- création des chaînes en parallèle en une passe par un balayage de l'image avec un pavé 3 x 3 ; les chaînes sont créées par mise à jour itérative par de simples jeux de pointeurs ;
- fusion des chaînes obtenues afin de minimiser leur nombre ;
- élimination de chaînes suivant un critère.

L'originalité de cette approche par rapport aux nombreux algorithmes de chaînage existant est de proposer une structure de données "propre" permettant d'effectuer le chaînage et de représenter son résultat. En effet il faut noter que dans les travaux antérieurs, l'aspect structure de données n'avait pas été abordé aussi clairement. Une méthode d'approximation polygonale peut permettre ensuite d'obtenir à partir des chaînes de contours une approximation polygonale des contours (voir le chapitre 3).

Chapitre 3

Segmentation de contours

Les chaînes de contours peuvent être partitionnées dans des segments de courbes qui ont une description analytique connue telles des lignes droites et des coniques. La plus simple description d'une courbe est la ligne droite mais dans de nombreux cas ce n'est pas suffisant et l'utilisation de courbes du deuxième ordre s'avère alors nécessaire. Ce chapitre est en grande partie basé sur les travaux de Pavlidis & Horowitz [147].

3.1. Segmentation d'une chaîne

Soit $C = \{c_i, i = 0, 1, \cdots, n\}$ l'ensemble des points d'une chaîne où $c_i = [x_i, y_i]$ est le $i^{ème}$ point de la chaîne avec ses coordonnées dans l'image. Ce point n'a en principe que deux voisins (sauf pour le premier et dernier point) : c_{i-1} et c_{i+1}. Soit S une partition de C. Soit M un sous-ensemble de C contenant les points de cassure correspondant à la partition S. On a :

$$S = \{s_0, s_1, \cdots, s_{N-1}\} \; et \; M = \{m_0, m_1, \cdots, m_N\}$$

Le segment s_k commence au point m_k et se termine au point m_{k+1}. Les éléments m de M appartiennent également à C. Chaque m correspond donc à un c : $m_k = c_{i_k}$. Un segment s_k est donc un sous ensemble de points de C tel que (figure 3.1.) :

$$s_k = \{c_j \mid i_k \leq j \leq i_{k+1},\; c_{ik} = m_k,\; c_{ik+1} = m_{k+1}\}$$

La segmentation d'une chaîne de contours posera donc deux problèmes :

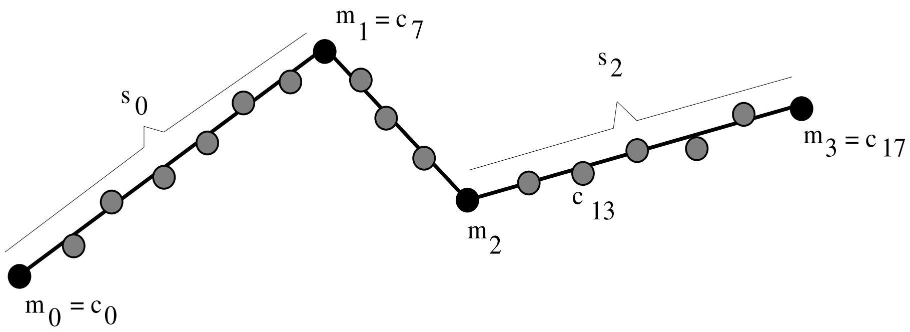

Figure 3.1. *Un exemple de segmentation.*

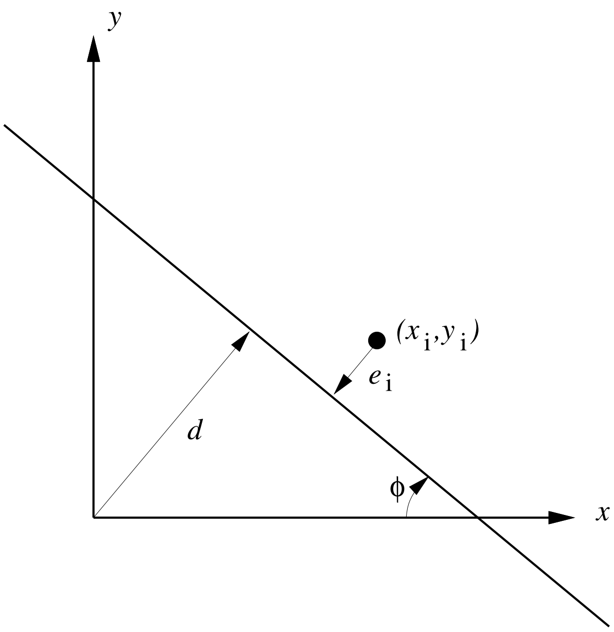

Figure 3.2. *La paramétrisation d'une droite utilise la distance de la droite à l'origine et l'inclinaison de la droite par rapport à l'axe horizontal.*

1. trouver un partionnement de la chaîne en segments $(s_0, \cdots, s_k, \cdots)$ et
2. trouver pour chaque segment s_k la meilleure approximation analytique.

Nous allons commencer par donner une solution simple d'approximation d'un ensemble de points par un segment de droite et par un arc de cercle. Ensuite nous allons décrire quelques méthodes de partionnement d'une chaîne de points en segments.

3.2. Approximer un segment par une droite

Soit l'équation d'une droite (figure 3.2.) :

$$x \ \sin\phi + y \ \cos\phi = d \tag{3.1}$$

La distance d'un point $[x_i, y_i]$ à cette droite est :

$$e_i = \mid x_i \ \sin\phi + y_i \ \cos\phi - d \ \mid$$

Nous voulons trouver la droite (les paramètres ϕ et d) qui minimise la grandeur :

$$E = \sum_{i=1}^{n} e_i^2 \tag{3.2}$$

L'ensemble des paramètres de la droite est la solution du système d'équations :

$$\left\{ \begin{array}{rcl} \frac{\partial E}{\partial \phi} & = & 0 \\ \frac{\partial E}{\partial d} & = & 0 \end{array} \right. \tag{3.3}$$

Après calculs on obtient pour d et ϕ :

$$d = V_x \sin\phi + V_y \cos\phi \tag{3.4}$$

$$\phi = \frac{1}{2} \arctan \frac{2V_{xy}}{V_{yy} - V_{xx}} \tag{3.5}$$

avec :

$$\begin{array}{rcl} V_x & = & \frac{1}{n} \sum_{i=1}^{n} x_i \\ V_y & = & \frac{1}{n} \sum_{i=1}^{n} y_i \end{array}$$

$$\begin{aligned} V_{xx} &= \sum_{i=1}^{n}(x_i - V_x)^2 \\ V_{yy} &= \sum_{i=1}^{n}(y_i - V_y)^2 \\ V_{xy} &= \sum_{i=1}^{n}(x_i - V_x)(y_i - V_y) \end{aligned}$$

3.3. Approximer un segment par un cercle

Soit l'équation d'un cercle :

$$(x-a)^2 + (y-b)^2 = r^2 \qquad [3.6]$$

Avec les notations suivantes :

$$\begin{aligned} A &= -2a \\ B &= -2b \\ C &= a^2 + b^2 - r^2 \end{aligned}$$

L'équation du cercle peut s'écrire à nouveau :

$$x^2 + y^2 + Ax + By + C = 0 \qquad [3.7]$$

La distance d'un point à un cercle s'écrit :

$$d_i = \mid \sqrt{(x_i - a)^2 + (y_i - b)^2} - r \mid \qquad [3.8]$$

Cependant ceci conduit à un problème d'optimisation non linéaire. La distance euclidienne d'un point à un cercle peut être remplacée par la distance algébrique :

$$\begin{aligned} e_i &= \mid (x_i - a)^2 + (y_i - b)^2 - r^2 \mid \\ &= \mid x_i^2 + y_i^2 + Ax_i + By_i + C \mid \end{aligned} \qquad [3.9]$$

Le critère à minimiser est :

$$E = \sum_{i=1}^{n} e_i^2 \qquad [3.10]$$

La solution pour A, B et C est donnée par le système d'équations suivant :

$$\left\{ \begin{array}{rcl} \frac{\partial E}{\partial A} & = & 0 \\ \frac{\partial E}{\partial B} & = & 0 \\ \frac{\partial E}{\partial C} & = & 0 \end{array} \right. \qquad [3.11]$$

Après dérivation on obtient :

$$\begin{array}{rcl} U_{xx}\, A + U_{xy}\, B + U_x\, C & = & -U_{xxx} - U_{xyy} \\ U_{xy}\, A + U_{yy}\, B + U_y\, C & = & -U_{xxy} - U_{yyy} \\ U_x\, A + U_y\, B + n\, C & = & -U_{xx} - U_{yy} \end{array} \qquad [3.12]$$

Avec :

$$U_{xy} = \sum_{i=1}^{n} x_i y_i$$

et des définitions similaires pour U_x, U_{xxy}, U_{xyz}, etc.

On obtient donc un système linéaire à trois inconnues, A, B et C, à partir desquelles on retrouve facilement les paramètres du cercle.

3.4. Algorithme de découpage récursif

Supposons pour l'instant qu'on veuille approximer une chaîne de points avec des segments de droite. Dans ce cas un algorithme de découpage récursif peut être le suivant :

1. pour une chaîne de points: est-ce un segment de droite ?
2. si oui – aller en fin ;
3. sinon – diviser la chaîne en deux sous-chaînes et répéter
l 'algorithme pour chaque sous-chaîne ;
4. fin.

Avec les notations précédentes, considérons une sous-chaîne s_k dont les extrémitées sont m_k et m_{k+1}. Si cette sous-chaîne n'est pas correctement approximée par un segment de droite (au sens du critère quadratique énoncé précédemment), une façon "naturelle" d'améliorer cette approximation est de chercher parmi les points de la sous-chaîne s_k celui qui est le plus éloigné de la corde définie par les extrémités de s_k. On casse alors s_k en deux sous-chaînes, s'_k et s''_k, et on leur applique le même traitement...

La figure 3.3. illustre une implantation efficace de l'algorithme de découpage récursif. La procédure utilise deux tableaux, CLOSED et OPEN. Initialement ces tableaux contiennent respectivement le premier et dernier points de la chaîne, A et B. Si la chaîne comprise entre A et B ne peut être approximée par un segment de droite (ce qui est le cas), le point le plus éloigné de la corde AB (C) est choisi comme point de cassure. Ce point est rajouté dans le tableau OPEN. A chaque pas d'itération de la procédure on considère la chaîne comprise entre le dernier point entré dans CLOSED et le dernier point entré dans OPEN. Lorsqu'un point de cassure est trouvé, il est rajouté à la fin du tableau OPEN. Lorsqu'une chaîne peut être approximée par un segment, le dernier point entré dans OPEN passe dans CLOSED.

CLOSED	OPEN	Segment considéré	Sommet créé
A	B	AB	C
A	B, C	AC	D
A	B, C, D	AD	–
A, D	B, C	DC	E
A, D	B, C, E	DE	–
A, D, E	B, C	EC	F
A, D, E	B, C, F	EF	–
A, D, E, F	B, C	FC	–
A, D, E, F, C	B	CB	G
A, D, E, F, C	B, G	CG	–
A, D, E, F, C, G	B	GB	H
A, D, E, F, C, G	B, G	GH	–
A, D, E, F, C, G, H	B	HB	–
A, D, E, F, C, G, H, B	–	–	–

Figure 3.3. *Un exemple de procédure de découpage récursif.*

3.5. Algorithme de découpage et union

Supposons maintenant qu'on veuille approximer une chaîne par une séquence de segments de droite et d'arcs de cercle. L'algorithme de découpage récursif devient alors :

1. pour une chaîne de points : est-ce un segment de droite ?

2. si oui – aller en fin ;

3. sinon – est-ce un cercle ?

4. si oui – aller en fin ;

5. sinon – diviser la chaîne en deux sous-chaînes et répéter l'algorithme pour chaque sous-chaîne ;

6. fin.

Cet algorithme pose trois problèmes :

1. étant donné un ensemble de points approximé correctement (au sens des moindres carrés) aussi bien par une droite que par un cercle, quelle approximation choisir ? Ce problème est illustré par la figure 3.4. Dans cet exemple nous sommes tentés de choisir un cercle. Nous introduisons un facteur d'évaluation e :

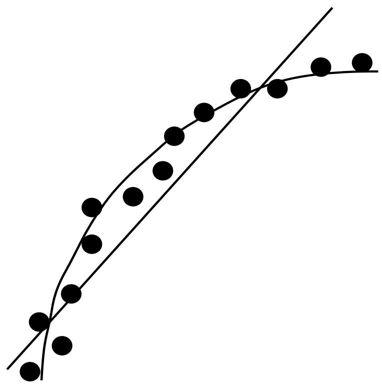

Figure 3.4. *Cette chaîne peut aussi bien être une droite qu'un cercle.*

$$e = \frac{k}{n}\,(1 - \frac{p}{n}) \qquad [3.13]$$

avec :

- n est le nombre de points de la chaîne ;
- k est le nombre de changements de signe de la séquence des vecteurs d'erreur ;
- p est la longueur de la plus grande séquence de vecteurs d'erreurs ayant le même signe.

Dans l'exemple traité, k est grand et p est petit pour un cercle, alors que c'est le contraire pour une droite. Dans ce cas on aura donc :

$$e_{cercle} > e_{droite}$$

2. les points de cassure choisis par l'algorithme de découpage récursif peuvent couper un arc de cercle en deux parties, figure 3.5. Il faudrait donc accompagner la procédure de découpage par une procédure d'union de deux chaînes adjacentes. Une procédure de découpage-et-union est donnée en exemple sur la figure 3.6.

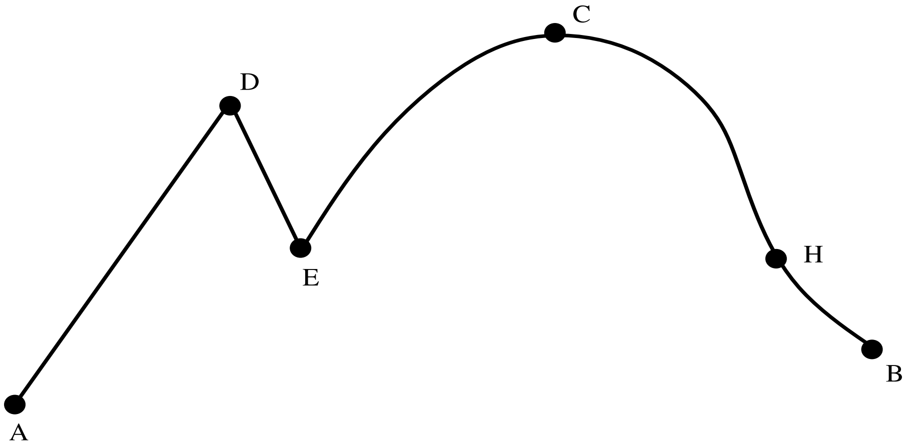

Figure 3.5. *La procédure de découpage récursif a coupé un arc de cercle en deux morceaux.*

3. lors du choix d'un point de cassure, sa position peut être mauvaise par rapport à une position théorique. C'est souvent le cas lorsqu'une droite est tangente à un cercle, le point de tangence étant difficile à localiser, figure 3.7. Soit E_k et E_{k+1} les erreurs d'approximation associées à deux segments adjacents et soit m_{k+1} leur point de cassure. On veut trouver la position d'un nouveau point de cassure m'_{k+1} tel que :

$$\begin{aligned} |m'_{k+1} - m_{k+1}| &< D_{max} \\ E'_k &< E_k \\ E'_{k+1} &< E_{k+1} \end{aligned}$$

CLOSED	OPEN	Segments considérés	Sommet enlevé
A	D, E, C, H, B	ADE	–
A, D	E, C, H, B	DEC	–
A, D, E	C, H, B	ECH	C
A, D, E	H, B	EHB	–
A, D, E, H	B	–	–
A, D, E, H, B	–	–	–

Figure 3.6. *Un exemple de découpage-et-union.*

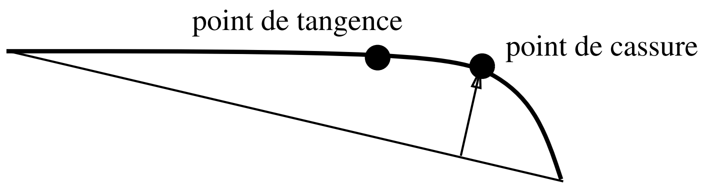

Figure 3.7. *Une droite tangente à un cercle.*

On définit donc un intervalle autour du point de cassure initial à l'intérieur duquel on doit se déplacer. Le nouveau point de cassure doit être tel que les erreurs associées aux deux segments adjacents soient diminuées.

3.6. Un exemple

Nous allons illustrer à l'aide d'un exemple simple la segmentation en lignes droites obtenues par un algorithme de découpage et union tel qu'il vient d'être décrit. La figure 3.8. montre une image comportant beaucoup de lignes droites (gauche) ainsi que le résultat de l'extraction de contours (droite). La figure 3.9. montre le résultat de l'approximation polygonale. On remarque que dans ce cas le choix de l'erreur maximale d'approximation n'est pas du tout critique.

3.7. La transformée de Hough

Il est possible de "grouper" des points de contour colinéaires sans passer par l'étape de chaînage. On peut utiliser pour cela une méthode qui consiste à faire

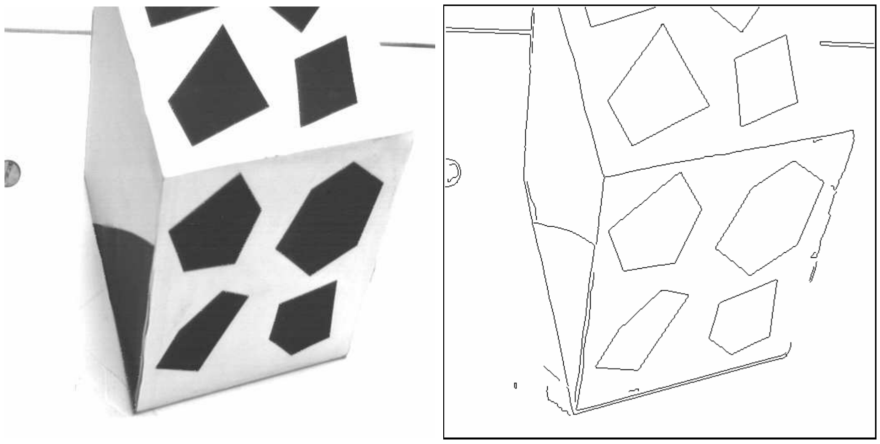

Figure 3.8. *Une image à niveaux de gris et les contours extraits de cette image utilisant le filtre de Deriche.*

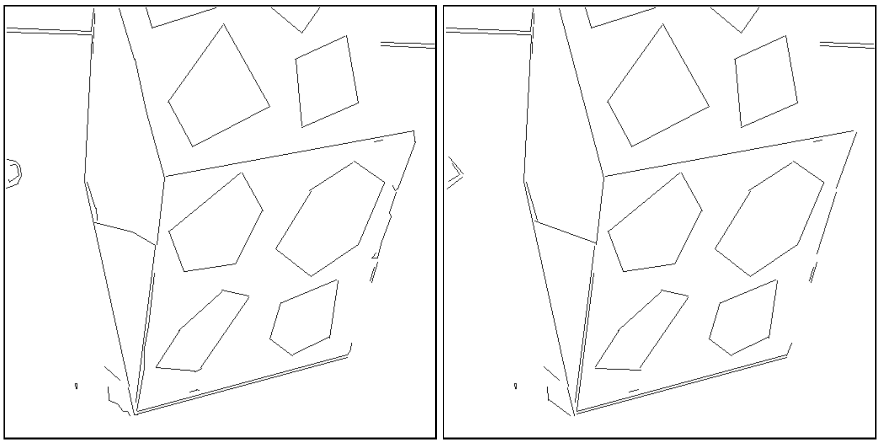

Figure 3.9. *Les segments de droite obtenus avec un algorithme de decoupage et union. Dans l'image de gauche nous avons utilisé une erreur maximale d'approximation de 0.4 pixels alors que dans l'image de droite nous avons utilisé une erreur de 5 pixels.*

correspondre à des points de contour du plan image des points d'un espace de paramètres. Cette technique s'appelle "transformée de Hough", du nom de son inventeur. Conceptuellement elle peut s'étendre pour détecter des courbes de forme quelconque. On trouvera un traitement complet de la tranformée de Hough dans [14].

Soit l'équation d'une droite telle que nous l'avons déjà utilisée :

$$x \sin \phi + y \cos \phi = d$$

Si de plus $\phi \in [0, \pi]$, les paramètres de la droite sont uniques. A chaque droite du plan $x - y$ correspond un point dans le plan $\phi - d$.

Considérons maintenant un point de contour du plan image. Ce point est caractérisé par sa position dans l'image, $[x_i, y_i]$ *et* par son orientation ϕ_i. On peut donc définir une droite qui passe par ce point et qui a la direction du contour :

$$x_i \sin \phi_i + y_i \cos \phi_i = d_i$$

A chaque point-contour du plan image correspond donc un point dans le plan $\phi - d$. Soient n points-contour colinéaires. On aura : $\phi_1 = \cdots = \phi_n$ et $d_1 = \cdots = d_n$. A tous ces points colinéaires il correspond *le même point* dans le plan des paramètres.

Afin d'utiliser pratiquement cette technique nous devons discrétiser le plan des paramètres en N×M cellules. A chaque point-contour du plan image correspond donc une cellule du plan $\phi - d$. De plus on associe un compteur à chaque cellule. Chaque fois qu'une cellule reçoit un "vote" de l'image on incrémente son compteur. Des points de contour colinéaires voterons pour la même cellule, le compteur correspondant contiendra par conséquent une valeur élevée par rapport au compteur d'une cellule qui ne correspond pas à des points de contour colinéaires, figure 3.10.

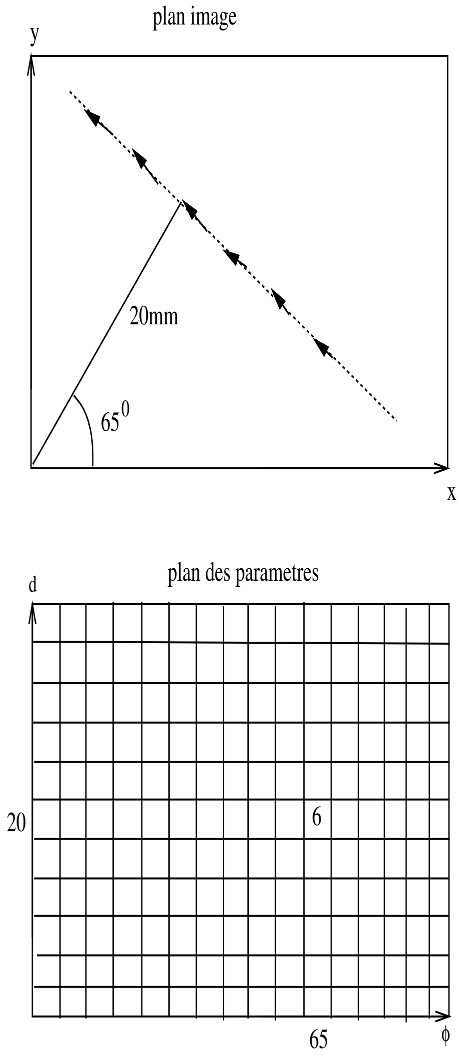

Figure 3.10. *Détection de droites avec la transformée de Hough.*