Significant points in a digital contour: characterization of cavities and dilation

Gilles Mathieu

Centre SIMADE, Ecole des Mines
158 cours Fauriel 42023 Saint-Etienne Cedex 2 France
tel 77 42 01 75 - fax 77 42 66 66
email mathieu@emse.fr

Abstract: *an iterative method to extract significant points from the Freeman code of a closed digital contour is proposed. The approach is syntactic and provides with convexity and concavity points. Although these points do not exactly characterize a digital contour, they supply with a knowledge of the cavities shape by means of a degree notion. An analyser builds a hierarchy of cavities from the string of significant points. Besides a shape description aspect, these points prove to be useful in image processing: they allow an easy incremental deduction of the object's dilation from its contour code.*
Keywords: digital xy-convexity, cavities, shape description, dilation

1. Introduction

Un aspect essentiel de la description de formes est l'extraction de points caractéristiques. De tels points présentent un intérêt immédiat pour la classification [Fre78]. Ils peuvent également s'avérer précieux dans un traitement d'image, notamment pour décomposer les objets. Ainsi les points de concavité que l'on peut considérer marquer la région de superposition de deux convexes [AC74], sont de bons candidats comme points critiques. Enfin la donnée de points caractéristiques constitue la base d'une représentation approchée d'un contour discret. Dans [CM91] trois grandes classes de méthodes sont distinguées:

- l'approximation par découpage récursif;
- l'approximation itérative: la décomposition du contour en segments de droite discrets en est un bon exemple;
- l'utilisation des variations de direction dans la chaîne de Freeman pour la recherche de points de forte discontinuité dans la courbure. Localement si la chaîne est parcourue pour détecter des configurations particulières de codes [HK82], ou d'arêtes (une arête est une séquence de codes égaux) [AC74]. Le point de vue est moins local lorsqu'on fait appel à une notion de courbure incrémentale fonction des différences angulaires successives [Fre78] [CT89]. Enfin une autre méthode extrait globalement les points dominants à l'aide d'un algorithme parallèle travaillant sur la liste de signes des changements de direction [SS78].

Nous proposons dans cet article une méthode d'extraction de points significatifs qui relève à la fois des deux dernières classes. Itérative dans son principe, elle travaille à partir d'un code de Freeman. Outre leur application naturelle à la caractérisation de formes, ces points permettent une dilatation rapide, parce qu'incrémentale, du contour. L'objectif à terme est, dans un contexte de modélisation en synthèse d'images, d'adopter la même démarche en 3D pour réaliser la dilatation d'un objet voxelisé.

2. Notion de points significatifs

On considère une image binaire. L'objet O (réunion des pixels pleins) et le fond $\overline{O}$ (réunion des pixels vides) sont considérés comme étant respectivement 4- et 8-connexe. De plus $\overline{O}$ entoure O qui ne présente qu'une composante connexe.

On s'intéresse plus particulièrement au contour C de l'objet: l'ensemble des points de O qui sont 4-adjacents à un point de $\overline{O}$. C s'obtient par un algorithme de suivi de contour [CM91] qui fournit en sortie une courbe supposée simple et 8-connexe. De cette courbe qui passe par les centres des pixels de contour, on tire C' celle qui longe les arêtes de pixels en contact avec le fond. Une autre façon d'obtenir C' à partir de l'animal de O (énumération des centres de ses pixels) est d'appliquer l'algorithme décrit dans [TT92].

La courbe C' est une liste circulaire d'arêtes horizontales ou verticales. Trois types de **charnières** articulent les pixels de contour: **creux** lorsque l'angle droit entre pixels est dans le vide; **plat** si cet angle est plat; enfin **plein** si l'angle droit est dans la matière. C' peut donc être décrite à l'aide d'un mot de Freeman M restreint à un alphabet de trois lettres { CREUX, PLAT, PLEIN }.

Dans ce mot toutes les charnières ne sont pas significatives de la forme. L'ensemble des codes PLAT peut déjà être trivialement supprimé. Et même parmi les CREUX et PLEIN restants, si certains correspondent effectivement à un point de convexité (VEX) ou de concavité (CAV), d'autres ne sont que des effets de la discrétisation.

Plus précisément, le mot des charnières d'un contour fermé est décomposé en **segments**, délimités par les points significatifs. Chaque segment est lui-même constitué d'une alternance de **sections parallèles** et **perpendiculaires**. Les termes parallèle et perpendiculaire ont été choisis non pour désigner deux directions privilégiées du plan, mais pour signifier parallèle ou perpendiculaire à la **direction** du segment. Cette dernière est l'une des quatre suivantes: **droite**, **basse**, **gauche** ou **haute**. Sur l'exemple de la figure 1 le contour se décompose en 8 segments. Les points de convexité sont figurés par des disques; les points de concavité par des carrés. Les flèches indiquent la direction de chaque segment.

L'extraction des charnières significatives se fait selon le pseudo-code suivant:

> *début* = VEX;
> *pixel* = plus haut pixel plein dans la colonne non vide la plus à gauche;
> *direction* = droite;
> tant que (contour non bouclé) {
>   *fin* = *un_segment*(*début*, *pixel*, *direction*);
>   /* on a extrait un segment de type *début-fin* */
>   *direction* = (*fin* == VEX) ? *suivante*() : *précédente*()
>   *début* = *fin*
>   }

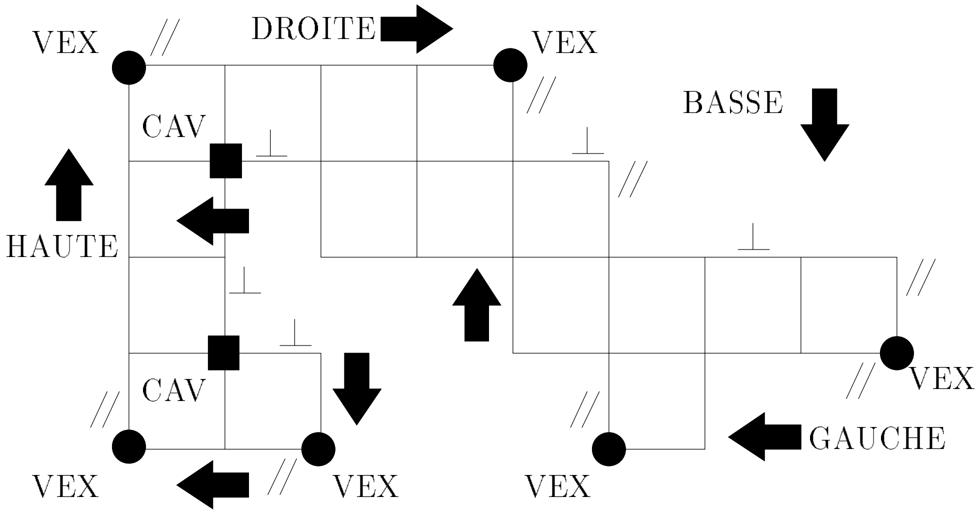

Figure 1 : *Décomposition d'un contour en segments*

Le contour est parcouru dans le sens horaire, aussi les fonctions *suivante*() et *précédente*() retournent pour droite, basse, gauche et haute respectivement { basse, gauche, haute, droite} et { haute, droite, basse, gauche} (voir figure 1). La procédure *un_segment()* renvoie à l'automate de la figure 2: selon le type VEX ou CAV du point *début* du segment, à partir d'un *pixel* donné et suivant la *direction* courante, elle explore les charnières successives jusqu'à la rencontre d'un point significatif de convexité ou de concavité.

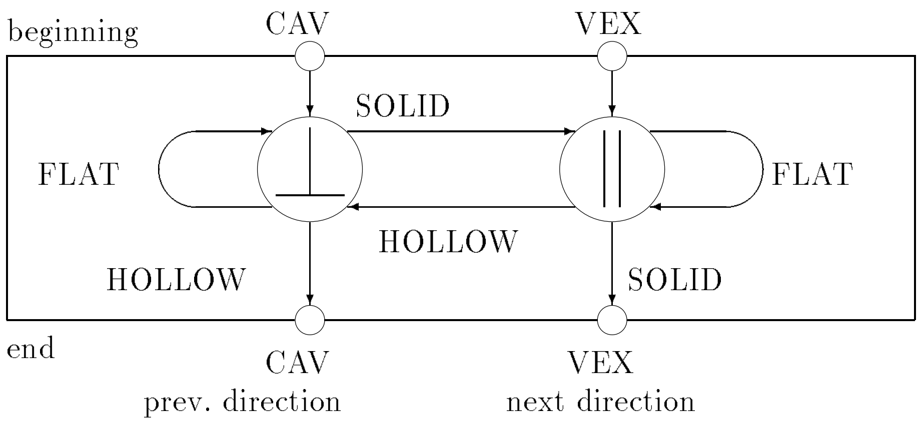

Figure 2 : *Description d'un segment*

Autrement dit, sections parallèles et perpendiculaires s'alternent sauf aux points significatifs. Il y a succession de deux sections parallèles aux points de convexité et succession de deux sections perpendiculaires aux points de concavité.

A partir de ces définitions un segment sera dit **faiblement convexe** en l'absence de concavité au sens de l'automate ci-dessus, donc s'il répond à l'expression rationnelle

$$VEXVEX \ = \| \ (\ \perp \| \ )^{*}$$

En termes imagés un segment faiblement convexe est un “escalier” dont les sections parallèles sont les “marches” et les sections perpendiculaires les “contremarches”. Un palier est un escalier dégénéré, sans contremarche; et un escalier digne de ce nom ne se termine jamais par une contremarche!

Un contour fermé sera convexe x-y [MF82] s'il est constitué de quatre segments droit, bas, gauche et haut faiblement convexes. Le mot de ses points significatifs se réduit alors à $VEX\,VEX\,VEX\,VEX$.

Si le segment n'est pas faiblement convexe, il y a une **cavité**. Elle débute nécessairement par un segment $VEXCAV$, comprend éventuellement un (ou des) segments(s) $CAVCAV$ et se termine par un segment $CAVVEX$.

$$VEXCAV = \| \; (\perp \| \;)^* \perp \qquad CAVCAV = \perp (\; \| \perp)^* \qquad CAVVEX = \perp (\; \| \perp)^* \; \|$$

3. Description de cavités

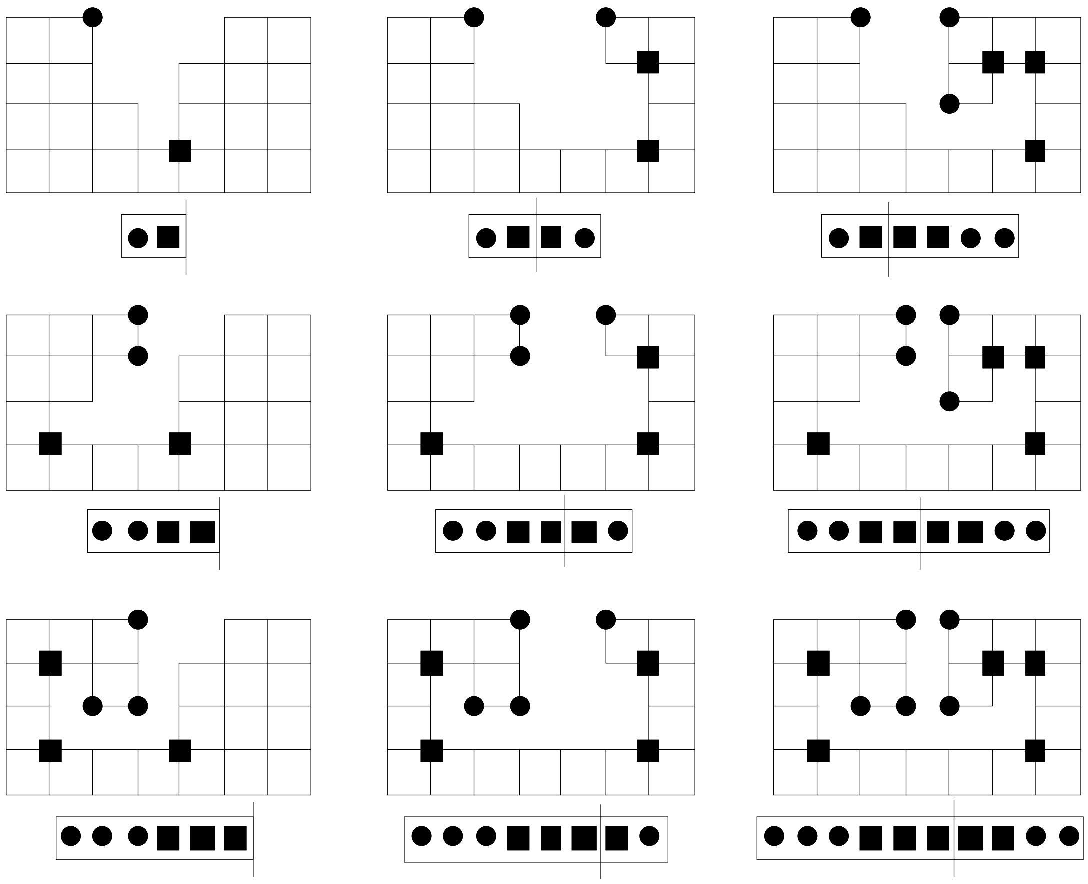

Figure 3: *Enroulement et déroulement d'une cavité simple: pour chaque ligne de haut en bas, e vaut respectivement 0, 1 et 2; pour chaque colonne de gauche à droite, d vaut respectivement 0, 1 et 2.*

Ainsi on a réduit le mot M des positions relatives (charnières) des arêtes successives du contour en un mot M' de charnières significatives de convexité ou de concavité sur l'alphabet $\{VEX, CAV\}$. Dans ce mot M', une **cavité simple** est définie syntaxiquement comme un motif

$$Cavit\acute{e}\_simple(e,d) = VEX^{e+1}\ CAV^{e+1}\ CAV^{d}\ VEX^{d} \tag{1}$$

où les entiers naturels e et d sont les **degrés** d'**enroulement** et de **déroulement** de la cavité. La figure 3 illustre différentes combinaisons et donne les motifs correspondants.

Le nombre d'occurences de la séquence $VEX\ CAV$ dans M' donne le nombre de cavités simples du contour. La figure 4 montre que pour un même mot, différents *appariements d'un motif* de cavité simple sont possibles: la cavité appartient au segment de direction droite en (4a); basse en (4b), (4c) et (4d); et gauche en (4e).

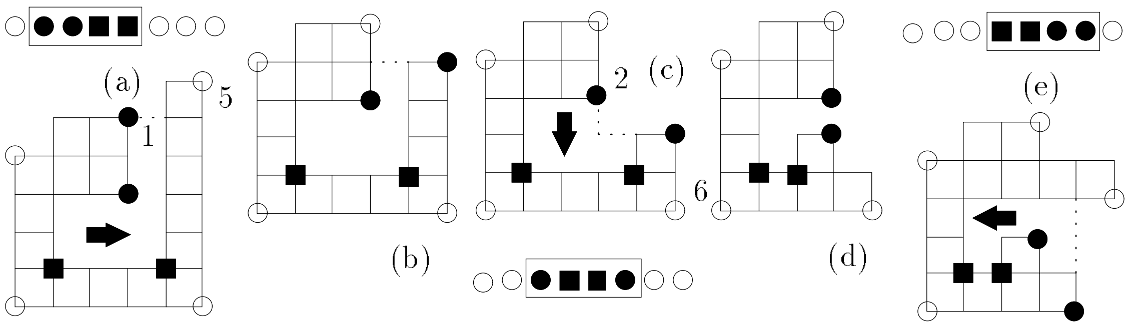

Figure 4: *Différentes configurations pour un même mot*

Ces différentes possibilités d'appariement d'un motif de cavité simple peuvent être vues comme le *glissement d'une fenêtre* sur une sous-chaîne CAV^{e+1+d} de M'. D'après (1) la somme des degrés d'enroulement et de déroulement est invariante sur les différentes configurations.

Les coordonnées du pixel sous-jacent à chaque lettre VEX ou CAV sont également stockées. Si le mot M' avec mémorisation des pixels ne constitue pas une représentation exacte du contour, il permet de discriminer entre les possibilités d'appariement et de déterminer à quel segment faiblement convexe appartient la cavité.

La fonction *plus_haut_que* compare (modulo la direction de segment courante) le premier pixel significatif du motif avec le premier qui suit le motif. Par exemple sur la figure (4a): 5 *plus_haut_que* 1 (modulo droite) et le premier appariement correspond à la cavité simple; en (4b), (4c) et (4d) la cavité simple est réduite après une opération *faire-glisser*: 1 *plus_haut_que* 5 et 6 *plus_haut_que* 2 (modulo basse); en (4e) deux telles opérations sont nécessaires car 1 *plus_haut_que* 5 et 2 *plus_haut_que* 6.

Les cavités simples s'organisent en une **hiérarchie** définie opérationnellement par l'algorithme itératif suivant qui les réduit en ordre postfixé:

```
faire_glisser()
{
    ++premier; ++dernier; ++suivant;        --e; ++d;
}
quelle_cavité_réduire()
{
    position = début de M';
    tant que (1) {
        apparier dans M' un motif Cavité_simple(e, d) au delà de position;
            /* on appelle respectivement premier, dernier et suivant
            la première lettre, dernière lettre et lettre suivant le motif */
        tant que ( premier ≠ CAV ) que ( suivant ≠ CAV )
          et que ( premier  plus_haut_que  suivant )
            faire_glisser();
        si ( suivant = CAV )        position = suivant -1;
        sinon                       retourner( premier, dernier );
        }
}
réduire(premier, dernier)
{
    supprimer dans M' la sous-chaîne [premier, dernier];
}
hiérarchie()
{
    tant que (M' ≠ VEXVEXVEXVEX ) {        /* il reste une cavité */
        premier, dernier = quelle_cavité_réduire();
        réduire( premier, dernier);
        }
}
```

Il est possible au passage (si l'on dispose pour chaque charnière significative de son indice dans le mot de Freeman d'origine M), de fermer chaque cavité au moment de sa réduction. C'est un moyen d'obtenir une succession d'approximations, de régularisations de l'objet O. Le résultat de la fermeture de la dernière cavité est l'enveloppe convexe x-y de l'objet.

D'autres sous-produits faciles à obtenir sont les mesures de concavité qui sont proposées dans [Skl72].

Mais l'intérêt de cette extraction de points significatifs ne se limite pas à l'aspect description de forme. Elle permet aussi d'effectuer rapidement sur le contour la dilatation au sens de la morphologie mathématique [Ser82].

4. Dilatation

Le contour du dilaté de l'objet (de l'épaisseur d'un pixel) peut très facilement se déduire du contour initial. Pour cela on translate chaque segment, selon sa direction haute, droite, basse ou gauche, respectivement d'un vecteur (-1,1), (1,1), (1,-1) ou (-1,-1), comme indiqué figure 5.

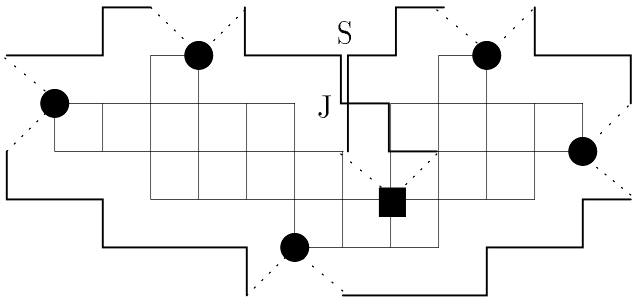

Figure 5: *Dilatation d'un contour: translation de ses segments*

Au niveau des points de convexité, il manque deux arêtes dans le prolongement du translaté de l'ancien segment: il suffit de rajouter deux charnières PLATes à la fin de son code. Au niveau des points de concavité, les translatés des deux anciens segments s'intersectent maintenant. On détermine le point de *jonction* en suivant à rebours le code *avant* la charnière CAV et dans le sens direct le code *après* cette même charnière. On continue ainsi jusqu'à la *séparation* des translatés: tout ce qui a été parcouru jusque là disparaît dans le code du dilaté et il suffit de rajouter la charnière correspondant au nouveau raccord de segments (indiquée en paramètre de la procédure *séparation* dans le pseudo-code ci-dessous). Les diverses possibilités pour *avant* et *après* peuvent être étudiées exhaustivement en fonction du type *parallèle* ou *perpendiculaire* des sections.

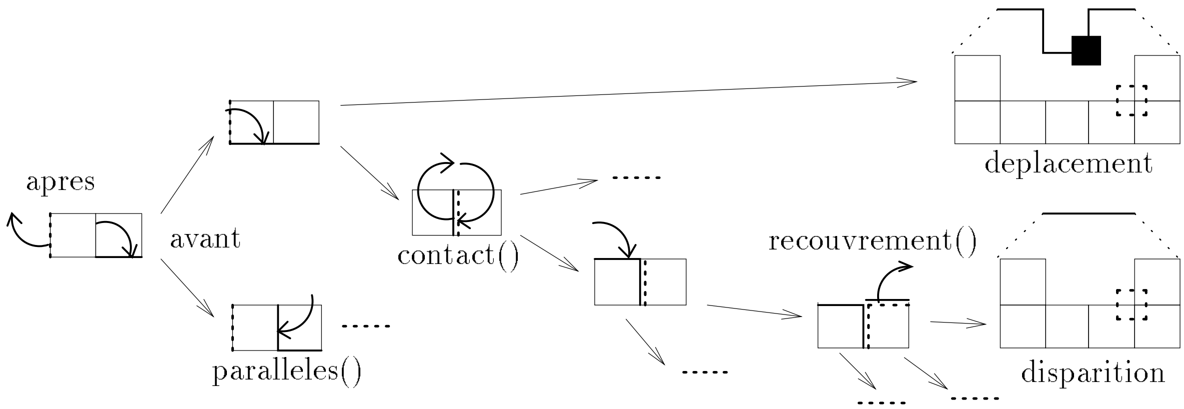

Figure 6: *Arbre (partiel) d'analyse de cas pour les charnières avant et après.*

L'arbre complet a 14 feuilles: 8 appels aux fonctions correspondant aux trois configurations types *recouvrement*, *contact* et *parallèles*; et 6 configurations d'arrêt relevant de deux types: *disparition* d'un CAV et d'un VEX lorsque une cavité est comblée par l'opération de dilatation; *déplacement* dans le cas contraire où la charnière est seulement déplacée. Toutes ces fonctions et variables s'organisent selon l'algorithme suivant:

```
recouvrement()
{
 switch( après ) {
  case PLEIN:
     disparition();
  case PLAT:
     ++après; recouvrement();
  case CREUX:
     déplacement( CAV );
  }
}
contact()
{
 switch( avant−− ) {
  case PLEIN: switch( après++ ) {
     case PLEIN: séparation( PLAT );
        recouvrement();
     case PLAT: séparation( CREUX );
        déplacement( CAV );
     }
  case PLAT: switch( après++ ) {
     case PLEIN: séparation( CREUX );
        recouvrement();
     case PLAT: contact();
     }
  }
}
dilate()
{
 début = 0;                /* début du segment avant la charnière significative */
 pour chaque charnière significative C {
  avant = charnière courante du segment avant C;
  après = charnière courante du segment après C;
  si ( C convexe ) {
    recopier de début jusqu'à avant;  ajouter( PLAT, PLAT, PLEIN ); déplacement( VEX )
   }
  sinon /* C concave */ {
    switch( avant−− ) {
     case PLAT: jonction();
      switch( avant−− ) {
       case PLAT: séparation( CREUX ); déplacement( CAV );
       case CREUX: contact();
       }
     case CREUX:
      parallèles();
     }
  début = après;
  }
}
```

```
parallèles()
{
 switch( avant−− ) {
  case PLEIN: jonction();
   switch( après++ ) {
    case PLEIN:−−avant; séparation( avant +1 );
     recouvrement();
    case PLAT:
     switch( avant−− ) {
      case PLAT:
       séparation( CREUX ); déplacement( CAV );
      case CREUX: contact();
      }
    }
  case PLAT:
   switch( après++ ) {
    case PLEIN: jonction();
     switch( après++ ) {
      case PLEIN:
       séparation( CREUX ); disparition();
      case PLAT:
       séparation( CREUX ); recouvrement();
      case CREUX: contact();
      }
    case PLAT: parallèles();
    }
  }
}
```

L'algorithme doit aussi gérer les changements de topologie qui se produisent dans des cas comme celui de la figure 7. Une table de hachage est créée pour chaque cavité afin d'y stocker les coordonnées de ses arêtes. Grâce à elle on trouve facilement le point de séparation (point S sur la figure 7).

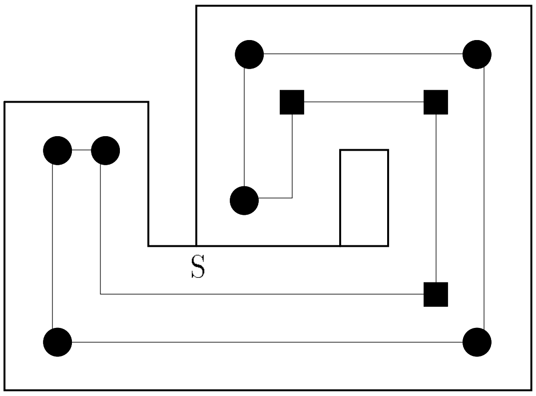

Figure 7 : *Changement de topologie.*

Ainsi la dilatation s'effectue incrémentalement, ne faisant appel au décodage du contour qu'aux seuls points de concavité, et ce sur des longueurs minimes. En l'absence de cas pathologiques où les cavités sont en spirale et de degrés élevés l'algorithme ne parcourt qu'une faible fraction du code initial. Le mot des charnières significatives du dilaté se déduit aussi incrémentalement de M'.

5. Conclusion et perspectives

On a proposé une méthode syntaxique d'extraction de points significatifs dans un contour discret. Outre l'aspect description de forme, ces points se révèlent utiles au traitement de l'objet: ils permettent de déduire facilement du code d'un contour celui de son dilaté.

Il semble intéressant d'adopter la même démarche dans le cas 3D pour un objet voxelisé. L'équivalent du codage de Freeman est alors un **graphe non orienté**, **connexe** et **4-régulier** dont les **nœuds** sont les **pixels de surface**. Les **arêtes** du graphe sont les **charnières** qui articulent ces pixels. Dans [Mat94] un algorithme a été proposé pour construire un tel graphe. La méthode est dirigée par les charnières significatives de deux tranches successives de l'objet.

L'objectif à terme est, dans un contexte de modélisation en synthèse d'images, de réaliser une sorte d'éditeur de voxels qui permette le modelage ou la retouche d'un objet de façon interactive, l'opération primitive étant la dilatation partielle (limitée à l'intérieur d'un contour désigné par l'utilisateur). Le travail est en cours concernant la déduction incrémentale du graphe des pixels de la structure du dilaté.

Références

[AC74] C. Arcelli and L. Cordella. Concavity point detection by iterative arrays. *Computer graphics and image processing*, 3:34–47, 1974.

[CM91] J.M. Chassery and A. Montanvert. *Géométrie discrète en analyse d'images*. Hermes, 1991.

[CT89] R. Chin and C. Ten. On the detection of dominant points on digital curves. *IEEE Transactions on PAMI*, 11(8):859–872, 1989.

[Fre78] H. Freeman. Shape description via the use of critical points. *Pattern recognition*, 10:159–166, 1978.

[HK82] S. Hung and T. Kasvand. Critical points on a perfectly 8- or 4-connected thin binary line. In *6th ICPR*, pages 531–533. IEEE Comp. Soc. Press, 1982.

[Mat94] G. Mathieu. Un algorithme de construction du graphe de surface pour un objet faiblement convexe. In *2èmes journées AFIG-GROPLAN*, pages 201–210. AFIG, December 1994.

[MF82] D. Montuno and A. Fournier. Finding the x-y convex hull of a set of x-y polygons. Technical Report CSRG-148, University of Toronto, November 1982.

[Ser82] J. Serra. *Image analysis and mathematical morphology*. Academic Press, 1982.

[Skl72] J. Sklansky. Measuring concavity on a rectangular mosaic. *IEEE transactions on computers*, C21(12):1355–1364, December 1972.

[SS78] P. Sankar and C. Sharma. A parallel procedure for the detection of dominants points on a digital curve. *Computer graphics and image processing*, 8:136–143, 1978.

[TT92] J. Tatari and E. Tosan. Représentation des polyominos. In *Journées GROPLAN*, pages 211–218. IRIN Nantes, November 1992.