide, [7, 8] ; les cyclides de Dupin sont les enveloppes des sphères tangentes à deux cercles-droites d'un plan, les centres des sphères décrivant l'une des coniques déférentes ; les cyclides de Dupin sont les enveloppes de sphères de rayons R et centrées en un point M sur une conique de foyer F telles que la distance $FM + R$ soit constante (définition de Maxwell) ; les cyclides de Dupin sont les surfaces enveloppes des sphères tangentes à trois sphères fixes ; les cyclides de Dupin sont les projections stéréographiques du tore de Clifford inclus dans la sphère S^3.

Une cyclide de Dupin de degré 4 est définie à l'aide de quatre paramètres $a,\ b,\ c$ et μ avec $a \geq c$ et $b = \sqrt{a^2 - c^2}$. Une cyclide de Dupin possède une équation paramétrique, Formule (2.7), et deux équations implicites équivalentes, Formule (2.8) et (2.9), [24, 14], résultat obtenu par [32] et aussi par [8] en explicitant l'enveloppe des sphères définissant une cyclide de Dupin. Selon les différentes valeurs des paramètres, il existe trois familles de cyclides de Dupin, ring cyclide ou cyclide en anneau, horned cyclide ou cyclide à croissant externe, spindle cyclide ou cyclide à croissant interne. Les propriétés des cyclides de Dupin ont été très étudiées [4, 5, 6, 7, 8, 10, 11, 23, 24, 25, 26, 2, 1, 30].

$$\Gamma(\theta,\psi) = \begin{cases} x(\theta,\psi) &= \dfrac{\mu(c - a\cos\theta\ \cos\psi) + b^2\cos\theta}{a - c\cos\theta\ \cos\psi} \\ y(\theta,\psi) &= \dfrac{b\sin\theta\ \times (a - \mu\cos\psi)}{a - c\cos\theta\ \cos\psi} \\ z(\theta,\psi) &= \dfrac{b\sin\psi\ \times (c\cos\theta - \mu)}{a - c\cos\theta\ \cos\psi} \end{cases} \tag{2.7}$$
$$\theta \in [0; 2\pi],\ \psi \in [0; 2\pi]$$

$$\left(x^2 + y^2 + z^2 - \mu^2 + b^2\right)^2 = 4\left(ax - c\mu\right)^2 + 4b^2y^2 \tag{2.8}$$

$$\left(x^2 + y^2 + z^2 - \mu^2 - b^2\right)^2 = 4\left(cx - a\mu\right)^2 - 4b^2z^2 \tag{2.9}$$

3 Cercles et courbes de Bézier Rationnelles quadriques

Nous allons commencer ce paragraphe par un rappel sur la détermination du centre d'un cercle connaissant trois points distincts. Soit A, B et C trois points d'un cercle C de centre O. Alors O est le point d'intersection des médiatrices Δ_1 et Δ_2 des segments $[AB]$ et $[AC]$. Nous souhaitons modéliser un arc de cercle par une courbe de Bézier Rationnelle quadrique. Tout diamètre d'un cercle étant axe de symétrie de ce cercle, le point P_1, Figure 1, doit donc appartenir au plan médiateur $\mathcal{P}$ du segment $[P_0P_2]$. Le théorème 1 donne les caractéristiques du cercle passant par P_0 et P_2 et ayant comme tangente (P_0P_1) et (P_2P_1). Réciproquement, le théorème 2 donne la construction du point P_1 pour que la courbe de Bézier Rationnelle quadrique soit l'arc de cercle de centre donné et passant par P_0 et P_2. Le théorème montre la modélisation d'un arc de cercle par une courbe de Bézier Rationnelle quadrique.

Théorème 1 *: Cercle déterminé par deux points et les tangentes en ces points*
Soit C le cercle de centre O_0 et de rayon R passant par P_0 et P_2 et ayant comme tangente (P_0P_1) et (P_2P_1), les

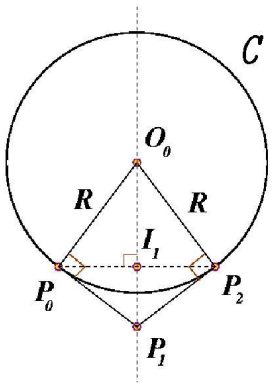


FIG. 1 – Modélisation d'un arc de cercle par une courbe de Bézier Rationnelle quadrique standard.

points P_0, P_1 et P_2 n'étant pas alignés.
Soit I_1 le milieu du segment $[P_0P_2]$. Soit $\mathcal{P}$ le plan médiateur du segment $[P_0P_2]$. Alors

1. *Le cercle C existe si et seulement si le point P_1 appartient au plan $\mathcal{P}$.*
2. *On suppose que le cercle C existe.*
   – *Le centre O_0 est donné par la Formule (3.1) et le rayon est $R = O_0P_0$.*

$$\overrightarrow{P_1O_0} = t_0\overrightarrow{P_1I_1}, \qquad t_0 = \frac{\overrightarrow{P_0P_1}^2}{\overrightarrow{I_1P_1} \bullet \overrightarrow{P_0P_1}} \tag{3.1}$$

   – *Dans le plan déterminé par le cercle C, la mesure de l'angle géométrique $\widehat{P_0O_0P_2}$ est inférieure à π ce qui veut dire que si l'on prend une paramétrisation γ usuelle du cercle (en cosinus et sinus) tel que $P_0 = \gamma(\theta_0)$, $P_2 = \gamma(\theta_1)$, on a $|\theta_0 - \theta_1| < \pi$.*

Théorème 2 *: Construction du point de contrôle P_1 connaissant le centre du cercle*
Soit C le cercle de centre O_0 et de rayon R passant par P_0 et P_2. Soit I_1 le milieu du segment $[P_0P_2]$.
Pour que la courbe de Bézier Rationnelle quadrique standard soit l'arc de cercle de C passant par les deux points distincts P_0 et P_2, le point P_1 est défini par la Formule (3.2).

$$\overrightarrow{I_1P_1} = t_1\overrightarrow{O_0I_1} \qquad t_1 = \frac{\overrightarrow{O_0P_0} \bullet \overrightarrow{I_1P_0}}{\overrightarrow{O_0P_0} \bullet \overrightarrow{O_0I_1}} \tag{3.2}$$

Il reste à déterminer le poids w de la courbe de Bézier Rationnelle quadrique standard pour que celle-ci corresponde à l'arc de cercle C désiré délimité par P_0 et P_2, Figure 1.

Théorème 3 *: Modélisation d'un arc de cercle par une courbe de Bézier Rationnelle quadrique quasi-standard*
Soit P_0, P_1 et P_2 trois points non alignés de l'espace euclidien $\mathcal{E}$ et I_1 le milieu du segment $[P_0P_2]$.
La courbe de Bézier Rationnelle quadrique quasi-standard γ de poids w est un arc de cercle C de centre O_0 et de rayon R définie par les points P_0, P_1 et P_2, théorème 1, si et seulement si w vérifie la Formule (3.3).

$$|1 + w| = |O_0I_1 + w0_0P_1| \tag{3.3}$$

Le morceau de l'arc de courbe déterminé par le poids w est donné par le tableau 1.

petit arc de cercle	grand arc de cercle
$w = \dfrac{O_0I_1 - R}{R - O_0P_1} = \dfrac{O_0I_1 - O_0P_0}{O_0P_0 - O_0P_1} > 0$	$w = -\dfrac{O_0I_1 + R}{R + O_0P_1} = -\dfrac{O_0I_1 + O_0P_0}{O_0P_0 + O_0P_1} < 0$

TAB. 1 – Arc de cercle en fonction du poids.

4 Conversion de cyclides de Dupin en carreaux de Bézier Rationnels biquadriques

Les courbes tracées sur une surface de Bézier Rationnelle biquadrique obtenues avec une des variables u ou v constante sont des coniques. Les lignes de courbures des cyclides de Dupin sont des cercles et donc des coniques particulières. Il est donc possible de convertir des cyclides de Dupin en carreaux de Bézier Rationnels biquadriques. L'algorithme de conversion proposé par M. Pratt dans [24] permet de déterminer facilement le carreau de Bézier Rationnel biquadrique représentant une cyclide de Dupin. Les coordonnées des points de contrôles et les poids sont calculés à partir de l'équation paramétrique d'une cyclide de Dupin, Equation (2.7), en fonction des paramètres de la cyclide de Dupin et des bornes délimitant la partie à convertir. Dans cet algorithme, les formules de calcul des points de contrôles du carreau de Bézier Rationnel biquadrique sont données en exploitant les relations trigonométriques liant les fonctions sinus et cosinus à la fonction tangente, Formule (4.1). La Figure 2 montre le résultat de la conversion d'un morceau de cyclide de Dupin en un carreau de Bézier Rationnel biquadrique.

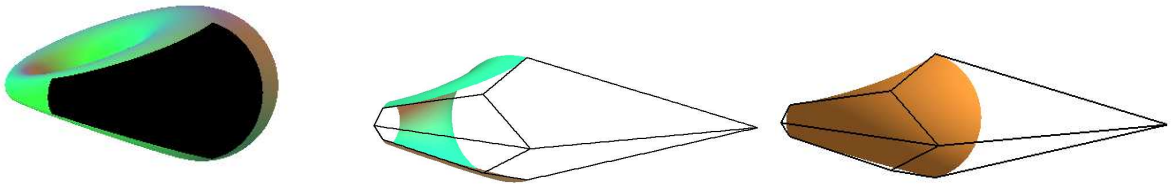
$$\forall \theta \in \mathbb{R}\pi\mathbb{Z}, \quad \cos(\theta) = \frac{1-\tan^2\left(\frac{\theta}{2}\right)}{1+\tan^2\left(\frac{\theta}{2}\right)}, \quad \sin(\theta) = \frac{2\tan\left(\frac{\theta}{2}\right)}{1+\tan^2\left(\frac{\theta}{2}\right)} \tag{4.1}$$

Morceau de cyclide de Dupin à convertir

Carreau de Bézier, méthode de Pratt

FIG. 2 – Conversion d'un morceau de cyclide de Dupin en un carreau de Bézier Rationnel biquadrique en utilisant la méthode de Pratt.

$(\theta_0, \theta_1, \psi_0, \psi_1) \in (\mathbb{R} - (\pi + 2\pi\mathbb{Z}))^4$ car la fonction tan n'est pas définie sur l'ensemble $\frac{\pi}{2} + \pi\mathbb{Z}$, donc pour certaines valeurs de θ ou ψ, la conversion de la cyclide de Dupin n'est pas possible. Certaines lignes de courbure ne peuvent donc pas être choisies comme bord du carreau de Bézier Rationnel biquadrique à convertir. La Figure 3 illustre un cas de disfonctionnement de cet algorithme : la discontinuité de la fonction tan en $\frac{\pi}{2}$ modulo π fait en sorte que cet algorithme ne fonctionne pas correctement lorsque $\pi \in]\theta_0; \theta_1[$ ou lorsque $\pi \in]\psi_0; \psi_1[$. La figure de gauche montre la portion de la cyclide à convertir. La figure du centre montre que le carreau de Bézier Rationnel biquadrique résultant ne correspond pas au morceau choisi. Nous améliorons l'algorithme précédent, pour obtenir la figure de droite, en changeant simplement le calcul des poids en prenant la valeur absolue de la formule originale.



Portion de la cyclide à convertir

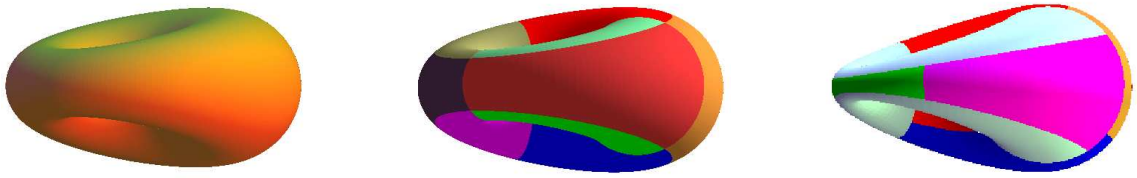
conversion, algorithme de Pratt

Conversion, amélioration de l'algorithme de Pratt

FIG. 3 – Insuffisance de l'algorithme de Pratt pour convertir un morceau de cyclide de Dupin en un carreau de Bézier Rationnel biquadrique.

Les valeurs trigonométriques θ_0, θ_1, ψ_0 et ψ_1 sont utilisées pour déterminer les points de contrôles et les poids. Si on a $\theta_0 = 0$ et $\theta_1 = 4\frac{\pi}{3}$, les points de contrôles seront "entre" $-\frac{2\pi}{3}$ et 0. Nous devons donc avoir $|\theta_0 - \theta_1| < \pi$

et $|\psi_0 - \psi_1| < \pi$. En tenant compte de cette contrainte, nous obtenons plusieurs conversions possibles : la Figure 4 illustre une cyclide de Dupin convertie en 9 carreaux de Bézier Rationnels biquadriques en utilisant l'algorithme amélioré. La combinaison des deux algorithmes permet de réduire à 6 le nombre minimal de carreaux nécéssaires pour assurer la conversion de toute la cyclide de Dupin, image de droite.



La cyclide | 9 carreaux de Bézier | 6 carreaux de Bézier

FIG. 4 – Conversion d'une cyclide de Dupin en carreaux de Bézier Rationnels biquadriques.

Cette amélioration permet de convertir toute une cyclide de Dupin en carreaux de Bézier Rationnelles biquadriques. Il ne permet pas d'avoir comme "bord" une courbe obtenue avec un des paramètres ayant une valeur de π. De plus, il ne fonctionne pas tout le temps, Figure 5. Le problème vient du poids w_{12} qui devrait être négatif. Ainsi, dans l'algorithme de Pratt amélioré, la courbe de Bézier Rationnelle quadrique modélise le grand arc du cercle supérieur délimitant la partie de la cyclide à convertir et la courbe de Bézier Rationnelle quadrique modélise le petit arc du cercle inférieur délimitant la partie de la cyclide à convertir. Nous allons développer un autre algorithme basé sur le calcul barycentrique et les propriétés circulaires des cyclides de Dupin.

Morceau de cyclide à convertir | Conversion en carreaux de Bézier

FIG. 5 – Problème de l'algorithme de Pratt amélioré pour effectuer certaine conversion d'un morceau de cyclide de Dupin en un carreau de Bézier Rationnel biquadrique.

5 Nouvel algorithme de conversion

5.1 Propriétés barycentriques des surfaces de Bézier Rationnelles biquadriques

On considère la surface de Bézier Rationnelle biquadrique standard S_0, de points de contrôles $(P_{ij})_{0\leq i,j\leq 2}$ et de poids $(w_{ij})_{0\leq i,j\leq 2}$ avec $w_{00} = w_{02} = w_{20} = w_{22} = 1$. Pour que la surface S_0 modèlise une surface à courbure sphérique, il faut que P_{01} appartienne au plan médiateur de $[P_{00}P_{02}]$, P_{10} appartienne au plan médiateur de $[P_{00}P_{20}]$, P_{21} appartienne au plan médiateur de $[P_{20}P_{22}]$ et P_{12} appartienne au plan médiateur de $[P_{02}P_{22}]$. Nous allons donner quelques propriétés barycentriques des surfaces de Bézier Rationnelles biquadriques quasi-standard.

Théorème 4 *: Propriétés barycentriques des surfaces de Bézier Rationnelles biquadriques quasi-standard*

Soit S une surface de Bézier Rationnelle biquadrique quasi-standard de points de contrôles $(P_{ij})_{0\leq i,j\leq 2}$ et de poids $(w_{ij})_{0\leq i,j\leq 2}$ avec $w_{00} = w_{02} = w_{20} = w_{22} = 1$. Un point $M(u,v)$, $(u,v) \in [0;1]^2$ appartient à la surface S si et seulement si $M(u,v)$ vérifie la Formule (2.5).

– *Soit I_0 le milieu du segment $[P_{00}P_{02}]$, J_0 le milieu du segment $[P_{00}P_{20}]$, I_2 le milieu du segment $[P_{20}P_{22}]$ et J_2 le milieu du segment $[P_{02}P_{22}]$. On a alors les relations de la Formule (5.1).*

$$\begin{array}{ll} \overrightarrow{OM\left(0,\frac{1}{2}\right)} = \frac{1}{1+w_{01}}\left(\overrightarrow{OI_0} + w_{01}\overrightarrow{OP_{01}}\right) & \overrightarrow{OM\left(1,\frac{1}{2}\right)} = \frac{1}{1+w_{21}}\left(\overrightarrow{OI_2} + w_{21}\overrightarrow{OP_{21}}\right) \\ \overrightarrow{OM\left(\frac{1}{2},0\right)} = \frac{1}{1+w_{10}}\left(\overrightarrow{OJ_0} + w_{10}\overrightarrow{OP_{10}}\right) & \overrightarrow{OM\left(\frac{1}{2},1\right)} = \frac{1}{1+w_{12}}\left(\overrightarrow{OJ_2} + w_{12}\overrightarrow{OP_{12}}\right) \end{array} \tag{5.1}$$

– *Soient G_0 l'isobarycentre des points P_{00}, P_{02}, P_{20}, P_{22}, G_2 le barycentre des points pondérés (P_{10}, w_{10}), (P_{01}, w_{01}), (P_{12}, w_{12}), (P_{21}, w_{21}) et $w = w_{01} + w_{10} + w_{12} + w_{21}$. Soit G_1 le barycentre des points pondérés $(G_0, 2)$, (G_2, w).*
– *Le point $M\left(\frac{1}{2}, \frac{1}{2}\right)$ vérifie les deux Formules (5.2) et (5.3). De la Formule (5.3), on déduit que le point P_{11} appartient à la droite $\left(M\left(\frac{1}{2}, \frac{1}{2}\right) G_1\right)$.*

$$\overrightarrow{OM\left(\frac{1}{2},\frac{1}{2}\right)} = \frac{1}{2+w+2w_{11}}\left((2+w)\,\overrightarrow{OG_1} + 2w_{11}\overrightarrow{OP_{11}}\right) \tag{5.2}$$

$$w_{11}\overrightarrow{M\left(\frac{1}{2},\frac{1}{2}\right)P_{11}} = \frac{2+w}{2}\overrightarrow{M\left(\frac{1}{2},\frac{1}{2}\right)G_1} \tag{5.3}$$

– *Le point $M\left(\frac{1}{2}, \frac{1}{4}\right)$ vérifie les deux Formules (5.4) et (5.5) où G_3 est le barycentre des points pondérés $(P_{00}, 9)$, $(P_{20}, 9)$, $(P_{02}, 1)$, $(P_{22}, 1)$, $(P_{01}, 6w_{01})$, $(P_{21}, 6w_{21})$, $(P_{10}, 18w_{10})$, $(P_{12}, 2w_{12})$, et $W_1 = 20 + 6w_{01} + 18w_{10} + 2w_{12} + 6w_{21}$. De la Formule (5.5), on déduit que le point P_{11} appartient à la droite $\left(G_3 M\left(\frac{1}{2}, \frac{1}{4}\right)\right)$*

$$\overrightarrow{OM\left(\frac{1}{2},\frac{1}{4}\right)} = \frac{1}{W_1+12w_{11}}\left(W_1\overrightarrow{OG_3} + 12w_{11}\overrightarrow{OP_{11}}\right) \tag{5.4}$$

$$(W_1+12w_{11})\,\overrightarrow{G_3M\left(\frac{1}{2},\frac{1}{4}\right)} = 12w_{11}\overrightarrow{G_3P_{11}} \tag{5.5}$$

– *Le point $M\left(\frac{1}{4}, \frac{1}{2}\right)$ vérifie les deux Formules (5.6) et (5.7) où G_4 est le barycentre des points pondérés $(P_{00}, 9)$, $(P_{20}, 1)$, $(P_{02}, 9)$, $(P_{22}, 1)$, $(P_{10}, 6w_{10})$, $(P_{12}, 6w_{12})$, $(P_{01}, 18w_{01})$, $(P_{21}, 2w_{21})$, et $W_2 = 20 + 6w_{10} + 18w_{01} + 2w_{21} + 6w_{12}$. De la Formule (5.7), on déduit que le point P_{11} appartient à la droite $\left(G_4 M\left(\frac{1}{4}, \frac{1}{2}\right)\right)$.*

$$\overrightarrow{OM\left(\frac{1}{4},\frac{1}{2}\right)} = \frac{1}{W_2+12w_{11}}\left(W_2\overrightarrow{OG_4} + 12w_{11}\overrightarrow{OP_{11}}\right) \tag{5.6}$$

$$(W_2+12w_{11})\,\overrightarrow{G_4M\left(\frac{1}{4},\frac{1}{2}\right)} = 12w_{11}\overrightarrow{G_4P_{11}} \tag{5.7}$$

5.2 Algorithme de conversion proposé

L'idée et de développer une méthode gardant les symétries circulaires le long des courbes coordonnées de la cyclide de Dupin en utilisant l'algorithme 1. Une cyclide n'est pas une surface de révolution, nous devrons donc nous assurer en plus que les droites $(G_1\Gamma(\theta_2, \varphi_2))$, $(G_3\Gamma(\theta_2, \varphi_3))$ et $(G_4\Gamma(\theta_3, \varphi_2))$ ont bien le même point d'intersection. Ceci est vrai à condition que θ_0 et θ_1 (ou ψ_0 et ψ_1) soient symétriques par rapport à 0 ou π sur le cercle trigonométrique. L'algorithme 1 permet de convertir la cylide de Dupin en carreaux de Bézier Rationnels biquadriques quasi-standards, en utilisant les formules des théorèmes 3 et 4.

5.3 Résultats

La Figure 6 illustre deux conversions d'une partie d'une cyclide de Dupin en carreaux de Bézier Rationnels biquadrique quasi-standard. Dans le second exemple, nous avons $\theta_1 = \pi$. Pour l'image du dessus à droite, nous avons utilisé un carreau de Bézier Rationnel biquadrique standard avec les poids positifs. Concernant l'image du dessous à droite, nous avons utilisé un carreau de Bézier Rationnel biquadrique quasi-standard, les poids sont positifs le long des lignes de courbures obtenues avec θ constant, négatifs le long des lignes de courbures obtenues avec ψ constant.

Algorithm 1 Modélisation d'un carreau de cyclide de Dupin par un carreau de Bézier Rationnel biquadrique quasi-standard

1. Nous choisissons un repère orthonormé de l'espace euclidien $\mathcal{E}$ de tel façon que l'équation de la cyclide de Dupin S soit définie par la nappe paramétrée Γ de la Formule (2.7), les lignes de courbure soient choisies pour $\theta = \theta_0$, $\theta = \theta_1$, $\psi = \psi_0$ et $\psi = \psi_1$ telles que θ_0 et θ_1 (ou ψ_0 et ψ_1) soient symétriques par rapport à 0 ou π sur le cercle trigonométrique.
2. les sommets des carreaux sont $P_{00} = \Gamma(\theta_0, \varphi_0)$, $P_{02} = \Gamma(\theta_1, \varphi_0)$, $P_{20} = \Gamma(\theta_0, \varphi_1)$, $P_{22} = \Gamma(\theta_1, \varphi_1)$.
3. Déterminer le centre des quatre cercles de courbure en utilisant trois points sur chaque ligne de courbure (dont les deux sommets).
4. Déterminer les points de contrôle P_{10}, P_{01}, P_{21} et P_{12} dans les plans médiateurs correspondants et dans les plans des lignes de courbures correspondants à l'aide des tangentes aux cercles correspondants en utilisant le théorème 2.
5. Calculer les poids w_{10}, w_{01}, w_{21} et w_{12} en utilisant le tableau 1 du théorème 3.
6. Déterminer la première courbe coordonnée γ_u qui est une courbe de Bézier Rationnelle quadrique standard de points de contrôles P_{00}, P_{10}, P_{20} et de poids w_{10}.
   Déterminer la seconde courbe coordonnée γ_v qui est une courbe de Bézier Rationnelle quadrique standard de points de contrôles P_{00}, P_{01}, P_{02} et de poids w_{01}.
7. Trouver θ_2 solution de l'équation $\gamma_u\left(\frac{1}{2}\right) = \Gamma(\theta_2, \varphi_0)$. Trouver φ_2 solution de l'équation $\gamma_v\left(\frac{1}{2}\right) = \Gamma(\theta_0, \varphi_2)$. Trouver φ_3 solution de l'équation $\gamma_v\left(\frac{1}{4}\right) = \Gamma(\theta_0, \varphi_3)$.
8. Soit G_0 l'isobarycentre des points P_{00}, P_{02}, P_{20}, P_{22}. Soit G_2 le barycentre des points pondérés (P_{10}, w_{10}), (P_{01}, w_{01}), (P_{12}, w_{12}), (P_{21}, w_{21}). Soit G_1 le barycentre des points pondérés $(G_0, 2)$, (G_2, w).
   Soit G_3 le barycentre des points pondérés $(P_{00}, 9)$, $(P_{20}, 9)$, $(P_{02}, 1)$, $(P_{22}, 1)$, $(P_{01}, 6w_{01})$, $(P_{21}, 6w_{21})$, $(P_{10}, 18w_{10})$, $(P_{12}, 2w_{12})$. Soit $M = \Gamma(\theta_2, \varphi_2)$.
   Déterminer P_{11} intersection des droites $(G_1\Gamma(\theta_2, \varphi_2))$ et $(G_3\Gamma(\theta_2, \varphi_3))$.
9. Déterminer le poids w_{11} défini par la Formule (5.3).

6 Comparaison des algorithmes de conversions

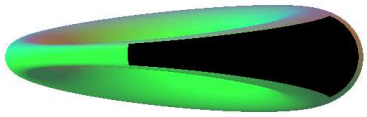
Pour convertir une cyclide de Dupin en utilisant notre algorithme, nous n'avons besoin que de quatre carreaux de Bézier Rationnels biquadriques alors qu'il en faut neuf en utilisant l'algorithme amélioré de Pratt et six en combinant l'algorithme de Pratt et sa version améliorée, Figure 7. Notre algorithme permet de plus de convertir une cyclide de Dupin le long d'une ligne de courbure obtenue avec un des paramètres valant π, ce qui n'est réalisable ni avec l'algorithme de Pratt, ni avec sa version améliorée. Cette valeur π est indispensable lors de jointure cylindre-plan modélisant un récipient.

7 Conclusion

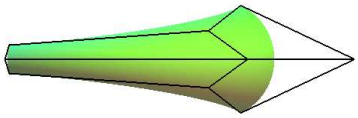
Dans cet article, nous avons développé un algorithme permettant de convertir des cyclides de Dupin en carreaux de Bézier Rationnels biquadriques quasi-standards. Notre point de départ a été le travail de M. Pratt. Comme celui-ci ne permettait pas de convertir toute une cyclide de Dupin, nous l'avons amélioré afin de résoudre ce problème. Il existait encore des valeurs interdites pour la conversion. Nous avons développé un nouvel algorithme basé sur les propriétés barycentriques des carreaux de Bézier Rationnels biquadriques quasi-standards et les propriétés circulaires des cyclides de Dupin. La combinaison de l'algorithme de Pratt est de sa version amélioré nécessite six carreaux pour représenter une cyclide de Dupin tandis que notre nouvel algorithme n'en demande que quatre. De plus, notre algorithme permet de convertir des morceaux de cyclides de Dupin impossible jusqu'à ce jour.

Références

[1] M. Berger. *Géométrie 2*, volume 5. Cedic-Nathan, 2ème edition, 1977-1978.

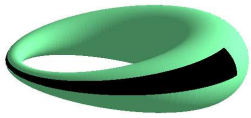


Cyclide de Dupin

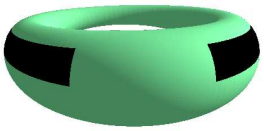


Carreau de Bézier

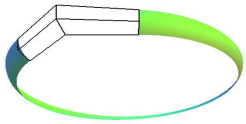
Poids positifs le long des lignes de courbures avec ψ constant, colonne de gauche, tableau 1.



Cyclide de Dupin, vue 1



Cyclide de Dupin, vue 2



Carreau de Bézier

Poids négatifs le long des lignes de courbures avec ψ constant, colonne de droite, tableau 1, et $\theta_1 = \pi$.

FIG. 6 – Conversion d'un morceau de cyclide de Dupin en un carreau de Bézier Rationnel biquadrique quasi-standard en utilisant l'algorithme 1.

[2] M. Berger and B. Gostiaux. *Géométrie différentielle : variétés, courbes et surfaces*. PUF, 2ème edition, avril 1992.

[3] P. Bézier. *Courbe et surface*, volume 4. Hermès, Paris, 2 edition, Octobre 1986.

[4] V. Chandru, D. Dutta, and C. M. Hoffmann. On the geometry of dupin cyclides. *CSD-TR-818*, November 1988.

[5] V. Chandru, D. Dutta, and C. M. Hoffmann. Variable radius blending using dupin cyclides. Technical Report CSD–TR–851, Purdue University, January 1989.

[6] G. Darboux. *Thèse à la faculté des sciences de Paris*. Annales scientifiques de l'école normale, 1866.

[7] G. Darboux. *Sur une Classe Remarquable de Courbes et de Surfaces Algébriques et sur la Théorie des Imaginaires*. Gauthier-Villars, 1873.

[8] G. Darboux. *Principe de géométrie analytique*. Gauthier-Villars, 1917.

[9] G. Demengel and J.P. Pouget. *Mathématiques des Courbes et des Surfaces. Modèles de Bézier, des B-Splines et des NURBS*, volume 1. Ellipse, 1998.

[10] Ch. P. Dupin. *Application de Géométrie et de Méchanique à la Marine, aux Ponts et Chaussées, etc* . Bachelier, Paris, 1822.

[11] D. Dutta, R. R. Martin, and M. J. Pratt. Cyclides in surface and solid modeling. *IEEE Computer Graphics and Applications*, 13(1) :53–59, January 1993.

[12] G. Farin. *Curves And Surfaces*. Academic Press, 3ème edition, 1993.

[13] J. Foley, A. Van Dam, D. Freiner, and J. Hughes. *Computer Graphics : Principles and Practice*. Addison Wesley, 2 edition, 1990.

[14] L. Garnier, M. Neveu, and S. Foufou. Jointure d'un cône et d'une sphère par une cyclide de dupin. *AFIG 2001*, pages 133–142, november 2001.

[15] G.Darboux. *Principe de Géométrie Analytique*. Gauthier-Villars, 1917.

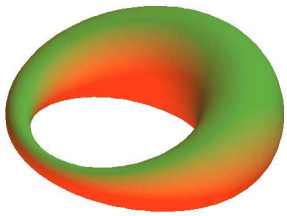
[16] M. Gourion. *Mathématiques, Terminales C et E, tome 2*. Fernand Nathan, 1983.

[17] C. Hoffman. Geometric and solid modeling : An introduction, 1989.

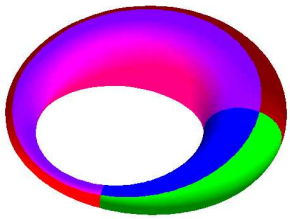
[18] J.M. Arnaudies J. Lelong-Ferrand. *Cours de Mathématiques : Géométrie et cinématique*, volume 3. Dunod, 2ème edition, Octobre 1991.

[19] F. Jaar, L. Garnier, and M. Neveu. The profiler : a tool for variational surface deformation. *Swiss Conference of CAD/CAM'99*, pages 179–184, février 1999.

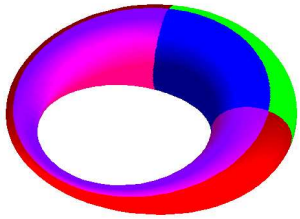
[20] M. Mantyla. *An Introduction to Solid Modeling*. Computer Science Press, Rockville, Md, 1998.



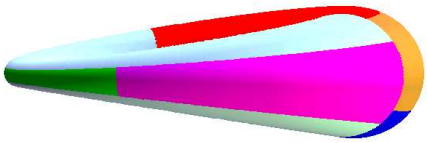
La cyclide



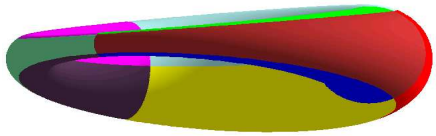
Une première conversion avec notre algorithme, 4 carreaux



Une seconde conversion avec notre algorithme, 4 carreaux



Algorithme de Pratt et algorithme de Pratt amélioré, 6 carreaux



Algorithme de Pratt amélioré, 9 carreaux

FIG. 7 – Comparaison des algorithmes pour effectuer la conversion d'une cyclide de Dupin en un carreau de Bézier Rationnel biquadrique.

[21] Moutard. *Annales de Mathématiques*. Bulletin de la société Philomatique et compte rendu de l'Accadémie, 1804.

[22] M Paluszny and W Boehm. General cyclides. *Computer Aided Geometric Design*, 15 :699–710, 1998.

[23] U. Pinkall. *Dupin hypersurfaces*. prepint, Bonn, 1985.

[24] M. J. Pratt. Cyclides in computer aided geometric design. *Computer Aided Geometric Design*, (7) :221–242, 1990.

[25] M. J. Pratt. Dupin cyclides and supercyclides. *Mathematics of Surfaces VI (Proc. of the VI. IMA Conference on the Mathematics of Surfaces, Brunel University*, 1994.

[26] M. J. Pratt. Cyclides in computer aided geometric design II. *Computer Aided Geometric Design*, 12(2) :131–152, 1995.

[27] M. J. Pratt. Quartic supercyclides I : Basic theory. *Computer Aided Geometric Design*, 14(7) :671–692, 1997.

[28] A. Requicha and J. Rossignac. Solid modelling and beyond, 1992.

[29] Ching-Kuang Shene. Blending two cones with dupin cyclides. *Computer Aided Geometric Design*, 15(7) :643–673, 1998.

[30] Ching-Kuang Shene. Do blending and offsetting commute for dupin cyclides. *Computer Aided Geometric Design*, 17(9) :891–910, 2000.

[31] D. Terzopoulos. Dynamic 3d models with local and global geometric deformations : Deformable superquadrics. *IEEE Trans on PAMI*, 13(7) :703–714, 1991.

[32] Fordyth A. R. Z. *Lecture on Differential Geometry of Curves and Surfaces*. Cambridge University Press, 1912.

Rétroconception en modélisation à base topologique

Franck Ledoux

LaMI, UMR 8042, 523 Place des Terrasses, 91000 Évry Cedex
fledoux@lami.univ-evry.fr

Résumé : *La description intuitive des opérations géométriques peut se faire de plusieurs façons : soit informellement, soit mathématiquement, soit par un algorithme. La première véhicule l'intuition sous-jacente aux opérations mais ne fournit ni rigueur, ni précision. Ceci est apporté par un algorithme ou une définition mathématique. L'approche algorithmique est appréciée par les développeurs, mais elle impose une implantation type. De plus elle rend difficile la comparaison des approches qui diffèrent souvent sur des configurations particulières. L'approche mathématique qui consiste à décrire de manière systématique le résultat des opérations, est plus abstraite. Elle n'a donc pas le biais précédent. Cependant, ces définitions ne reflètent généralement pas l'intuition des opérations, ce qui les rend souvent obscures et difficiles à manipuler pour des opérations complexes. Sur ce constat, nous proposons de définir différemment les opérations géométriques. Les définitions restent mathématiques, mais on tire parti des propriétés du modèle pour n'expliciter que le minimum de modifications à effectuer. De cette façon, nous obtenons un niveau de définition plus abstrait véhiculant toute l'intuition des opérations et du modèle topologique. Ce résultat a été obtenu en utilisant une méthode de rétro-définition basée sur l'utilisation des propriétés du modèle. Nous avons mené une formalisation complète du modèle des n-G-cartes et de l'opération de chanfreinage.*

Mots-clés : Topologie, n-G-cartes, abstraction, spécifications formelles, chanfreinage.

1 Introduction

En modélisation à base topologique, les objets sont représentés en distinguant leur topologie, c'est-à-dire leur décomposition en volumes, faces, arêtes et sommets, et leur géométrie, c'est-à-dire leur position et leur forme dans l'espace. Dans ce cadre, de nombreux modèles mathématiques permettent de représenter une classe particulière d'objets. Par exemple, les *cartes généralisées de dimension* n [Lie91], ou *n-G-cartes*, modélisent des quasi-variétés ouvertes ou fermées, orientables ou non, et de dimension quelconque. Pour ce faire, les n-G-cartes disposent de propriétés que toute opération de transformation d'objets doit préserver. Usuellement, la définition d'opérations géométriques peut se faire de trois façons différentes :

1. *informellement* - Le plus souvent les descriptions informelles sont très intuitives. Elles s'appuient sur des exemples et décrivent le résultat des opérations sans préciser la façon de calculer ce résultat. Malheureusement, elles sont souvent incomplètes et imprécises. Par exemple, on dira d'une opération de collage entre deux objets volumiques, qu'elle identifie des faces de chaque objet.
2. *mathématiquement* - Par nature, les définitions mathématiques sont précises et en pratique elles sont toujours complètes. Malheureusement, les définitions mathématiques des opérations topologiques sont très peu intuitives. Elles décrivent le résultat de manière systématique, sans même différencier les parties des objets modifiées ou non par les opérations. Elles restent cependant relativement abstraites, puisqu'elles décrivent le résultat et non le moyen de le calculer. Par exemple, pour le collage de deux volumes, on posera comme précondition que les faces à identifier soient isomorphes, et on décrira complètement l'objet résultant de ce collage.
3. *algorithmiquement* - On explique comment effectuer effectivement l'opération. Cette approche est appréciée par les développeurs, mais elle impose une implantation type. En effet, pour écrire réellement l'algorithme, il est nécessaire de s'appuyer sur la structure de données réellement utilisée. Si cette structure change, la définition n'est plus valable. Par exemple, si un objet est représenté comme une liste d'arêtes liées entre elles par des relations d'adjacence, un algorithme pour l'opération de collage pourra être basé sur le parcours de la liste des arêtes composant chacun des objets à coller et sur l'identification des arêtes à coller.

Définir informellement une opération offre l'avantage de présenter intuitivement ce que doit faire l'opération, mais cette approche n'apporte ni rigueur, ni précision. L'utilisation d'un algorithme pallie ces problèmes, mais dépend trop de l'implantation. Elle rend donc impossible la comparaison des approches qui diffèrent souvent sur des configurations particulières. Une définition mathématique est rigoureuse, précise et ne tient pas compte de l'implantation. Elle semble donc la plus adaptée pour définir une opération géométrique. Cependant, elle a le désavantage de ne véhiculer aucune intuition liée au modèle manipulé.

Partant de ce constat, nous proposons de définir différemment les opérations géométriques. Les définitions restent mathématiques, mais on tire parti des propriétés du modèle pour n'expliciter que le minimum de modifications à effectuer. Certaines modifications implicites découlent alors des propriétés du modèle. On obtient de cette façon des définitions mathématiques plus abstraites que celles habituellement proposées, qui sont exhaustives, souvent obscures et difficiles à manipuler. Cette approche permet de formaliser fidèlement les descriptions intuitives des opérations géométriques. Elles sont donc plus faciles à écrire et ont moins de risque d'être erronées.

L'obtention de ces nouvelles définitions plus abstraites et plus intuitives, découle d'une étude complète de formalisation menée à l'aide du langage de spécification algébrique CASL [CAS00, Mos99, BM00]. Ce langage dispose d'un grand pouvoir d'expression qui permet de formaliser l'ensemble des concepts topologiques et d'utiliser un syntaxe très proche des notations mathématiques habituelles. Néanmoins, afin de nous concentrer sur la méthode de rétroconception proposée, nous présentons nos définitions abstraites sous la forme mathématique usuelle et non leurs spécifications qui peuvent être consultées dans [LAGB01, LA01].

Dans cet article, nous nous intéressons au modèle des n-G-cartes [Lie89, Lie91], à ses opérations de base et à des opérations plus complexes telles que le chanfreinage [Elt94]. Nous commençons par présenter le modèle des n-G-cartes ainsi que l'opération de base de couture. Cette opération nous sert à introduire notre méthode de rétroconception que nous appliquons ensuite sur l'opération de chanfreinage.

2 Cartes généralisées et rétroconception

Nous présentons dans cette section le modèle topologique des n-G-cartes, ainsi que l'opération de couture qui permet de coller deux objets de même dimension entre eux.

2.1 Cartes généralisées de dimension n

Une carte généralisée de dimension n ou n-G-carte est un modèle définissant la topologie d'une subdivision d'espace de dimension n. Ce modèle est un modèle ordonné qui est défini à partir d'un type unique d'éléments abstraits, les *brins*, sur lesquels sont définis des relations d'adjacence. Il permet de représenter des quasi-variétés orientables ou non de dimension n en fournissant une définition homogène à toutes les dimensions. Avant d'introduire la définition mathématique des n-G-cartes, présentons les intuitivement en décomposant successivement les différentes cellules d'un objet (voir figure 1). Les brins sont représentés visuellement par des segments ou demi-arêtes. Les liens entre brins sont modélisés mathématiquement par des applications ayant la propriété d'être des involutions[1].

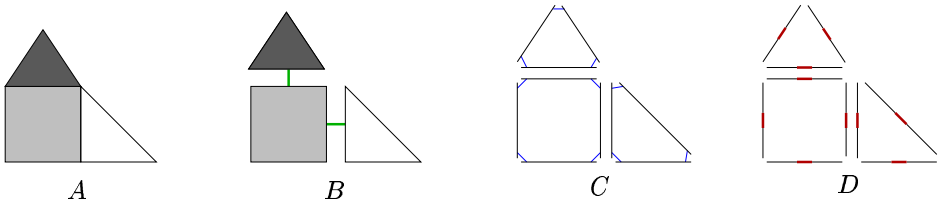


FIG. 1 – *Décomposition d'un objet $2D$ en un ensemble de brins.*

Considérons l'objet géométrique $2D$ de la figure $1-A$. À la figure $1-B$, le même objet est décomposé de façon à mettre en évidence les relations d'adjacence entre les faces le composant. À la figure $1-C$, on poursuit la décomposition de l'objet initial en décomposant chaque face en objets de dimension inférieure, c'est-à-dire en des arêtes liées par leurs relations d'adjacence. Enfin, à la figure $1-D$, chaque arête est découpée en deux demi-arêtes qui représentent des brins. Pour retrouver totalement l'objet initial à partir des brins, il reste à reporter les différentes relations d'adjacence sur ces éléments afin de retrouver les arêtes, les faces puis l'objet initial tout entier (voir figure $2-A$). On dénote les arêtes en indiquant une relation entre deux brins. Cette relation est une involution notée α_0, car elle relie des brins appartenant à des sommets différents, c'est-à-dire des entités de dimension 0.

[1]Une application $f : D \to D$ est une involution si et seulement si pour tout élément x appartenant à D, on a $xf^2 = x$, où f^2 est la composition de f par f.

Les faces sont reconstruites en introduisant l'involution α_1 qui relie deux brins appartenant à des arêtes différentes (dimension 1). Enfin, l'objet initial est complètement déterminé en reliant des brins de deux faces différentes (dimension 2) par α_2.

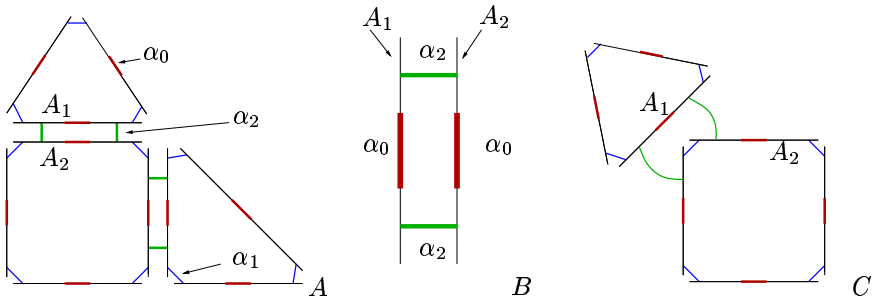


FIG. 2 – *Liaisons et invariants dans une 2-G-carte*

En résumé, chacune des relations d'adjacence est modélisée par une involution α_i dont l'indice i correspond à la dimension de la relation d'adjacence. Le fait d'utiliser des involutions assure que si un brin b_1 a pour image par α_i un brin b_2, alors l'image de b_2 par α_i est b_1, ce qui traduit un lien "physique" réciproque entre deux entités. Cependant, ceci n'est pas suffisant pour garantir la cohésion de la représentation de l'objet. Si nous reprenons l'exemple de la figure 1 et que nous reportons l'ensemble des relations d'adjacence dessus, nous obtenons la représentation de la figure $2-A$. Pour obtenir ce résultat, nous avons relié les faces adjacentes le long de leurs arêtes. Comme une arete est formée de deux brins, une liaison entre deux faces est constituée de deux liaisons α_2. Ces liaisons doivent être cohérentes. Ce qui exclut l'objet de la figure $2-C$. La contrainte utilisée ici est que $\alpha_0\alpha_2$ soit une involution. Ainsi, si les arêtes A_1 et A_2 sont définies relativement à une face, on relie leurs brins par α_2 pour obtenir l'arête complète (voir figure $2-B$). Ce type de contraintes se généralise simplement aux dimensions supérieures [Lie91].

DÉFINITION 1 (CARTE GÉNÉRALISÉE) *Une carte généralisée de dimension n, n≥-1, ou* **n-G-carte**, *est une algèbre G=(D, α_0, ... , α_n), où :*

- *D est un ensemble de* **brins**, (C_1)
- $\alpha_0, \ldots, \alpha_n$ *sont des* **involutions** *sur D telles que :* (C_2)

  $\forall i \in \{0, ..., n-2\}, \forall j \in \{i+2, \ ... \ , n\}, \ \alpha_i\alpha_j$ *est une involution.* (C_3)

Si les différentes cellules[2] ont bien été mises en évidence lors de nos deux exemples de construction, la définition des n-G-cartes ne s'appuie pas sur une notion explicite de cellules. Pour retrouver les cellules composant une n-G-carte, il suffit de regrouper les brins en fonction des dimensions des liaisons. Cette notion de familles de brins se traduit par la notion d'orbite .

DÉFINITION 2 (ORBITE) *Soit une n-G-carte $G = (D, \alpha_0, ..., \alpha_n)$, un brin $d \in D$ et P un ensemble de m permutations*[3] *$\{p_1, \ldots, p_m\}$ sur D. On appelle* **orbite** *de d par rapport à P l'ensemble*

$$\{d' | \exists p'_1 \in P, \ldots, \exists p'_k \in P, \text{ avec } d' = dp'_1 \ldots p'_k\} \text{ (on peut avoir } p'_i = p'_j \text{ pour } i \neq j)$$

et on note cette orbite $< p_1, \ldots, p_m > (d)$.

L'orbite $< p_1, \ldots, p_k > (d)$ peut être vue comme l'ensemble des brins atteignables à partir de d par la composition des fonctions $p_1, \ldots, p_k$. Les cellules sont des orbites particulières.

DÉFINITION 3 (CELLULE) *Soient une n-G-carte $G = (D, \alpha_0, ..., \alpha_n)$, un brin $d \in D$, et un entier i inférieur à n. La* **i-cellule** *contenant d est l'orbite $< \alpha_0, ..., \alpha_{i-1}, \alpha_{i+1}, ..., \alpha_n > (d)$.*

Une i-cellule contenant un brin d est donc constituée de l'ensemble des brins atteignables à partir de d par toute composition d'involutions différentes de α_i. L'involution α_i relie entre elles deux cellules de dimension i. Elle permet donc de "passer" d'une i-cellule à sa voisine. La figure 3 présente les différentes cellules d'une 3-G-carte :

[2] Sommets, arêtes, faces, volumes, etc.

[3] Une fonction p est une permutation sur l'ensemble D si pour tout brin $d \in D$, il existe un entier $k > 0$ tel que $dp^k = d$.

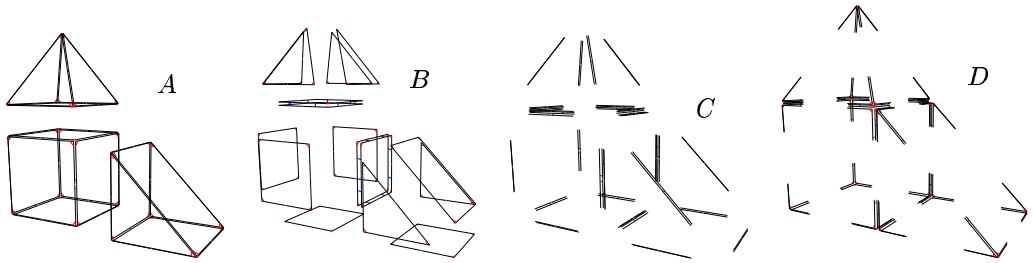


FIG. 3 – *Cartes des cellules.*

en A, les volumes ou 3-cellules ; en B, les faces ou 2-cellules ; en C, les arêtes ou 1-cellules ; en D, les sommets ou 0-cellules.

2.2 Couture et rétroconception

L'opération de couture est une opération de base qui permet de coller deux objets de même dimension[4]. Nous introduisons sa description informelle puis nous présentons sa définition mathématique classique pour finalement détailler notre approche.

2.2.1 Description informelle

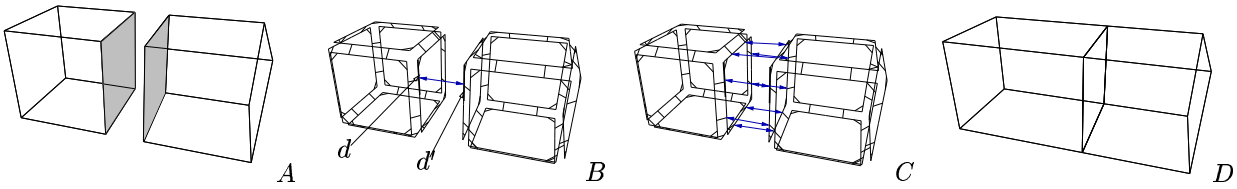


FIG. 4 – *Couture de deux cubes le long d'une face.*

Considérons deux cubes que l'on veut coller le long d'une face (voir figure 4). Naturellement, on souhaite désigner les deux faces à coller ensemble. En l'occurrence, les faces grisées de la figure $4-A$. Comme nous utilisons le modèle des n-G-cartes, cela revient simplement à désigner deux brins d et d' qui appartiennent à ces deux faces et seront liés par la couture (voir figure $4-B$). Puisque nous voulons coller deux volumes, les brins d et d' seront cousus par α_3, ainsi que tous les autres brins des deux faces (voir figure $4-C$). Nous obtenons ainsi l'objet géométrique de la figure $4-D$. Notons qu'une précondition de cette opération est que les brins désignés ne soient pas déjà cousus par α_3. C'est le cas ici, et on dira que d et d' sont 3-libres.

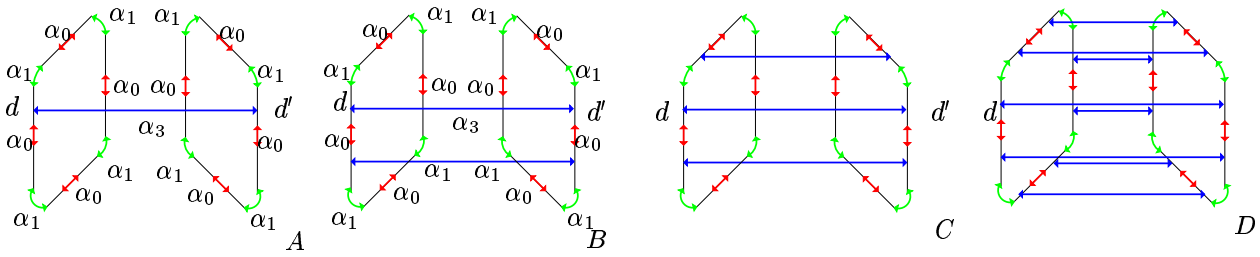


FIG. 5 – Comment coudre deux faces ensemble.

Du fait des propriétés des n-G-cartes, la désignation de deux brins et d'une dimension suffit à déterminer la couture. En effet, attardons-nous sur les brins des deux faces à coller (voir figure 5). À la figure $5-A$, le lien par α_3 entre

[4]D'un point de vue géométrique, l'opération de couture correspond à l'opération classique d'identification d'arêtes en B-Rep.

les deux brins d et d' est établi. Le fait que d et d' soient liés par α_3 impose que les brins $d\alpha_0$ et $d'\alpha_0$ soient aussi liés par α_3 pour assurer la propriété que $\alpha_0\alpha_3$ est une involution (voir figure $5-B$). De même, les deux brins $d\alpha_1$ et $d'\alpha_1$ doivent être liés par α_3 pour assurer que $\alpha_1\alpha_3$ est une involution (voir figure $5-C$). Par propagation, tous les brins des deux faces doivent aussi être liés par α_3 pour obtenir une 3-G-carte cohérente (voir figure $5-D$). Notons que si les deux faces n'avaient pas été "identiques", il n'aurait pas été possible de préserver les propriétés du modèle.

La description intuitive de l'opération de couture se résume donc simplement à dire que l'on lie deux brins ensemble. Les autres liens sont imposés par des propriétés du modèle.

2.2.2 Définition mathématique de la couture

Usuellement, une opération est définie de manière constructive soit par une définition mathématique, soit par un algorithme. Dans le modèle des n-G-cartes, les définitions mathématiques existantes énumèrent l'ensemble des brins et des liens du nouvel objet. Ces définitions ne traduisent donc pas directement l'intuition. Par exemple, une définition mathématique de la couture [Elt94] est :

DÉFINITION 4 (COUTURE) *Soient une n-G-carte $G = (D, \alpha_0, ..., \alpha_n)$, deux brins $d, d' \in D$, $i \leq n$ un entier naturel et φ un isomorphisme de $< \alpha_0, \ldots, \alpha_{i-2}, \alpha_{i+2}, \ldots, \alpha_n > (d)$ dans $< \alpha_0, \ldots, \alpha_{i-2}, \alpha_{i+2}, \ldots, \alpha_n > (d')$ tel que $\varphi(d) = d'$. Nous définissons la n-G-carte $G' = (D', \alpha'_0, ..., \alpha'_n)$ par :*

1. $D' = D$
2. $\forall j, 0 \leq j \leq n, j \neq i, \alpha'_j = \alpha_j$
3. $\alpha'_i(e) = \begin{cases} \varphi(e) & \text{si } e \in < \alpha_0, \ldots, \alpha_{i-2}, \alpha_{i+2}, \ldots, \alpha_n > (d) \quad (3.1) \\ \varphi^{-1}(e) & \text{si } e \in < \alpha_0, \ldots, \alpha_{i-2}, \alpha_{i+2}, \ldots, \alpha_n > (d') \quad (3.2) \\ \alpha_i(e) & \text{sinon.} \quad (3.3) \end{cases}$

G' est la n-G-carte résultant de la i-couture de d et d' dans G.

Dans cette définition, la notion d'isomorphisme utilisée est définie comme suit.

DÉFINITION 5 (ISOMORPHISME DANS UNE n-G-CARTE) *Soient une n-G-carte $G = (D, \alpha_0, \ldots, \alpha_n)$ et un sous-ensemble $I = \{\alpha'_1, \ldots, \alpha'_i\}$ de $\{\alpha_0, \ldots, \alpha_n\}$. Étant donnés deux brins d et d' de D, une fonction $\varphi : D \to D$ réalise un isomorphime de $< I > (d)$ sur $< I > (d')$ si et seulement si*

1. *φ réalise une bijection de $< I > (d)$ sur $< I > (d')$;*
2. *pour toute involution $\alpha'_j \in I$, pour tout brin d_1 de $< I > (d)$, on a*

$$d_1\alpha'_j\varphi = d_1\varphi\alpha'_j.$$

La définition 4 mérite quelques explications :

– l'isomorphisme φ de $< \alpha_0, \ldots, \alpha_{i-2}, \alpha_{i+2}, \ldots, \alpha_n > (d)$ dans $< \alpha_0, \ldots, \alpha_{i-2}, \alpha_{i+2}, \ldots, \alpha_n > (d')$ permet de représenter les nouveaux liens ajoutés par la couture. Le choix de ces deux orbites est compréhensible si l'on se souvient des contraintes régissant le modèle des n-G-cartes. En particulier, $\alpha_i\alpha_j$ est une involution pour tout $0 \leq i + 1 < j \leq n$. Puisque d et d' sont cousus par α_i, il est nécessaire de coudre aussi les brins $d\alpha_j$ et $d'\alpha_j$ par α_i pour tout $j \in \{0, 1, \ldots, i-2, i+2, \ldots, n\}$ afin d'assurer cette contrainte. Par propagation, tous les brins des deux orbites se trouvent donc liés par α_i.
– les points (2) et (3.3) de la figure 4 indiquent quels sont les liens qui restent inchangés.
– les points (3.1) et (3.2) introduisent les nouveaux liens. Le point (3.2) assure que la fonction α_i est une involution.

Pour conclure sur cette définition, on peut remarquer qu'elle ne s'appuie pas sur les propriétés des n-G-cartes pour définir implicitement la nouvelle n-G-carte mais qu'elle énumère la totalité des éléments composant la nouvelle n-G-carte.

2.2.3 Rétro-définition

La définition mathématique 4 fournit une description exhaustive et systématique de la couture. Par contre elle ne retranscrit pas l'intuition véhiculée par la définition informelle qui est que l'on n'explicite que le lien entre les

deux brins désignés par l'utilisateur et qu'on laisse "travailler" les propriétés du modèle. Nous nous proposons d'apporter à la fois rigueur et intuition en fournissant une définition mathématique abstraite au sein de laquelle on caractérise complètement le résultat de l'opération tout en explicitant le minimum de changements.

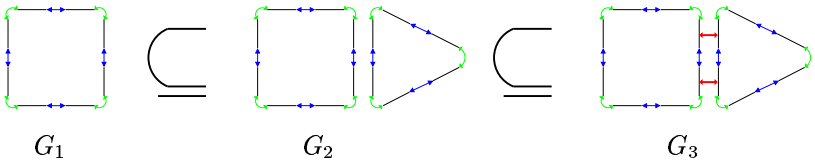


FIG. 6 – Le principe d'inclusion entre n-G-cartes.

Informellement, la couture de deux brins d et d' par α_i dans une n-G-carte implique que les brins des orbites $< \alpha_0, \ldots, \alpha_{i-2}, \alpha_{i+2}, \ldots, \alpha_n > (d)$ et $< \alpha_0, \ldots, \alpha_{i-2}, \alpha_{i+2}, \ldots, \alpha_n > (d')$ sont obligatoirement cousus correctement pour valider les propriétés du modèle. Cette explication informelle de la couture repose sur un fait implicite important qui est que les seules modifications effectuées sur la n-G-carte initiale sont le lien explicité entre d et d' et les liens résultant de ce lien pour satisfaire les propriétés du modèle. En d'autres termes, aucun lien supplémentaire n'a été effectué, et aucun brin n'a été ajouté ou supprimé. Pour définir rigoureusement cette notion de changement minimum, nous introduisons le principe d'inclusion entre n-G-cartes.

DÉFINITION 6 (INCLUSION ENTRE n-G-CARTES) *Une n-G-carte $G = (D, \alpha_0, \ldots, \alpha_n)$ est* **incluse** *dans une n-G-carte $G' = (D', \alpha'_0, \ldots, \alpha'_n)$ si et seulement si*
– *D est un sous-ensemble de D',*
– *pour tous brins d_1 et d_2 de D tels que $d_1 \neq d_2$, si $d_1\alpha_i = d_2$, $i \in [0, n]$, alors $d_1\alpha'_i = d_2$.*

Ainsi le passage d'une n-G-carte à une autre par inclusion, consiste à ajouter des brins et/ou des liens. Dans l'exemple de la figure 6, le passage de G_1 à G_2 se fait en ajoutant les brins du triangle et leurs liaisons, et le passage de G_2 à G_3 se fait en liant le carré au triangle. Dans le modèle des n-G-cartes, les brins i-libres sont représentés par un point fixe de l'involution α_i. Comme l'indique la définition 6, seules les liaisons de ces points fixes sont susceptibles de changer par inclusion. À l'aide de la notion d'inclusion, nous redéfinissons la couture comme suit.

DÉFINITION 7 (COUTURE) *Soient une n-G-carte $G = (D, \alpha_0, ..., \alpha_n)$, un entier naturel $i \leq n$ et deux brins i-libres $d, d' \in D$. La plus petite n-G-carte $G' = (D', \alpha'_0, ..., \alpha'_n)$ telle que*

1. *$G \subseteq G'$*
2. *$d\alpha'_i = d'$*

est la n-G-carte résultant de la i-couture de d et d' dans G.

Comme la n-G-carte obtenue contient la n-G-carte initiale, nous conservons tous les brins et liens présents avant la couture. Le fait que les deux brins désignés soient liés dans la nouvelle n-G-carte assure non seulement la présence de ce lien mais aussi celle de tous les liens nécessaires à la cohérence des n-G-cartes le long de l'orbite adéquate. Enfin, comme nous considérons la plus petite n-G-carte contenant la n-G-carte initiale G et telle que les deux brins désignés d et d' sont cousus par α_i, nous assurons que les seules différences entre la n-G-carte obtenue et la n-G-carte G sont dues à la propagation des invariants. L'existence de cette n-G-carte peut facilement être prouvée[5]. À titre d'exemple, considérons la figure 7. En A, nous avons l'objet initial sur lequel est matérialisé la liaison par α_2 à effectuer entre d et d'. En l'état actuel, ce n'est donc pas une 2-G-carte. En B, la 2-G-carte résultant de la couture de d et d' par α_2 contient seulement une liaison supplémentaire par α_2. La présence de cette liaison est la conséquence immédiate de la préservation des invariants du modèle : comme d et d' sont liés par α_2, les brins $d\alpha_0$ et $d'\alpha_0$ doivent l'être aussi pour assurer que $\alpha_0\alpha_2$ est une involution. En C, une 2-G-carte où la 2-couture entre d et d' a été effectuée correctement, mais qui contient des brins et des liaisons supplémentaires. De tels objet sont évités en ne conservant que la plus petite 2-G-carte. Enfin en D, la couture est aussi effectuée correctement mais la 2-G-carte initiale a été tronquée, elle ne contient donc pas la 2-G-carte de départ.

[5]Une façon simple de prouver l'existence et l'unicité de cette n-G-carte est de montrer qu'elle est la plus petite n-G-carte vérifiant la définition 6 de la couture.

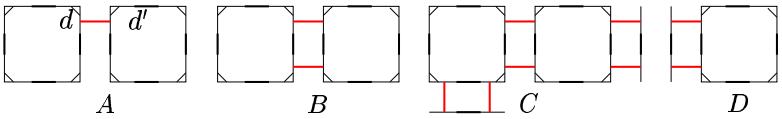


FIG. 7 – *Exemple de couture.*

Les définitions 6 et 7 ainsi que le modèle des n-G-cartes et des opérations comme l'isomorphisme entre deux orbites, ont toutes été formalisées en CASL [LAGB01]. Plus précisément, c'est de ce travail de spécification que découlent les définitions précédentes. Il en va de même pour l'opération de chanfreinage que nous abordons dans la section suivante.

3 Rétroconception du chanfreinage

Dans cette section, nous présentons tout d'abord une définition informelle du chanfreinage pour ensuite introduire notre définition abstraite.

3.1 Qu'est-ce que le chanfreinage ?

Prenons l'exemple de la figure 8. En A, nous avons la topologie d'un cube dont nous chanfreinons l'arête A et le sommet S. Dans le cas du chanfreinage du sommet S, on obtient une face triangulaire, car le sommet S était initialement bordé par trois faces (voir figure $8-B$) alors que l'arête A est remplacée par une face à deux côtés (voir figure $8-C$).

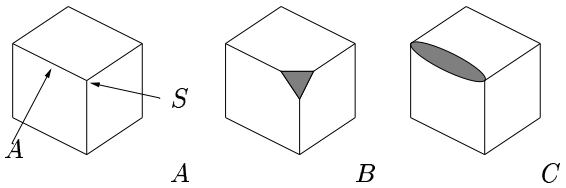


FIG. 8 – *Exemples de chanfreinage.*

Plus généralement, étant donnée une subdivision S de dimension n, chanfreiner une de ses cellules, notée C, consiste à remplacer cette cellule par une cellule de dimension n dont le nombre de cellules de dimension $n-1$ la bordant est égal au nombre de cellules de dimension n incidentes à C.

Notons dès à présent qu'en tant que définition informelle, la définition du chanfreinage donnée précédemment n'explicite pas ce que signifie le remplacement d'une cellule par une autre. En particulier comment connecte-t-on la nouvelle cellule au reste de l'objet qui est resté inchangé ? L'écriture d'une définition mathématique ou d'un algorithme permet de répondre à ce type de questions en décrivant complètement le résultat.

3.2 Définition abstraite du chanfreinage

Comme la couture, le chanfreinage a été défini mathématiquement et de façon exhaustive [Elt94]. Par souci de concision, nous ne donnons pas cette définition ici pour nous concentrer sur notre approche. Nous cherchons à identifier les modifications explicites et celles à laisser implicites lors du chanfreinage d'une cellule. Pour cela, nous traduisons la définition informelle du chanfreinage dans le modèle des n-G-cartes. Supposons que l'on dispose d'une n-G-carte $G = (D, \alpha_0, \ldots, \alpha_n)$ et d'une i-cellule $C \subset G$ à chanfreiner, on doit :

1. construire une n-cellule C' telle que le nombre de $(n-1)$-cellules lui étant incidentes soit égal au nombre de n-cellules précédemment incidentes à C.
2. remplacer la cellule C par la nouvelle cellule C'.

La première étape consiste à construire une n-cellule à partir d'une i-cellule en respectant certaines contraintes sur le nombre de cellules incidentes. Cette étape est effectuée en considérant la notion de cellule duale.

DÉFINITION 8 (CELLULE DUALE) *Soit une n-G-carte $G = (D, \alpha_0, \ldots, \alpha_n)$ telle que $\alpha_i = id_{|D}$, alors la n-G-carte $G' = (D', \alpha'_0, \ldots, \alpha'_n)$ définie par :*
– $D' = D$,
– $\forall d \in D,\ \forall k \in [0, n],\ k \neq i$
  – *si* $k < i,\ d\alpha'_k = d\alpha_k$,
  – *si* $k > i,\ d\alpha'_{k-1} = d\alpha_k$,
est appelée la n-G-carte **duale** *de G. Elle a pour propriété que $\alpha'_n = id_{|D'}$.*

Une i-cellule C incluse dans une n-G-carte G peut elle-même être considérée comme une n-G-carte telle que $\alpha_i =$id. Grâce à la définition précédente, on peut construire à partir d'une i-cellule C, une n-cellule dont le nombre de $n-1$ cellules incidentes soit égal au nombre de n-cellules incidentes à C. Par exemple, à la figure 9, la cellule duale d'un sommet en A est obtenue en B, les liaisons α_1 et α_2 (qui ont toutes les deux une dimension supérieure à 0, c'est-à-dire à celle de la cellule considérée) deviennent respectivement des liaisons α_0 et α_1. La cellule duale d'une arête en C est donnée en D. Une arete est de dimension 1, les liaisons α_0 restent donc inchangées, mais les liaisons α_2 deviennent des liaisons α_1. Ces deux exemples sont traités dans une 2-G-carte.

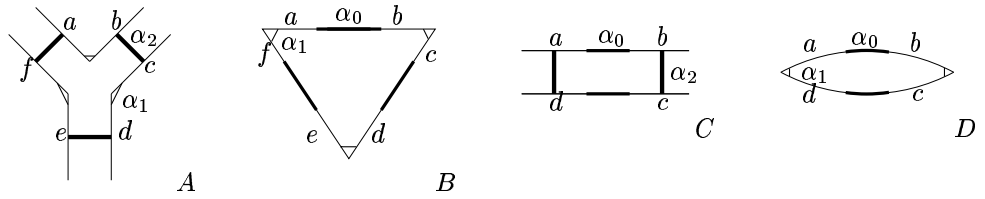


FIG. 9 – *Deux exemples de cellules duales dans une 2-G-carte.*

Si la création de la n-cellule duale se fait aisément, il faut maintenant pouvoir l'introduire dans la n-G-carte initiale. Il va de soi que cela ne peut se faire en remplaçant littéralement la i-cellule à chanfreiner par la n-cellule construite précédemment sous peine de ne pas pas pouvoir "recoller" la cellule au reste de la n-G-carte. Il faut en fait éclater la cellule de départ, construire la nouvelle cellule et les relier entre elles. C'est ce que nous faisons avec la définition suivante qui est commentée juste après.

DÉFINITION 9 (CHANFREINAGE) *Soient une n-G-carte $G = (D, \alpha_0, ..., \alpha_n)$ et une i-cellule $C =< \alpha_0, ..., \alpha_{i-1}, \alpha_{i+1}, ..., \alpha_n > (d)$ de G. Nous notons D_i l'ensemble des brins de C, et nous introduisons la famille $(D_j)_{j\in[i+1,n]}$ d'ensembles de brins tels que pour tout $j \in [i+1, n]$,*
– $D \cap D_j = \emptyset$,
– *pour tout* $k \in [i+1, n],\ k \neq j,\ D_j \cap D_k = \emptyset$,
– *il existe une bijection* $\varphi^j : D_i \to D_j$.
Nous considérons en outre la n-G-carte éclatée $L = (D^L, \alpha_0^L, ..., \alpha_n^L)$ qui est la plus grande n-G-carte incluse dans G et telle que $\forall d \in D_i,\ d\alpha_{i+1}^L = d$. Nous définissons alors la n-G-carte chanfreinée $I = (D^I, \alpha_0^I, ..., \alpha_n^I)$ comme la plus petite n-G-carte contenant L et telle que :

1. $D^I = D \cup \bigcup_{j\in[i+1,n]} D_j$
2. *pour tout* $d \in D_i$, $d\alpha_{i+1}^I = d\varphi^{i+1}$
3. *pour tout* $d \in D_i$, *pour tout* $k \in [i, n-1]$, $d\varphi^k\alpha_{k+1}^I = d\varphi^{k+1}$
4. *la cellule formée des brins* $d'\varphi^n$ *avec* $d' \in D_i$, *est la cellule duale de* C.

La n-G-carte I est le résultat du chanfreinage de la cellule C dans la n-G-carte G.

Considérons la figure 10 où un sommet C est chanfreiné dans une 2-G-carte G. Sur la ligne du haut, nous représentons l'objet dans sa globalité, tandis que sur la ligne du bas, nous effectuons un zoom sur la cellule à chanfreiner.
– En A, l'objet de départ est représenté. À titre d'exemple, nous détaillons les transformations liées au brin d.
– En B, la cellule C est éclatée, c'est-à-dire que l'ensemble des brins de la cellule sont décousus par α_1. Le "trou" ainsi formé permettra d'introduire la nouvelle face duale de C. Notons que c'est cette n-G-carte (et non la n-G-carte initiale) qui est incluse dans la n-G-carte finale. En effet, initialement (voir la figure 10-A) le brin d est lié

à son voisin, alors qu'au final (voir la figure 10-F) il est lié à un autre brin introduit ci-dessous. Cette n-G-carte éclatée (figure 10-B) est notée L dans la définition 9 et est elle-même définie comme incluse dans la n-G-carte de départ (figure 10-A).

– En C, nous présentons le traitement local au brin d. Pour ce brin on ajoute une chaîne de deux brins $d\varphi^1$ et $d\varphi^2$ tels que $d\alpha_1 = d\varphi^1$ et $d\varphi^1\alpha_2 = d\varphi^2$. Ce traitement est défini par les points 2 et 3 de la définition 9. Les fonctions $\varphi^{i+1}, \ldots, \varphi^n$ permettent d'ajouter et de nommer les nouveaux brins dans la n-G-carte.
– En D, nous présentons l'ajout de la chaîne de brins pour tous les brins de la cellule C. La nouvelle face commence ainsi à prendre forme.
– En E, sont présentées les coutures de la nouvelle face, qui est la duale du sommet C de départ à chanfreiner. Cette face est constituée de tous les brins $d'\varphi_2$ introduits pour chaque brin d' du sommet de départ. On effectue donc les coutures entre ses brins selon la définition 8 qui est utilisée au point 4 de la définition 9.
– Le résultat de la figure $10-E$ n'est pas une 2-G-carte car certains liens manquent aux niveaux des brins intermédiaires entre la cellule C et sa cellule duale. Ces dernier liens résultent en fait des propriétés du modèle et ne sont donc pas explicites dans la définition 9. On obtient finalement le résultat de la figure $10-F$.

Notons que plus la dimension de la n-G-carte est élevée, plus le nombre de coutures effectuées implicitement est grand.

Nous proposons ainsi une formalisation directe de la définition intuitive du chanfreinage. Seules les imprécisions de la description intuitive ont été comblées, en particulier la manière de lier la nouvelle face à la cellule de départ. Nous obtenons ainsi une définition rigoureuse, complète et intuitive du chanfreinage. Bien entendu, la complexité intrinsèque de l'opération de chanfreinage transparaît dans sa définition. La définition 9 ne peut donc pas être qualifiée de claire et concise. Cependant les gains sur ces points par rapport à la définition précédemment proposée par H. ELTER [Elt94] sont significatifs. De plus, la proximité avec la description intuitive de l'opération en fait une référence de correction beaucoup plus sûre.

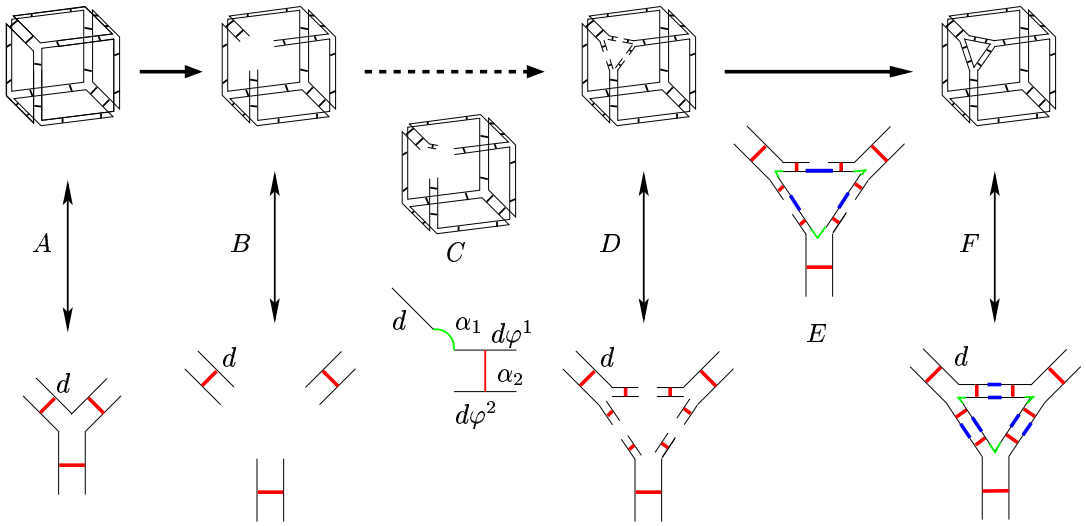


FIG. 10 – *Arrondi fermé d'un sommet dans une* 3*-G-carte plongée.*

3.3 Retombées algorithmiques

Même si cela peut paraître paradoxal, la définition 9 a des retombées algorithmiques directes. En fait, la recherche de critères minimaux nous permet d'isoler un comportement local à chaque brin et le travail des invariants du modèle. Dans le cas de la couture, le traitement local est simplement de lier deux brins ensemble, tandis que pour le chanfreinage, il consiste à ajouter une chaîne de $n-i$ brins. Ensuite, les invariants travaillent directement dans le cas de la couture et sont amorcés par les liaisons de la cellule duale dans le cas du chanfreinage. À titre d'exemple, l'algorithme de chanfreinage que nous avons implanté suit cette démarche : on parcourt l'ensemble des brins de la cellule à chanfreiner dans un ordre quelconque ; pour chaque brin d, on ajoute une chaîne de $n-i$ brins ; si les brins voisins de d dans la cellule à chanfreiner sont déjà traités, on effectue les coutures de la cellule duale et on explicite la propagation des invariants du modèle entre d et ses voisins.

Finalement, nous obtenons une définition mathématique du chanfreinage à la fois plus abstraite que la définition usuelle, mais aussi plus proche des algorithmes effectivement mis en œuvre qui sont locaux. Obtenir de tels algorithmes est important car ils permettent de s'affranchir de l'ordre de parcours des objets en simplifiant la programmation, ils suppriment presque totalement les préconditions en étant locaux et non plus globaux et ils se généralisent plus facilement à des dimensions quelconques.

4 Conclusion et perspectives

Dans le cadre de la modélisation géométrique à base topologique, nous proposons une nouvelle approche pour définir mathématiquement les opérations de transformations géométriques dans le modèle des n-G-cartes. La description proposée est basée sur l'inclusion entre n-G-cartes et repose sur la recherche de critères minimaux de modification.

Un intérêt majeur de notre approche est d'obtenir des définition mathématiques plus proche du niveau informel et donc plus faciles à valider. Ce point est important pour conférer un degré de confiance plus fort à une nouvelle opération géométrique. En outre, nous avons vu que l'écriture de telles définitions abstraites met en évidence des considérations d'ordre algorithmique. Les algorithmes locaux basés sur le parcours d'un ensemble de brins et le traitement local de ceux-ci se déduisent plus facilement de nos définitions où le traitement local à chaque brin est identifié et les parcours se déduisent des invariants du modèle.

Nous avons déjà appliqué notre approche sur des opérations telles que l'extrusion et la triangulation, mais il serait intéressant de s'intéresser à des opérations plus complexes comme le produit cartésien ou les opérations booléennes. En outre, il faudrait tester notre approche sur d'autres modèles topologiques comme les cartes [Cor75], les chaînes de cartes [EL94], ou les ensembles simpliciaux [Lan95]. La même démarche minimaliste doit y être applicable à condition de changer le critère de modification minimale d'un objet.

Références

[BM00] M. Bidoit and P. D. Mosses. A gentle introduction to CASL. Tutorial, CoFI Workshop at the 3rd European Joint Conferences on Theory and Practice of Software (ETAPS'2000), Berlin, Germany, April 2000.

[CAS00] CoFI (common framework initiative) task group on language design. casl the common algebraic specification language summary, June 2000. ftp ://ftp.brics.dk/Projects/CoFI.

[Cor75] R. Cori. Un code pour les graphes planaires et ses applications. *Astérisque*, 27, 1975.

[EL94] H. Elter and P. Lienhardt. Cellular complexes as structured semi-simplicial sets. In *International Journal of Shape Modeling*, volume 1, pages 191–217. 1994.

[Elt94] H. Elter. *Étude de structures combinatoires pour la représentation de complexes cellulaires*. PhD thesis, Université Louis-Pasteur de Strasbourg, 1994.

[LA01] F. Ledoux and A. Arnould. Geospec : specification libraries for geometric modelling, sept. 2001. http ://www.sic.sp2mi.univ-poitiers.fr/GL/GeoSpec.

[LAGB01] F. Ledoux, A. Arnould, P. Le Gall, and Y. Bertrand. Geometric Modelling with CASL. In M. Cerioli and G. Reggio, editors, *Recents Trends in Algebraic Development Techniques*, number 2267 in Lecture Notes in Computer Science, pages 176–200. Springer Verlag, April 2001.

[Lan95] V. Lang. *Une étude de l'utilisation des ensembles simpliciaux en modélisation géométrique interactive*. thèse, Université Louis Pasteur, L.S.I.I.T, Strasbourg, octobre 1995.

[Lie89] P. Lienhardt. Subdivisions of n-dimensional spaces and n-dimensional generalized maps. In *Annual A.C.M. Symposium on Computational Geometry*, pages 228–236, Saarbrücken, R.F.A, June 1989.

[Lie91] P. Lienhardt. Topological models for boundary representations : a comparison with n-dimensional generalized maps. *Computer-Aided Design*, 23(1) :59–82, 1991.

[Mos99] P. D. Mosses. CASL : A guided tour of its design. *LNCS*, 1589 :216–240, 1999.

Deux algorithmes d'intersection des surfaces de subdivision

S. Lanquetin, S. Foufou, H. Kheddouci, M. Neveu

LE2I, FRE CNRS 2309/Université de Bourgogne BP 47870
21 078 Dijon cedex

sandrine.lanquetin@u-bourgogne.fr

Résumé : *Le calcul des intersections de surfaces est un problème fondamental en modélisation. Toute opération Booléenne peut être vue comme un calcul d'intersection suivi d'une sélection des parties à conserver pour construire la surface de l'objet résultant. Dans cet article, nous nous intéressons aux calculs des intersections de surfaces de subdivision (surfaces générées par le schéma de subdivision de Loop). Nous présentons trois variantes d'algorithmes de calcul différent. La première variante calcule cette intersection après une classification des faces des objets en couples d'intersection et de non intersection. La seconde variante se base sur le 1-voisinage des faces en intersection. La troisième variante utilise la notion de graphe biparti.*

Mots-clés : Opérations booléennes, courbes d'intersection, surface de subdivision, principe de Loop.

1. Introduction

Les méthodes de génération de surfaces occupent une place très importante en Infographie et en Conception et Fabrication Assistée par Ordinateur (CFAO). La modélisation basée sur les surfaces de subdivision dispose de deux avantages principaux. Elle s'applique à des maillages de topologie arbitraire (comme la modélisation polygonale) et elle a un comportement local (comme la modélisation par les NURBS ou les B-Splines) puisqu'elle s'appuie uniquement sur un petit nombre de points de contrôle.
Les surfaces de subdivision sont maintenant largement utilisées. Le succès de ces surfaces provient de leur capacité à générer des surfaces lisses à partir de maillages initiaux arbitraires et leur implémentation relativement aisée grâce à leur concept simple. Elles sont définies par un maillage initial de type arbitraire et des règles de raffinement. Ces règles sont composées de règles géométriques déterminant les positions des nouveaux points de contrôle à partir des positions des anciens et de règles topologiques qui décrivent la procédure de raffinement de la connectivité du polyèdre de contrôle et de ce fait les propriétés de la surface. A partir d'un maillage polygonal, appelé réseau de contrôle, l'application répétée des règles de raffinement produit des nouveaux maillages polygonaux comprenant de plus en plus de faces. La suite de maillages ainsi constituée converge vers une surface lisse, appelée surface limite (par exemple, B-spline ou Box-spline), topologiquement similaire au réseau de contrôle initial. La Figure 1 montre un exemple de surface de subdivision. De la gauche vers la droite, la surface est de plus en plus lisse.

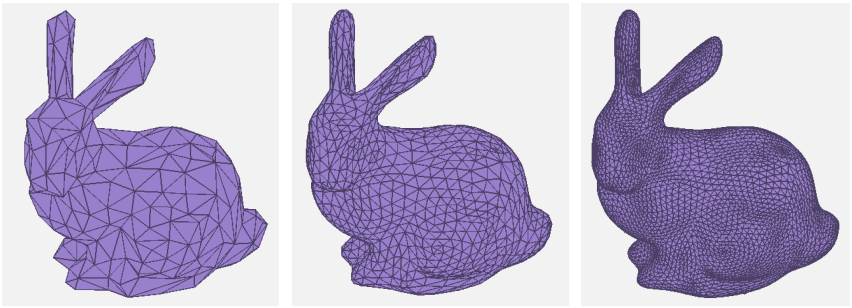


Fig. 1. Exemple de surface de subdivision.

Depuis l'introduction des surfaces de subdivision en 1978 par Catmull-Clark [Cat78] et Doo-Sabin [Doo78] de nombreux principes (ou schémas) de subdivision ont été proposés [Loo87, Zor00, Kob00, Vel00].
Le tableau 1 présente une synthèse des principes les plus connus. Pour chaque schéma, on peut constater le type de maillage sur lequel il peut s'appliquer (triangulaire, quadrilatéral…) et le type de la surface limite.

Schéma	Type de maillage	Type de surface limite
Doo-Sabin	polygonal	B-spline quadratique
Catmull-Clark	quadrilatéral	B-spline cubique
Loop	Triangulaire	B-spline triangulaire quartique

Tab. 1. Classification des schémas de subdivision les plus courants.

L'utilisation croissante de ces surfaces nécessite de reconstruire les outils préalablement connus pour d'autres types de surfaces ou de solides. Le calcul des opérations booléennes, par exemple, est fondamental pour la construction d'objets complexes à partir d'objets plus simples. Un objet CSG est généré par combinaison de plusieurs opérations booléennes (intersection, union, différence) entre des primitives élémentaires [Kri94]. Les primitives peuvent être des formes simples (cube, cylindre, tore…) ou des formes plus complexes construites à l'aide d'un ensemble de primitives simples ou généré par des surfaces plus compliquées. La Figure 2 montre un exemple d'opérations booléennes pouvant être effectuées sur deux objets simples.

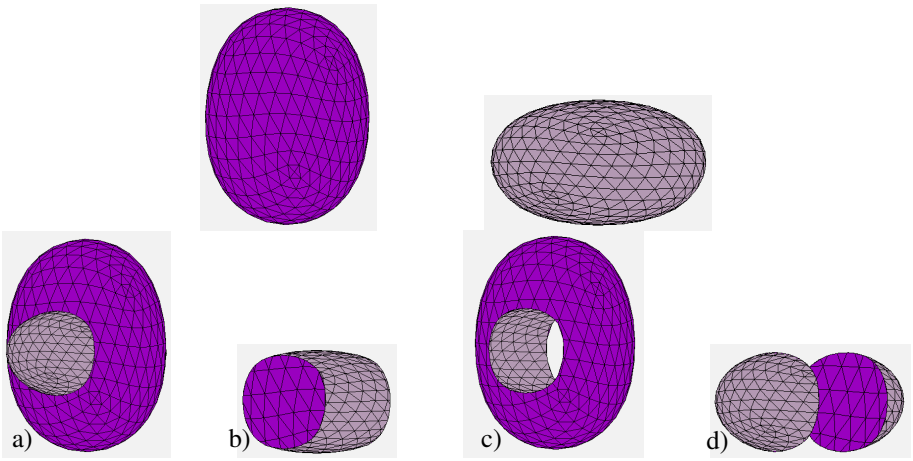


Fig. 2. Union (a), intersection (b) et différence ((c) et (d)) entre les deux surfaces.

De façon générale, une opération booléenne se déroule en deux étapes :

- Calcul des courbes d'intersection entre les surfaces des deux primitives impliquées dans l'opération : des points d'intersection sont trouvés, classés puis connectés pour approcher la courbe d'intersection.
- Conservation des parties des primitives nécessaire à la construction de la surface de l'objet résultant selon l'opération booléenne envisagée.

En modélisation géométrique, le calcul des intersections de surfaces est un problème récurrent et fondamental largement étudié pour les surfaces algébriques et paramétriques [Boe91, Pat93, Abd96]. Selon les algorithmes impliqués dans les différentes taches fondamentales, les méthodes de calcul d'intersections peuvent être classées parmi les quatre catégories principales suivantes. Les méthodes analytiques [Pat93, Cha87] lorsqu'une des surface est exprimée sous forme implicite et l'autre sous forme paramétrique. Les méthodes de discrétisation réduisent le problème d'intersection surface/surface à un ensemble de problèmes d'intersection courbe/surface [Bar87]. Les méthodes de suivi nécessitent de connaître au moins un point de la courbe d'intersection, appelé point de départ, pour générer une suite de points sur la courbe d'intersection en exploitant les propriétés géométriques locales des surfaces en intersection [Baj88]. Les méthodes de subdivision utilisent le raffinement des surfaces pour trouver une approximation polygonale des courbes d'intersection. La fiabilité de ces méthodes dépend du niveau de subdivision et des différents outils utilisés pour contrôler l'arbre de subdivision.
Dans cet article, nous nous intéressons particulièrement au calcul des intersections des surfaces de subdivision dans un contexte d'algèbre de solides modélisés par des surfaces de subdivision. Pour des raisons pratiques, nous ne considérons que les surfaces de subdivision générées par le principe de Loop. Ce principe s'appliquant uniquement sur des maillages triangulaires, il permet de travailler sur des faces triangulaires, et par conséquent planes. Nous présentons et nous comparons trois variantes d'un algorithme de calcul : la première variante calcule cette intersection après une classification des faces des objets en couples d'intersection et de non intersection. La seconde variante se base sur le 1-voisinage des faces en intersection. La troisième variante utilise la notion de graphe biparti. Pour appliquer les deux dernières variantes, il est nécessaire de connaître les courbes d'intersection au premier niveau. Elles seront donc calculées à l'aide de l'algorithme naturel d'intersection de maillages. La principale différence entre la version voisinage et la version graphe repose sur les ensembles de faces considérés. La première considère un ensemble de faces par objet alors que la seconde fait intervenir plusieurs sous ensemble de l'ensemble précédent par l'intermédiaire d'un graphe biparti.

2. Concepts de base.

La deuxième et la troisième variante de notre algorithme se basent sur les notions de 1-voisinage, de 2-voisinage et de graphe biparti. La Figure 3 illustre les termes de 1-voisinage et de 2-voisinage. Le 1-voisinage $\mathcal{V}_1(F)$ d'une face F contient cette face et toutes les faces voisines à cette face par un sommet, il est délimité par les petits pointillés : $\mathcal{V}_1(F) = \{F\} \cup \{F_i / \exists s \in S(F) \cap S(F_i)\}$ où $S(F)$ est l'ensemble des sommets de la face F. Le 2-voisinage $\mathcal{V}_2(F)$ est en fait le 1-voisinage de $\mathcal{V}_1(F)$, il est borné par les grands pointillés : $\mathcal{V}_2(F) = \{\mathcal{V}_1(F_i) / F_i \in \mathcal{V}_1(F)\}$. L'ensemble des faces obtenues par une subdivision du 1-voisinage de F est noté $\mathcal{W}_2(F)$, $\mathcal{W}_2(F) = \{F_i / F_i \in Loop(\mathcal{V}_1(F))\}$. $\mathcal{W}_1(F)$ est un sous-ensemble de faces de $\mathcal{W}_2(F)$ restreint aux faces ayant un sommet commun avec l'une des sous faces résultant de la subdivision de F, $\mathcal{W}_1(F) = \{F_i / F_i \in Loop(\mathcal{V}_1(F)) \,\&\, s \in S(Loop(F)) \cap S(F_i)\}$ où $S(Loop(F))$ est l'ensemble des sommets obtenus par la subdivision de F.

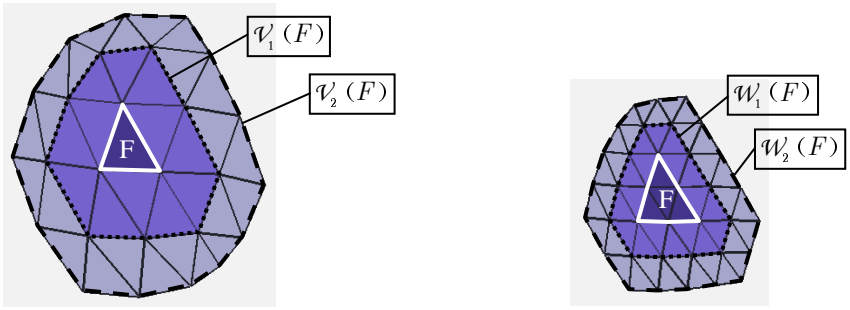


Fig. 3. 1-voisinage et 2-voisinage : $\mathcal{V}_1(F)$, $\mathcal{V}_2(F)$, $\mathcal{W}(F)$ et $\mathcal{W}_2(F)$.

Un graphe biparti est un graphe dans lequel :

- Les sommets sont répartis en deux groupes I_1 et I_2. Dans notre algorithme, ces groupes seront respectivement formés par les faces intersectantes de chaque surface.
- Chaque arête a une de ses extrémités dans chacun de ces groupes. Dans notre cas, une arête symbolisera la présence d'une intersection entre deux faces.
- Aucune arête ne peut relier deux sommets d'un même groupe.

3. La variante naturelle

Le calcul des courbes d'intersection s'effectue en plusieurs étapes. Tout d'abord, les faces des deux surfaces sont répertoriées en deux catégories : les faces en intersection et les autres. Seul les couples de faces en intersection sont considérés dans la suite de l'algorithme. L'intersection face/face est calculée pour obtenir les points d'intersection qui seront ensuite triés et reliés par des segments de manière à obtenir des approximations linéaires par morceaux des courbes d'intersection.

3.1. Détection des intersection entre deux faces

Cette étape préliminaire consiste à utiliser les tests de collision des boites englobantes des faces des deux maillages pour faire un premier filtrage qui éliminera les faces clairement disjointes de toute investigation future. Ensuite, les intersections de toutes les faces restantes vont être calculées. La complexité de l'algorithme devient en $O(m_1 \times n_1)$ avec m_1 et n_1 respectivement inférieurs aux nombre de faces m et n des surfaces S_1 et S_2.

3.2. Déterminer les points d'intersection entre deux faces

Plusieurs méthodes peuvent être envisagées pour calculer les points d'intersection. O'Brien et Manocha [Obr00] calculent l'intersection en effectuant l'intersection des plans porteurs des faces puis en prenant la restriction aux faces. Ils sont donc obligés de distinguer deux cas : le cas où les deux points d'intersection appartiennent aux segments d'une même face et le cas où ils sont portés par des segments de faces différentes (Figure 4).

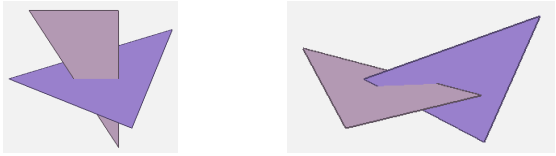


Fig. 4. Deux cas d'intersection face/face.

Une autre solution consiste à calculer les intersections de tous les segments d'une face avec l'autre face. L'intersection droite plan est tout d'abord calculée. L'équation de la droite étant écrite sous forme barycentrique, la restriction au segment s'effectue en vérifiant que la valeur du paramètre est dans l'intervalle $[0,1]$. Ensuite il ne reste plus qu'à vérifier l'appartenance du point d'intersection à la face, ce qui se fait aisément en comparant l'aire de la face avec la somme des aires des triangles formés par le point d'intersection et les sommets de la face. C'est cette seconde méthode que nous avons choisi d'implémenter.

3.3. Tri des points d'intersection et évaluation de la courbe polygonale d'intersection.

Le tri des points est facilement réalisé grâce à la structure du point d'intersection qui stocke les coordonnées du point, les faces de chaque objet à l'origine de ce point et l'arête qui porte ce point. Ainsi les extrémités des segments d'intersection correspondent aux points d'intersection contenant deux faces identiques. Ensuite, les segments sont reliés entre eux en utilisant la structure winged edge [Bau72].

4. La variante de voisinage.

Le calcul de l'intersection au niveau initial pour cet algorithme se fait à l'aide de l'algorithme naturel. Les faces en intersection des deux surfaces sont donc connues par la suite. La Figure 5 représente la courbe d'intersection et les faces en intersection au niveau initial.

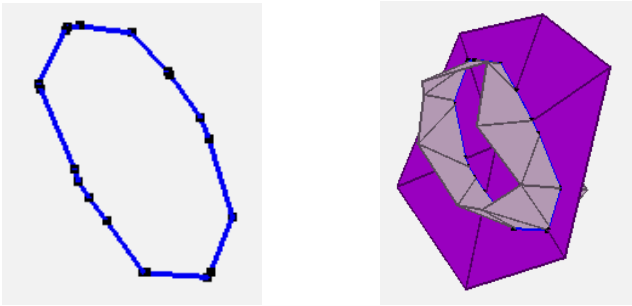


Fig. 5. Courbe d'intersection au premier niveau avec les faces en intersection.

Soit I_i l'ensemble des faces participant à l'intersection de la surface S_i. La première étape de cet algorithme consiste à récupérer le 1-voisinage $\mathcal{V}_1(I_i)$. Rappelons que ce 1-voisinage contient l'ensemble I_i ainsi que toutes les faces voisines aux faces de I_i par un sommet. Les ensembles $\mathcal{V}_1(I_1)$ et $\mathcal{V}_1(I_2)$ sont représentés par la Figure 6.a.

Puis les ensembles $\mathcal{V}_1(I_i)$ sont subdivisés en appliquant le principe de Loop. Lors de cette étape de raffinement, seul le 1-voisinage des ensembles $\mathcal{V}_1(I_i)$ est conservé. En effet, pour obtenir une subdivision correcte du 1-voisinage en entier, le 2-voisinage serait nécessaire. Sur la Figure 6.a, les faces obtenues par subdivision de I_i sont représentées en foncé et celles du 1-voisinage en clair sur la Figure 6.b.

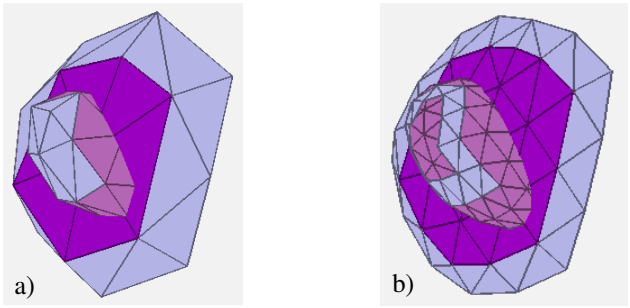


Fig. 6. Faces en intersection. a) 1-voisinage $\mathcal{V}_1(I_i)$. b) 1-voisinage de la subdivision $\mathcal{W}_1(I_i)$.

En réalité, on conserve tout de même le 2-voisinage $\mathcal{V}_2(I_i)$ de I_i afin de calculer correctement le 2-voisinage $\mathcal{W}_2(I_i)$ de l'ensemble des sous faces des faces I_i obtenu par la subdivision de Loop. En effet, dans certains cas, il est nécessaire d'avoir les faces de $\mathcal{W}_2(I_i)$ en mémoire pour récupérer le voisinage en entier au niveau suivant. Ensuite, l'algorithme naturel est à nouveau utilisé pour tester les intersections entre $\mathcal{W}_1(I_1)$ et $\mathcal{W}_1(I_2)$. Les courbes d'intersection obtenues au niveau suivant peuvent être tracées (Figure 7).

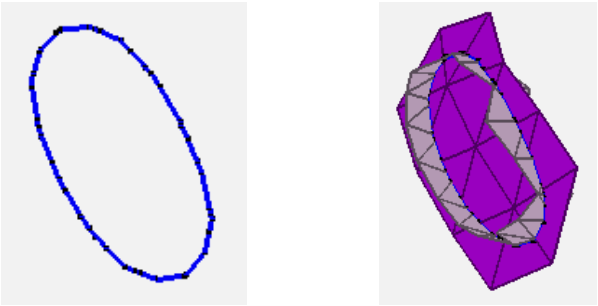


Fig. 7. Intersection au niveau 1. Gauche : la courbe obtenue. Droite : les faces participant à l'intersection.

Pour récupérer les courbes d'intersection au niveau de subdivision x donné, il suffit d'appliquer x fois ce processus. La Figure 8 montre les résultats obtenus au niveau 2. De gauche à droite, on a : les 1-voisinages $\mathcal{V}_1(I_i)$, les 1-voisinages de la subdivision $\mathcal{W}_1(I_i)$, les faces de $\mathcal{W}_1(I_i)$ en intersection et la courbe d'intersection.

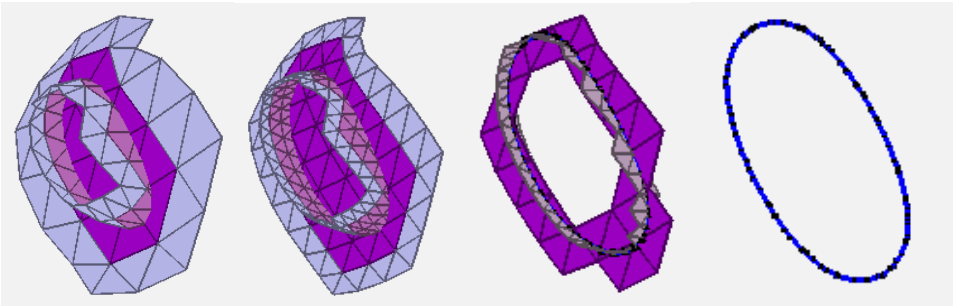


Fig. 8. Voisinage, subdivision et courbe d'intersection au niveau 2.

Cet algorithme réduit de manière significative le nombre de faces intervenant dans le calcul d'intersection, il est par conséquent beaucoup plus rapide que l'algorithme naturel.

5. Algorithme utilisant un graphe biparti

Cet algorithme repose sur l'utilisation d'un graphe biparti, il permet de réduire le nombre d'intersections à tester. Dans l'exemple présenté sur la Figure 9, on construit le graphe d'intersection d'une bande (40 faces) avec un lapin (694 faces).

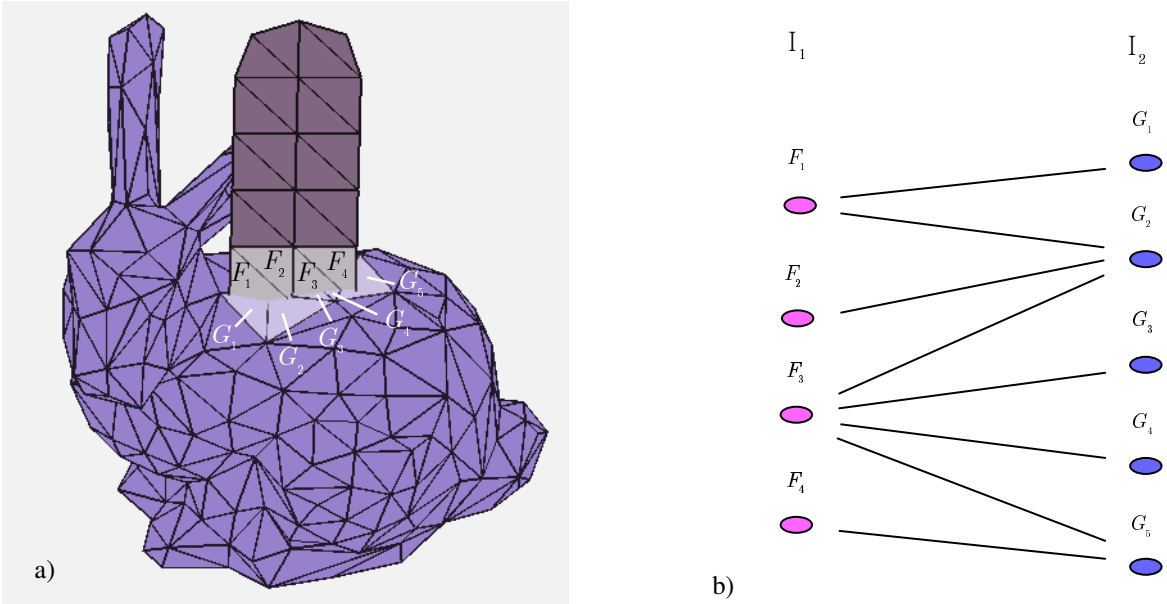


Fig. 9. Exemple de construction d'un graphe où F_i et G_i sont respectivement des éléments de I_1 et de I_2.

Les faces intersectantes de la bande sont appelées de gauche à droite F_1 à F_4 et celle du lapin G_1 à G_5 (Figure 9.a). Les nœuds de la première partie du graphe, disposés en colonne représentent les faces de l'objet 1 (la bande) et ceux de la seconde partie représentent les faces de l'objet 2 (le lapin). Les faces de l'objet 1 qui intersectent l'objet 2 sont reliées par des arêtes (Figure 9.b).
Une fois ce graphe constitué, les voisinages des faces des 2 objets sont ajoutés au graphe. L'intersection de chaque face de $\mathcal{V}_1(I_1)$ avec toutes les faces de $\mathcal{V}_1(I_2)$ n'est plus testée, l'idée de l'algorithme consiste à calculer seulement les intersections entre des sous-groupes de ces ensembles. En effet, à chaque face intersectante F_i de la surface S_1, le graphe d'intersection (graphe biparti) fait correspondre certaines faces G_i de la surface S_2 en intersection avec cette face, on va donc calculer uniquement les intersections entre $\mathcal{V}_1(F_i)$ et $\mathcal{V}_1(G_k)$ étant reliés par une arête.

L'entrée de cet algorithme est un graphe biparti $G^0(I_1^0, I_2^0, E^0)$ où I_1^0 et I_2^0 sont les ensembles de sommets du graphe (faces intersectantes) et E^0 l'ensemble des arêtes (couples de faces en intersection). Sa sortie est également un graphe biparti $G^1(I_1^1, I_2^1, E^1)$ où I_1^1 et I_2^1 sont les ensembles de sommets (faces intersectantes) et E^1 l'ensemble des arêtes. Ce graphe représente le graphe biparti au niveau suivant de subdivision (couples de faces en intersection).
L'algorithme opère en 4 étapes consécutives de la façon suivante :

1. Pour chaque nœud du graphe, les 1-voisinages des faces F_i de l'objet 1 et des faces G_k de l'objet 2 sont récupérés, ils sont respectivement notés $\mathcal{V}_1(F_i)$ et $\mathcal{V}_1(G_k)$.
2. Les $\mathcal{V}_1(F_i)$, $F_i \in I_1$ et $\mathcal{V}_1(G_k)$, $G_k \in I_2$ sont ensuite subdivisés partiellement de manière à conserver uniquement le 1-voisinage de la subdivision de F_i et de G_k et notés $\mathcal{W}_1(F_i)$ et de $\mathcal{W}_1(G_k)$.
3. L'algorithme naturel est utilisé pour déterminer les couples de faces intersectantes entre $\mathcal{W}_1(F_i)$ et $\mathcal{W}_1(G_k)$ (cf. section 3).
4. Le nouveau graphe biparti $G^1(I_1^1, I_2^1, E^1)$ est construit à l'aide de ces nouveaux couples de faces intersectantes de la même manière qu'à la Figure 9.

Ci-dessous, nous appliquons l'algorithme précédent sur un exemple simple. Pour réduire le nombre de sommets du graphe, cette fois on considère l'intersection du lapin (694 faces) avec une plus petite bande (10 faces).
Les Figures 10 à 13 illustrent successivement la construction du graphe biparti initial, l'étape 1 de récupération du voisinage, l'étape 2 de subdivision partielle, l'étape 3 de détermination des couples de faces en intersection et l'étape 4 de construction du nouveau graphe d'intersection.

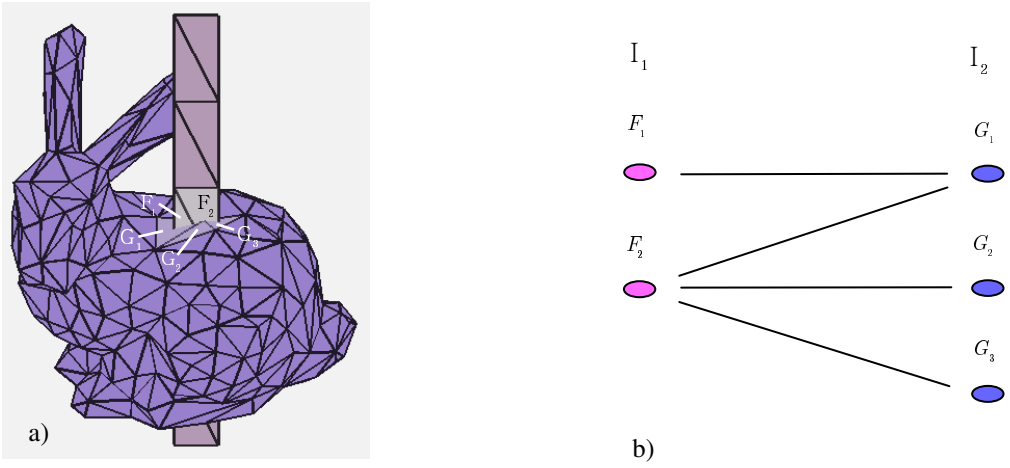


Fig. 10. Construction du graphe biparti initial

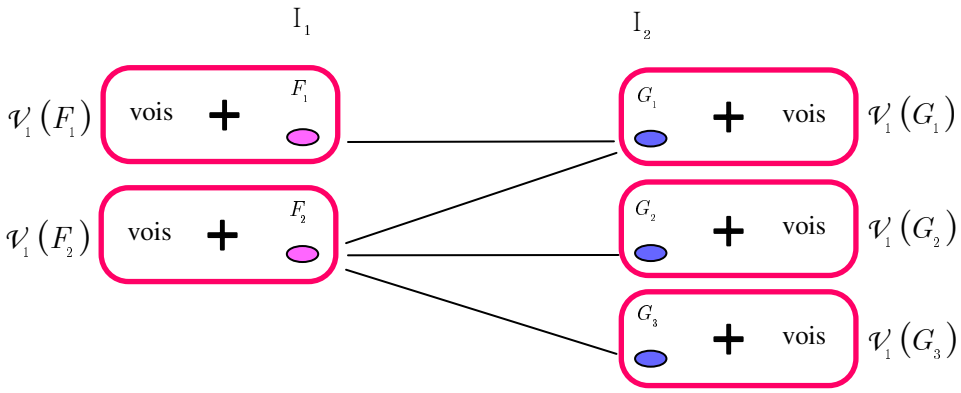


Fig. 11. Etape 1 : Récupération des 1-voisinages $\mathcal{V}_1(F_i)$ et $\mathcal{V}_1(G_k)$.

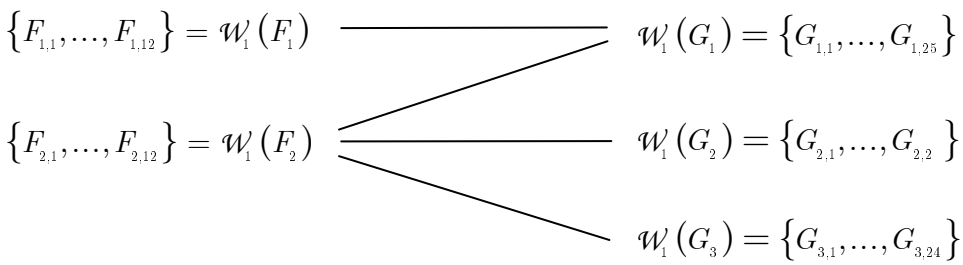


Fig. 12. Etape 2 : Obtention des $\mathcal{W}_1(F_i)$ et de $\mathcal{W}_1(G_k)$ par subdivision partielle de $\mathcal{V}_1(F_i)$ et $\mathcal{V}_1(G_k)$.

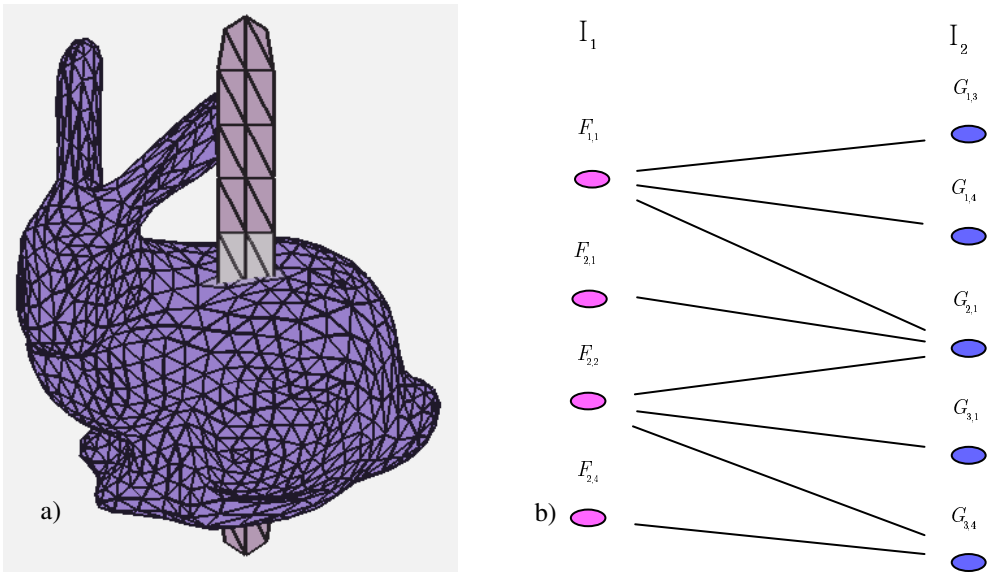


Fig. 13. Construction du nouveau graphe biparti à l'aide des couples de faces en intersection de $\mathcal{W}_1(F_i)$ et de $\mathcal{W}_1(G_k)$.

Cet algorithme est plus efficace puisqu'il se base sur un graphe biparti donnant un minimum de faces intersectantes à chaque subdivision. Actuellement, l'implémentation est en cours afin de confirmer cette affirmation. En effet, l'idée d'ordonner les sommets de I_1 et de I_2 est envisagée afin d'éviter de générer plusieurs fois les même faces en calculant les 1-voisinages de ces sommets.

6. Conclusion

Dans cet article, nous avons décrit trois algorithmes permettent de calculer les courbes d'intersection entre deux objets modélisés par des surfaces de subdivision. L'algorithme naturel est une variante peu optimisée qui peut

être intéressante à utiliser lorsque les surfaces des objets à intersecter ont un nombre réduit de faces. Les deux autres algorithmes proposés reposent sur les notions de 1-voisinage et de graphe biparti. Ils permettent de calculer les courbes d'intersection entre deux objets plus rapidement qu'avec l'algorithme naturel notamment lorsque le nombre de faces en présence est très élevé. Il reste maintenant à intégrer cette amélioration dans le cadre des opérations booléennes. L'implémentation de ces deux algorithmes est en cours. Une étude comparative sera envisagée par la suite avec d'autres algorithmes existants.

Références

[Abd96] K. Abdel-Malek, H. J. Yeh. "Determining intersection curves between surfaces of two solids". Computer Aided Design, vol. 28-6/7, pp 539-549, 1996.

[Baj88] C. L. Bajaj, C. M. Hoffmann, J. E. Hopcroft, R. E. Lynch. "Tracing surface intersections". Computer Aided Geometric Design, vol. 5, pp 285-307, 1988.

[Bar87] R. E. Barnhill, G. Farin, M. Jordan, B. R. Piper. "Surface/surface intersection". Computer Aided Geometric Design, vol. 4-3, pp 3-16, 1987.

[Bau72] Bruce G. Baumgart, "Winged edge polyhedron representation", Technical Report CS-TR-72-320, pp 5, 1972.

[Boe91] E. Boender. "A survey of intersection algorithms for curved surfaces". Computer & Graphics, vol. 15-1, pp 99-115, 1991.

[Cat78] E. Catmull, J. Clark. "Recursively generated B-spline surfaces on arbitrary topological meshes". Computer Aided Design, vol. 9-6, pp 350-355, 1978.

[Cha87] V. Chandru, B. S. Kochar. "Geometric Modeling: Algorithms and NEW Trends". Chapter Analytic Techniques for Geometric Intersection Problems, pp 305-318, SIAM, Philadelphia, PA, 1987.

[Doo78] D. Doo, M. Sabin. "Behaviour of recursive subdivision surfaces near extraordinary points". Computer Aided Design, vol. 9-6, pp 356-360, 1978.

[Kob00] L. Kobbelt. "Sqrt(3)-Subdivision". Computer Graphics Proceedings, Annual Conference Series, pp. 103-112, July 2000.

[Kri94] S. Krishnan, A. Narkhede, D. Manocha. "Boole: A System to Compute Boolean Combinations of Sculptured Solids". Technical Report, Department of Computer Science, University of North California, 1994.

[Loo87] C. Loop. "Smooth Subdivision Surfaces Based on Triangles". Master's thesis, University of Utah, Department of Mathematics, 1987.

[Obr00] D. A. O'Brien, D. Manocha. "Calculating Intersection Curve Approximations for Subdivision Surfaces", 2001. http://www.cs.unc.edu/~obrien/courses/comp258/project.html

[Pat93] N. M. Patrikalakis. "Surface-to-surface intersections". IEEE Computer Graphics & Applications, vol. 13-1, pp 89-95, January 1993.

[Vel00] L. Velho, D. Zorin. "4-8 Subdivision". Computer Aided Geometric Design, volume 18-5, pp 397-427, 2000.

[Zor00] D. Zorin. "Subdivision Zoo". SIGGRAPH 2000 Course Notes, Subdivision for Modeling and Animation, Chap. 4, pp 65-98, 2000.