Opérateur matériel de chaînage de contours temps réel et flot de données

Philippe LAMATY **Didier DEMIGNY** **Gérard BARBIER**

Equipe Traitement d'Images et du Signal
ENSEA/UCP; 6, av. du Ponceau; 95014 CERGY-PONTOISE Cedex; France
demigny@ensea.fr

Service d'Etudes Techniques Electroniques
Aérospatiale Division Engins Tactiques; 9 à 18, rue Béranger; 92322 CHATILLON Cedex; France

RESUME

Nous proposons un algorithme qui extrait des chaînes de segments d'une image de type contour en vue d'une intégration matérielle temps réel et flot de données. Cet algorithme se décompose en trois phases qui sont : l'extraction des chaînes de segments, l'approximation polygonale de chaque chaîne, l'élimination des chaînes inférieures à une tolérance. Les deux dernières étapes réduisent pour l'une le nombre de segments de chaque chaîne, et pour l'autre le nombre de chaînes. L'algorithme obtient les meilleurs résultats de chaînage pour des contours d'épaisseur un presque partout.

ABSTRACT

We propose an algorithm which extracts lines segments chains of edge image in order to design a real-time and data-flow hardware architecture. The algorithm can be broken down into three steps : a lines segments chains extraction step, a polygonal approximation step, and an elimination of little chain smaller than a threshold. The two last phases reduce for the first one, the number of lines segments of each chain, and for the second one the number of chains. An edge map of one pixel thickness obtains the best results of line segments chaining.

1 Introduction

Le chaînage construit une approximation polygonale des contours d'une image binaire. Certaines méthodes d'approximation consistent à détecter un point de contour et à parcourir le contour pour en extraire ses attributs [FRE78]. D'autres approches permettent d'extraire des chaînes de pixels correspondant aux contours de l'image et effectuent ensuite une fusion des pixels voisins [CHA81] [GIR87]. Quelques méthodes ont, par ailleurs, été appliquées sur des machines à base de transputers [LEG92].

Nous proposons une solution pouvant être intégrée dans un composant de type ASIC avec un fonctionnement flot de données. En outre, au lieu d'extraire des chaînes de pixels [GIR87], l'algorithme extrait des chaînes de segments pour ensuite faire une approximation polygonale [PAV74] [WAL84] [COC96]. Ceci améliore les résultats de cette dernière, et réduit la taille des ressources.

2 Chaînage de contours

L'algorithme de chaînage de contours présenté consiste dans un premier temps à extraire d'une image binaire les segments de droite correspondant aux contours. Les pixels de l'image traitée (contour ou binaire) se définissent par deux types d'informations : contour ou fond. L'étape suivante regroupe les segments connexes dans un même ensemble dit chaîne de segments (Figure 1). Le traitement suivant, nommé polygonisation, réalise une approximation polygonale des chaînes. Un paramètre règle l'amplitude de l'approximation qui réduit le nombre d'informations de chaque chaîne. Un algorithme de type fusion/découpage est généralement utilisé [GAR92]. La dernière étape consiste à éliminer les chaînes dont la taille est inférieure à un seuil.

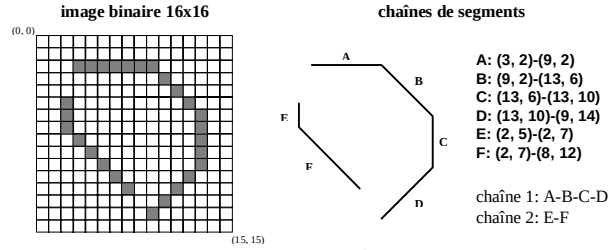

Figure 1: Exemple de chaînage de contours

La simplicité de l'algorithme présenté (Figure 2) permet son intégration dans un système temps réel, flot de données de type composant spécifique (ASIC). Des études avaient déjà montré la faisabilité temps réel et/ou flot de données [GIR87] [CHR94] du chaînage de contours.

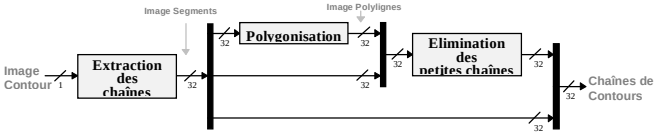

Figure 2: Synoptique du chaînage de contours

Dans la suite de cet article, nous décrirons les différentes étapes réalisant l'opération de chaînage de contours.

2.1 Extraction des chaînes de segments

Deux étapes sont nécessaires à la réalisation de cette opération. La première est l'extraction des segments de

l'image contour et la suivante le regroupement des segments connexes appelé chaînage.

- ***Extraction des segments***

Les contours sont supposés fins (épaisseur un presque partout), donc un automate 3x3 (Figure 3) est suffisant pour l'extraction. Des contours plus épais ne dégradent que faiblement les performances de l'automate.

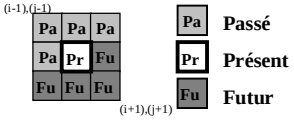

Figure 3: Automate 3x3

L'automate contient trois types d'informations : passé, présent, futur. Le passé représente les pixels déjà traités, le présent celui en cours de traitement, le futur les pixels non traités. Chaque élément du masque contient l'information contour ou fond de l'image, le passé et le présent contiennent en plus une indication sur l'origine du segment. Cette information est de type coordonnées (x, y). Pour s'affranchir des problèmes d'effets de bord, une bordure virtuelle de un pixel de type fond est placée autour de l'image.

L'automate exécute des traitements si et seulement si le présent est de type contour. Ceci réduit le nombre de cas possibles à 256. A chacun de ces cas est associée une action, qui peut être de trois types : début de segment(s), continuité de trait, fin de segment(s).

- **début de segment(s)** :

  se définit par tous les passés de type fond ou détection d'une intersection (Figure 4 (a) et (b)).

  Les coordonnées (*x*, *y*) du présent sont affectées avec la position du masque dans l'image.

- **continuité de trait** :

  se définit par trois éléments de type contour alignés, le présent faisant partie des trois. Dans le cas des deux directions à ±45°, les autres éléments doivent être de type fond. Pour 0° et 90°, seuls les quatre coins peuvent être de type fond ou contour, les autres doivent être de type fond (Figure 4 (c)).

  La position du passé aligné est recopiée dans le présent.

- **fin de segment(s)** :

  se définit par tous les futurs de type fond ou détection d'une intersection (Figure 4 (b) et (d)).

  La détection d'une fin de segment(s) entraîne la création d'un ou plusieurs segments. Le ou les passés permettent de connaître l'origine de départ des segments. Le présent indique leur fin, et il est affecté à la position du masque dans l'image.

Une intersection (Figure 4 (b)) est couramment définie par au moins deux passés, ou au moins deux futurs, ou encore un passé et un futur non aligné de type contour (rupture de continuité). Le présent est pour tous les cas de type contour.

La création d'un segment se décompose en deux opérations : - ① - *détection d'un début de segment.*
- ② - *détection d'une fin de segment(s).*

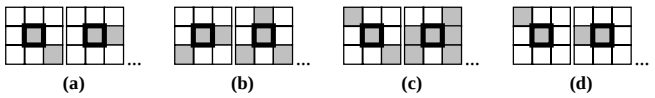

Figure 4: Exemples de (a) début, (b) intersection, (c) continuité, (d) fin de segments

L'automate peut créer un ou deux segments au maximum (Figure 5). Or si plus de deux passés sont de type contour, l'automate devrait créer autant de segments. Mais les résultats d'extraction obtenus avec cette limitation sont plus que satisfaisants, puisque la reconstruction d'une image discrète à partir des chaînes de segments donne une image strictement identique à celle de départ.

Figure 5: Création de segment(s), (a) 1 segment, (b) 2 segments

Pour la réalisation (Figure 6), une look-up table (ou une logique câblée) peut être utilisée pour coder les actions à effectuer. Une information sur les passés à prendre en compte est ajoutée à chaque action. Une machine d'états décode les actions, initialise l'information du pixel courant (présent), et indique si zéro, un ou deux segments ont été créés. Un système à base de files d'attente (LIFO) peut être utilisé pour faire circuler les informations segments (A, B) de la machine d'états sur un seul bus.

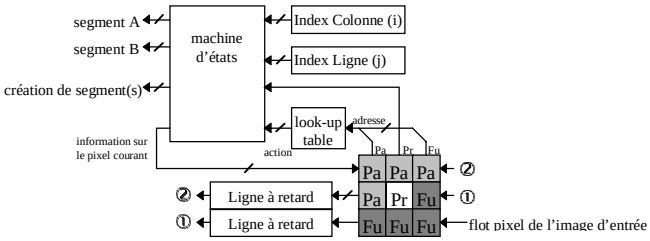

Figure 6: Architecture de l'extraction des segments

- ***Chaînage des segments connexes***

Comme pour un étiquetage de régions[ROS66], le chaînage attribue à chaque chaîne de segments un numéro d'identification (Figure 1). Deux étapes sont nécessaires. La première est effectuée pendant l'extraction des segments. Pour cela, l'automate associe aux actions les traitements suivants : début ou intersection → nouvelle chaîne, continuité de trait ou rupture de continuité → recopie du numéro d'identification, fin de segments → pas d'opération. Un problème dû au balayage séquentiel de l'image décompose des contours en plusieurs chaînes. Une machine d'états utilisant une table d'équivalence lie ces chaînes en une seule. Cette table est affectée par l'automate lors de la création de deux segments (Figure 5 (b)) avec tous les futurs de type fond. Ceci oblige à mémoriser les informations des chaînes. Or, dans une chaîne de segments, des données sont communes. Une mise en forme (Figure 7) permet de réduire de la (moitié - 1) le nombre de coordonnées.

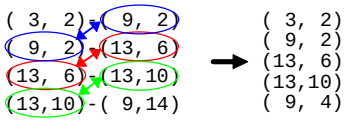

Figure 7: Exemple de mise en forme

Lorsqu'une chaîne est complète et que ses données ont été mises en forme, ses coordonnées sont émises vers l'algorithme d'approximation polygonale ou d'élimination des petites chaînes. La chaîne émise est précédée par le numéro d'identification et le nombre de coordonnées.

2.2 Approximation polygonale

Elle réduit le nombre d'informations de chaque chaîne de segments en fonction d'une tolérance [PAV74] [KUR81] [WALL84]. L'approximation polygonale est aussi appelée polygonisation (Figure 8).

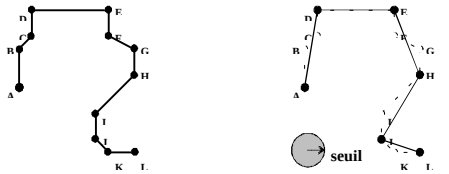

Figure 8: Exemple de polygonisation, (a) référence, (b) référence + polylignes

La méthode d'approximation la plus utilisée est le découpage récursif [PAV74]. Mais des algorithmes d'approximation itératifs [KUR81] [WALL84] sont préférés aux méthodes récursives, car plus simple à intégrer. En général, les méthodes itératives sont moins performantes que les méthodes de découpage récursif. De plus, des travaux ont montré qu'une recherche des points de courbure [ATT54] permet d'obtenir un résultat « fidèle » à l'image (Figure 9).

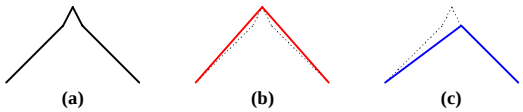

Figure 9: (a)chaîne d'origine, (b)approx. recherchée, (c)approx. détériorante

Pour valider le choix de l'algorithme de polygonisation, une analyse statistique a été effectuée entre différentes méthodes. A savoir, une récursive [PAV74] (**rec**) choisie comme référence, et deux itératives [WAL84] (**wal**) et (**opn**). La dernière méthode fait intervenir une recherche des points de courbure suivie de deux approximations itératives, la première entre points de courbure, et la seconde sur toute la chaîne. Les calculs sont effectués sur des nombres entiers, sauf pour le découpage récursif qui sert de référence et dont les opérations arithmétiques se font sur des flottants.

Les courbes (Figure 10) et (Figure 11) montrent l'évolution moyenne de la dégradation des contours et du nombre de coordonnées en fonction du seuil de polygonisation.

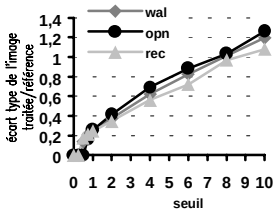

Figure 10: Déformation de l'image

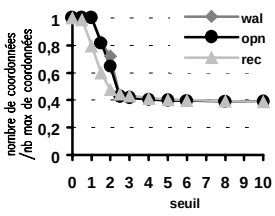

Figure 11: Evolution du nombre de coordonnées

Les deux remarques les plus importantes sont que l'évolution du nombre de coordonnées devient minime pour un seuil supérieur à deux, et que les versions (**wal**) et (**opn**) ont une déformation proche d'un découpage récursif (Figure 12).

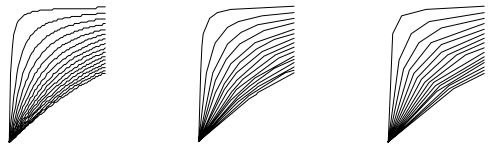

Figure 12: Exemple de déformation avec un seuil de 1,5; (a) image référence, (b) image traitée version rec, (c) image traitée version opn

- ***Points de courbure***

Ils correspondent aux points où le contour change de directions (Figure 13). Les contours peuvent être décrits par huit directions (Figure 13 (c)) : droite (⇨), haut-droite (⇗), haut (⇧), haut-gauche (⇖), gauche (⇦), bas-gauche (⇙), bas (⇩), bas-droite (⇘). La direction d'un segment AB est obtenue par le calcul des signes et des distances nulles de $x_A - x_B$ et $y_A - y_B$. Les directions ⇨⇧⇦⇩ sont neutres dans le cas où le signe de la distance non-nulle est identique à celui de la distance du segment précédent. Elles apportent une information dans le cas où la distance perpendiculaire des segments précédent et suivant sont de signes opposés (ex: ⇗⇨⇩, ⇘⇩⇖, etc.).

Un point de la chaîne est un point de courbure si les segments connexes en ce point ont des directions différentes. Si un segment a une direction neutre significative alors ses extrémités sont des points de courbure. Ceci permet d'obtenir les mêmes points quel que soit le sens de parcours.

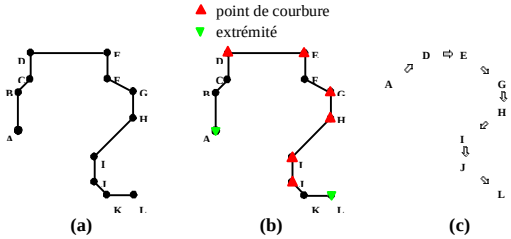

Figure 13: (a) image source, (b) points de courbure, (c) orientations

La simplicité de l'algorithme de recherche des points de courbure permet une intégration flot de données (Figure 14).

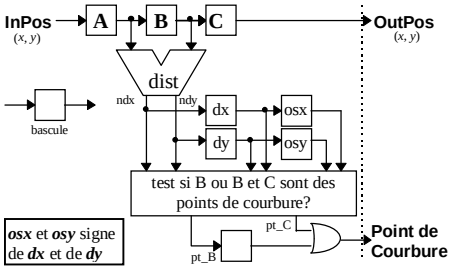

Figure 14: Architecture de la recherche des points de courbure

- ***Approximation itérative***

Les algorithmes consistent à fusionner les points tant que le cumul des erreurs est inférieur à un seuil. Une architecture de l'algorithme de [WAL84] est donnée (Figure 15).

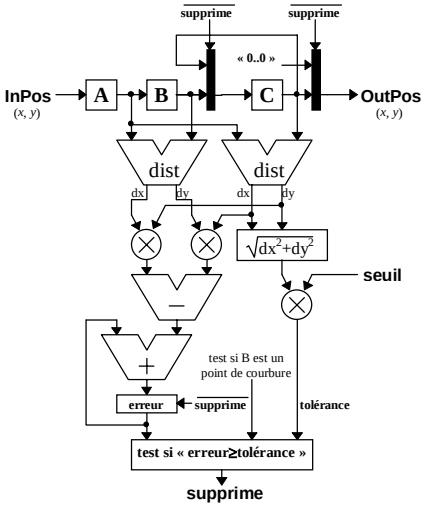

Figure 15: Architecture de l'approximation polygonale

Deux problèmes apparaissent avec les algorithmes d'approximation itérative. Ils sont souvent dus au cumul de l'erreur. Le premier est la situation des points de courbure au-delà des points souhaités et le deuxième est lié au sens de parcours de la chaîne. Mais l'utilisation des chaînes de segments à la place des chaînes de pixels atténue fortement les problèmes de ce genre.

2.3 Elimination des petites chaînes

Elle permet d'éliminer les chaînes dont la taille est inférieure à un seuil. Cet opérateur somme la taille des segments d'une chaîne et compare la taille obtenue à un seuil. Si elle est inférieure, il supprime la chaîne de la liste. Ceci permet, par exemple, d'éliminer les pixels isolés qui n'apportent aucune information dans certaines applications (exemple: recalage d'images, reconnaissance de forme).

La courbe (Figure 16) montre l'évolution moyenne du nombre de chaînes en fonction du seuil d'élimination.

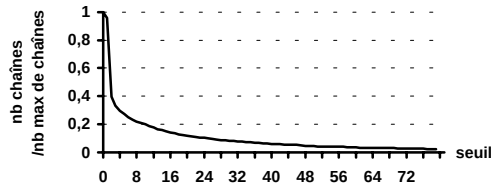

Figure 16: Evolution du nombre de chaînes

L'architecture, déduite de l'algorithme, (Figure 17) utilise une file d'attente pour temporiser les données. Si la chaîne doit être supprimée, ses données sont écrasées par les données de la chaîne suivante. Sinon, elle est émise sur le bus de sortie.

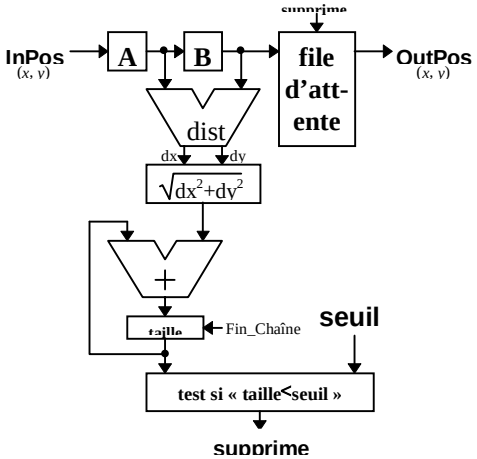

Figure 17: Architecture de l'élimination des petites chaînes

L'opérateur « dist » utilisé par les algorithmes d'approximation polygonale (Figure 14, Figure 15) et d'élimination des petites chaînes (Figure 17) calcule les distances sur l'axe des abscisses et des ordonnées du segment défini par ses extrémités.

3 Conclusion

L'exemple (Figure 18) montre le type de résultat obtenu à la sortie des différents blocs de l'opérateur de chaînage de contours. Les critères définis par [GAR92] ont été utilisés pour vérifier les résultats de l'approximation polygonale. Par ailleurs, les statistiques réalisées sont une moyenne de résultats obtenus sur une dizaine d'images du GDR 134.

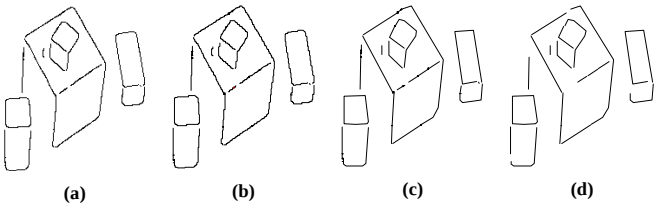

Figure 18: (a) image source, (b) chaînes de segments, (c) polylignes (seuil 1,5), (d) grandes polylignes (seuil 10)

Elles ont permis de déterminer la taille des différentes ressources utiles au bon fonctionnement de l'algorithme. Par exemple, la gestion de 4000 chaînes, soit environ 32.000 segments, utilise une mémoire 180 Kbits.

Pour éviter une trame de retard due à l'identification des chaînes lors de l'extraction, l'opérateur utilise les pixels de type fond pour réaliser le chaînage des segments. Le problème de cette solution provient des données qui peuvent rester dans la mémoire de mise en forme en fin d'image.

L'opérateur de chaînage de contours réalisé occupe une surface d'environ 0,6 cm² en 0,6 µm, fonctionne à 40Mhz et peut traiter des images de 1024x1024 pixels.

4 Références

[ATT54] Attneave *"Some informational aspects of visual perception"*, Psychological Review, vol. 61, pp. 183-193, 1954

[ROS66] Rosenfeld A. and Pfaltz J.L. *"Sequential operations in digital picture processing"*, Journal of ACM, vol. 13, n° 4, pp. 471-494, 1966

[PAV74] Pavlidis T. and Horowitz S.L. *"Segmentation of planes curves"*, IEEE Transaction on Computers, vol. 23,n° 8, pp. 860-870, 1974

[FRE78] Freeman H. *"Computer processing of line drawing images"*, Computing Surveys, vol 6, pp. 57-97, 1978

[CHA81] Chakravarty I. *"A single-pass, chain generating algorithm for region boundaries"*, Computer Graphics and Image Processing, vol 15, pp. 182-193, 1981

[KUR81] Kurozumi Y. and Davis W. *"Polygonal approximation by the minmax method"*, Computer Graphics and Image Processing, vol. 19, pp. 248-264, 1981

[WAL84] Wall K. and Danielsson P.E. *"A fast sequential method for polygonal approximation of digitized curves"*, CVGIP, vol. 28, pp. 220-227, 1984

[GIR87] Giraudon G. *"*Chaînage efficace de contours*"*, Rapport de recherche n°605 INRIA, Mars 1987

[GAR92] Garnesson P. and Giraudon G. *"L'approximation polygonale, bilans et perspectives"*, Rapport de recherche n°1621 INRIA, Février 1992

[LEG92] Legrand P. and Derutin J.P. *"Contour based image segmentation process on a parallel vision machine"*, Procedings of MVA'92-IAPR Workshop on Machine Vision Applications, Décembre 1992

[CHR94] Christiaen P. and Loosfelt P. *"Chaînage de contours"*, Conférence Adéquation Algorithmes Architectures, pp.21-28, Janvier 1994

[COC96] Cocquerez J.P. and Philipp S. *"Analyse d'images: filtrage et segmentation"*, pp.54-61, 1996