Les di fférents codages de f ormes

Sommaire

Le code de Freeman [Fre74] a comme but premier la compression des données. En effet, il permet de passer d'une image binaire, dont on mémorise la valeur du niveau de gris (0 ou 1) de chaque pixel, à une chaîne de codants d'une part et aux coordonnées d'un point du contour de l'objet d'autre part. Il est également possible d'utiliser le code de Freeman différentiel compressé grâce au code de Huffman [Liu] pour augmenter encore le taux de compression.

Dans ce chapitre, nous présenterons d'abord les codes de contour les plus connus, puis d'autres codes visant à compresser les images binaires et enfin le code de Freeman en détail.

1.1 Quelques codes de contour

1.1.1 La tortue de Papert

Description :

Une des représentations de contour a été proposée par S. Papert [Pap] en 1973. L'alphabet de ce code utilise uniquement deux symboles illustrés figure 1.1 : le 1, qui correspond à un virage à gauche pour sortir de l'objet étudié et le 0 associé à un virage à droite pour entrer dans l'objet étudié.

Algorithme 1 Obtention du code de Papert

```
1 Localisation d'un premier point de la frontière interne de l'objet (au sens voisinage V8,
  voir figure 1.13 page 12)
2 si le pixel courant appartient à l'objet alors
3    on tourne à gauche et on avance d'un pixel
4 sinon
5    on tourne à droite et on avance d'un pixel
6 fin si
```

L'algorithme 1 présente la méthode d'obtention du code d'un contour selon le code de Papert. Dans cet algorithme, à la ligne 1, la localisation du premier point de la frontière interne peut se faire, grâce à la détection du premier point allumé par exemple au sens du balayage éléctronique (Voir figure 1.21 page 17). Du fait du choix de tourner à gauche lorsqu'on se trouve dnas l'objet et à droite lorsqu'on se trouve à l'extérieur, le parcours de la frontière se fait dans le sens horaire.

La figure 1.2 page ci-contre représente le codage d'une forme binaire par cette tortue de Papert.

FIGURE 1.1 – Alphabet du code de Papert.

Propriétés :

Le codage par la tortue de Papert possède les propriétés suivantes :

Il est **invariant par rotation** d'un angle multiple de 90°.

Il est **invariant par translation** (il suffit de translater le point de départ).

En transformant les 1 en 0 et réciproquement, on obtient le code du complémentaire de l'objet.

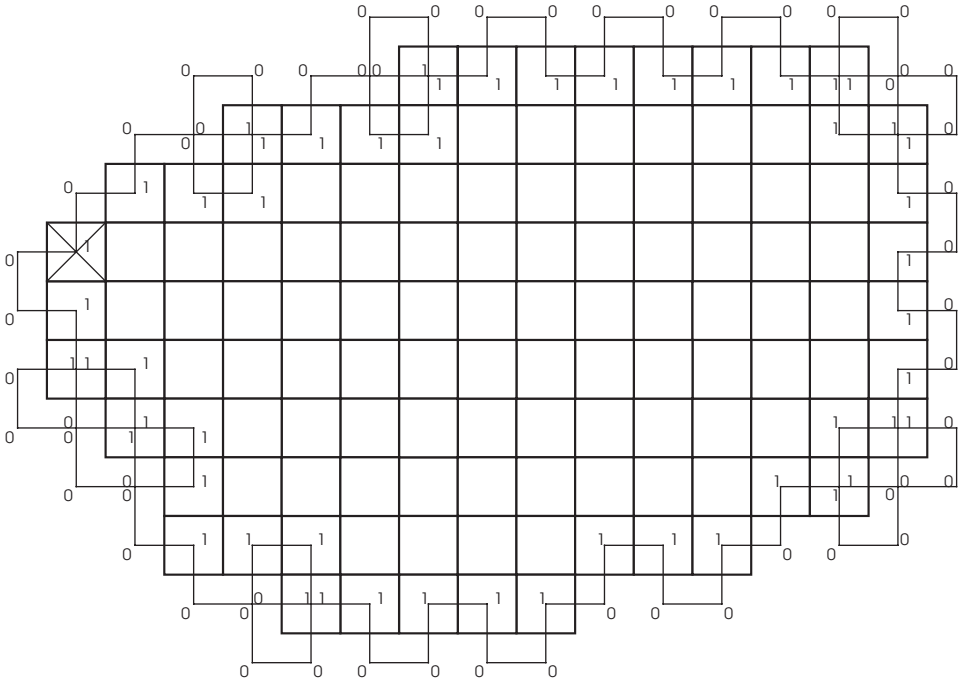

FIGURE 1.2 – Codage d'un objet binaire par la tortue de Papert.

Le code de l'objet obtenu par symétrie centrale (ou axiale) est simplement le même code, mais parcouru en sens inverse.

Le code est **indépendant du point de départ** : deux codages du même objet commençant à deux points différents seront identiques à une permutation circulaire près.

1.1.2 Le code de Bribiesca

1.1.2.1 En trois dimensions

Description :

Le code introduit par E. Bribiesca ([Bri99] et [Bri00]) permet de coder une surface discrète en trois dimensions afin d'en obtenir les caractéristiques. Cette surface ayant été extraite auparavant, on la considère comme une courbe faisant le tour de l'objet de départ (Voir l'illustration figure 1.4 page 5 issue de la publication de [Bri00]). On parle alors de courbe ou de surface par abus de langage.

Cette courbe est composée de segments non épais de taille égale à un côté de pixel. Les frontières d'un objet quelconque en trois dimensions peuvent être représentées par ce genre de chaîne. L'alphabet de cette représentation est composé uniquement de cinq symboles correspondant aux cinq variations possibles de directions orthogonales dans l'espace. Chacun de ces symboles est représenté par un nombre et illustré sur la figure 1.3 page suivante.

Deux segments définissent un changement de direction et deux changements de direction définissent un élément de la chaîne. Les éléments possibles sont les suivants :

- (a) le 0 indique qu'il n'y a pas de changement,

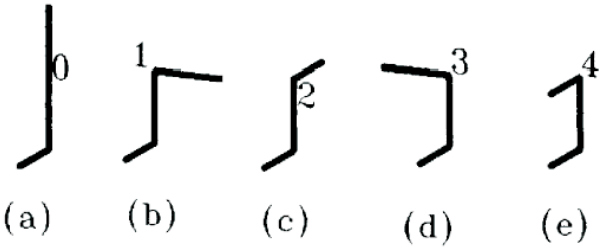

FIGURE 1.3 – Les cinq changements orthogonaux de directions dans l'espace, alphabet du code de Bribiesca en 3D.

- (b) le 1 est un virage vers la droite,
- (c) le 2 est un basculement vers l'avant,
- (d) le 3 symbolise un virage vers la gauche,
- (e) le 4 représente un retour en arrière.

La chaîne représentative d'une courbe est obtenue en calculant les changements relatifs de direction autour de la courbe. On obtient donc une chaîne composée d'un nombre fini d'éléments que l'on pourra coder en base 5. La longueur L de la courbe sera donc $L = (n+2)p$ avec n le nombre de codes dans la chaîne et p la longueur d'un segment (ici égal à un côté de pixel).

Propriétés :

Ce code possède plusieurs propriétés :

Indépendance par rotation : une courbe ainsi codée aura le même code après rotation d'angles multiples de 90°.

L'inverse d'une chaîne (voir § 2.1.1 page 26) peut être facilement calculé simplement en inversant l'ordre des codes. Attention, si des ″0″ sont présents dans la chaîne, il faudra alors inverser les codes précédant et suivant ces ″0″.

Indépendance du point de départ pour une courbe fermée.

Invariance après une symétrie par rapport à un plan, excepté l'inversion des 1 et des 3 dans la chaîne.

La comparaison de courbes est alors aisée car deux courbes identiques auront exactement le même code.

Une surface en trois dimensions représentée par un ensemble de voxels peut ainsi être codée par la courbe entourant cette surface. Un exemple issu de la publication de [Bri00] est donné figure 1.4 page ci-contre où l'on voit à gauche l'objet en volume à coder et à droite la succession des arêtes par lesquelles la courbe décrit la surface de cet objet.

Pistes de recherche :

Nous voyons dans ces travaux quelques améliorations possibles :

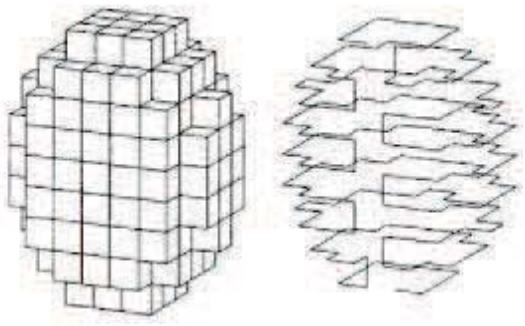

FIGURE 1.4 – Exemple de représentation du code de Bribiesca en trois dimensions

- à partir de représentations de surfaces en trois dimensions codées par ce processus, il semble possible, d'extraire le code du contour d'une coupe de l'objet 3D afin d'en étudier les paramètres géométriques.
- il est également envisageable de compresser encore plus ce code grâce à un code de Huffman ou une compression du nombre de 0 dans la chaîne.

1.1.2.2 En deux dimensions

Description :

L'étude de E. Bribiesca [Bri00] a été poursuivie en 2005 par H. Sanchez-Cruz [San05] en n'utilisant plus que trois des cinq symboles initiaux afin de rester dans le plan. Une représentation graphique de l'alphabet de ce code correspondant à ces trois changements de direction est donnée figure 1.5. Voici ce à quoi ils correspondent :

- Le codant 0 n'induit pas de changement de direction, c'est à dire que l'on continue dans la même direction que le codant précédent.
- Le codant 1 indique un changement de direction en avant par rapport au codant précédent.
- Le codant 2 indique un retour en arrière par rapport au codant pécédent.

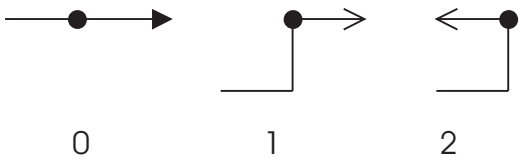

FIGURE 1.5 – Alphabet du code de Sanchez-Cruz, projection sur le plan des codes de Bribiesca.

Propriétés :

Les propriétés géométriques de ce code sont :

- l'**invariance par translation**,
- l'**invariance par rotation**,

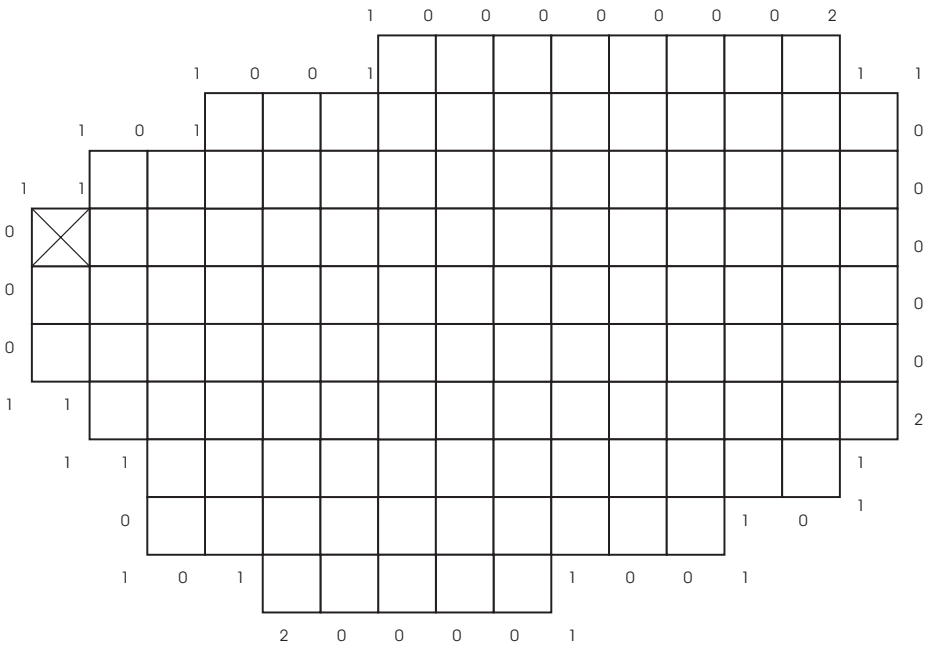

FIGURE 1.6 – Exemple de représentation du code de Sanchez-Cruz

- l'**invariance par symétrie par rapport à un point et par rapport à un axe**.

La figure 1.6 donne l'exemple d'un objet codé grâce à ce code.

1.1.3 Le Vertex Chain Code

Description :

En 1999, E. Bribiesca [Bri99] a présenté un autre moyen de coder les contours de formes : le Vertex Chain Code ou VCC. Le principe est de compter le nombre de sommets à l'intérieur de l'objet, et touchant le sommet considéré. Pour améliorer la compression, chacun de ces nombres est décrémenté d'une unité.

Dans le cas des pixels d'une image en trame carrée, il existe trois possibilités décrites respectivement par les symboles 0, 1 et 2. L'alphabet de ces symboles, correspondant respectivement à un, deux et trois sommets à l'intérieur de l'objet, est représenté figure 1.7.

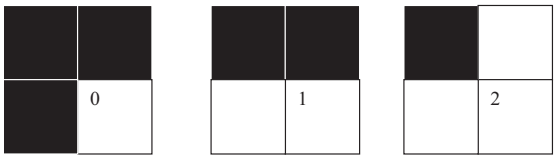

FIGURE 1.7 – Alphabet du Vertex Chain Code en trame carrée.

Propriétés :

Le Vertex Chain Code possède les propriétés suivantes :

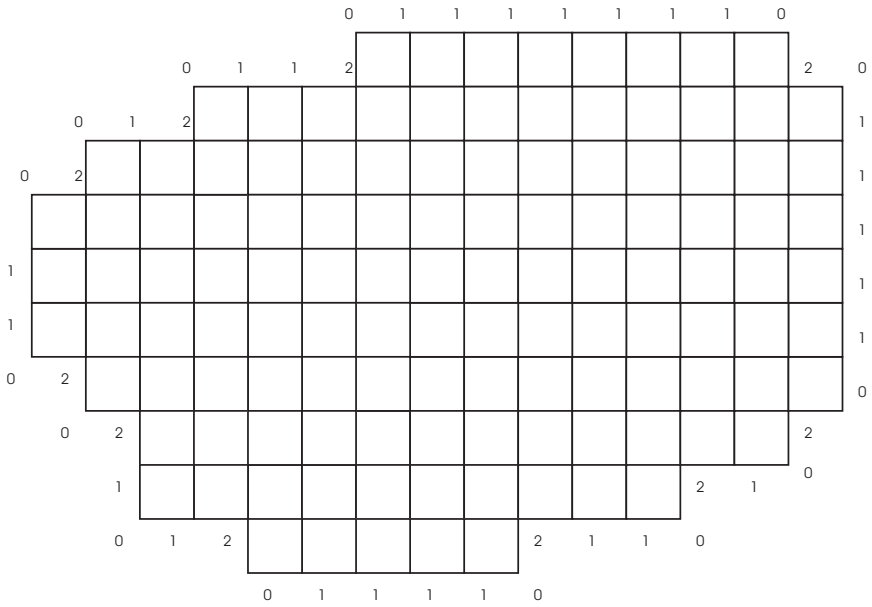

FIGURE 1.8 – Exemple de codage utilisant le Vertex Chain Code.

- Il est **invariant par rotation**.
- Il est **invariant par translation**.
- Il est **invariant par symétrie par rapport à un axe ou par rapport à un point**.

La figure 1.8 montre un objet codé grâce au Vertex Chain Code.

1.1.4 Le code de Liu et Zalic

Description :

Liu et Zalic [Liu] ont proposé en 2005 une variante du code de Freeman : on ne code plus la position relative du voisin du pixel courant, mais la variation d'angle entre le codant précédent et le codant suivant.

Ainsi, une variation d'angle de $+45°$ sera codée par le codant 1, et une variation de $-45°$ sera codée par le codant 2.
Le codant 3 représente une variation de $+90°$, le 4 : $-90°$, le 5 et le 6 respectivement $+135°$ et $-135°$.
Enfin le 7 symbolise un retour en arrière, c'est-à-dire une variation de $180°$ et le 0 le prolongement de la dernière direction.

Une illustration de ces huit variations de directions est présentée figure 1.9 page suivante ; il s'agit en fait de l'alphabet des symboles utilisés pour ce codage. Un exemple d'objet ainsi codé est présenté figure 1.10 page suivante.

Ce code proposé par Liu et Zalic est en fait une généralisation à huit directions du code de H. Sanchez-Cruz [San05] présenté au paragraphe 1.1.2.2 page 5.

Propriétés :

Le code de Liu et Zalic possède les propriétés suivantes :

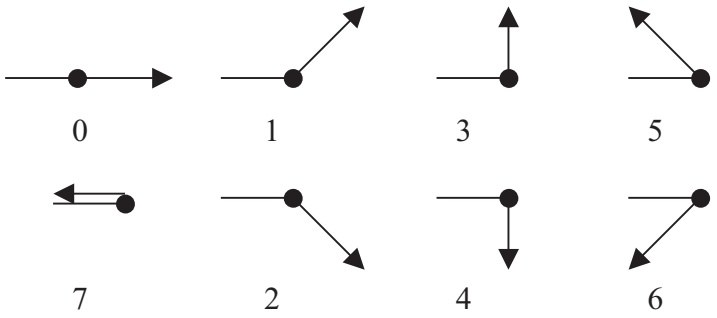

FIGURE 1.9 – Alphabet du code de Liu.

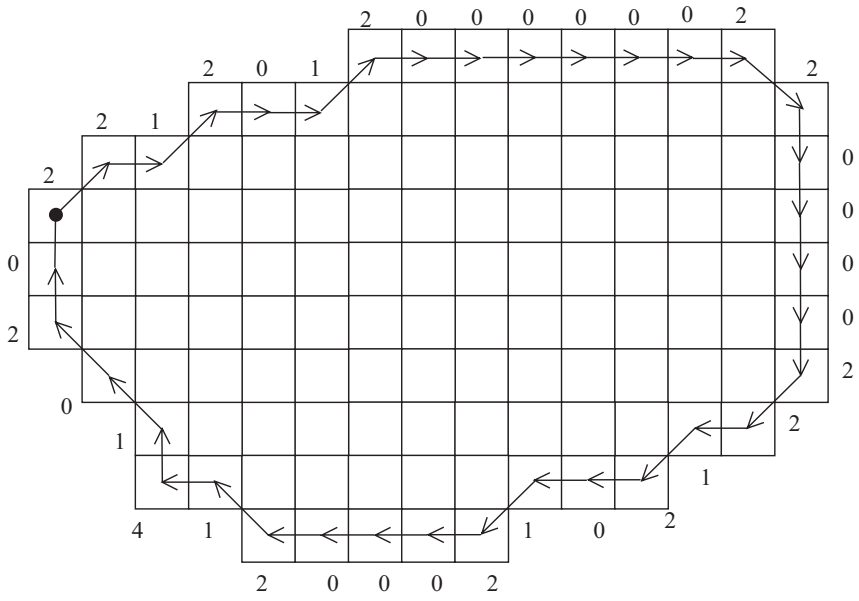

FIGURE 1.10 – Exemple d'objet représenté grâce à son code de Liu.

- **indépendance par rotation** d'angles multiples de 90°,
- **indépendance par translation**,
- **indépendance du point de départ**.
- la symétrie par rapport à un axe est obtenue en parcourant le code à l'envers et en inversant les 1 avec les 2, les 3 avec les 4, et les 5 avec les 6.

1.1.5 Le code de Zeng et Vasseur

Il existe un autre code développé pour décrire les frontières d'un objet. Il s'agit du code présenté par X. Zeng et C. Vasseur [Zen]. Ce code nécessite des connaissances a priori de l'objet étudié : détection de **segments**, détection de **portions de cercles**, connaissance de la **frontière** de l'objet et **périmètre**.

Le code du contour ne décrit donc que les segments et les portions de cercles. Une portion de cercle est codée par la lettre b alors qu'un segment de droite est codé par la lettre a.

Après détection du contour, les segments de droite de celui-ci sont codés de la façon suivante :

$$(a-m)(c-p)(d-\theta) \tag{1.1}$$

La première partie de ce code : $(a-m)$, signifie que le segment nommé m est une ligne (m est en fait un nombre entier repérant le numéro du segment). La seconde partie $(c-p)$ signifie que la longueur de ce segment représente $p\%$ du périmètre de l'objet (supposé fermé) et la dernière $(d-\theta)$ signifie que l'angle formé par ce segment et l'axe des x est de θ degrés dans le sens indirect.

Les arcs de cercle (AB de centre C) sont codés ainsi :

$$\begin{array}{c}(b-n_1)(c-p)(d-\theta_1)\\(b-n_2)(d-\theta_2)\end{array} \tag{1.2}$$

avec $n_2 = n_1 + 1$, $(b-n_1)$ et $(b-n_2)$ représentent les segments AC et BC, p est le pourcentage de la longueur AB dans le périmètre de l'objet, θ_1 et θ_2 sont respectivement les angles formés par l'axe des x et les segments AC et BC.

En plus des connaissances a priori qu'elle requiert, cette méthode est en quelque sorte une *vectorisation* et ne code plus tous les pixels de la frontière de l'objet. Pour ces raisons, l'obtention du code n'est pas aussi simple et rapide que dans les cas vus précédemment et nous avons décidé de ne pas approfondir l'étude de ce code.

1.1.6 Synthèse

Le tableau 1.11 page suivante résume les différents codes et leurs propriétés. Ils sont classés par ordre croissant de l'efficacité calculée par [San07]. Dans ce tableau, les codes donnés pour invariants par symétrie axiale, le sont à un détail près : il faut parcourir le code en sens inverse.

De plus, la longueur du code nous donne une indication sur la taille relative des codes obtenus. Ainsi, le code de Freeman à 4 directions, le VCC et le code introduit par Sanchez-Cruz d'une part et le code de Liu et Zalic et le code de Freeman à 8 directions d'autre part ont exactement la même longueur.

Enfin, les invariances par rotation ne prennent en compte que les rotations multiples de 90°.

Une caractéristique supplémentaire de ces codes est qu'ils sont tous indépendants du point de départ sur la frontière, à une permutation circulaire près du code.

Code	nombre de symboles	longueur du code	invariance par		
			rotation	translation	symétrie
Liu et Zalic	8	le plus court	oui	oui	non
Sanchez-Cruz	3	moyenne	oui	oui	oui
Vertex Chain Code	3	moyenne	oui	oui	oui
Freeman 4 directions	4	moyenne	non	oui	non
Tortue de Papert	2	le plus long	oui	oui	oui
Freeman 8 directions	8	le plus court	non	oui	non

FIGURE 1.11 – Différents codages de contour présentés et leurs propriétés.

1.2 Autres codes

1.2.1 Run-Length Encoding

Le code RLE (Run-Length Encoding) est un algorithme de compression d'images binaires sans perte de données. Il a pour but de coder ligne par ligne le nombre de pixels consécutifs ayant la même couleur. On l'utilise par exemple pour compresser l'image correspondant à une page de texte scannée ou pour l'envoi de fax. Comme il est fréquent sur de telles images d'avoir sur la même ligne plusieurs pixels de la même couleur (noir ou blanc) ce code est bien une compression de l'image de départ.

Prenons pour exemple une partie d'une ligne de fax représentée par des B pour les pixels noirs et des W pour les pixels blancs. On pourra avoir ce genre de séquence :
WWWWWWWWWWWWBBBWWWWWWWWBWWWWWWWWWWWWBBBWW qui sera alors remplacée par : 11W 3B 7W 1B 11W 3B 1W.

1.2.2 Le code de Huffman

Le code de HUFFMAN (1952) consiste à associer aux différents symboles des codes d'autant plus courts que ces symboles sont fréquents : il minimise la longueur moyenne pour une distribution de probabilité donnée. La perte de performance provient de la quantification des données : le catalogue formé doit être associé à l'objet compressé, il est bien sûr comptabilisé dans la taille totale du fichier.

Une étude ayant été faite dans le but d'utiliser la compression de Huffman, Liu et Zalic [Liu] ont démontré (en utilisant le code présenté sur la figure 1.9 page 8) sur un échantillon de 1000 objets que le codant 0 représente 45% des codants. Viennent ensuite les codants 1 et 2 avec chacun 25%, puis 3 et 4 avec 2%, et enfin 5, 6 et 7 avec moins de 1%. L'utilisation du code de Huffman est donc justifiée et apporte une compression de l'ordre de 40% dans les exemples donnés lors de cette étude.

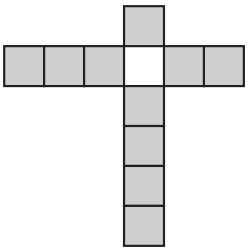

FIGURE 1.12 – Vecteur donnant le nombre de pixels de l'objet dans les quatre directions : $V = (1\ 2\ 4\ 3)$.

1.2.3 Neighbourhood Code

Une autre approche pour décrire les formes (l'étude porte actuellement sur les caractères écrits à la main) est celle présentée par I.J. Tsang [Tsa]. On ne code plus ici uniquement les pixels de la frontière de l'objet, mais tous les pixels de l'objet.

Le but de cet article est de reconnaître des caractères écrits à la main à l'aide d'un code utilisant les relations entre pixels voisins. Pour cela, chaque pixel de l'image est transformé en un vecteur donnant une information sur le nombre de pixels voisins dans les quatre directions (nord, est, sud, ouest). Ces vecteurs de voisinage sont transformés en un code donnant des informations sur la frontière de l'objet. On associe à chaque pixel un vecteur contenant le nombre de voisins dans les quatre directions nord, est, sud et ouest : $V = (n\ e\ s\ o)$. Considérant le pixel blanc de la figure 1.12, nous avons le vecteur $V = (1\ 2\ 4\ 3)$.

L'ensemble de ces vecteurs pour tous les pixels décrit parfaitement l'image.

Ce code présente une propriété d'invariance par translation et par rotation d'angles multiples de 90 degrés. Ces rotations induiront un décalage dans les coordonnées des vecteurs.

On utilise ensuite une foncion Φ de transformation des vecteurs $V = (n\ e\ s\ o)$ en un code $C = \Phi(n\ e\ s\ o)$. Cette transformation dépend de la taille de l'image $N \times N$. La plus grande valeur que puissent prendre n, e, s ou o est $N - 1$. La transformation a alors les conditions limites suivantes :

$$\begin{aligned} n + s &< N \\ e + o &< N \end{aligned} \tag{1.3}$$

Soient α et β les fonctions de transformation pour les paires (n, s) et (e, o). Sans tenir compte des conditions aux limites, on a :

$$\begin{aligned} \alpha &= nN + s \\ \beta &= eN + o \end{aligned} \tag{1.4}$$

En respectant les conditions aux limites, on obtient :

$$\begin{aligned} \alpha &= nN + s - \tfrac{n(n-1)}{2} \\ \beta &= eN + o - \tfrac{e(e-1)}{2} \end{aligned} \tag{1.5}$$

En posant par la suite $\gamma = \alpha K + \beta$ et $K = \beta_{max} + 1 = \frac{N^2+N}{2}$, on a :

$$\gamma = \alpha \frac{N^2 + N}{2} + \beta \tag{1.6}$$

Nous avons donc ici pour une imagette 16×16 pixels 18496 codes possibles, cette imagette étant de taille suffisante pour reconnaître un caractère.

Ce code peut ensuite être réduit pour obtenir la taille d'image désirée, par exemple pour passer d'une image 128×128 à une image 16×16.

Une fois la bonne taille obtenue, il suffit de comparer le code de l'imagette avec une base de données et ainsi retrouver le caractère présent dans l'image. Le caractère retenu est celui pour lequel l'erreur (par exemple l'erreur quadratique moyenne) est la plus faible.

1.3 Le code de Freeman

1.3.1 Description et implémentation

1.3.1.1 Description

En dimension 2, sur un maillage carré, on peut étudier les propriétés des objets en utilisant le voisinage à quatre points noté $V_4(i, j)$ ou le voisinage à huit points noté $V_8(i, j)$ d'un pixel (i, j) où i et j représentent respectivement le numéro de ligne et le numéro de colonne du pixel dans l'image. Ces voisinages sont représentés sur la figure 1.13. Ils correspondent respectivement aux quatre et huit pixels les plus proches du pixel (i, j).

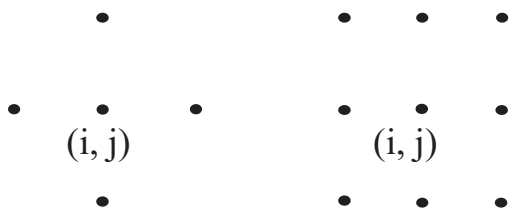

FIGURE 1.13 – Voisinages V4 (à gauche) et V8 (à droite) d'un pixel (i,j), i et j représentant respectivement le numéro de ligne et le numéro de colonne.

En considérant le cas des pixels carrés, le voisinage V_4 a la particularité suivante : tous les points du voisinage sont à une distance égale du point central. Cette distance vaut 1 pixel. On parle alors de 4-connexité. (Un objet pour lequel le passage d'un pixel à l'un de ses voisin ne se fait qu'à la verticale ou l'horizontale est 4-connexe.)

Pour le voisinage V_8, cette distance vaut également 1 pixel pour les pixels situés horizontalement ou verticalement par rapport au pixel central et $\sqrt{2}$ pixel pour les pixels situés en diagonale du point central. On parle alors de 8-connexité. (Un objet pour lequel le passage d'un pixel à l'un de ses voisin se fait à la verticale, l'horizontale ou en diagonale est 8-connexe.)

On remarque l'inclusion suivante :

$$V_4(i, j) \subset V_8(i, j) \tag{1.7}$$

3	2	1
4	X	0
5	6	7

	1	
2	X	0
	3	

FIGURE 1.14 – Code de Freeman à 4 ou 8 directions. X désigne le point courant et les chiffres correspondent aux 4 ou 8 directions possibles pour lesquelles X a un voisin appartenant à la frontière de l'objet.

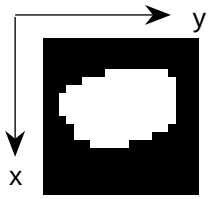

FIGURE 1.15 – Exemple d'objet digital.

Le code de Freeman, introduit la première fois en 1961 dans [Fre61a], a pour but de coder le contour d'un objet par :

- une information ***absolue*** correspondant aux coordonnées d'un point de départ,
- une chaîne de codants donnant la position ***relative*** du point suivant du contour de l'objet selon une des représentations présentées figure 1.14 (codage utilisant 4 ou 8 directions). Ainsi, en utilisant 8 directions, le codant 0 signifie que le pixel suivant du contour de l'objet se situe à droite du pixel courant, et le codant 5 désigne un pixel suivant en bas à gauche du pixel courant.

Chacune des 4 et 8 directions peut être codée respectivement sur 2 et 3 bits, induisant une forte compression sans perte de l'image.

Traditionnellement, ces directions sont comptées dans le sens trigonométrique (sens direct). La figure 1.15 présente un objet A blanc sur fond noir. Ainsi, pour le code utilisant 8 directions, les codants pairs lient deux pixels par une distance d'un pixel alors que les codants impairs lient deux pixels par une distance de $\sqrt{2}$ pixel.

On désire obtenir le code de Freeman d'un objet A considéré. Pour cela, on sélectionne un point appartenant à la frontière interne de A (par exemple le premier pixel rencontré par balayage électronique), on mémorise les coordonnées de ce premier point (ici, (3,8)) puis on cherche son plus proche voisin appartenant à A (au sens d'un voisinage V_4) selon un sens de rotation donné (ici, le sens direct). On réitère ensuite cette dernière opération jusqu'à revenir au point de départ.

Ainsi, de proche en proche, on reconstitue la forme de l'objet A en donnant le codage de Freeman [Fre74] de la frontière de A.

Cet exemple donne la chaîne 667760700001001012222234444444544545. Un schéma un peu plus détaillé du code obtenu sur cet objet est présenté figure 1.16 page suivante.

On peut également utiliser le codage différentiel qui donne

- les coordonnées du premier pixel du contour interne de l'objet (information absolue),
- le premier codant,
- la variation d'un codant à l'autre (information relative).

L'exemple ci dessus donne :
(3,8) 60+10-1+2-1+1000+1-10+1-1+1+10000+1+1000000+1-10+1-1+1.

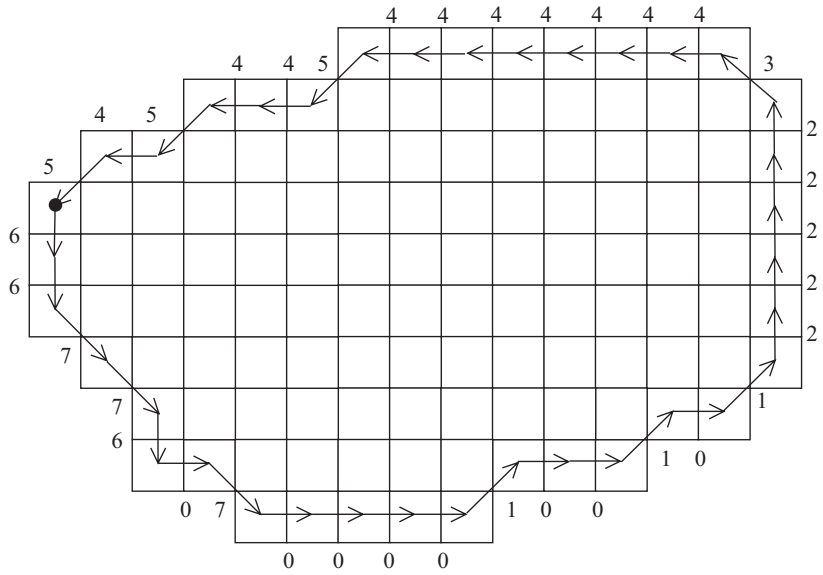

FIGURE 1.16 – Exemple d'objet et du code de Freeman à huit directions de son contour.

La frontière interne discrète de l'objet A est notée $FI(A)$, et est définie de la façon suivante :

$$x \in FI(A) \Leftrightarrow \begin{cases} x \in A \\ x \text{ a au moins un voisin dans } A^C \text{ le complémentaire de A} \end{cases} \tag{1.8}$$

De la même manière, on définit la frontière externe $FE(A)$:

$$x \in FE(A) \Leftrightarrow \begin{cases} x \notin A \\ x \text{ a au moins un voisin dans } A \end{cases} \tag{1.9}$$

Pour ces deux définitions, il faut préciser si le voisinage est considéré au sens V_4 ou au sens V_8.

On préfèrera $FI_{V4}(A)$ à $FI_{V8}(A)$ car les directions autorisées pour le suivi de la frontière sont plus nombreuses (8 au lieu de 4) et permettent ainsi un meilleur suivi de la frontière.

Connaissant un objet par un pixel de sa frontière interne et le code de Freeman à quatre directions des points de cette frontière, il est facile d'obtenir un code de Freeman à huit directions. En effet, il suffira d'associer deux à deux les codants (quatre directions) successifs différents et de leur faire correspondre le codant huit directions suivant le tableau donné

figure 1.18. Par exemple on pourra remplacer 32 (4 directions) par 5 (8 directions). Pour les autres cas, il suffit d'associer le codant huit directions correspondant au codant quatre directions.

Un tableau résumant tous les cas possibles est présenté dans ce même tableau.

Notons que le remplacement des codants 23 (4 directions) par 5 (8 directions) induirait la perte d'un pixel (voir figure 1.17). Il en est de même par rotation pour les autres cas présentés dans le tableau 1.18. L'odre des codants dans ce tableau est donc important.

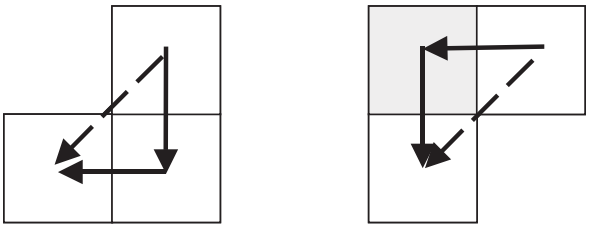

FIGURE 1.17 – Remplacement de 32 par 5 lors du passage de 4 à 8 directions, le remplacement de 23 par 5 induirait la perte du pixel grisé.

En revanche, la transformation d'un code utilisant 8 directions à un code n'utilisant que 4 directions n'est possible que si l'on met en évidence une connexion diagonale ou si l'objet est 4-connexe. Sinon, le passage d'un code de Freeman à 8 directions à un code de Freeman à 4 directions augmentera le nombre d'objets de l'image.

Codant quatre directions		**Codant correspondant**
paire de codants	**codant simple**	**(huit directions)**
3 2		5
0 3		7
1 0		1
2 1		3
	0	0
	1	2
	2	4
	3	6

FIGURE 1.18 – Passage d'un code à quatre directions à un code à huit directions.

A partir du code à huit directions décrivant le polygone passant par le milieu des pixels de la frontière interne d'un objet (flèches en pointillés sur la figure 1.19 page suivante), il est également possible d'obtenir le code à quatre directions décrivant le contour extérieur de ces mêmes pixels (flèches en gras sur la figure 1.19 page suivante).

A chaque codant a_i est associé un vecteur (a_{ix}, a_{iy}) correspondant aux déplacements en x et en y introduits par ce codant. La figure 1.20 page suivante présente ces vecteurs dans le cadre de notre étude. Le cas 1 donne ces déplacements pour les axes orientés en x croissant vers le bas et y croissant vers la droite, le cas 2 pour les axes orientés classiquement en x croissant vers la droite et en y croissant vers le haut.

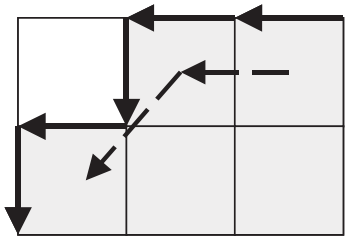

FIGURE 1.19 – Codage par les centres des pixels (flèches en pointillé) et par les contours des pixels (flèches en gras).

	Cas 1		Cas 2	
Codant	a_{ix}	a_{iy}	a_{ix}	a_{iy}
0	0	1	1	0
1	-1	1	1	1
2	-1	0	0	1
3	-1	-1	-1	1
4	0	-1	-1	0
5	1	-1	-1	-1
6	1	0	0	-1
7	1	1	1	-1

FIGURE 1.20 – Déplacements en x et en y induits par chaque codant.

1.3.1.2 Implémentation

Nous avons implémenté le code de Freeman à 4 et à 8 directions. Nous travaillons ici avec un seul objet connexe. Voici le fonctionnement du programme réalisé pour le code à 8 directions :

La fonction a les paramètres suivants :
- en entrée :
  - l'image binaire contenant l'objet dont on désire le code de Freeman
- en sortie :
  - les coordonnées (x, y) du point de départ
  - le code de Freeman de la forme, dans un tableau d'une ligne et n colonnes (n étant le nombre de codants)

L'algorithme développé est l'algorithme 2 page ci-contre.

Cet algorithme ne nécessite pas d'extraction de la frontière avant d'être exécuté et a donc l'avantage de travailler sur un objet binaire complet, et non pas uniquement sur la frontière de l'objet.

1.3.1.3 Remarques :

- Chaque codant est noté ici a_i. Un ensemble de codants forme une chaîne, et une chaîne de codants notée $a_1a_2...a_i...a_n$ donne la représentation du contour d'un objet. n représente le nombre de codants nécessaires pour revenir au point de départ.

Algorithme 2 Obtention du code de Freeman

1 recherche du premier point courant $P(x_0, y_0)$
2 recherche du premier codant à partir du premier point courant (voir quatrième remarque ci-après)
3 mise en mémoire du premier codant
4 nouveau point courant = point codé
5 **tant que** nouveau point courant $\neq P(x_0, y_0)$ **faire**
6 recherche du codant suivant
7 mise en mémoire du codant
8 **fin tant que**
9 **retourner** le chaîne de codants et $P(x_0, y_0)$

- On travaille ici en trame carrée, mais le code de Freeman s'obtient de façon identique si le pas en x est différent du pas en y.

- Les axes des x et des y pour les images étudiées ne sont pas orientés de façon usuelle ; la première coordonnée représente le numéro de la ligne et la seconde le numéro de la colonne de l'image. Ainsi, l'axe des x est vertical et dirigé vers la bas, et l'axe des y est horizontal et dirigé vers la droite, comme représenté sur la figure 1.15 page 13.

- La recherche du premier codant à partir du premier point courant est dissociée du reste du code car elle est un peu particulière. En effet, compte tenu de la méthode d'obtention du premier point de l'objet, par balayage de haut en bas et de gauche à droite, le point suivant ne peut se trouver qu'avec les codants 6, 7, 0 ou 1. Cette particularité est due au fait qu'un pixel est d'abord repéré par son numéro de ligne et ensuite par son numéro de colonne. Certains logiciels comme Matlab utilisent d'ailleurs ce balayage par défaut. Sur l'exemple de la figure 1.21, le premier codant sera donc 6.

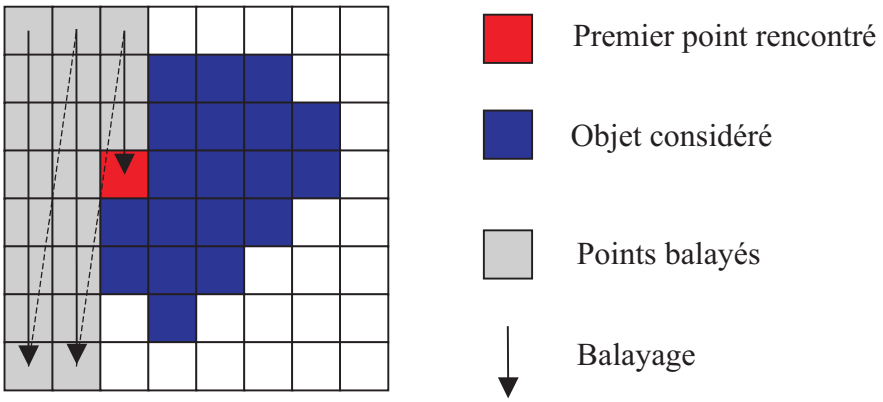

FIGURE 1.21 – Exemple d'obtention du premier point et du premier codant. Le balayage se fait de haut en bas et de gauche à droite.

- La recherche du codant suivant est faite comme suit : comme on désire tourner autour de la forme dans le sens trigonométrique, cela signifie que, dans ce sens de parcours, la forme sera toujours à gauche. Pour cette raison, si le dernier codant trouvé est a, on recherchera le suivant en commençant par $a-2[8]$ (où $[8]$ signifie modulo 8). Si le point

suivant n'est pas à cette position, on le cherchera alors en $(a-2)+1$, puis $(a-2)+2$, et ainsi de suite jusqu'à trouver le point suivant et donc le codant suivant. Illustrons ceci par l'exemple de la figure 1.22.

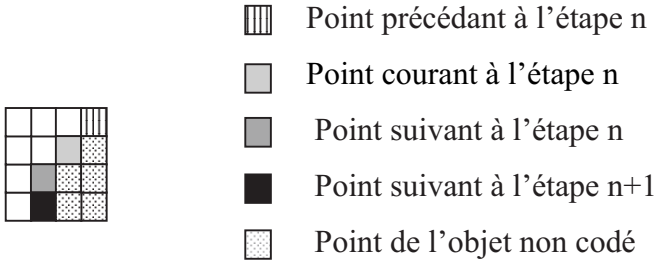

FIGURE 1.22 – Exemple d'obtention d'un codant.

Sur cet exemple, considérant le point courant à l'étape n, le code précédent est 5. D'après l'algorithme, on cherchera le point suivant à partir du codant 3 puis 4 ; on trouvera à nouveau le codant 5. A l'étape $(n+1)$, on cherchera à partir du codant 5, et on trouvera comme prochain codant 6. A l'étape $(n+2)$, on cherchera donc à partir du codant 3 et ainsi de suite jusqu'à revenir au point de départ.

- Avec le sens de rotation choisi au départ, la recherche du codant suivant a se fait bien à partir de $(a-2)$ et non pas $(a-3)$ car un codant a ne peut pas être suivi par un codant $(a-3)$. Montrons ceci par le contre-exemple suivant : la représentation du code ...4524... est illustrée sur la figure 1.23.

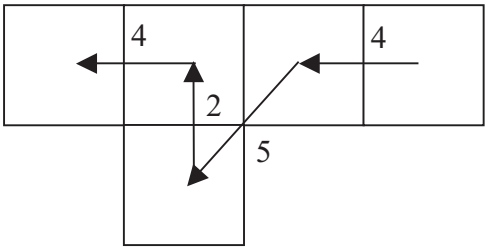

FIGURE 1.23 – Exemple de code impossible avec le sens de rotation trigonométrique choisi au départ : 4524 sera remplacé par 444.

En revanche le code correspondant à cette partie de frontière est ...444... Le codant 5 ne peut donc pas être suivi du codant 2. De plus, le codant 0 ne peut également pas être suivi par le codant 6. En effet, la représentation de la chaîne 06 se code par le codant 7.

Les combinaisons suivantes sont donc impossibles : 05, 06 et 16 ainsi que toutes les rotations multiples de 90° correspondantes.

- Une vérification doit être faite lorsque l'on est revenu au point de départ : l'objet entier a-t-il été codé ? En effet, si le pixel de départ (premier pixel trouvé au sens d'un balayage illustré figure 4.7 page 99) est un pixel se trouvant sur la frontière à deux endroits différents, c'est-à-dire que l'objet n'a qu'une épaisseur d'un pixel à cet

endroit, il ne faut pas arrêter le codage de la forme. Par conséquent, de retour au point de départ, si l'objet n'a pas été codé complètement, il faut coder le reste. L'exemple d'un tel cas est donné sur la figure 1.24 où le premier pixel au sens du balayage utilisé est marqué d'une croix, la partie de l'objet codée sans prendre de précaution est en blanc et la partie à ajouter est en gris.

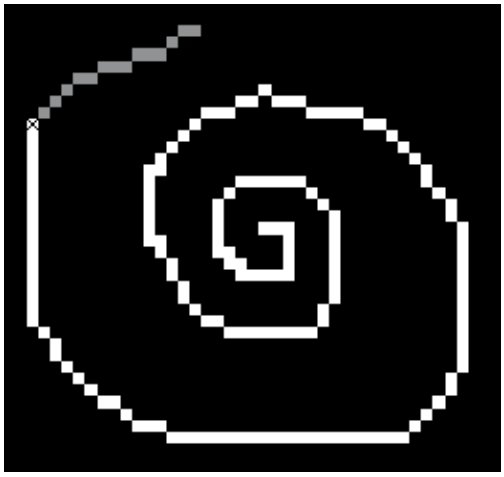

FIGURE 1.24 – Cas particulier de codage : le pixel de départ est un pixel où le code va passer deux fois.

- Voici un exemple de code obtenu sur une roue dentée (voir figure 1.25) :
  Point initial : $x_0 = 36$, $y_0 = 8$
  code : (276 codants) 66666660007000766654555476776701001007766566567077072111210007666766600000002221222100076777610110123223221100700701211214333432221000100022222224443444322210111032332344545443322212212343343655565444322322244444446665666544432333254554567667665544343456556507770766654445444

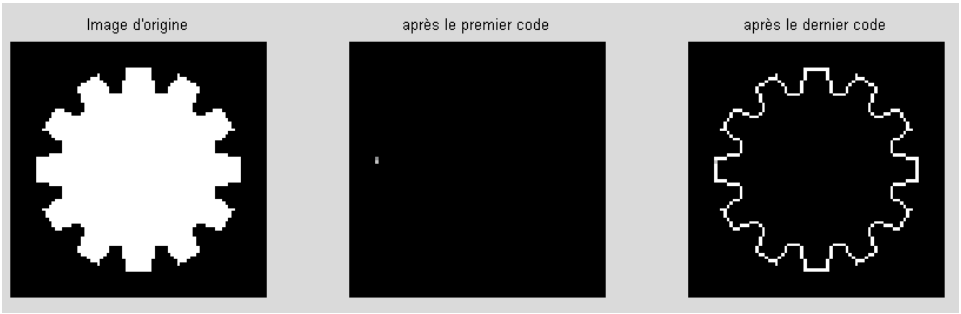

FIGURE 1.25 – Exemple de code de Freeman.

- Ce programme a été amélioré pour prendre en compte les objets touchant le bord de l'image. En effet, la recherche du codant suivant se fait en parcourant les pixels voisins afin de trouver le premier pixel allumé. Si le point courant est un pixel du bord de l'image, la recherche se fera sur un pixel hors de l'image, et le programme se terminera suite à une erreur de dimension. Pour remédier à ce problème, si l'objet touche le

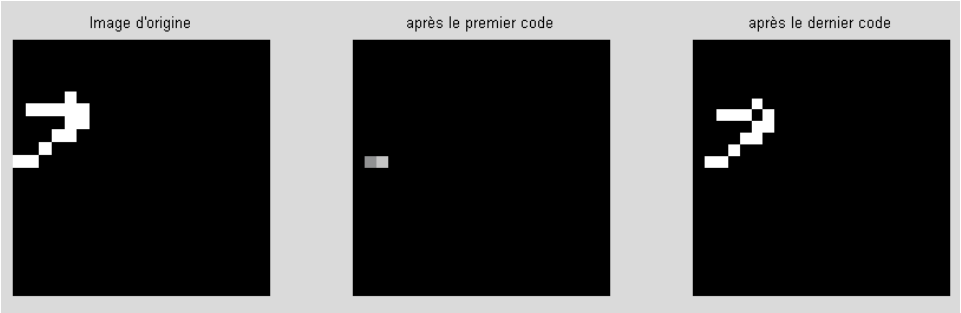

FIGURE 1.26 – Exemple d'objet touchant le bord de l'image. Point initial : $(x0, y0) = (10, 1)$. Code : (17 codants) 0 1 1 0 1 2 3 5 4 4 0 0 7 5 5 5 4.

bord, nous ajoutons une bande noire de largeur un pixel, tout autour de l'image, avant d'effectuer le codage. Bien sur, en fin de codage, l'erreur induite par cette méthode sur les coordonnées du premier point est corrigée. Un exemple de résultat obtenu, avec un objet touchant le bord est donné figure 1.26.

Nous sommes alors en mesure d'utiliser le codage de Freeman afin d'estimer certains paramètres de l'objet.

1.3.2 Utilisations du code de Freeman

1.3.2.1 Reconnaissance de caractères chinois

Le code de Freeman est utilisé pour la reconnaissance de caractères chinois écrits à la main par A. Amin [Ami] : après une numérisation de l'écriture, le squelette des caractères est extrait par la méthode des boules maximales(voir[Blu]) et analysé grâce au codage de Freeman. Si nécessaire, ce code de Freeman du contour est simplifié par filtrage. Les caractéristiques sont ensuite extraites et la forme est classifiée.

Ici, A. Amin utilise le squelette ; cependant l'opération de squelettisation n'est pas une application continue dans l'espace des formes muni de la distance de Hausdorff. Deux formes voisines peuvent avoir des squelettes très différents et la reconnaissance d'objet par cette méthode n'est donc pas robuste.

Il est cependant théoriquement possible de revenir à l'objet compact de départ à partir du squelette si pour chaque point de celui-ci est conservé en mémoire le rayon de la boule maximale centrée en ce point contenue dans la forme de départ. Ainsi, par union de toutes ces boules, l'objet initial peut être retrouvé.

1.3.2.2 Correction de caractères arabes manuscrits

Une application similaire est réalisée par [Mad] afin d'effectuer une correction géométrique des caractères arabes manuscrits. L'utilisation du code de Freeman est cependant différente. Ici, après détection du contour du caractère et obtention de son code de Freeman, on réalise une transformée de Fourier discrète (FFT). En effet, à partir du code de Freeman obtenu, on génère une fonction temporelle périodique sur laquelle le calcul de la FFT est réalisable.

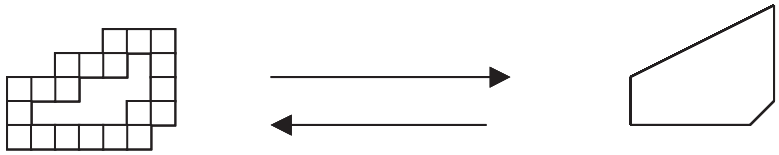

FIGURE 1.27 – Exemple d'objet discret et sa polygonalisation par Breton.

La notion de temps est ici associée à la durée du parcours du contour de sorte que le temps nécessaire pour parcourir les k premiers arcs qui relient les points du contour est :

$$t_k = \sum_{i=1}^{k} \Delta t_i \tag{1.10}$$

avec $\Delta t_i = 1$ pour un codant pair et $\sqrt{2}$ pour un codant impair. Les distances parcourues à l'instant t_k sont décrites par x_k (projection du contour sur l'axe des X) et y_k (projection du contour sur l'axe des Y) avec :

$$\begin{array}{l} x_k = \sum_{i=1}^{k} \Delta x_i \\ y_k = \sum_{i=1}^{k} \Delta y_i \end{array} \tag{1.11}$$

où $\Delta x_i \in \{-1, 0, 1\}$ suivant la direction donnée par le code de Freeman. Ainsi, le contour peut être décrit par la fonction discrète (x_k, y_k) sur laquelle on calculera la transformée de Fourier.

1.3.2.3 Simplification de polygones

Ce code est également utilisé dans le codage et la simplification de polygones. O. Rogalla [Rog] utilise la programmation par la démonstration, afin de programmer des robots. Une des étapes est l'extraction de contours de formes, puis le codage de Freeman des frontières et une simplification afin d'obtenir un polygone pour classifier et utiliser la forme simplifiée obtenue.

L. Dorst [Dor] utilise les propriétés du codage de Freeman pour extraire d'une forme quelconque la (n)-caractérisation, la (n_e, n_o)-caractérisation, la (n_e, n_o, n_c)-caractérisation et la (n, p, q, s)-caractérisation. Celles-ci donnent respectivement le nombre d'éléments de la chaîne (n), le nombre d'éléments pairs (n_e) et impairs (n_o) de la chaîne, le nombre de "coins" dans la chaîne (n_c). Les paramètres n, p, q, s ont été introduits en 1984 dans [Dor84].

J. Kim [Kim] présente la notion d'approximation de formes par des polygones. Il utilise le code de Freeman pour coder la frontière des objets avant de réaliser l'approximation.

R. Breton [Bre] travaille également avec des polygones. Il propose un algorithme de vectorisation (reconstruction euclidienne) d'une forme discrète 4-connexe inversible et proche d'un résultat intuitif. Le but de cet algorithme est illustré par la figure 1.27.

La méthode est la suivante : à partir d'un point de la courbe, on recherche le segment auquel ce point appartient. On réitère ensuite le processus jusqu'à obtention de tous les segments constituant l'objet que l'on peut alors reconstruire.

Le code de Freeman est utile ici afin de déterminer les points de rebroussement. Une perspective de ce travail est maintenant d'étudier ces points de rebroussement car l'algorithme échoue en cas de tels points. Une autre perspective consiste en une adaptation vers la 3D.

Amélioration possible : La détection de segments de droites doit pouvoir se faire à partir du code de Freeman du contour de l'objet. En effet, un segment de droite a un code de Freeman périodique. En travaillant sur la recherche de périodes dans le code (Recherche systématique, Fourier...) on peut déterminer les segments de droites composant le polygone que l'on cherche à caractériser.

Nous envisageons ici une polygonalisation d'objets binaires à partir du code de Freeman et des algorithmes de Debled-Renneson et Reveillès (voir [Rev] et [Deb]).

1.3.2.4 Reconnaissance de profils de visages humains

Il existe plusieurs solutions pour reconnaître des visages parmi lesquelles :
- Le codage de Freeman pur du contour du visage, muni d'une distance entre 2 codes,
- La vectorisation du contour du visage muni d'une distance,
- La caractérisation de points particuliers et leur comparaison.

Ce dernier point a été étudié par D. Odin [Odi] en 1995. L'étude se déroule en plusieurs points :

- Cadrage du profil :
  Le profil utilisé est le profil gauche afin d'être en harmonie avec les fichiers anthropométriques de la police. Le cadrage doit être relativement serré afin que l'ensemble de la tête soit visible. Le sujet doit être imberbe afin de ne pas fausser les mesures. On doit voir l'oreille dans sa totalité afin de pouvoir caractériser une ellipse qui l'englobe.

- Prétraitement de l'image :
  On réalise un filtrage puis un seuillage de l'image afin d'obtenir le contour du visage, puis une reconstruction. On détecte ensuite le contour grâce au code de Freeman. On obtient ainsi le code du profil gauche de l'individu dont on veut extraire les caractéristiques physiques.

- Détection et caractérisation de neuf points de la silhouette du profil :
  Ces points sont les suivants : le bout du nez, le bout du menton, le bas du nez, le coin de la bouche, le bas de la bouche, la lèvre supérieure, la lèvre inférieure, le front et le bas du front. Ces produits sont recherchés comme des points extrémaux locaux sur le code de Freeman du contour du profil.

- Détermination et caractérisation du coin de l'oeil :
  On utilise ici une méthode empirique qui consiste en une suite d'étapes visant à rapprocher un point courant du véritable coin de l'oeil. Elle repose en partie sur la caractérisation des points du profil.

- Détection et caractérisation des grains de beauté éventuels :
  Cette détection repose sur le fait que les grains de beauté, supposés assimilables à des cercles, ont une certaine uniformité des niveaux de gris le long d'un cercle donné.

- Détection et caractérisation d'une ellipse d'approximation de l'oreille :
  La méthode utilisée a été développée à partir du concept de "'templates"' déformables.

1.4 Conclusion

Nous avons décrit les codages de contours les plus connus existant actuellement. Notre étude porte maintenant sur l'utilisation du code de Freeman. Nous avons choisi d'approfondir ce codage car il est apparu dans les premiers en 1974 et nous paraît plus adapté à l'étude des paramètres de formes, du fait d'une part de l'information absolue donnant les coordonnées du premier point du contour, et d'autre part de l'information relative donnant les positions des pixels voisins.

Enfin, il est possible de passer du code de la tortue de Papert, ou de celui de Sanchez-Cruz, du VCC ou du code de Liu et Zalic au code de Freeman de façon bijective. Une étude similaire de ces codes n'est donc pas nécessaire.