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).

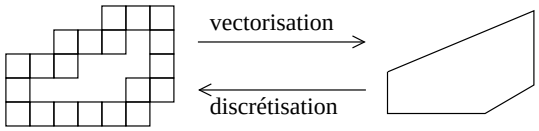

FIG. 1 – Courbe discrète de départ, courbe réelle reconstruite et courbe réelle discrétisée.

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.

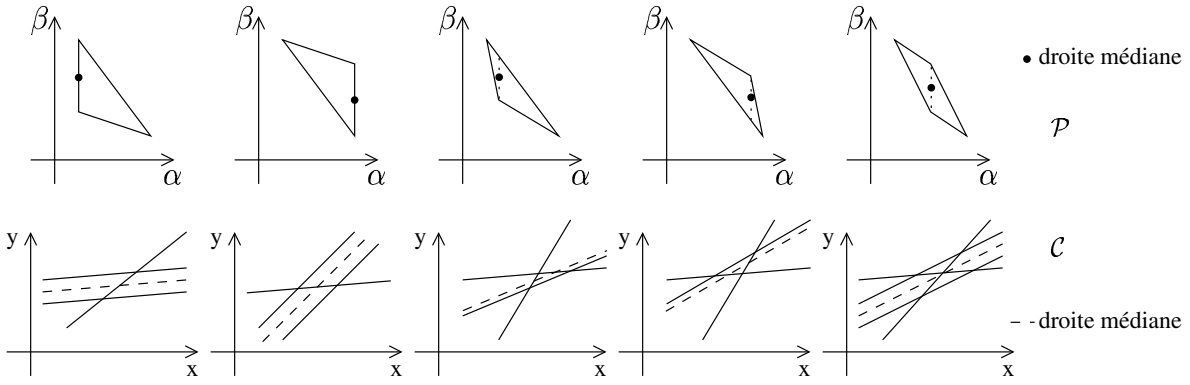

FIG. 2 – Les cinq formes possibles de l'ensemble de solutions et la solution choisie dans chaque cas.

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.

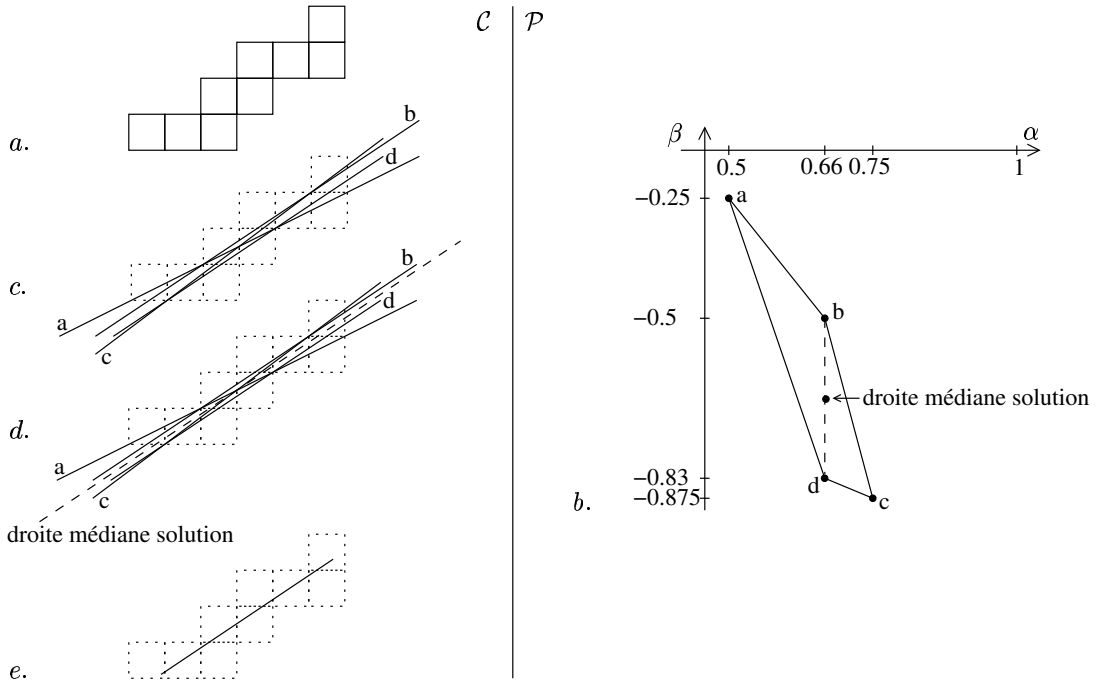

FIG. 3 – Résultat de l'algorithme de Vittone sur un segment.

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.

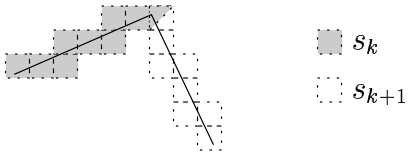

FIG. 4 – Intersection triviale.

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*.

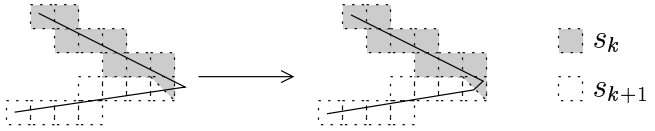

FIG. 5 – Intersection hors du pixel commun aux deux segments ⇒ ajout d'un joint.

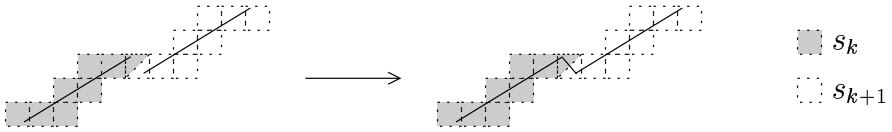

FIG. 6 – Intersection hors du pixel commun aux deux segments ou aucune intersection ⇒ ajout d'un joint.

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.

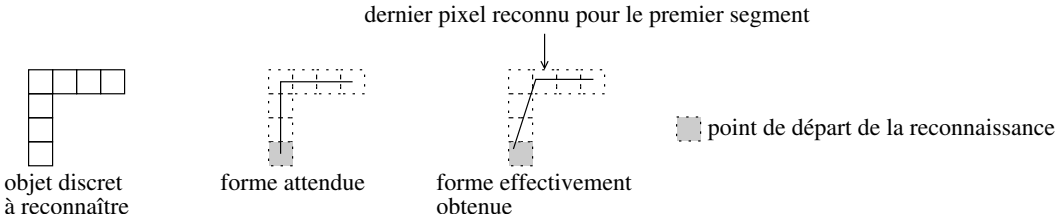

FIG. 7 – Problème de la reconnaissance maximale.

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.

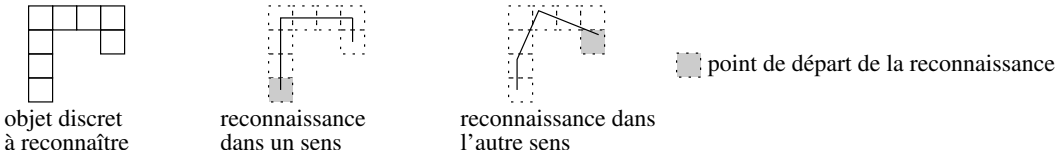

FIG. 8 – Problème du retrait systématique du dernier pixel reconnu.

• **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.

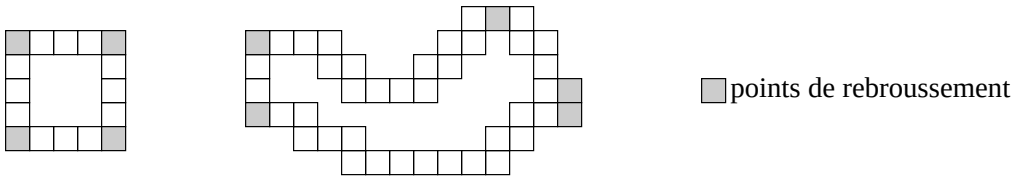

FIG. 9 – Exemple de points de rebroussement sur deux courbes.

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).

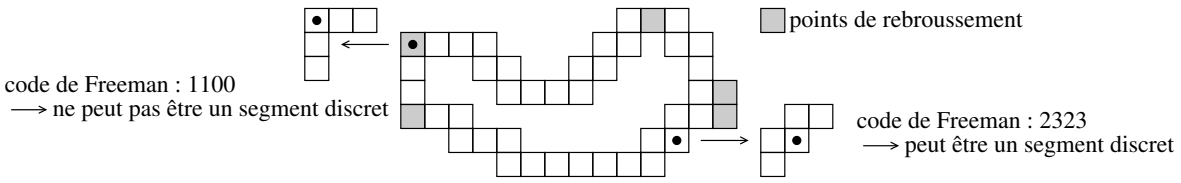

FIG. 10 – Détermination des points de rebroussement.

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.

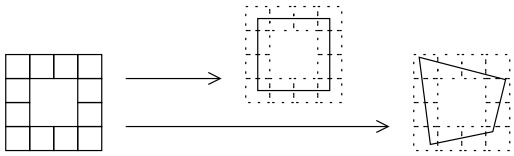

FIG. 11 – Plusieurs courbes réelles correspondent à une même courbe discrète.

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).

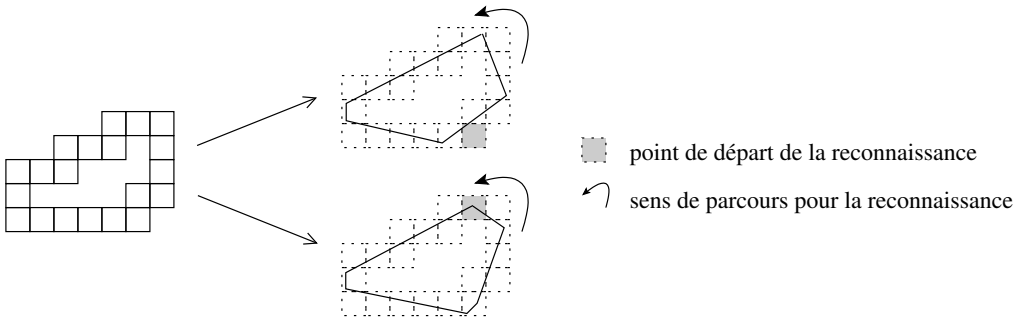

FIG. 12 – La même courbe discrète, reconnue en partant de deux points différents.

• **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.

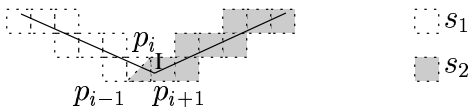

FIG. 13 – Intersection dans un pixel voisin du pixel commun aux deux segments discrets.

• **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.

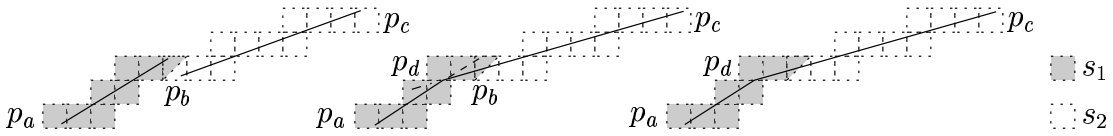

FIG. 14 – Elimination d'un joint grâce à une seconde reconnaissance.

• **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 $[p_{b-1}, p_{b+1}]$.

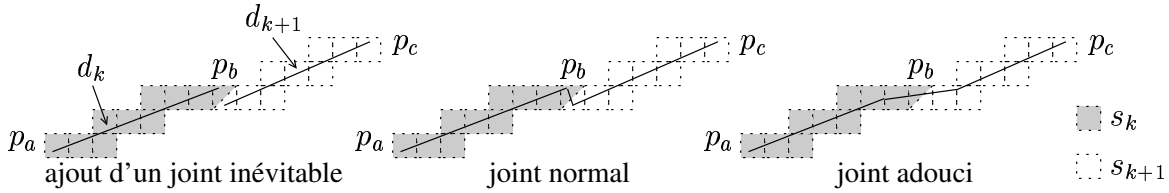

FIG. 15 – Cas d'un ajout de joint inévitable.

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.

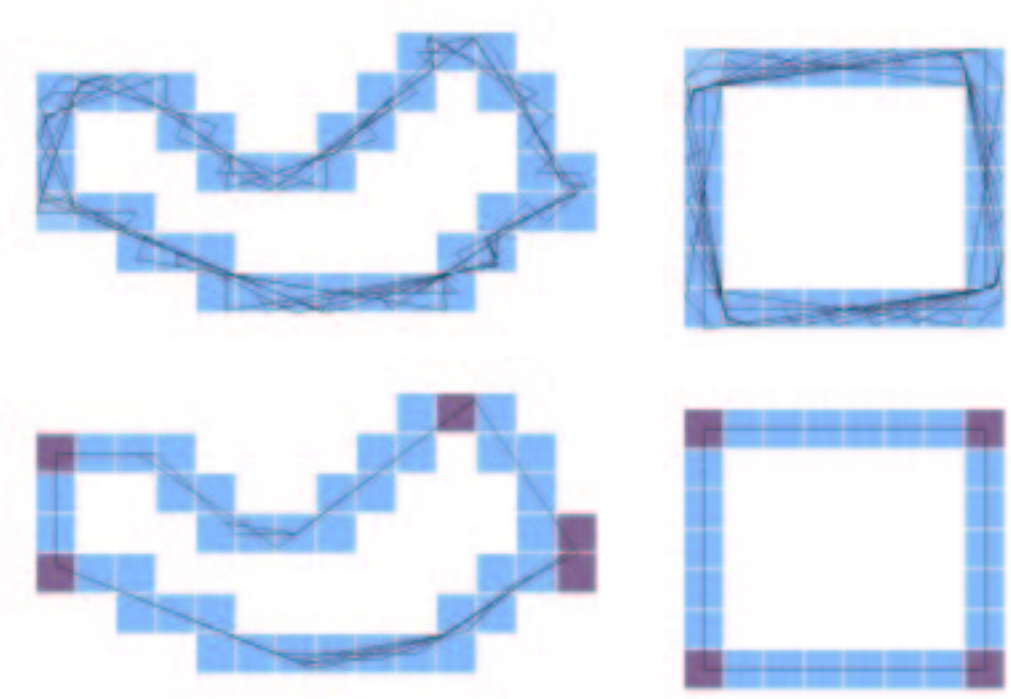

FIG. 16 – En haut, résultats obtenus avec la première version de l'algorithme, en bas, avec la version actuelle. En foncé, les points de rebroussement.

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.