4/00

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique
Département d'Electronique

Mémoire de Fin d'Etudes pour l'Obtention du Diplôme d'Ingénieur d'Etat en Electronique

Les Algorithmes Génétiques Appliqués à la Segmentation Des Images de Textures

Proposé par :

L. HAMAMI

Réalisé par :

K. RAMDANI
M. A. ROULA

Examiné par :

C. LARBES — A. BELOUCHRANI — B. BOUSSEKSOU

1999/2000

DEDICACES

A mes chers parents, à ma soeur,
A ma grande famille,
A tous les membres de ma famille à Alger,
A tous mes amis,

Je dédie ce modeste travail
Mohammed Ali

A mes très chers parents,
A toute ma famille,
A tous mes amis, en particulier Samir,

Je dédie mon travail
Krimo

Remerciements

Ce modeste travail a été réalisé sous la direction scientifique de Madame ***L. HAMAMI*** *chargée de cours à l'ENP, à qui nous présentons l'expression de notre profonde gratitude pour tous les conseils et les encouragements, qu'elle nous a prodigués pendant toute la durée de ce travail.*

Nous remercions très vivement Monsieur ***C. LARBES,*** *PhD chargé de cours à l'ENP, de l'honneur qu'il nous fait en acceptant de juger notre travail et de présider le jury de thèse.*

Nous sommes très reconnaissants également à Monsieur ***A. BELOUCHRANI,*** *docteur chargé de cours à l'ENP, pour l'intérêt qu'il a bien voulu porter à ce travail, en acceptant de l'examiner.*

Que Monsieur ***B. BOUSSEKSOU*** *chargé de cours à l'ENP soit vivement remercié pour l'intérêt qu'il manifeste pour ce travail en participant à ce jury de thèse.*

A toutes les personnes qui ont contribué de près ou de loin, directement ou indirectement à l'aboutissement de ce travail, en particulier Messieurs A. BOUDAIEB, A. AMIRA, K. CHERCHALI et d'autres qui se reconnaîtront, que tous, trouvent ici le témoignage de notre profonde reconnaissance.

ملخص:

إن تقسيم الصور يعد مرحلة هامة في عملية التعرف على الأشكال. في حالة الصور ذات النسيج المركب. التقسيم يحتاج الى وضع نموذج رياضي و الى أحسن تقييم لمعا ملات النموذج المختار. إن الخوارزميات الجينية، هي خوارزميات تحسين قوية، مستوحات من التطور البيولوجي. في هذا العمل تستعمل هذه الأخيرة في تقسيم الصور ذات التركيب المصاغة في إطار الحقول العشوائية لماركوف، و كذلك في إستشعار الحيز الخارجي بإستعمال المميزات الكسورية و نموذج التراجع الذاتي للصورة.

الكلمات المفتاحية: الخوارزميات الجينية، التقسيم، إستشعار الحيز الخارجي، النسيج الصورة، حقول ماركوف، البعد الكسوري، نموذج التراجع الذاتي.

Résumé :

La segmentation d'images est une étape essentielle pour la reconnaissance de formes. Dans le cas d'images de textures, la segmentation requière une modélisation mathématique et une estimation optimale des paramètres du modèle choisi. Les algorithmes génétiques sont des algorithmes puissants d'optimisation inspirés de l'évolution biologique. Dans cette application il sont utilisés pour la segmentation de textures modélisées par les champs aléatoires de Markov, et aussi, pour la détection de contours par utilisation d'attributs fractals et du modèle A R de l'image.

Mots clés : *algorithmes génétiques, segmentation, détection de contours, textures, champs de Markov, dimension fractale, modèle AR-2*

Summary:

Image segmentation is an important step for pattern recognition. For textured images, the segmentation needs mathematics modelling and optimal estimation of model parameters. Genetic algorithms are powerful optimisation algorithms inspired from biologic evolution. In this application, they are used for segmentation of textures modelled by Markov random fields, and also, for edge detection using fractal attributes and AR model.

Keywords: *genetic algorithms, image segmentation, edge detection, texture, Markov random fields, fractal dimension, AR model.*

Sommaire

Sommaire
Introduction générale 1
Chap I : Généralités & Prétraitement 3
I.1- Introduction 3
I.2- Généralités sur l'image 4
I.2.1- Définition 4
I.2.2- L'image numérique 4
I.2.3- Système de traitement d'images 5
I.2.4- La segmentation 7
I.2.5- Les textures 7
I.2.6- Les attributs d'une image 7
I.3- Le prétraitement et la détection de contours 11
I.3.1- La modification d'histogramme 11
I.3.2- Filtrage du bruit 14
I.3.3- Le filtrage de Dériche 15
I.3.4- Résultats sur le prétraitement 16
I.4- Conclusion 20
Chap II : Les Algorithmes Génétiques 21
II.1- Introduction 21
II.2- Principes généraux 22
II.2.1- Etapes suivies dans l'exécution d'un AG 23
II.2.2- Lexique général 23
II.2.2- Description 25
a- Codage des données 25
b- Génération aléatoire de la population initiale 25
c- Gestion des contraintes 25
d- Opérateur de croisement 26
e- Opérateur de mutation 27
g- Principes de sélection 28
II.3- Améliorations classiques 29
II.3.1- Le Scaling 29
II.3.2- Le Partage (sharing) 32
II.3.3- Association des AG avec des méthodes locales 35
II.3.4- Le parallélisme 35
II.4- Description rapide d'un algorithme génétique 37
II.4.1- Mutation $X_k \xrightarrow{Mutation} Y_k$ 38
II.4.2- Croisement $Y_k \rightarrow Z_k$ 38
II.4.3- Sélection $Z_k \rightarrow X_{k+1}$ 38
II.5- Conclusion 39
Chap III : Segmentation par Champs de Markov et Algorithmes Génétiques 40
III.1- Introduction 40

III.2- Le modèle Markovien de textures 41
III.2.1- Définitions 41
III.2.2- Théorème de *Hammersley-Clifford* 42
III.2.3- Approximation de la fonction de partition 43
III.3- La relaxation sélectionniste 44
III.3.1- Les unités 45
III.3.2- L'algorithme génétique 45
III.4- Un algorithme supervisé pour la segmentation des images texturées et non texturées 46
III.4.1- Description 46
III.4.2- Prise d'échantillons 48
III.4.3- Phase de segmentation 48
III.5- Conclusion 49
Chap IV : Segmentation par Algorithmes Génétiques et Attributs Fractals **50**
IV.1- Introduction 50
IV.2- Caractérisation de la texture 51
IV.3- La détection des régions contours candidates 53
IV.4- L'algorithme génétique 54
IV.4.1- Le Codage 54
IV.4.2- L'opérateur de croisement 55
IV.4.3- L'opérateur de mutation 56
IV.4.4- La fonction d'adaptation 56
IV.5- Optimisation des régions candidates par algorithme génétique 56
IV.6- Conclusion 57
Chap V : Résultats et interpretations **58**
V.1- Présentation du Logiciel 58
V.2- Aspects de programmation 59
V.3- Les résultats 60
V.3.1- Méthode de la relaxation sélectionniste 60
V.3.2- L'algorithme supervisé de segmentation 64
Conclusion generale **68**
Annexe 1 : Approximation de la fonction de partition **69**
Annexe 2 : Démonstration et Implantation du filtre de Dériche **71**
Annexe 3 : Eléments de la théorie fractale et multifractale **77**
Bibliographie **80**

Introduction générale

Le traitement numérique d'images s'est imposé ces dernières années comme une discipline à part entière. Cela est dû essentiellement à l'évolution spectaculaire du matériel informatique et la nécessité –de plus en plus croissante- de remplacer l'être humain dans beaucoup d'activités. Cette évolution s'est accompagnée par le développement des applications du traitement d'images dans des domaines très variés. En effet, les techniques de reconnaissance de formes et d'analyse numérique d'images se sont révélées très fructueuses dans des disciplines telles la robotique, la reconnaissance des caractères imprimés et des visages, l'imagerie médicale ainsi que dans la télédétection (Images SAR) et l'astronomie.

La segmentation constitue une étape essentielle pour la reconnaissance de formes, car les algorithmes de reconnaissance opèrent souvent sur des objets ou sur des contours bien séparés du fond de l'image. La segmentation consiste, en fait, à extraire l'information utile –à reconnaître– du signal aléatoire bidimensionnel qu'est l'image.

On sait, depuis longtemps, segmenter et détecter les contours des images dont les régions sont différenciées par leurs niveaux de gris. Il n'en est pas de même pour les images de textures qui restent un véritable défi technique. Les articles dans ce domaine innovent chaque fois des algorithmes plus efficaces, plus rapides et plus robustes. Il s'agit, dans la majorité des cas, de modéliser mathématiquement les textures. Le problème est alors d'estimer les paramètres du modèle choisi de façon optimale et avec le moins d'informations, a priori, possibles et c'est ici que peut intervenir l'optimisation par les algorithmes génétiques.

L'évolution biologique a engendré des systèmes vivants autonomes extrêmement complexes qui peuvent résoudre des problèmes très difficiles, tels l'adaptation continuelle à un environnement complexe, incertain et en constante transformation. Pour cela, les êtres vivants supérieurs, comme les mammifères, sont dotés de capacités inégalées de reconnaissance de formes, d'apprentissage et d'intelligence. La grande variété des situations auxquelles la vie s'est adaptée, laisse penser que le processus de l'évolution est capable de résoudre –en trouvant les meilleures solutions- de nombreux problèmes, c'est à dire qu'il est robuste. L'idée était de simuler la nature pour résoudre des problèmes mathématiques. Les algorithmes génétiques fonctionnent en générant une population de solutions possibles, en la

faisant évoluer en privilégiant les meilleurs individus, nous obtiendrons à la fin la meilleure solution à notre problème : dans notre cas celui de la segmentation.

Cette thèse est organisée de la façon suivante :

Le premier chapitre traite du problème de la segmentation d'images en général. Au début nous allons décrire les différents attributs qui caractérisent les images, l'accent sera mis sur les attributs de textures (statistiques, fractales...). Nous avons préféré inclure dans ce même chapitre la partie prétraitement dans laquelle des opérateurs de lissage et de réduction de bruit y sont présentés ainsi que les détecteurs de contours, notamment le filtre de Dériche.

Dans le deuxième chapitre, nous abordons les algorithmes génétiques, où seront décrits les principes généraux de l'optimisation par les AG, les opérateurs de base tels la sélection, la mutation et le croisement ainsi que les manières qui sont susceptibles d'améliorer leurs performances en termes de convergence et de temps de calcul, comme par exemple la mise en échelle, le partage et le parallélisme. Nous avons exprès évité d'entrer dans des détails théoriques sur les algorithmes génétiques. En effet, il est difficile de faire une théorie générale car leur utilisation dans un cas sera très différente dans un autre cas.
Par la suite nous décrirons deux algorithmes de segmentation :

Le premier, au chapitre III, est un algorithme non supervisé qui utilise la discrimination des textures en les modélisant par les champs de *Markov*. L'estimation des paramètres de textures et l'attribution des labels de région se fait par un algorithme génétique distribué. Dans ce même chapitre nous proposerons un algorithme supervisé pour la segmentation d'images texturées et non texturées.

Le deuxième algorithme, présenté au chapitre IV, est un détecteur de contours qui utilise l'estimation de la dimension fractale (calculée avec d'autres paramètres) et la fixation d'un critère de décision (contour ou non contour). L'arrangement des régions contours se fait par un algorithme génétique combinatoire.

Les résultats obtenus, ainsi que leurs interprétations, seront donnés au chapitre V.

Chapitre I

Généralités & Prétraitement

I.1- Introduction

Nous aborderons dans ce chapitre quelques généralités sur l'image et son traitement à commencer par l'acquisition et la numérisation.

La segmentation des images - en particulier les images texturées- nécessite une modélisation mathématique de la distribution aléatoire des pixels. Selon la modélisation choisie, on a besoin de mesurer –ou d'estimer- des attributs sur les régions de façon à pouvoir les discriminer. Avant d'aborder les algorithmes de segmentation, nous présenterons dans ce chapitre les principales grandeurs -calculables- sur l'image et exploitables par la suite dans les algorithmes des chapitres III et IV.

Les images issues de l'acquisition sont souvent affectées de bruits divers causés par le système d'acquisition lui-même ou bien par des déformations inhérentes à l'environnement de prise de vue. En plus des bruits, l'image brute peut avoir une dynamique de luminance très réduite et être trop sombre ou trop éclairée, c'est à dire peu contrastée. A cause de toutes ces déformations, un prétraitement s'impose. Celui-ci a pour but de réduire les bruits autant que possible et de rehausser le contraste de l'image pour que le traitement soit le plus efficace possible. Nous aborderons quelques opérateurs de réduction de bruits ainsi les techniques de rehaussement les plus utilisées, en particulier celles qui agissent sur l'histogramme de l'image.

La détection de contours, bien que faisant partie de la segmentation (Approche Contours), est mentionnée traditionnellement avec le prétraitement. Nous avons testé les opérateurs classiques de détection de contours (*Sobel, Roberts, Laplace*...), mais nous ne citerons que le filtre de *Dériche* du fait que les autres sont souvent mentionnés dans les thèses de traitement d'images.

I.2- Généralités sur l'image

I.2.1- Définition

Dans une scène visuelle ou dans la représentation de celle-ci dans le plan focal d'un système optique, on peut associer à tout point une fonction : $L(x, y, t, \lambda)$.

où :

x,y :sont les coordonnées du point noté s.

t : représente le temps.

λ :est la longueur d'onde (la couleur).

Cette fonction représente l'image de la scène en question. Elle peut être réduite dans la majorité des cas à une fonction A(x,y) s'il n'y a pas d'évolution temporelle, comme c'est le cas pour les images animées. En l'absence de couleur c'est à dire pour les images en niveaux de gris, on parle alors d'images statiques monochromatiques. Ce dernier type constitue l'objet de notre étude.

I.2.2- L'image numérique

Les dispositifs de traitement du signal les plus efficaces sont numériques. Cela englobe les supports de stockage et de communication, ainsi que les méthodes de compression et les instruments de calcul. Afin pouvoir utiliser ces dispositifs pour le traitement d'images il faudra *numériser* notre image. Cela se fait par un *échantillonnage* puis une *quantification* de l'intensité lumineuse du signal image. Après ces deux étapes, on obtient une image digitale qu'on peut représenter par une matrice $(A_{I,j})$ dont les éléments sont l'intensité lumineuse d'un faisceau de lumière monochromatique réfléchie par l'objet et qu'on appelle *niveaux de gris*. Dans ce travail on a utilisé les notations suivantes : $A[s], A[i, j]$ pour représenter l'image numérique.

Le nombre de ces niveaux de gris dépend des applications, il peut aller de 2 pour les images en noir et blanc jusqu'à plusieurs milliers pour la haute définition. La taille de la matrice dépend aussi des applications et du système de prise de vue. Les valeurs courantes sont : 256*256,512*512.

a- Echantillonnage bidimensionnel

Il est effectué sur la fonction continue de luminance, en prélevant, à des intervalles réguliers des échantillons selon les directions x et y. Ces échantillons doivent être suffisamment rapprochés pour représenter assez bien la fonction continue d'origine. En effet, le théorème d'échantillonnage de *Shanon* s'applique pour les signaux bidimensionnels. Il stipule que :

« L'information sur l'image n'est pas perdue si les fréquences d'échantillonnage suivant i et j sont supérieurs au double des plus hautes fréquences contenues dans l'image. »- il s'agit bien entendue de fréquences spatiales.

b- La Quantification

Cette opération est effectuée en divisant la gamme dynamique de la luminance d'une image en un nombre fini d'intervalles et en attribuant à toute valeur de l'intervalle une seule valeur de luminance. Le nombre d'intervalles dépend des caractéristiques du système visuel humain ainsi que du support physique sur lequel on désire reproduire l'image quantifiée. En

effet il est inutile de coder l'image sur 1024 niveaux de gris si un écran ou une imprimante ne peut en restituer que 256.

La distribution des intervalles de quantification peut être uniforme. Dans ce cas, on parle de quantification linéaire où la même importance est donnée à toutes les gammes de luminances. L'œil étant très sensible dans l'obscurité –vision nocturne par bâtonnets –on peut donc utiliser une loi de quantification plus serrée vers le noir et plus espacée vers le blanc. On utilise alors une loi non linéaire pour s'adapter à l'observateur la loi la plus utilisée est la loi logarithmique.

I.2.3- Système de traitement d'images

Le traitement d'images est né de l'idée de remplacer l'observateur humain par la machine. La vision intervenant dans un très grand nombre d'activités, le champ d'application du traitement d'images est devenu très vaste.

Généralement les systèmes de traitement numérique d'images se composent des éléments suivent :

1. - Un système d'acquisition et de numérisation qui permet d'effectuer l'échantillonnage et la quantification d'une image initialement acquise par un capteur (caméra, appareil photos, scanner, ...).
2. - Une mémoire de masse qui permet de stocker l'image digitale.
3. - Un système de visualisation, généralement composé d'un moniteur de télévision classique permettant de visualiser l'image.
4. - Un calculateur qui effectue les opérations de traitement d'images.

Dans notre application le calculateur et la mémoire de masse sont représentés par le micro-ordinateur, quant à l'acquisition se sont généralement des images photographiques scannées et sauvegardées dans des fichiers de format bien défini.

A partir de l'image numérique, il convient d'extraire l'information pertinente à l'égard de l'application concernée. Puis la traiter et l'interpréter en vue d'une reconnaissance ou d'une prise de décision.

Pour pouvoir reconnaître un objet dans une image, encore faut-il pouvoir le séparer du reste de l'image. C'est l'étape de segmentation qui consiste à faire une partition de l'image en sous-ensembles appelés régions. Cette opération devient particulièrement difficile dans le cas d'images bruitées ou hautement texturées.

Les figures I.1 et I.2 présentent les étapes essentielles d'un processus de traitement d'images.

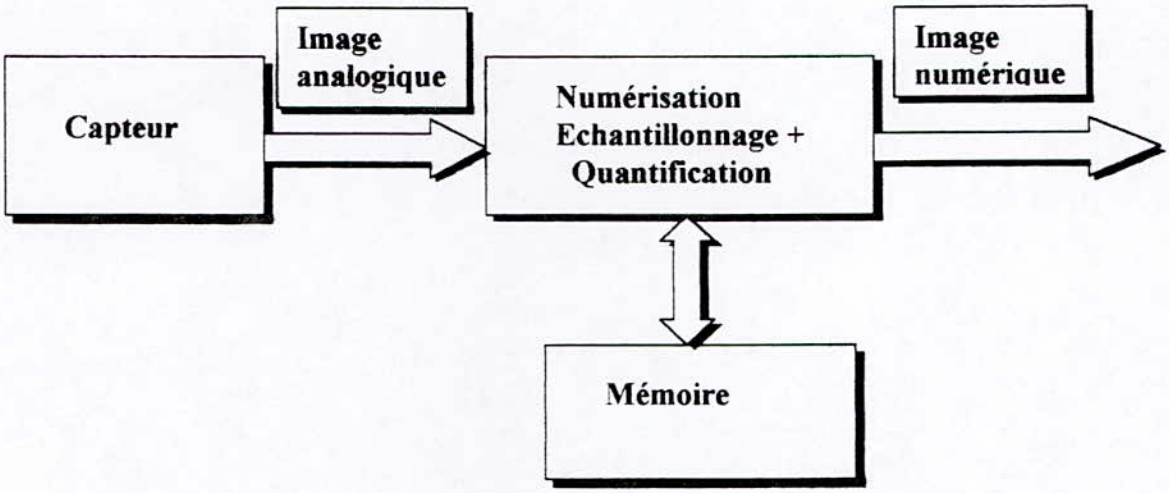


Figure I.1 : synoptique d'un système d'acquisition et de numérisation.

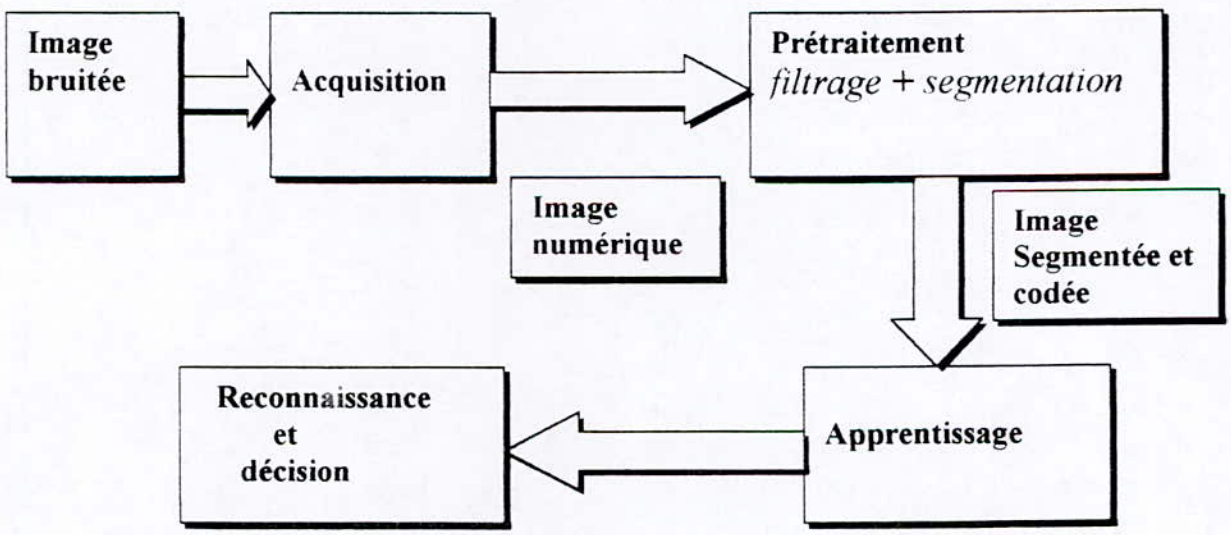


Figure I.2 : Principales étapes d'un processus de reconnaissance de formes.

I.2.4- La segmentation

La segmentation est le processus qui subdivise l'image en différentes régions connexes ayant des propriétés communes. Il s'agit pour la plus part des cas, de niveaux de gris voisins ou de textures. Pour cela, il existe généralement deux approches possibles. La première étant la segmentation par extraction de contours qui est considérée souvent comme une étape de prétraitement. Dans cette approche, les régions sont caractérisées par leurs frontières. La seconde approche est basée sur les propriétés intrinsèques des régions. C'est la segmentation en régions homogènes. L'homogénéité est un critère qui doit être défini avec précision selon les cas il peut être le niveau de gris, la couleur ou la texture. Ce dernier cas est l'un des problèmes les plus difficiles en traitement d'images.

I.2.5- Les textures

Une texture peut être définie comme un motif qui se répète plus ou moins aléatoirement, dans une région de l'image (voir figure V.3), ainsi dans une image texturée les différentes régions ne se différencient pas par leurs niveaux de gris mais par la distribution de ces niveaux dans l'espace bidimensionnel. La texture présente à une échelle donnée, le même aspect quelle que soit la zone observée, dans cette condition on la considère comme la réalisation d'un processus aléatoire stationnaire, c'est à dire que les statistiques calculées sont invariantes par translation.

Pour pouvoir segmenter (séparer) ces régions de l'image, il faut fixer un critère qui doit être calculable à partir de l'image. Il en existe plusieurs selon la modélisation choisie où le type d'images traitées, par exemple, dans les méthodes stochastiques – où l'image est modélisée par un champ aléatoire- on utilise les attributs stochastiques. Par contre, on utilise les attributs fractals dans les méthodes surfaciques (image modélisée par une surface),... etc.

On ne va pas décrire tous les attributs de l'image, néanmoins on va citer parmi les plus importants.

I.2.6- Les attributs d'une image

a- L'histogramme

L'histogramme d'une image est l'un des attributs les plus important pour caractériser une image et il entre dans beaucoup de méthodes de segmentation comme la méthode par *corrélation d'histogrammes locaux*. L'histogramme est un graphe représentant le nombre de pixels en fonction du niveau de gris. Il peut être interprété après normalisation comme une densité de probabilité. On peut aussi bien définir des histogrammes locaux qui sont calculés sur des fenêtres de tailles prédéfinis et constituer ainsi un critère de séparation entre régions : deux régions différentes auront des fonctions d'histogramme différentes.

b- Attributs stochastiques [1]

L'image est considérée ici comme la réalisation d'un processus aléatoire discret A[s] à valeur dans G={0,... , g} où g est le niveau de gris maximal correspondant au blanc. Et par ce fait, nous pouvons lui attribuer différentes grandeurs statistiques qui vont nous permettre de caractériser le processus A[s], à partir de ces différentes réalisations, parmi ces grandeurs :

La moyenne :

Dans le cas continu A(x), l'espérance ou la moyenne d'ensemble est définie par :

$$m_1(x) = E\{A(x)\} = \int_{-\infty}^{-\infty} a.f(a,x)da \quad \text{(I-1)}$$

où f est la densité de probabilité de A.
Dans le ces discret il suffit de passer d'intégrale à la somme :

$$m_1(x) = E\{A(s)\} = \sum_i P_i.a_i \quad \text{(I-2)}$$

où a_i est une réalisation de la variable A[s] et p_i est la probabilité $P(A[s] = a_i)$.

2.Les moments d'ordre k

Les moments généralisés d'ordre k sont définis comme suit :

$$E\{A^k[s]\} = \sum_i p_i.a_i^k \quad \text{(I-3)}$$

Les moments centrés d'ordre k sont définis par :

$$\mu_k(s) = E\{(A[s] - \mu[s])^k\} \quad \text{(I-4)}$$

De la même manière, on peut définir ces grandeurs sur une région bien particulière, en faisant les sommes à l'intérieur de ces régions, et cela va constituer une manière de discriminer entre les régions qui est très utilisée dans beaucoup de méthodes de segmentation.

Pour une région **R** de **k** pixels, on peut définir les différents moments d'espace centrés et non centrés.

Moment d'espace du premier ordre (moyenne) :

$$m_1 = \frac{1}{K}\sum_{s\in R} A[s] \quad \text{(I-5)}$$

Moment d'espace centré du second ordre ou variance :

$$m_2 = \frac{1}{K}\sum_{s\in R} (A[s] - m_1)^2 \quad \text{(I-6)}$$

L'écart type, noté σ, est défini comme la racine carrée de la variance. La moyenne est l'écart type sont en général les grandeurs statistiques les plus importantes, et sont les plus utilisés dans la segmentation. Néanmoins, certaines méthodes utilisent des statistiques d'ordre supérieur. De façon générale, le moment d'espace centré d'ordre k est défini :

$$m_k = \frac{1}{K}\sum_{s\in R} (A[s] - m_1)^k \quad \text{(I-7)}$$

c- Attributs fractals [1]

La théorie des fractals est sortie du domaine mathématique pur, pour devenir applicable dans des disciplines diverses, comme la géographie, la biologie ou la cosmologie.

Son aspect principal, est de travailler avec des dimensions non entières plutôt qu'avec les dimensions classiques (*1,2 et 3*). Cette nouvelle description, a ouvert un champ très large d'applications y compris en traitement d'images. Ont été développés, beaucoup de méthodes de compression basées sur la modélisation par fractals. Un autre aspect principal des objets fractals est leur auto-similarité par changement d'échelle. C'est à dire qu'ils ont la même apparence quelle que soit la résolution utilisée. Cette propriété correspond parfaitement à la description de certaines textures d'image. On peut donc légitimement penser à utiliser cette

propriété pour la segmentation d'images par mesure des attributs fractals *(la dimension est la plus importante)*.

Remarque : L'annexe 3 contient des éléments sur la théorie fractale et multifractale.

Calcul de la dimension fractale

Soit **R** une région dans laquelle on calcule les attributs. Dans tout ce qui suit, on supposera que la mesure utilisée est la somme des niveaux de gris de la région. On notera A[s] le niveau de gris du pixel s.

Il existe plusieurs méthodes pour calculer la dimension fractale d'un objet, mais nous allons en décrire que deux dans le cadre de ce projet. L'une d'elles, présentée dans le chapitre de la segmentation par la méthode des fractals, est nommée «méthode des triangles ». Elle consiste à calculer la surface variable de deux triangles construits par quatre points bien définis dans l'image. L'autre méthode, qu'on va décrire ci-après est appelée « méthode des boites ».

Pentland a montré que pour des textures homogènes, la surface d'un objet 3-D est fractale si et seulement si son image est fractale et que la dimension fractale reste la même. La dimension fractale d'une image est comprise entre 2 et 3 ; plus la texture est lisse (resp. rugueuse), plus la dimension fractale est proche de 2 (resp. 3).

La méthode de *Voss* a pour but d'estimer le nombre moyen, noté N(r), de boites cubiques de coté r fixé, nécessaires pour recouvrir l'image, considérée comme une surface dans l'espace $\Re^3$. Pour cela, on estime P(m,r), la probabilité qu'une boite de taille r, centrée sur un point arbitraire de la surface, contiennent m points de l'ensemble. On a donc :

$$\forall r, \sum_{m=1}^{N_p} P(m,r) = 1 \tag{I-8}$$

où N_p est le nombre de points possibles dans le cube.

Si N(r,m) est le nombre moyen de boites contenant m points et K le nombre de sites sur lesquels se fait le calcul (taille de la région) alors :

$$mN(r,m) = KP(m,r) \tag{I-9}$$

L'estimation du nombre moyen de boites disjointes nécessaires pour recouvrir la surface est :

$$N(r) = \sum_{m=1}^{N_p} N(m,r) = K \sum_{m=1}^{N_p} \frac{P(m,r)}{m} \tag{I-10}$$

L'estimation aux moindres carrés de la pente du nuage de points $(\ln(r), -\ln N(r))$, obtenu avec des boites de taille r croissante, donne l'estimation de la dimension fractale. L'algorithme suivant présente ce calcul :

Initialisation :
POUR r=1 à r_{max} et m=1 à r^3 FAIRE
P(m,r)=0

POUR tout site s de l'image FAIRE
DEBUT
POUR r=1 à r_{max} FAIRE
DEBUT
centrer un cube de côté r sur le pixel [s,A(s)]
compter le nombre de pixels de l'image qui appartiennent à ce cube
incrémenter P(m,r) de 1

FIN

FIN

POUR $r=1$ *à* r_{max} *FAIRE*

$$N(r) = \sum_{m=1}^{N_p} \frac{P(m,r)}{m}$$

Estimer par la méthode des moindres carrés la pente D de la courbe $\left(\ln(r), -\ln N(r)\right)$.

d- Modèle AR

D'une manière générale, un modèle autorégressif bidimensionnel [5] (causal et stationnaire), le cas d'une image, est décrit par l'équation aux différences suivante :

$$A(i,j) = \sum_{(m,n)\in D} a(m,n)A(i-m, j-n) + e(i,j) \qquad \text{(I-11)}$$

où :

$e(i,j)$: est l'entrée du modèle qui peut être un bruit blanc ou une séquence aléatoire de fonction de densité spectrale connue σ_e.

$a(m,n)$: sont les coefficients du modèle.

$A(i,j)$: est le champ des données.

Le champ $A(i,j)$ est supposé aléatoire, gaussien, stationnaire, de moyenne nulle et de variance finie. Les paramètres $a(m,n)$ peuvent être calculés en minimisant la variance de l'erreur de prédiction $e(i,j)$:

$$E\left[e(i,j)^2\right] = E\left[\left(A(i,j) - \sum_{(m,n)\in D} a(m,n)A(i-m, j-n)\right)^2\right] \qquad \text{(I-12)}$$

par rapport aux paramètres $a(m,n)$, c'est à dire :

$$\frac{\partial E\left[e(i,j)^2\right]}{\partial a(m,n)} = 0, \quad \forall (m,n) \in D \qquad \text{(I-13)}$$

Ce qui implique la relation d'orthogonalité suivante :

$$E\left[e(i,j)A(i-k, j-l)\right] = 0, \quad \forall (k,l) \in D \qquad \text{(I-14)}$$

et en remplaçant $e(i,j)$ par sa valeur, et en passant par l'autocorrélation on obtient le système d'équations normales :

$$\Gamma(k,l) - \sum_{(m,n)\in D} a(m,n)\Gamma(k-m, l-n) = \sigma_e \delta(k,l), \ \forall (k,l) \in D \qquad \text{(I-15)}$$

En passant à l'écriture matricielle on aura :

$$R_{MN}\,\overline{a} = \sigma_e^2\,\overline{1}_{MN} \qquad \text{(I-16)}$$

où :

$$R_{MN} = \begin{bmatrix} R_0 & R_{-1} & \cdots & R_{-N} \\ R_1 & & & \\ \vdots & & & R_{-1} \\ R_N & \cdots & R_1 & R_0 \end{bmatrix}$$: est la matrice d'autocorrélation de l'image. Elle est de type Toeplitz bloc Toeplitz, telle que :

$$R_k = \begin{bmatrix} \Gamma(0,k) & \Gamma(-1,k) & \cdots & \Gamma(-M,k) \\ \Gamma(1,k) & & & \\ \vdots & & & \Gamma(-1,k) \\ \Gamma(M,k) & \cdots & \Gamma(1,k) & \Gamma(0,k) \end{bmatrix} = R_{-k}$$

$\bar{a}$: est le vecteur paramètres donné par :

$$\bar{a}^T = \left(\bar{a}_0^T, \bar{a}_1^T, \cdots, \bar{a}_n^T, \cdots, \bar{a}_N^T\right)$$

où : $\bar{a}_0^T = \left(1, -a(1,0), \cdots, -a(M,0)\right)$ et $\bar{a}_n^T = \left(-a(0,n), \cdots, -a(M,n)\right)$

Le vecteur $\bar{1}_{MN}$ est le vecteur unitaire tel que :

$\bar{1}_{MN} = \left(\bar{1}_M^T, \bar{0}^T, \cdots, \bar{0}^T\right)$ et $\bar{1}_M^T = (1,0,\cdots,0)$

On peut résoudre la relation (I-16) par l'algorithme de *Levinson* bidimensionnel [5].

I.3- Le prétraitement et la détection de contours

Le but du prétraitement est de rendre la segmentation plus efficace, car on a souvent a faire à des images bruitées où très peu contrastées. Le prétraitement renforce la ressemblance entre les pixels appartenants aux même régions et par opposition accentuer la dissemblance entres les pixels appartenant à des régions différentes. Le prétraitement inclus :

- L'amélioration du contraste (par modification d'histogramme).
- Le filtrage du bruit.

La détection de contours est considérée ici, avec le prétraitement, car elle utilise le même procédé que les filtres réducteurs de bruit (filtres linéaires et stationnaires). Mais il faut garder à l'esprit que cette technique vient souvent après la segmentation ou faisant partie de celle-ci.

I.3.1- La modification d'histogramme

a- Expansion de dynamique

Cette méthode simple [11], consiste à utiliser au mieux l'échelle de niveaux de gris et d'étaler la dynamique de l'image sur un plus large intervalle, en général, entre le blanc et le noir.

Soit $A[i,j]$ l'image de départ et $A'[i,j]$ l'image après transformation. Soient $[a_0, a_1]$ l'intervalle de niveaux de gris dans l'image d'origine et $[a_{\min}, a_{\max}]$ l'intervalle disponible (selon le système d'affichage).

$a_{\min}$ correspondant au noir maximum (en général égale à 0).

a_{max} correspondant au blanc maximum (en général égale à 255). L'expansion de dynamique est la transformation linéaire suivante :

$$a'_s = \alpha\, a_s + \beta \tag{I-17}$$

tel que :

$$\forall\, a \in [a_0, a_1],\ a \xrightarrow{T} a' \in [a_{min}, a_{max}]$$

On aura deux équations à deux inconnus. Il est donc, facile de calculer α et β. On obtient :

$$\beta = \frac{a_{min}.a_1 - a_{max}.a_0}{a_1 - \alpha_0} \tag{I-18}$$

$$\alpha = \frac{a_{max} - a_{min}}{a_1 - \alpha_0} \tag{I-19}$$

Cette transformation améliore seulement l'aspect visuel de l'image. Cependant, ces performances sont limitées car on peut avoir une image (figure I.3) dont la dynamique occupe tout l'intervalle de niveaux de gris, mais de façon non égale, où la majorité des pixels sera par exemple, proche du noir. L'expansion ne va pas agir efficacement sur cette image (figure I.4) car :

$$a_0 \approx a_{min} \text{ et } a_1 \approx a_{max}.$$

b- Expansion quadratique

Pour palier ce problème on a développé un opérateur inspiré du premier. Cependant, la transformation appliquée n'est pas linéaire mais quadratique la différence est qu'on utilise trois points au lieu de deux pour avoir un polynôme du deuxième ordre on a :

$$a'_s = \alpha\, a_s^2 + \beta\, a_s + \gamma \tag{I-20}$$

En plus des conditions sur le maximum et le minimum, qui doivent devenir noir et blanc respectivement, on impose que la transformation de la moyenne de l'image $m_1(A)$ soit $\frac{a_{max} - a_{min}}{2}$ on aura donc :

$$a_0 \xrightarrow{T} a_{min} \qquad a_1 \xrightarrow{T} a_{max} \qquad m_1(A) \xrightarrow{T} \frac{a_{max} - a_{min}}{2}$$

On a trois points on peut donc faire une interpolation polynomiale et on aura :

$$\alpha = \frac{\begin{vmatrix} a_{min} & a_0 & 1 \\ \frac{a_{max} - a_{min}}{2} & m_1 & 1 \\ a_1 & a_1 & 1 \end{vmatrix}}{\Delta} \qquad \beta = \frac{\begin{vmatrix} a_0^2 & a_{min} & 1 \\ m_1 & \frac{a_{max} - a_{min}}{2} & 1 \\ a_1^2 & a_1 & 1 \end{vmatrix}}{\Delta} \qquad \gamma = \frac{\begin{vmatrix} a_0^2 & a_0 & a_{min} \\ m_1^2 & m_1 & \frac{a_{max} - a_{min}}{2} \\ a_1^2 & a_1 & a_1 \end{vmatrix}}{\Delta}$$

(I-21)

$$\text{où : } \Delta = \begin{vmatrix} a_0^2 & a_0 & 1 \\ m_1^2 & m_1 & 1 \\ a_1^2 & a_1 & 1 \end{vmatrix}$$

On peut voir l'efficacité de cette méthode sur la figure I.5.

c- Egalisation d'histogramme

Cette transformation [1], consiste à rendre le plus plat possible, l'histogramme de niveaux de gris de l'image.

L'image $A[i, j]$ est considérée dans ce cas comme l'ensemble de réalisations d'une variable aléatoire A admettant une densité de probabilité $f(a)$ non nulle sur l'intervalle, et admettant aussi une fonction de répartition F(a). L'histogramme - qui est le nombre de pixels en fonction du niveau de gris- peux être interprété –vu le grand nombre de pixels- comme une densité de probabilité à une normalisation prés. Donc, dans la pratique f représente l'histogramme normalisé (de façon à ce qu'il devienne une densité de probabilité) et F est l'histogramme cumulé, défini comme suit :

$$F(a) = \sum_{i=a_{\min}}^{a} f(i) \tag{I-22}$$

On cherche une transformation T continue, dérivable –au moins par morceaux- et strictement croissante telle que la variable aléatoire B=T(A) soit uniformément répartie dans l'intervalle $[b_{\min}, b_{\max}]$. Soit g(b) la densité de probabilité de B et T' la fonction dérivée de T, on obtient :

$$g(b) = \begin{cases} f(a)\dfrac{1}{T'(a)} = \dfrac{1}{b_{\max} - b_{\min}} & \text{pour } b_{\min} < b < b_{\max} \\ 0 & \text{ailleurs} \end{cases} \tag{I-23}$$

avec :

$$b = T(a) \quad et \quad a \in [a_{\min}, a_{\max}]$$

ceci équivaut à :

$$T'(a) = (b_{\max} - b_{\min}) f(a)$$

La transformation T est alors définie par :

$$T(a) = (b_{\max} - b_{\min}) F(a) + b_{\min} \quad \text{pour } a \in [a_{\min}, a_{\max}] \tag{I-24}$$

En considérant de petites fluctuations autour du niveau de gris a, on peut rendre la transformation T linéaire. L'amplitude des fluctuations est multipliée par le facteur :

$$T'(a) = (b_{\max} - b_{\min}) . f(a) \tag{I-25}$$

L'égalisation à pour effet d'amplifier les fluctuations dans les zones où celles-ci sont faibles. Ces zones ont des valeurs de densité de probabilité élevées. Le fait que la transformation soit continue et monotone, le sens et la position des transitions sont conservés : si la région 1 est un peu plus claire que la région 2 elle le sera encore plus après transformation (figure I.6).

I.3.2- Filtrage du bruit

Le filtrage regroupe un ensemble de méthodes qui ont pour but d'atténuer les effets indésirables du bruit. Ici, il faudra distinguer filtrage au sens large et strict du terme. Car, le bruit peut être tous ce qui ne nous intéresse pas dans l'image est dans ce cas le filtrage se confond avec la segmentation ou avec la détection de contours. Le bruit qui nous intéresse ici est toute modification subite par l'image originale -(*l'image physique*- ça peut être causé par le système d'acquisition –*cameras, appareils photos, câbles de transmission*...-) ou peut être inhérent à l'environnement de prise de vue (*obstacles divers, atmosphère dans le cas d'images satellites*).

Dans tous les cas le bruit dans l'image est considéré comme un champ aléatoire, on le caractérise au premier ordre par sa densité de probabilité f ou sa fonction de répartition, au deuxième ordre par sa densité spectrale ou par sa fonction de corrélation. En général la caractérisation d'un bruit va rarement au-delà du deuxième ordre.

L'échelle spatiale des fluctuations du bruit est relativement faible par rapport à la taille des régions. Le bruit est donc du type « haute fréquence ». On a donc le problème de traitement de signal suivant : retrouver par filtrage les niveaux d'intensité de chaque région. Les techniques les plus simples utilisent des filtres linéaires et stationnaires du type passe bas utilisant principalement la forte redondance spatiale de l'image. En effet, les pixels voisins d'une image ont pratiquement les mêmes caractéristiques et correspondent à des fréquences basses (*par rapport au bruit*). Cependant, il existe des cas où cette approche devient limitée. On peut citer dans ce cas d'autres approches qu'on ne va pas détailler comme : les filtres non-linéaires, filtres homomorphiques, filtres adaptatifs,....

Nous présenterons, dans ce qui suit, quelques filtres très utilisés et faciles à mettre en œuvre.

a- Filtre moyenneur

Cette technique [10] est une opération linéaire implantée dans le domaine spatial. Le principe est de glisser une fenêtre, appelée aussi « masque de convolution », de taille NxN le long de l'image à traiter et remplacer la valeur du pixel par la moyenne des pixels pondérée par les coefficients du filtre se trouvant dans cette fenêtre. En générale, on utilise une matrice de taille (3x3) tel que :

Parmi les masques de convolution qui présentent la forme d'un filtre passe bas, nous citons :

$$M1=\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \qquad M2=\frac{1}{10}\begin{bmatrix}1 & 1 & 1\\ 1 & 2 & 1\\ 1 & 1 & 1\end{bmatrix} \qquad M3=\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} \tag{I-26}$$

Remarques :

- Lorsque la taille du masque augmente, il est préférable de calculer le produit de convolution indirectement par les techniques de filtrage de *Fourrier*.
- L'utilisation du filtre passe bas réduit le bruit (effet poivre et sel), mais elle engendre un effet de flou (appelé aussi Blurring).

b- Filtre médian

C'est un filtre non linéaire mais néanmoins simple. Développé par *Tuckey* en 1971, il est utilisé pour éliminer le bruit et préserver les contours dans une image.

Le principe est de ranger les éléments de la fenêtre d'analyse –*voisinage direct de chaque pixel*- par ordre croissant. C'est à dire les pixels de la fenêtre d'analyse, et à prendre l'élément de rang (L+1)/2 pour remplacer le pixel traité. L est impair représente la taille de la fenêtre (pour une fenêtre 3×3 L = 9).

I.3.3- Le filtrage de Dériche

Nous allons développer dans ce paragraphe le filtre de *Dériche* [1], avec sa partie lissage et détection de contours. L'annexe 2 contient les développements mathématiques que nous avons jugés inutile d'inclure dans cette partie.

En utilisant l'opérateur optimal de *Dériche* (annexe 2), on peut mettre en œuvre un filtre bidimensionnel. On effectue d'abord un lissage de l'image pour améliorer l'immunité au bruit, ensuite on calcul le gradient (*détecteur de contours*) en chaque point de l'image.

a- Lissage

Le filtre de lissage de *Dériche* est la combinaison de deux filtres monodimensionnels dans les directions x et y. Le filtre monodimensionnel est l'intégrale du filtre optimal de *Dériche* h(x) (cf. annexe 2) :

$$f(x) = \int cxe^{-\alpha|x|}dx = b(\alpha|x|+1)e^{-\alpha|x|} \tag{I-27}$$

La constante b est calculée de façon donner une réponse égale à 1 pour un signal d'entrée constant de niveau 1 ; on obtient ainsi $b = \frac{\alpha}{4}$.

Remarque : Dans la pratique il faut normaliser le filtre par rapport à la valeur maximale du niveau de gris. C'est à dire qui correspond au blanc maximum qui dans notre cas est égale 255. Car il ne faudrait pas que la réponse de l'image au filtre pour certains pixels puisse dépasser cette valeur (cela ne voudrait rien dire pour l'ordinateur).

L'expression de la réponse impulsionnelle du filtre bidimensionnel séparable de lissage est donc de la forme :

$$f(x,y) = b^2(\alpha|x|+1)e^{-\alpha|x|}(\alpha|y|+1)e^{-\alpha|y|} \tag{I-28}$$

Si l'image originale est représentée par $A(x,y)$, l'image lissée sera donc :

$$B(x,y) = A * f(x,y) \tag{I-29}$$

On peut voir l'effet de cet opérateur sur une image bruité dans la figure I.8.

b- Calcul du gradient (détecteur de contours)

Le calcul du gradient se fait à partir des dérivées selon x et y du produit de convolution de l'image par le filtre de lissage f(x,y).

Compte tenu de la méthode de dérivation de l'opérateur de convolution et de la séparabilité du filtre $f(x,y)$ nous aurons :

$$\frac{\partial B}{\partial x}(x,y) = B_x(x,y) = \frac{\partial\{A * f\}}{\partial x}(x,y) = A * \frac{\partial f}{\partial x}(x,y) \tag{I-30}$$

et

$$\frac{\partial B}{\partial y}(x,y) = B_y(x,y) = \frac{\partial\{A * f\}}{\partial y}(x,y) = A * \frac{\partial f}{\partial y}(x,y) \quad \text{(I-31)}$$

ainsi on aura :

$$f_x(x,y) = \frac{\partial f}{\partial x}(x,y) = \eta x e^{-\alpha|x|}(\alpha|y|+1)e^{-\alpha|y|} \quad \text{(I-32)}$$

et

$$f_y(x,y) = \frac{\partial f}{\partial y}(x,y) = \eta y e^{-\alpha|y|}(\alpha|x|+1)e^{-\alpha|x|} \quad \text{(I-33)}$$

où η est une constante de normalisation calculée de façon à donner un maximum d'amplitude 1 en réponse à une transition unitaire. On obtient alors $\eta = -\frac{\alpha^3}{4}$. Ce qui donne :

$$f_x(x,y) = h(x)f(y)$$

Et par conséquent l'image de la dérivée directionnelle en x s'écrit :

$$B_x(x,y) = \big(A * h(x)\big) * f(y)$$

et celle en y :

$$B_y(x,y) = \big(A * h(y)\big) * f(x)$$

C'est à dire que la dérivée directionnelle selon x est le résultat d'un lissage suivant la direction y, suivi par une dérivation suivant x. Et l'inverse pour la dérivée selon y.

A la fin on calcule pour chaque pixel le module du vecteur aux composantes B_x et B_y. L'image finale aura donc la forme :

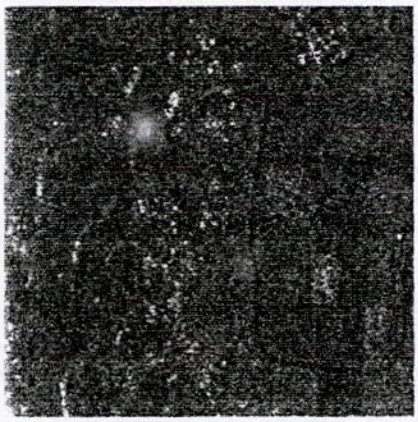
$$B\ (x,y) = \big(B_x(x,y)^2 + B_y(x,y)^2\big)^{\frac{1}{2}} \quad \text{(I-34)}$$

L'effet de cet opérateur sur une image de cellule est représenté dans la figure I.7.

N.B : Comme pour tous les détecteurs de contours on doit à la fin procéder à une binarisation de l'image résultat, par application d'un seuillage automatique ou manuel.

I.3.4- Résultats sur le prétraitement

Nous allons présenter ici quelques résultats obtenus par application des opérateurs de prétraitement, développés ci-dessus. En particulier, les opérateurs d'histogramme et les filtres de lissage et de détection de contours de *Dériche*.



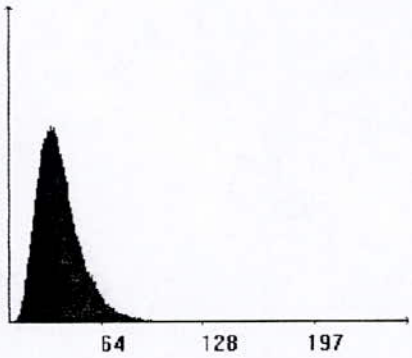
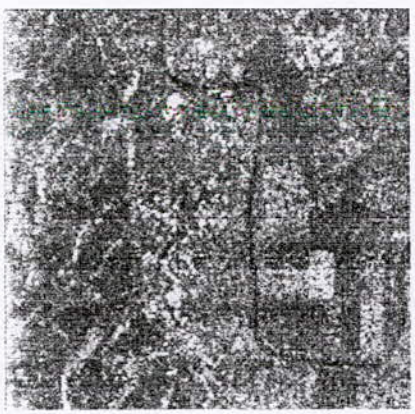


Figure I.3 : Image originale et son histogramme.



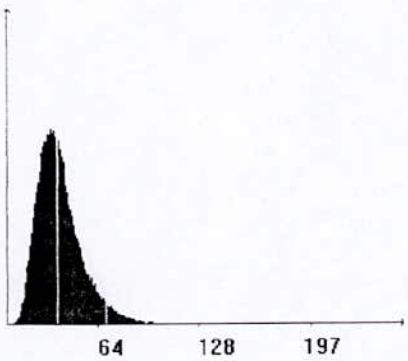


Figure I.4 : Image traitée par expansion dynamique et son histogramme.

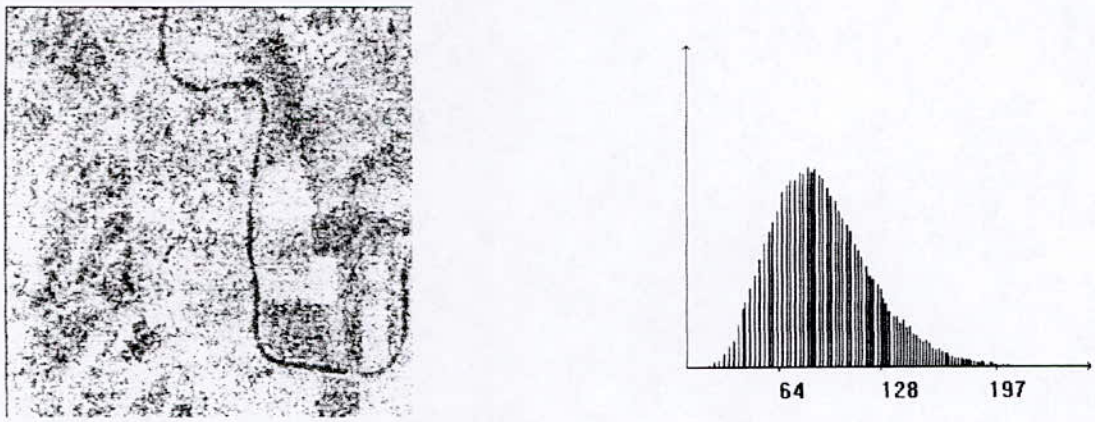


Figure I.5 : Image traitée par expansion quadratique et son histogramme.

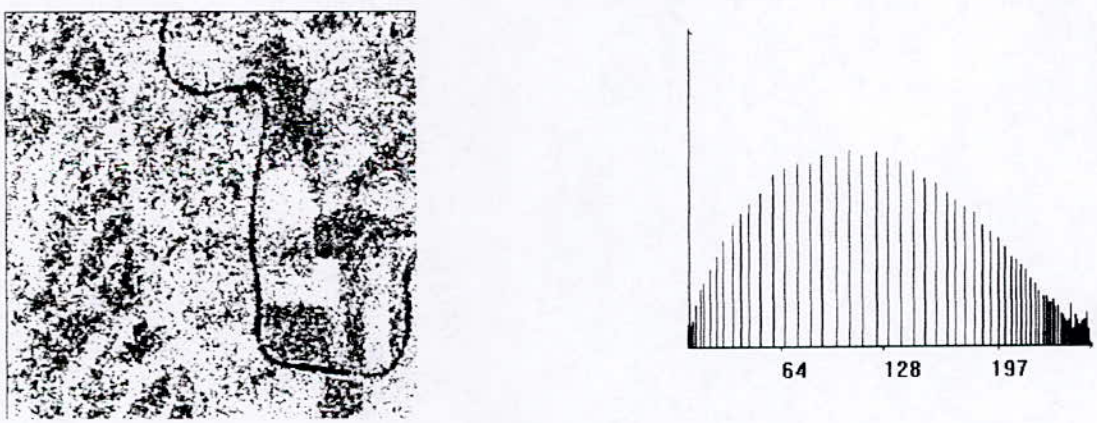
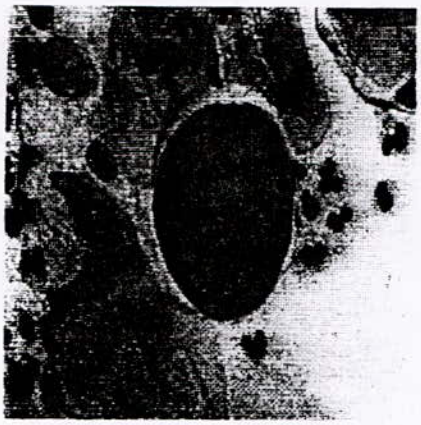


Figure I.6 : Image traitée par égalisation d'histogramme et son histogramme.



- a -

- b -

- c -

Figure I.7 :
a- Image originale.
b- Détection de contours par *Dériche.*
c- Détection de contours par *Sobel.*

- a -

- b -

Figure I.8 :
a- Image originale avec du bruit.
b- Image après lissage de *Dériche.*

I.4- Conclusion

On a présenté dans ce chapitre quelques attributs qui peuvent caractériser les régions de l'image en vu de les segmenter. Mais il en existe autant que de modèles pour décrire les régions. Nous avons citer seulement les attributs qui vont être utilisés dans les méthodes de segmentation décrites au chapitre III et IV.

Il est difficile de résumer tous les prétraitements qu'on peut effectuer sur les images. Les prétraitements dépendent du contexte de l'application. Ainsi dans l'imagerie radar, où on a une faible résolution, on accorde une importance particulière à la réduction du bruit - qui est souvent multiplicatif- en lui appliquant des opérateurs spéciaux (non abordés ici) comme le traitement morphologique et les filtres adaptatifs. Dans les applications de reconnaissance on s'intéresse à l'obtention de contours fermés et amincis. On a donc besoin d'appliquer - après détection de contours- des opérateurs d'amincissement et de fermeture comme l'opérateur de *Canny* [10], de l'érosion Morphologique, et même par utilisation des réseaux de neurones [1].

Enfin, on a présenté quelques résultats de prétraitement sur des images de nature différente pour tester l'efficacité de chaque opérateur.

Chapitre II

Les Algorithmes Génétiques

II.1- Introduction

Les mécanismes que l'on pense à l'origine de l'évolution reposent essentiellement sur le principe de la compétition qui sélectionne les individus les plus adaptés à leur milieu en leur assurant une descendance, et aussi sur une forme de coopération mise en œuvre par la reproduction sexuée. Les possibilités espérées de ces mécanismes ont conduit quelques chercheurs des années 1950 à vouloir les simuler pour les appliquer à l'ingénierie. Mais ces travaux n'ont pas été probants en raison des connaissances insuffisantes, à l'époque, de la génétique naturelle et aussi en raison des faibles performances des calculateurs qui étaient disponibles. D'autre part, l'extrême lenteur de l'évolution semblait très grande pour songer à exploiter utilement un tel processus.

Dans les années 60 et 70, trois écoles modélisant l'évolution de façon différente ont fait leur apparition indépendamment :

- Les algorithmes génétiques (*genetic algorithms*, GAs), développés par J.H. Holland.
- La programmation évolutionnaire (*evolutionary programming*, EP), de L.J. Fogel.
- Les stratégies d'évolution (*evolution strategies*, ESs), de H.P. Shwefel.

II.2- Principes généraux

Les trois approches évoquées ci-dessus, font parties de la classe des algorithmes évolutionnaires (*evolutionary algorithms*, EAs). Elles ne diffèrent que par l'absence ou la présence de tel ou tel opérateur ou encore par des détails d'implémentation des opérateurs qu'elles emploient. Bien que les buts originels de ces algorithmes aient été différents, ils sont aujourd'hui employés essentiellement pour accomplir des tâches d'optimisation. Les variantes que connaissent actuellement les algorithmes génétiques, recouvrent les principales caractéristiques qui différenciaient à l'origine ces différentes approches.

Les algorithmes génétiques sont des algorithmes d'optimisation qui utilisent des techniques (comme déjà évoqué) dérivées de la génétique et de l'évolution naturelle : croisement, mutation, sélection, etc.... . Les algorithmes génétiques ont déjà une histoire relativement ancienne puisque les premiers travaux de *John Holland* sur les systèmes adaptatifs remontent à 1962. L'ouvrage publié par *David Goldberg* [2] a largement contribué à les vulgariser et les expliquer clairement.

Un algorithme génétique recherche le ou les extremums d'une fonction définie sur un espace de données. Pour l'utiliser, on doit disposer des cinq éléments suivants :

- Un principe de codage de l'élément de population. Cette étape associe à chacun des points de l'espace d'état une structure de données. Elle se place généralement après une phase de modélisation mathématique du problème traité. De ce fait, le code utilisé joue un rôle très important dans le succès des algorithmes génétiques. Les codages binaires ont été très utilisés à l'origine. Les codages réels sont de plus en plus utilisés surtout lorsqu'il s'agit d'optimiser des problèmes à variables réelles.
- Un mécanisme de génération de la population initiale. Ce mécanisme doit être capable de produire une population d'individus non homogène qui servira de base pour les générations futures. Le choix de la population initiale est important car il peut jouer sur la convergence vers l'optimum global. Dans le cas où l'on ne connaîtrait rien du problème à résoudre, il est essentiel que la population initiale soit répartie sur tout le domaine de recherche.
- Une fonction à optimiser. Celle-ci retourne une valeur de R^+ appelée fonction objectif ou fonction d'évaluation de l'individu.
- Des opérateurs permettant de diversifier la population au cours des générations et d'explorer l'espace d'état. Ainsi, l'opérateur de croisement recompose les gènes d'individus existant dans la population et l'opérateur de mutation a pour but de garantir l'exploration de l'espace d'état.
- Des paramètres de dimensionnement : taille de la population, nombre total de générations ou critère d'arrêt de l'algorithme, probabilités d'application des opérateurs de croisement et de mutation (c'est à dire le taux de croisements et de mutations par population).

Le principe général du fonctionnement d'un algorithme génétique est représenté sur la figure II.1.

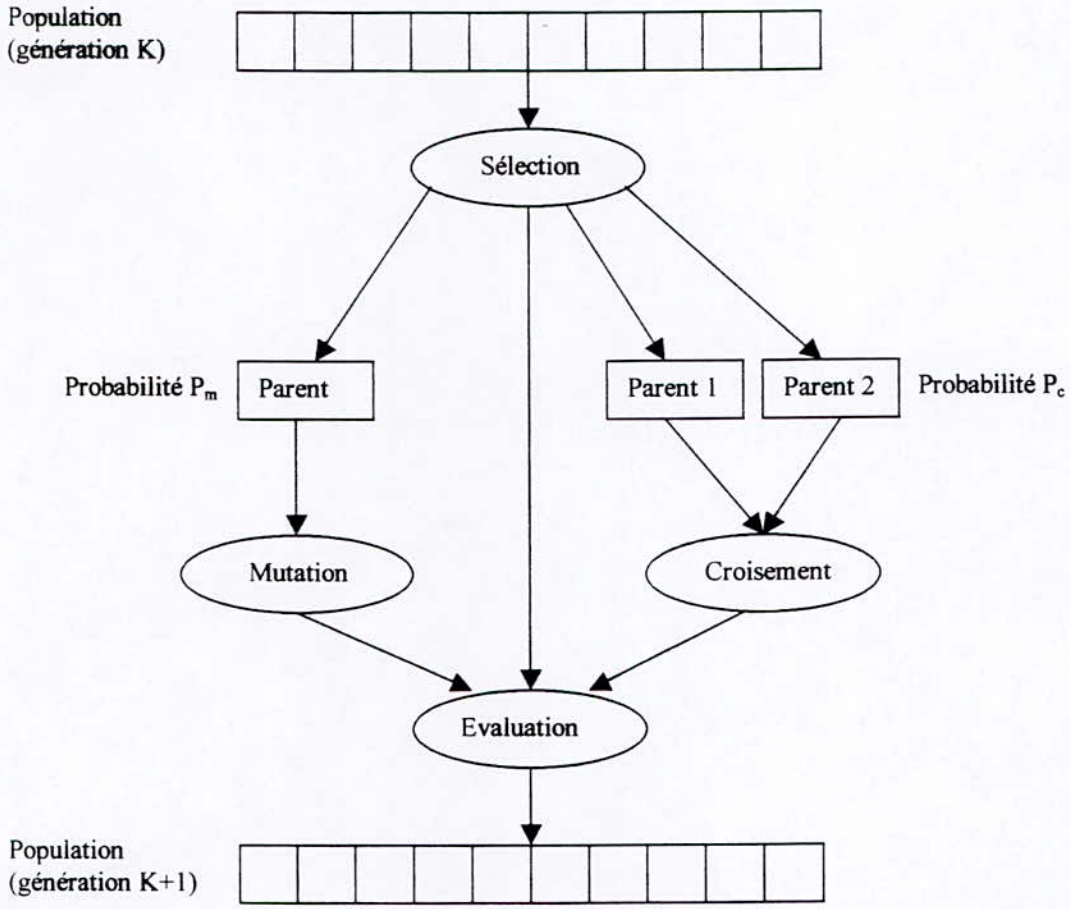


Figure II.1:Principe général des algorithmes génétiques

II.2.1- Etapes suivies dans l'exécution d'un AG

D'abord, on commence par générer une population d'individus de façon aléatoire. Pour passer d'une génération *k* à la génération *k+1*, les trois opérations suivantes sont répétées pour tous les éléments de la population k. Des couples de parents P1 et P2 sont sélectionnés en fonction de leurs adaptations. L'opérateur de croisement leur est appliqué avec une probabilité P_c (généralement autour de 0.6) et génère des couples d'enfants C_1 et C_2. D'autres éléments P sont sélectionnés en fonction de leur adaptation. L'opérateur de mutation leur est appliqué avec la probabilité P_m (P_m est généralement très inférieur à P_c) et génère des individus mutés P'. Le niveau d'adaptation des enfants (C_1,C_2) et des individus mutés P' sont ensuite évalués avant leur insertion dans la nouvelle population. Différents critères d'arrêt de l'algorithme peuvent être choisis :

- Le nombre de générations que l'on souhaite exécuter peut être fixé à priori. C'est ce qu'il faut faire pour obtenir une solution dans un temps limité.
- L'algorithme peut être arrêté lorsque la population évolue très lentement.

II.2.2- Lexique général

En premier lieu voici un résumé du vocabulaire nécessaire à la compréhension des algorithmes génétiques [2].

Mot	*Signification générale*	*Signification dans le contexte*
Stochastique	Qui fait appel à des méthodes non déterministes, qui met en jeu le hasard	idem
Biodiversité	Diversité physique des individus d'une population	idem
Gènes	Une information portée par les chaînes d'ADN qui forment les chromosomes	Un élément de l'alphabet de codage(dans le codage binaire c'est: 0 ou 1)
Phénotype	Les attributs physiques d'un individu directement liés à son patrimoine génétique (ses gènes)	la valeur réelle d'une variable
Génome	Ensemble des gènes d'un individu	L'ensemble des éléments codant les variables pour un individu donné.
Cross-over (croisement)	Opération de combinaison ayant lieu lors de la rencontre d'un gamète mâle et d'un gamète femelle pendant la reproduction	Combinaison des chromosomes des parents pour donner un fils.
Mutation	Une altération des gènes due Vraisemblablement à une erreur de copie de l'ADN	L'altération d'un gène lors de la reproduction
Population	Un ensemble d'individus	L'ensemble des individus qu'on considère et qu'on fait évoluer
Chaîne binaire	Une chaîne de bits (0 ou 1), codée dans la base 2	Idem
Performance	Capacités dont fait preuve un individu dans une situation donnée	Adaptation de l'individu dans l'univers de la fonction considérée, après la mise en échelle
Sélection	Tri des individus pour garder ceux qui vérifient un certain critère	Choix des individus pour la Reproduction, influencé par la performance de chaque individu
Fonction d'adaptation	La fonction qui permet de déterminer la performance d'un individu	La fonction qui mesure l'adéquation d'un élément de la population.
Allèle	Un certain nombre de gènes qui peut être interprété comme directement responsable d'un caractère physique	Un ensemble de gènes du même chromosome
Chromosome	Composé de chaînes d'ADN, contient une partie de l'information génétique	Une variable (sa définition en gènes et son équivalent en nombre réel)

II.2.2- Description

a- Codage des données

Historiquement, le codage utilisé par les algorithmes génétiques était représenté sous forme de chaînes de bits contenant toute l'information nécessaire à la description d'un point dans l'espace d'état. Ce type de codage a pour intérêt de permettre de créer des opérateurs simples de croisement et de mutation. C'est également en utilisant ce type de codage que les premiers résultats de convergence théorique ont été obtenus.
Cependant, ce type de codage n'est pas toujours bon comme le montrent les deux exemples suivants :

- Deux éléments voisins en terme de distance de *Hamming* ne codent pas nécessairement deux éléments proches dans l'espace de recherche. Cet inconvénient peut être évité en utilisant le code Gray.
- Pour des problèmes d'optimisation dans des espaces de grande dimension [7], le codage binaire peut rapidement devenir mauvais (nombre de bits superflus).Et généralement, chaque variable est représentée par une partie de la chaîne de bits et la structure du problème n'est pas bien reflétée, car si l'ordre des variables a une importance dans la structure du chromosome, il n'en a pas forcément dans la structure du problème. Pour remédier à cela, on a pensé à utiliser des vecteurs réels pour conserver les variables du problème dans le codage de l'élément de population sans passer par le codage binaire intermédiaire. La structure du problème est conservée dans ce codage.

b- Génération aléatoire de la population initiale

Le choix de la population initiale d'individus conditionne fortement la rapidité de l'algorithme. Si la position de l'optimum dans l'espace d'état est totalement inconnue, il est naturel de générer aléatoirement des individus en faisant des tirages uniformes dans chacun des domaines associés aux composantes de l'espace d'état en veillant à ce que les individus produits respectent les contraintes du problème. Si par contre, des informations a priori sur le problème sont disponibles, il parait bien évident de générer les individus dans un sous-domaine particulier, afin d'accélérer la convergence. En supposant que la gestion des contraintes ne peut se faire directement, les contraintes sont généralement incluses dans le critère à optimiser sous forme de pénalités. Il est clair qu'il vaut mieux, lorsque c'est possible ne générer que des éléments de population respectant les contraintes, c'est à dire sanctionner par exclusion tout élément illégal.

c- Gestion des contraintes

Un élément de la population qui viole une contrainte se verra attribuer une mauvaise fonction objectif [2], et aura une forte probabilité d'être éliminé par le processus de sélection.
Il peut cependant être intéressant de conserver, tout en les pénalisant, les éléments non admissibles car ils peuvent permettre de générer des éléments admissibles de bonne qualité (à l'aide de la mutation). Pour de nombreux problèmes, l'optimum est atteint lorsque l'une au moins des contraintes de séparation est saturée, c'est à dire sur la frontière de l'espace admissible.

Gérer les contraintes en pénalisant la fonction objectif est difficile, un "dosage" s'impose pour ne pas favoriser la recherche de solutions admissibles au détriment de la recherche de l'optimum ou inversement.

Disposant d'une population d'individus non homogène, la diversité de la population doit être entretenue au cours des générations afin de parcourir le plus largement possible l'espace d'état. C'est le rôle des opérateurs de croisement et de mutation.

d- Opérateur de croisement

Le croisement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes. Classiquement, les croisements sont entre deux parents et génèrent deux enfants. Initialement, le croisement associé au codage par chaînes de bits est le croisement à découpage de chromosomes (slicing crossover). Pour effectuer ce type de croisement sur des chromosomes constitués de M gènes, on tire aléatoirement une position dans chacun des parents. On échange ensuite les deux sous-chaînes terminales de chacun des deux chromosomes, ce qui produit deux enfants C_1 et C_2 (voir figure II.2).

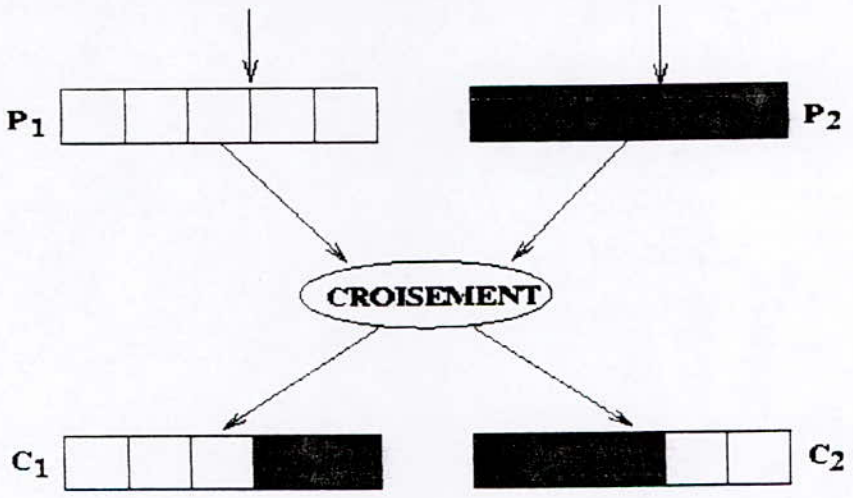


Figure II.2: Slicing crossover

On peut étendre ce principe en découpant le chromosome non pas en 2 sous-chaînes mais en 3, 4, etc... (voir figure II.3).

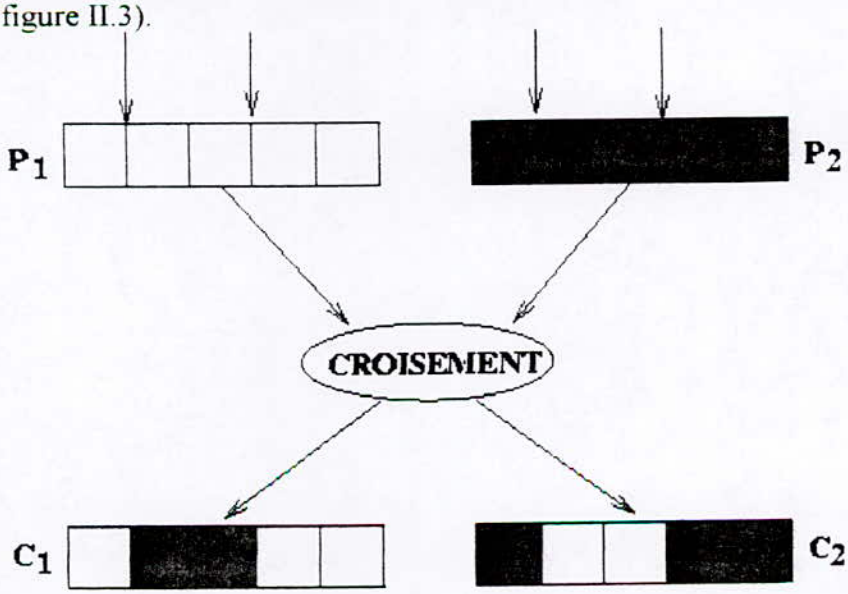


Figure II.3: Slicing crossover à 2 points

Ce type de croisement à découpage de chromosomes est très efficace pour les problèmes discrets. Pour les problèmes continus, un croisement "barycentrique" [7] est souvent utilisé. C'est à dire, deux gènes $P_1(i)$ et $P_2(i)$ sont sélectionnés dans chacun des parents à la même position i. Ils définissent deux nouveaux gènes $C_1(i)$ et $C_2(i)$ par combinaison linéaire :

$$\begin{cases} C_1(i) = \alpha P_1(i) + (1-\alpha) P_2(i) \\ C_2(i) = (1-\alpha) P_1(i) + \alpha P_2(i) \end{cases} \quad \text{(II-1)}$$

où α est un coefficient de pondération aléatoire adapté au domaine d'extension des gènes (il n'est pas nécessairement compris entre 0 et 1, il peut par exemple prendre des valeurs dans l'intervalle [-0.5,1.5] ce qui permet de générer des points entre ou à l'extérieur des deux gènes considérés).
Dans le cas particulier d'un chromosome matriciel constitué par la concaténation de vecteurs, on peut étendre ce principe de croisement aux vecteurs constituant les gènes (voir figure II.4) :

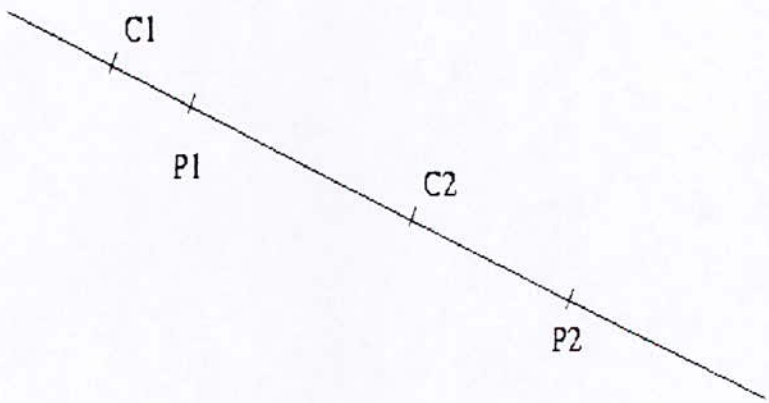


Figure II.4: Croisement barycentrique.

On peut imaginer et tester des opérateurs de croisement plus ou moins complexes sur un problème donné mais l'efficacité de ce dernier est souvent lié intrinsèquement au problème.

e- Opérateur de mutation

L'opérateur de mutation apporte aux algorithmes génétiques la propriété d'ergodicité de parcours d'espace. Cette propriété indique que l'algorithme génétique sera susceptible d'atteindre tous les points de l'espace d'état, sans pour autant les parcourir tous dans le processus de résolution. Ainsi en toute rigueur, l'algorithme génétique peut converger sans croisement, et certaines implémentations fonctionnent de cette manière, mais un algorithmes génétique ne peux pas se passer de mutation car c'est le mécanisme même de la recherche. Donc, théoriquement, les propriétés de convergence des algorithmes génétiques dépendent fortement de cet opérateur.

Pour les problèmes discrets, l'opérateur de mutation consiste généralement à tirer aléatoirement un gène dans le chromosome et à le remplacer par une valeur aléatoire (voir

figure II.5). Si la notion de distance existe, cette valeur peut être choisie dans le voisinage de la valeur substituée.

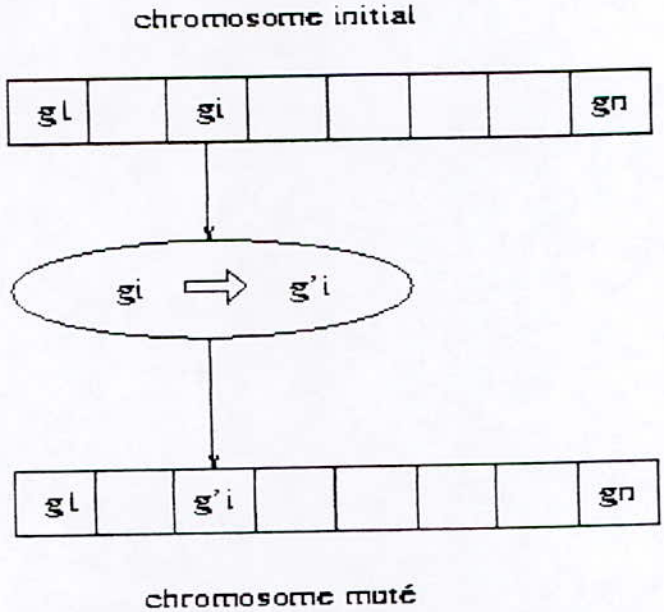


Figure II.5: Principe de l'opérateur de mutation

Dans les problèmes continus, on procède un peu de la même manière en tirant aléatoirement un gène dans le chromosome, auquel on ajoute un bruit généralement gaussien. L'écart type de ce bruit est difficile à choisir a priori (méthode utilisée dans ch. III).

g- Principes de sélection

A l'inverse d'autres techniques d'optimisation, les algorithmes génétiques ne requièrent pas d'hypothèses particulières sur la régularité de la fonction objective. L'algorithme génétique n'utilise notamment pas ses dérivées successives, ce qui rend très vaste son domaine d'application. De même sur le plan de continuité. Néanmoins, dans la pratique, les algorithmes génétiques sont sensibles à la régularité des fonctions qu'ils optimisent.
Le peu d'hypothèses requises permet de traiter des problèmes très complexes. La fonction à optimiser peut ainsi être le résultat d'une simulation.

La sélection permet d'identifier statistiquement les meilleurs individus d'une population et d'éliminer les mauvais. On trouve dans la littérature un nombre important de principes de sélection plus ou moins adaptés aux problèmes qu'ils traitent. Dans notre cas, nous citerons deux principes de sélection à savoir:

- Roulette wheel selection (technique classique) [2].
- Stochastic remainder without replacement selection [7].

Le principe de Roulette wheel exploite la métaphore d'une roulette de casino, qui comporterait autant de cases que d'éléments dans la population et où la taille de ces cases serait proportionnelle à la fonction objectif de chaque élément. Le jeu étant lancé, l'élément sélectionné est désigné par l'arrêt de la bille sur sa case. Si les cases sont déroulées sur un segment de droite, la sélection d'un individu revient à choisir aléatoirement un point du segment avec une distribution de probabilité uniforme. La variance de ce processus est élevée. Par malchance, il est possible à la limite qu'un élément de mauvaise qualité soit sélectionné pour la reproduction autant de fois qu'il y a d'individus remplacés. Il est aussi possible qu'un

individu ayant une bonne valeur d'adaptation (fonction objectif) ne soit jamais sélectionné. Ce phénomène est responsable de la ***dérive génétique*** qui permet à certains individus de survivre au détriment d'individus meilleurs. Pour limiter ce risque, la taille de la population doit être suffisamment grande. Lorsque la dimension de la population est réduite, il est difficile d'obtenir en pratique l'espérance mathématique de sélection en raison du peu de tirages effectués. Un biais de sélection plus ou moins fort existe suivant la dimension de la population.

La technique Stochastic remainder without replacement selection, évite ce genre de problème et donne de bons résultats. Ainsi, cette technique est décrite comme suit:

- Pour chaque élément i, on calcule le rapport r_i de sa fonction objectif sur la moyenne des fonctions d'adéquations.
- Soit $e(r_i)$ la partie entière de r_i, chaque élément est reproduit exactement $e(r_i)$ fois.
- La roulette wheel selection, précédemment décrite, est appliquée sur les individus affectés des fonctions objectifs $r(i)-e(r_i)$.

II.3- Améliorations classiques

Les processus de sélection présentés sont très sensibles aux écarts de fonctions objectifs et dans certains cas, un très bon individu risque d'être reproduit trop souvent et peut même provoquer l'élimination complète des autres; on obtient alors une population homogène contenant un seul type d'individus. Ainsi, dans l'exemple de la figure II.6 le second mode M2 risque d'être le seul représentant pour la génération suivante et c'est seulement la mutation qui pourra aider à atteindre l'objectif global M1 après plusieurs essais successifs.
Pour éviter ce comportement, il existe d'autres modes de sélection (ranking) ainsi que des principes (scaling, partage) qui empêchent les individus "forts" d'éliminer complètement les plus "faibles". On peut également modifier le processus de sélection en introduisant des tournois entre parents et enfants, basés sur une technique proche du recuit.

Enfin, on peut également introduire des recherches multi-objectifs, en utilisant la notion de dominance lors de la sélection.

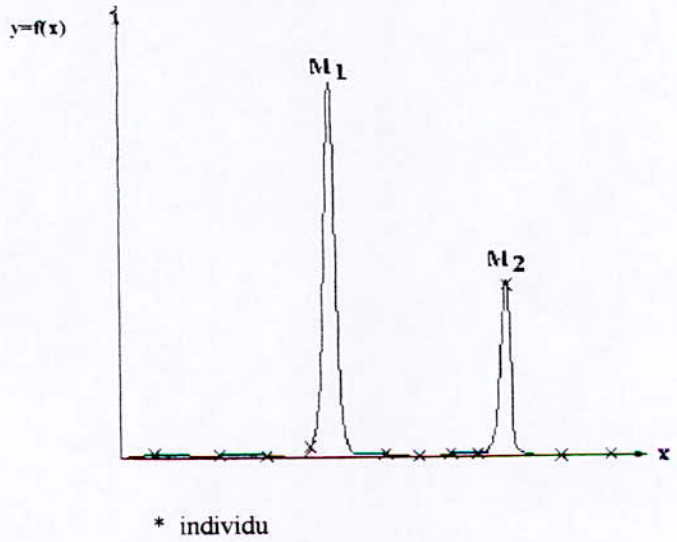


Figure II.6 : Un exemple où la sélection classique risque de ne produire qu'un seul individu.

II.3.1- Le Scaling

Le scaling ou mise à l'échelle, modifie les fonctions d'adaptations afin de réduire ou d'amplifier artificiellement les écarts entre les individus. Le processus de sélection n'opère plus sur la fonction objectif réelle mais sur son image après le scaling. Il y a plusieurs types de scaling, mais on présente ici les types les plus utilisés.
Soit f_r la fonction objectif avant le scaling et f_s la fonction objectif modifiée par le scaling.

a- Scaling linéaire

Dans ce cas la fonction de scaling est définie de la façon suivante [8] :

$$f_s = a f_r + b \tag{II-2}$$

$$a = \frac{\max' - \min'}{\max - \min} \tag{II-3}$$

$$b = \frac{\min' \max - \min \max'}{\max - \min} \tag{II-4}$$

En règle générale, le coefficient a est inférieur à 1, ce qui permet de réduire les écarts de fonctions objectif et donc de favoriser l'exploration de l'espace. Ce scaling est statique par rapport au numéro de la génération, et pénalise la fin de convergence lorsque l'on désire favoriser les modes dominants.

b- Scaling linéaire dynamique

Dans ce cas le paramètre b n'est pas constant mais variable [2], avec les générations. La fonction d'adaptation aura la forme suivante :

$$f_s = a f_r + b_t \tag{II-5}$$

b_t va dépendre de l'application.

c- Troncature en Sigma

Cette méthode a été proposée par *Forrest* pour améliorer la mise en échelle et aussi prendre en compte les valeurs négatives possibles (*car on ne peut pas interpréter une probabilité négative*). La formule proposée par *Goldberg* [2], [7], [8]est :

$$f_s = f_r - \left(\bar{f} - c \times \sigma\right) \tag{II-6}$$

où :
c est un petit entier,
σ est l'écart type de la population,
$\bar{f}$ est la moyenne de la fonction objectif f_r.

d- Scaling exponentiel

Gillies a suggéré un changement d'échelle en puissance [2], [7], [8], pour lequel la fonction d'adaptation est une certaine puissance de la fonction objectif. Elle est définie dans le cas général de la façon suivante:

$$f_s = (f_r)^{K(n)} \tag{II-7}$$

où :
n : est la génération courante.
K : dépend du problème traité. Dans une étude appliquée à la vision artificielle *Gillies* a chois K constant et égale à 1.005.

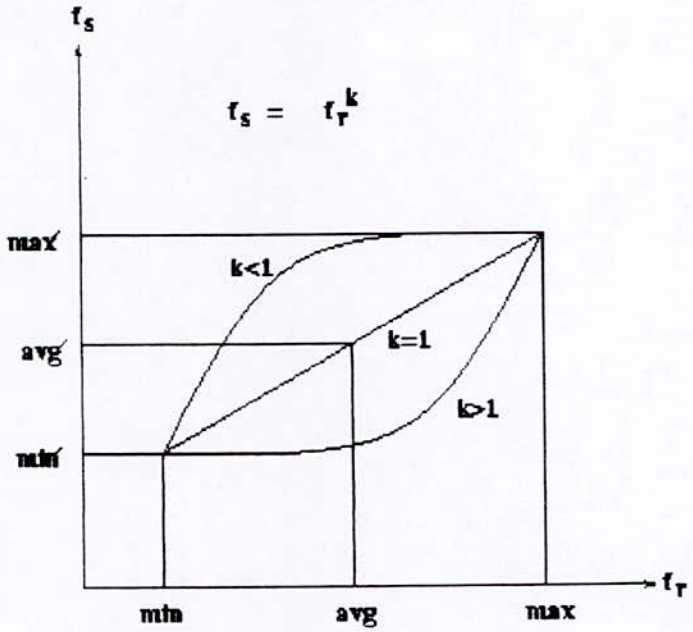


Figure II.7 : Fonction de scaling exponentiel

- Pour k proche de zéro, on réduit fortement les écarts de fonctions objectif. aucun individu n'est vraiment favorisé et l'algorithme génétique se comporte comme un algorithme de recherche aléatoire et permet d'explorer l'espace.
- Pour k proche de 1, le scaling n'a pas d'effets.
- Pour k > 1 les écarts sont exagérés et seuls les bons individus sont sélectionnés ce qui produit l'émergence des modes.

Dans la pratique, on fait généralement varier k des faibles valeurs vers les fortes valeurs au cours des générations. Pour cela on peut utiliser la formule suivante :

$$k = \left(tg\left[\frac{n}{N+1} \frac{\pi}{2} \right] \right) p \tag{II-8}$$

n étant la génération courante, N le nombre total de générations, p un paramètre à choisir. Dans la plupart des applications on choisit p=0.1. L'évolution de k en fonction de la génération n est donnée par la figure II.8.

Une autre définition possible du scaling exponentiel [2] est :

$$f_s = (a \times f_r + b)^{K(n)} \tag{II-9}$$

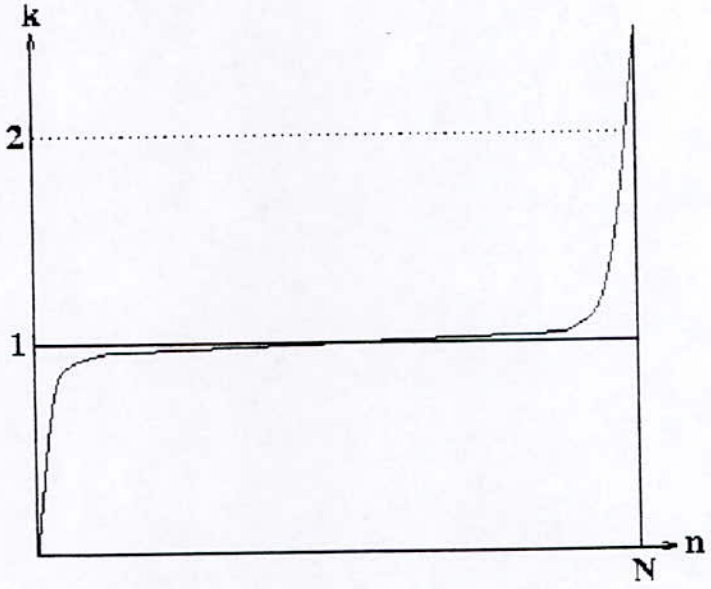


Figure II.8: Allure de l'évolution de k en fonction des générations

e- Scaling logarithmique

Cette méthode a été examinée par *Fitzpatrick* et *Grefenstette* [2], [7], [8], pour la mise en échelle de la fonction objectif surtout pour les problèmes de minimisation. La fonction d'adaptation aura la forme suivante :

$$f_s = b - \log(f_r) \qquad \text{(II-10)}$$

où b est choisi plus grand que n'importe quelle valeur de $\log(f_r)$.

II.3.2- Le Partage (sharing)

a- Introduction

L'objectif du partage [2] est de répartir sur chaque sommet de la fonction à optimiser un nombre d'individus proportionnel à la fonctions objectif associée à ce sommet. La figure II.9 présente deux exemples de répartitions de populations dans le cas d'une fonction à cinq sommets : le premier sans partage, le second avec partage.

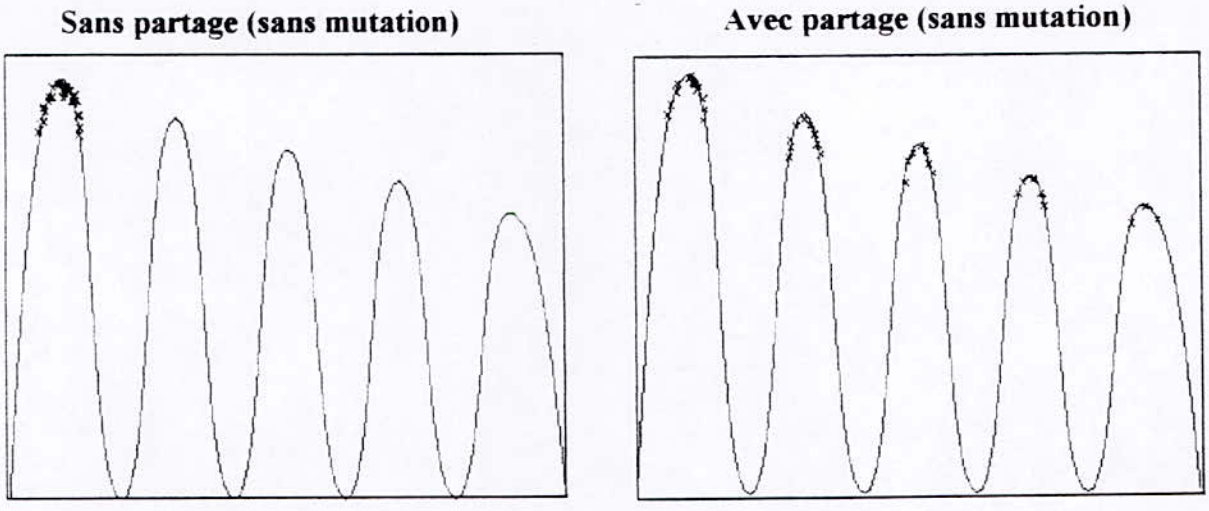


Figure II.9: Objectif du partage

b- Principe

Le partage consiste à modifier la fonction objectif utilisée par le processus de sélection (comme le scaling). Pour éviter le rassemblement des individus autour d'un sommet dominant, le partage pénalise les fonctions objectif en fonction du taux d'agrégation de la population dans le voisinage d'un individu. Il nécessite l'introduction d'une notion de distance. Dans la pratique, il faut définir une distance indiquant la dissimilarité entre deux individus. Cette distance est alors utilisée pour calculer la nouvelle fonction objectif de la façon suivante :

$$f_i' = \frac{f_i}{m_i'} \qquad \text{(II-11)}$$

avec:

$$m_i' = \sum_{j=1}^{N} S(d(x_i, x_j))$$

$$S(d) = 1 - \left(\frac{d}{\sigma_{share}}\right)\alpha \quad \text{si} \quad d < \sigma_{share}$$

$$S(d) = 0 \quad \text{si} \quad d > \sigma_{share}$$

Le paramètre σ_{share} permet de délimiter le voisinage d'un point et dépend du problème traité. La figure II.10 donne l'allure de S(d) pour différentes valeurs de α. Suivant la valeur donnée à α le partage sera plus ou moins efficace. Ainsi pour $\alpha < 1$, on pénalise les groupes très agglomérés.

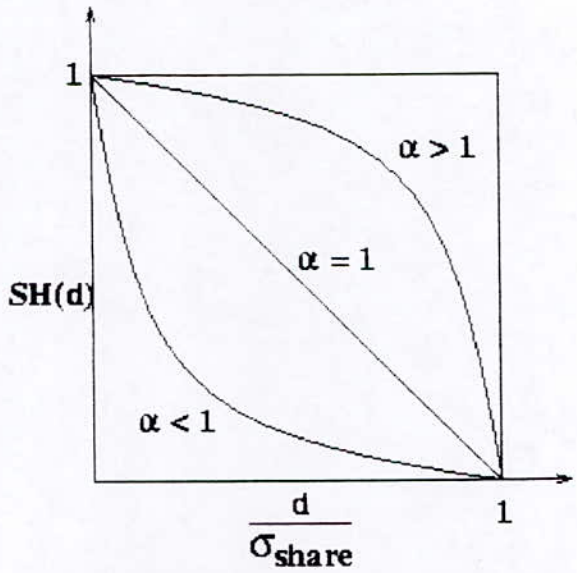


Figure II.10: Allure de $S\left(\frac{d}{\sigma_{share}}\right)$

Dans la pratique, ce type de partage donne effectivement de bons résultats mais au prix de N^2 calculs de distances entre chromosomes à chaque génération pour une population de taille N. Or les algorithmes génétiques induisent une complexité en N sans partage et le fait de passer en N^2 peut être pénalisant, notamment pour N grand.
Pour réduire ce nombre, on utilise un partage "clusterisé".

c- Partage clusterisé

Pour effectuer ce type de partage [2], on commence par identifier les différents "clusters" d'individus dans la population. Ce dernier utilise deux paramètres d_{min} et d_{max} respectivement pour fusionner ou pour créer des clusters. Initialement, chaque individu de la population est considéré comme le centre d'un cluster. On applique alors successivement les deux principes suivants :

- Si deux centres sont à une distance inférieure à d_{min}, on fusionne ces derniers dans un cluster unique dont le centre résultant est le barycentre des deux centres initiaux.
- Un nouvel individu est agrégé à un cluster si sa distance au centre le plus proche est inférieure à d_{max}. Dans ce cas, on recalcule le centre du cluster global. Sinon, on crée un nouveau cluster contenant ce seul individu.

Ce principe de fusion-agrégation permet d'obtenir un nombre de clusters fluctuant avec la répartition des individus dans l'espace d'état. On applique ensuite le principe de partage en modifiant les fonctions objectif de la façon suivante :

$$f_i' = \frac{f_i}{m_i'} \qquad \text{(II-12)}$$

Avec : $$m_i' = n_c\left[1-\left(\frac{d_{ic}}{2d_{max}}\right)\alpha\right]$$

avec:

- n_c: nombre d'individus contenus dans le cluster auquel appartient l'individu i.
- α : coefficient de sensibilité.
- d_{ic} : distance entre l'individu i et le centre du cluster c.

II.3.3- Association des AG avec des méthodes locales

La grande force des algorithmes génétiques est leur capacité à trouver la zone de l'espace des solutions contenant l'optimum de la fonction. En revanche, ils sont inefficaces lorsqu'il s'agit de trouver la valeur exacte de l'optimum dans cette zone. Or, c'est précisément ce que les algorithmes locaux d'optimisation réalisent le mieux.

Il est donc naturel de penser à associer un algorithme local à l'algorithme génétique de façon à trouver la valeur exacte de l'optimum. On peut aisément le faire en appliquant à la fin de l'algorithme génétique un algorithme local sur le meilleur élément trouvé. Cette technique est d'autant plus efficace que l'on utilise simultanément du clustering, et que l'algorithme local est appliqué à chaque meilleur élément de chaque cluster. En effet, on constate souvent que le meilleur élément trouvé par l'algorithme génétique ne reste pas le meilleur élément après amélioration par l'algorithme local de tous les meilleurs éléments de clusters.
Une autre technique consiste à utiliser un algorithme local associé à l'algorithme génétique pour calculer la fonctions objectifs d'un élément. On peut, par exemple dans un espace fortement combinatoire, rechercher avec l'algorithme génétique les zones intéressantes de l'espace en générant les éléments, l'adaptation de chaque élément étant calculée par un programme d'optimisation local (linéaire par exemple).

En fait, l'association AG-méthodes locales est une quasi-nécessité. Les deux méthodes sont complémentaires et ont des champs d'application différents. L'algorithme génétique permet de faire disparaître la combinatoire du problème, laissant alors le champ libre aux méthodes locales dans chacune des zones connexes qu'il pense susceptible de contenir l'optimum global.

II.3.4- Le parallélisme

L'intérêt de la parallélisation des algorithmes génétiques [12] est de gagner en temps de calcul. Il existe pour cela au moins deux méthodes classiques utilisées:

- La première consiste à diviser la population de taille n en N sous-populations et à les répartir sur l'ensemble des machines dont on dispose.
- La seconde maintient la population totale sur une seule machine mais les évaluations se font sur les autres machines afin qu'elles se fassent en même temps.

Dans les deux cas il est nécessaire d'avoir un mécanisme de communication inter-processus.

a- Parallélisme par îlots

L'idée ici est de faire fonctionner plusieurs algorithmes génétiques en parallèle avec une population réduite pour chacun. Le programme maître lance N occurrences du programme d'algorithmes génétiques appelées esclaves sur des machines différentes en passant à chacune les paramètres nécessaires à leur bon fonctionnement.

Ensuite chaque processus fait évoluer sa population indépendamment jusqu'à ce qu'il décide (selon une probabilité fixée à l'avance) de rassembler ses meilleurs individus pour en transmettre une certaine quantité (proportionnelle à la taille de la population) à une autre machine de son choix. La machine réceptrice intègre alors ces nouveaux éléments dans sa propre population en éliminant les moins bons de ses individus. L'intérêt du parallélisme par

îlots, est qu'il offre la possibilité de travailler sur de grandes populations (n_0) tout en donnant des résultats dans un temps raisonnable puisque la durée nécessaire est à peu de choses près celle qu'il faudrait pour une population de taille $\frac{n_0}{N}$.

Si N est le nombre d'ordinateurs disponibles et si l'on néglige les temps de communication.

La méthode introduit un clustering forcé car chaque îlot peut être considéré comme un cluster subdivisé en petits groupes. Chaque machine a la possibilité de converger vers des optima qui seront différents de ceux calculés sur les autres ce qui correspond au comportement introduit avec le clustering. D'autre part, le surcoût de temps passé pour les communications n'est pas excessif puisqu'il n'y a de communication que de temps en temps. Il faut tout de même garder à l'esprit qu'une subdivision en sous-populations de taille trop réduite risque de conduire à faire tourner des algorithmes génétiques non fiables statistiquement. En effet, il faut quand même qu'une population contienne suffisamment d'individus pour que l'espace d'état puisse être exploré de façon correcte, afin que les résultats aient une certaine valeur.

Des tests ont été faits par *Yann LeFablec* pour déterminer l'efficacité de la méthode (voir figure II.12).

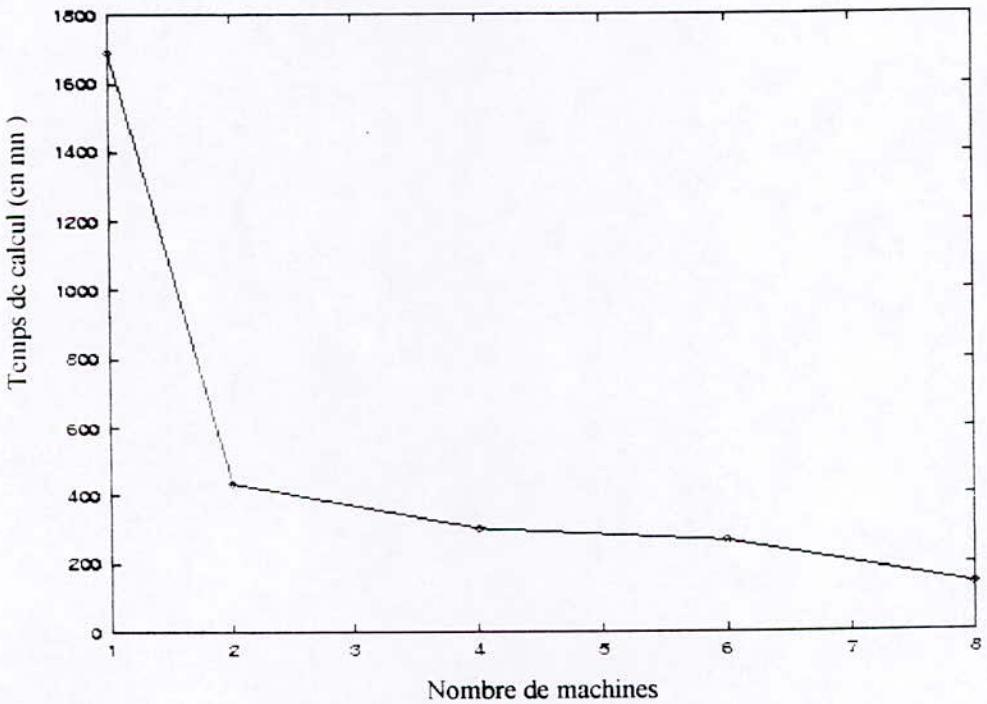


Figure II.12 : Evolution du temps de calcul

Il ne faut s'attacher qu'à la forme générale de la courbe, dans la mesure où les charges locales des machines peuvent modifier de façon sensible les résultats. L'efficacité de la méthode est cependant largement mise en évidence.

b- Parallélisation des calculs

Contrairement à la méthode qui vient d'être décrite qui divise la population totale, on utilise ici des démons de calcul de fonctions objectifs dont la seule fonction est de recevoir un individu et de renvoyer son adaptation. Pour utiliser la puissance de calcul parallèle offerte de manière optimale, il faut retarder au maximum les calculs pour les envoyer en blocs aux démons et faire en sorte qu'aucun ne reste inactif. Le principe se trouve résumé sur la figure II.13 : le programme maître se charge de faire la sélection, les croisements, etc.... En d'autres termes il fait évoluer la population, puis répartit les calculs dont il a besoin sur l'ensemble des démons. Enfin, dès qu'il a reçu tous les résultats, l'algorithme commence une nouvelle génération.

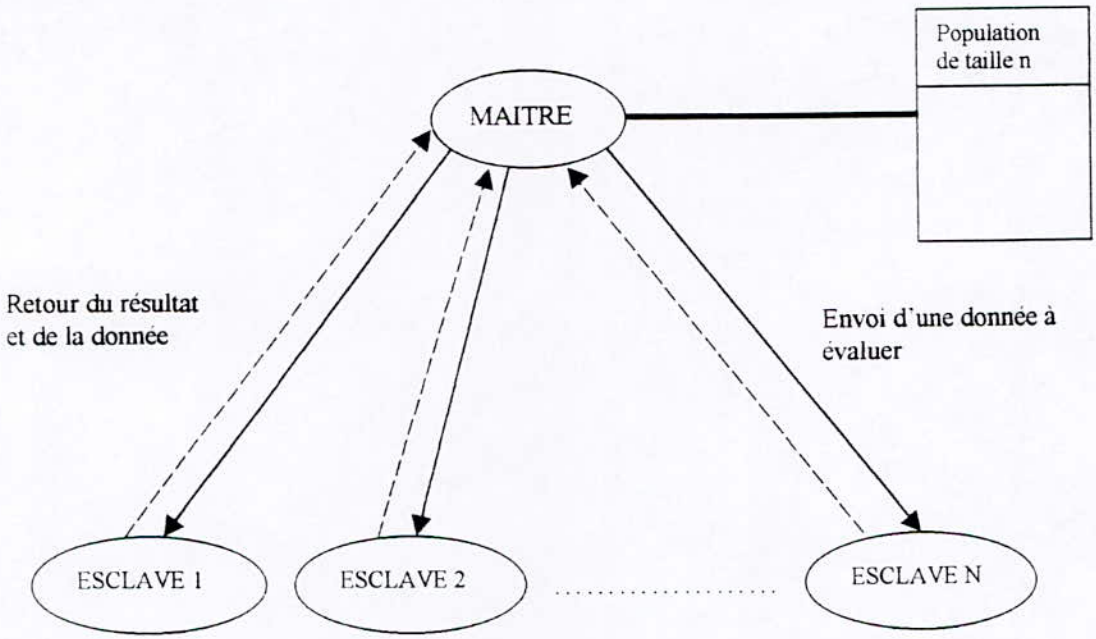


Figure II.13 : Principe de fonctionnement de la parallélisation des calculs.

Il faut noter que ce mécanisme demande un grand nombre de communications pour envoyer les données et les évaluations. La méthode n'est donc intéressante que si le temps passé pour un calcul d'adaptation est grand devant le temps de communication, elle sera par conséquent utilisée pour des problèmes dont les évaluations de fonctions objectifs prennent du temps, on pense essentiellement à des cas faisant appel à des réseaux de neurones ou à de gros calculs matriciels.

La parallélisation est extrêmement efficace pour accélérer les temps de résolution des algorithmes génétiques. Il faut certes bien étudier le problème afin d'utiliser le bon mécanisme de parallélisme, mais les gains en temps sont alors importants.

II.4- Description rapide d'un algorithme génétique

Soit N la taille (fixe) de la population, notons X_k la population de la génération k : il s'agit d'une matrice $X_k=(X_k^1,X_k^2,\ldots,X_k^N)$ de E^N dont les N éléments sont des chaînes de bits (chromosomes) de taille P. Le passage de la génération k à la génération k+1, c'est à dire de X_k à X_{k+1} se décompose en trois étapes :

$$X_k \xrightarrow{Mutation} Y_k \xrightarrow{Croisement} Z_k \xrightarrow{Selection} X_{k+1}.$$

Chacune de ces étapes peut être modélisée formellement.

II.4.1- Mutation $X_k \xrightarrow{Mutation} Y_k$

L'opérateur considéré ici est le suivant : pour chaque composante de chaque élément X_k^i, une variable de *Bernoulli* de paramètre P_m est tirée indépendamment et suivant le résultat l'élément binaire examiné est changé ou non. (0 est changé en 1 et 1 en 0).
La probabilité P_m de mutation doit être préalablement choisie et elle est généralement faible.
Comme nous le verrons par la suite, cet opérateur joue un rôle clé dans la convergence de l'algorithme génétique.

II.4.2- Croisement $Y_k \rightarrow Z_k$

L'opérateur étudié ici est l'opérateur à un point (slicing crossover). Ici encore, la probabilité de croisement P_c est fixée initialement. Pour construire la population Z_k, N/2 couples sont formés à partir de la population Y_k (par exemple en appariant les individus consécutifs de Y_k, ou bien en choisissant au hasard et uniformément des individus dans Y_k). Pour chaque couple, une variable de Bernoulli de paramètre P_c est tirée pour décider si le croisement a lieu. Si c'est le cas, un site de coupure est tiré au hasard, et les segments finaux des deux chromosomes sont échangés.
Une nouvelle paire d'individus est ainsi obtenue (identique à l'ancienne s'il n'y a pas eu de croisement) et est stockée dans la population Z_k. En général, le paramètre P_c est choisi grand.
Remarquons que les opérateurs de mutation et de croisement ne font pas intervenir la fonction f, ce sont des opérateurs stochastiques d'exploration. C'est le troisième et dernier opérateur, la sélection, qui guide la population vers les valeurs élevées de la fonction f.

II.4.3- Sélection $Z_k \rightarrow X_{k+1}$

Les N individus de la population X_{k+1} sont obtenus après sélection des individus de Z_k. On conserve ainsi les meilleurs individus de Z_k, indépendamment à l'aide d'une distribution de probabilité qui favorise les individus de Z_k les mieux adaptés.
Le choix le plus fréquent est l'unique distribution telle que la probabilité d'un individu soit proportionnelle à son adaptation, i.e. la probabilité de sélection de l'individu Z_k^i qui est :

$$P_i = P(Z_k^i) = \frac{f(Z_k^i)}{\sum_{j=1}^{N} f(Z_k^j)} \qquad \text{(II-13)}$$

En tirant les individus dans la population Z_k conformément aux probabilités P_i, on constitue la nouvelle génération X_{k+1}.

II.5- Conclusion

Dans ce chapitre nous avons présenté de façon très générale les algorithmes génétiques vu que le thème principal étant leurs applications dans les problèmes de segmentation, nous avons jugé qu'une description théorique trop exhaustive serait inutile. Dans ce domaine, la pratique devance encore la théorie, même si les mécanismes commencent à être plus clairs nombreuses interrogations demeurent cependant concernant les relations réelles entre les différents paramètres caractérisant l'algorithme génétique et les choix pratiques de ceux-ci. Il reste également à étudier les nombreux raffinements (tels que le scaling, partage, le clustering, l'élitisme) indispensables dans la pratique.

Chapitre III

Segmentation par Champs de Markov et Algorithmes Génétiques

III.1- Introduction

Dans cette application, nous utiliserons une modélisation des textures par les champs de *Gibbs-Markov* [14], [15], [16], qui est un modèle d'interaction spatiale. Il stipule que l'image d'entrée contient plusieurs textures chacune d'entre elles est la réalisation d'un champ de *Markov* (MRF). Il existe dans la littérature plusieurs modèles qui décrivent les (MRF) l'un des plus simples est le modèle généralisé d'*Ising*, lequel nous allons traiter en particulier.

La methode qu'on va décrire s'appelle «la relaxation séléctionniste » [3]. L'estimation des paramètres du modèle repose sur la maximisation d'une fonction de vraisemblance à plusieurs variables. L'algorithme génétique ne va pas seulement servir à optimiser cette fonction, mais constituer le cœur même du processus de segmentation, par l'attribution des labels et la fusion entre les pixel de même région. L'algorithme génétique fait - par un processus d'évolution -, parallèlement, l'estimation des paramètres et la segmentation des régions, et cela en codant de façon judicieuse les individus, et définissant de manière adéquate les opérateurs génétiques.

III.2- Le modèle Markovien de textures

III.2.1- Définitions

a- Les Champs de Markov

On considère un ensemble bidimensionnel de pixels qu'on va appeler S.
La texture A est une réalisation **x** du champ aléatoire X=$\{x_s, s \in S\}$ où x_s est une variable aléatoire prenant ses valeurs dans l'ensemble de tous les niveaux de gris G={0,...,g-1}
x prend donc ces valeurs dans $\Omega = G^{|S|}$, $|S|$ est le nombre de pixels dans S.
Un système de voisinage $N = \{N_s, s \in S\}$ est défini en associant à chaque pixel $s \in S$ un ensemble de voisins N_s tel que :

- $s \notin N_s$
- $s \in N_t \Leftrightarrow t \in N_t$

L'ensemble N_s est appelé le voisinage de s [1], [16], et t est dit voisin de s si $t \in N_s$,il est clair- *de par la définition- que* la relation de voisinage est symétrique.
Un champ aléatoire X est un champ de *Markov* [1], [3], associé au système de voisinage N si et seulement si :

- $P(x) > 0$
- $P(x_s / x_r, r \in S - \{s\}) = P(x_s / x_r, r \in N_s)$

Cela signifie que la connaissance d'un voisinage d'un pixel s, est suffisante pour calculer la loi marginale en chaque pixel.

b- Les cliques

Un sous-ensemble c de S est appelé clique relative au système de voisinage N si :

- C'est un singleton.
- Deux pixels quelconques reliés par une relation de voisinage (au sens de N).

Une clique peut donc être un seul pixel ou un ensemble de deux pixels mutuellement voisins. Par exemple si s et r sont voisin au sens de N on aura la clique $c =< s, r >$.
On peut avoir plusieurs types de cliques pour un même système de voisinage. L'ensemble de toutes les cliques d'un type i est noté C_i.
L'ensemble de toutes les cliques est donc égale à $C = \bigcup_{i=1}^{p} C_i$ où p est le nombre des différents types de cliques, ce nombre dépend de l'ordre du modèle de markov choisi.

c- La fonction potentiel

La dépendance spatiale entre les pixels de l'image est représentée par la fonction potentiel $(V_c(X = x), c \in C)$ associée aux cliques, c'est à dire qu'on va associer à chaque clique une fonction dont la valeur dépend des valeurs des deux pixel s, r qui constituent cette clique, c'est dans le choix de cette fonction que se différencient les modèles de *Markov* proposés pour décrire tel ou tel phénomène.

Pour sortir de l'abstraction on peut voir dans la (figure IV.1) le système de voisinage autour d'un pixel s, il est constitué d'un ensemble d'autre pixels r, pour un modèle d'ordre n on

choisit les pixels qui ont un chiffre- *dans le tableau*- inférieur ou égal à n. A titre d'exemple cette figure montre les différentes cliques d'un modèle d'ordre 2 :

Les cliques montrées en gris son celles qui ont un potentiel non nul c'est à dire qui vont intervenir effectivement.

Numérotation pour avoir le système de voisinage selon l'ordre du modèle choisi.

5	4	3	4	5
4	2	1	2	4
3	1	s	1	3
4	2	1	2	4
5	4	3	4	5

Système de voisinage d'ordre2

r	r	r
r	s	r
r	r	r

Les quatre types de cliques pour le système de voisinage d'ordre 2.

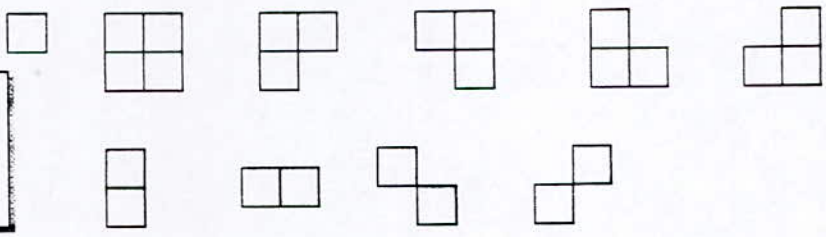


Figure III.1 : Système de voisinage et cliques d'ordre 2 correspondantes.

III.2.2- Théorème de *Hammersley-Clifford*

Le champ aléatoire X est un champ aléatoire de *Markov* sur S si pour un système de voisinage donné la distribution de probabilité est une distribution de *Gibbs* [1], [3],[14],[15],[16], c'est à dire :

$$\forall x \in \Omega, P(X = x) = exp(-E(x))/Z \qquad \text{(III-1)}$$

où :

Ω est l'ensemble de toutes les configurations possibles $\Omega = G^{|S|}$.

$E(x) = \sum_{c \in C} V_c(x)$ est l'énergie de la configuration x.

Z est une constate de normalisation (pour avoir $p \leq 1$) elle contient l'énergie de toutes les configurations possibles elle est appelée fonction de partition qui est définie par :

$$Z = \sum_{y \in \Omega} \exp\{-E(y)\} \qquad \text{(III-2)}$$

Dans le modèle généralisé d'*Ising*, seulement les cliques contenants 2 pixels ont un potentiel non nul. A chaque type de clique C_i est associé un paramètre β_i. Les β_i constituent le vecteur B (c'est lui qui différencie une texture d'une autre). Le nombre de composants est bien sur égale au nombre de types de cliques. Dans le modèle précédent c'est un vecteur à 4 éléments.
La fonction potentiel s'écrit $V_c(x) = \Delta_c(x)\beta_i$,
où :

$$\Delta_c = \begin{cases} -1 & si \quad x_r = x_s \\ +1 & si \quad x_r \neq x_s \end{cases} \tag{III-3}$$

tel que c={s,r}.

On peut écrire d'une autre manière la fonction énergie :

$$E(x, B) = B.K(x)^t \tag{III-4}$$

où le vecteur $K(x) = (k_1(x),...,k_p(x))$ est défini par :

$$k_i(x) = \sum_{c \in C_i} \Delta_c(x) \tag{III-5}$$

Pour résumer les choses, chaque texture est une réalisation aléatoire déterminée par son vecteur B. Pour connaître la probabilité d'avoir dans un pixel donné un niveau de gris donné, on utilise le théorème de *Hammersley-Clifford*. Ce dernier va faire intervenir la fonction énergie qui est la somme des potentiels des pixels dans un proche voisinage donc cette probabilité va dépendre de l'entourage du pixel s et du vecteur de texture B.
Mais dans la pratique le vecteur B n'est pas connu et dans les applications de segmentation on a souvent plusieurs textures dans l'image qu'il faut séparer, le problème revient donc à estimer les vecteurs B de chaque texture. Nous utiliserons, dans notre étude, un critère de maximum de vraisemblance.

III.2.3- Approximation de la fonction de partition

La fonction Z décrite dans le paragraphe précédent est appelée fonction de partition, elle permet d'avoir une probabilité de vraisemblance inférieure à 1. En calculant l'énergie de toutes les configurations de niveaux de gris possibles dans une fenêtre d'analyse w*w, dans la pratique une telle approche n'est pas possible, vu le très grand nombre de configurations possibles (a titre d'exemple pour une image à 256 niveaux de gris et en utilisant une fenêtre d'analyse de 8*8 on aura à calculer l'énergie de 256^{64} configurations !). C'est pour cela qu'on utilise une approximation fondée sur un développement au second ordre de chaque terme de la fonction de partition $Z_W(B)$ de la manière suivante :

$$\exp\{-E(y;B)\} \approx 1 - E(y;B) + \tfrac{1}{2}E(y;B)^2 \tag{III-6}$$

Les termes approximés sont sommés sur tout $y \in \Omega_W$, $\Omega = G^{|S|}$. Avec un développement mathématique (voir annexe 1), on aboutit à la relation suivante :

$$\widetilde{Z}_W(B) = g^{n-2}\left\{ g^2 - ng(g-2)\sum_{i=1}^{p}\beta_i + 2n(g-1)\sum_{i=1}^{p}\beta_i^2 + n^2/2(g-2)^2\left(\sum_{i=1}^{p}\beta_i\right)\right\} \tag{III-7}$$

où :

g est le nombre de niveaux de gris dans l'image.
$n = w \times w$ est le nombre de pixels dans la fenêtre d'analyse.
Les B_i sont les éléments du vecteur paramétrique d'une texture.

Il est intéressant de noter que cette approximation est valable quel que soit l'ordre du modèle de *Markov* utilisé, et c'est cette dernière formule qui sera finalement utilisée lors de l'implémentation de l'algorithme.

III.3- La relaxation sélectionniste

La relaxation sélectionniste [3], est une méthode de segmentation de textures non supervisée , c'est à dire que l'algorithme ne connaît aucune information a priori sur l'image. Le passage d'une image d'entrée contenant différentes textures à une image de sortie labelisée se fait par le biais d'un algorithme génétique distribué qui fait interagir une population d'unités, chaque unité correspond à un pixel dans l'image et elle est définie par deux donnés :

- Un vecteur candidat de paramètres B_s
- Un label L_s

Nous associons, par la suite, à chaque unité une fonction d'adaptation qui mesure à quel point les B_s de cette unité correspondent à la texture locale. Enfin, les opérateurs génétiques sont appliqués et la population d'unités va évoluer et ceux qui auront les meilleurs valeurs de la fonction d'adaptation vont être sélectionnés et croisés, et par conséquent vont remplacer les unités qui les entourent et vont proliférer dans leurs régions, la mutation va garantir une diversité dans la population. A la fin, chaque texture aura un seul label " gagnant " qui lui sera associé. La séparation des régions sera alors très facile.

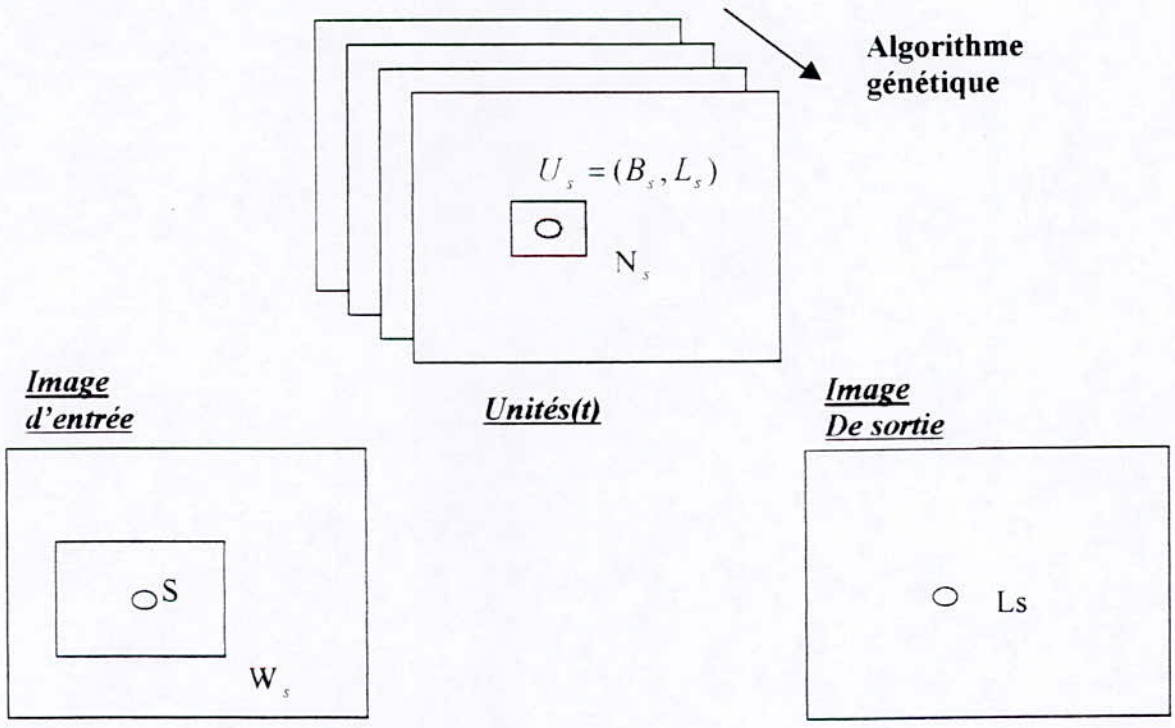


Figure III.2 : Présentation de l'algorithme

III.3.1- Les unités

Comme on l'a dit, à chaque pixel sera associé une unité $U_s = \{B_s, L_s\}$. Une collection d'unités est appelée population. Notre objectif est d'avoir à la fin une collection de labels. Comme on l'a vu dans ch. II, pour utiliser un algorithme génétique il faut définir une fonction d'adaptation qui doit mesurer la correspondance des paramètres B_s à la texture présente en U_s. Cette fonction est tout simplement la probabilité de vraisemblance du modèle généralisé d'*Ising*, calculée sur une fenêtre w_s de taille w*w centrée sur le pixel s on a donc :

$$f(U_s) = \exp\{-E(x_s, B_s)\} / \widetilde{Z}_{W_s}(B_s) \qquad \text{(III-8)}$$

où $\widetilde{Z}_{W_s}(B_s)$ est la fonction de partition donnée dans par (III-7).

III.3.2- L'algorithme génétique

a- L'initialisation

La population initiale est générée comme suit :
Pour chaque unité U_s chaque composant $B_{s,l}$ du vecteur de paramètres B_s est initialisé aléatoirement de façon uniforme dans l'intervalle $[-\delta, \delta]$. Comme on l'a déjà expliqué les paramètres B_s doivent être suffisamment petits pour que l'approximation de la fonction d'adaptation soit valable, expérimentalement le choix de $\delta = 1/w^2$ donne de bons résultats. Quant aux labels, ils doivent être tous différents. Donc on aura autant de labels que de pixels dans l'image et toutes les unités sont des contours.

Après l'initialisation de la population, la relaxation précédentes consiste en l'itération des opérateurs de sélection de croisement et de mutation sur chaque unité jusqu'à ce qu'un critère d'arrêt soit satisfait. Les opérateurs génétiques sont définis de la façon suivantes :

b- La sélection

La sélection se fait sous forme de tournoi locale. Pour chaque unité on choisit dans son voisinage N_s l'unité qui présente la plus grande fonction d'adaptation, celle-ci sera sélectionnée pour remplacer l'unité centrale U_s, cela dans le cas où l'unité en question serait non-contour. Si c'est un contour le tournoi ne se fait pas dans l'entourage direct de l'unité mais partout ailleurs dans l'image est cela pour permettre à des régions éloignées d'interagir entre elles. Dans ce dernier cas (unité contour) il n'y a pas de croisement ni de mutation.

c- Le croisement

C'est une variante du croisement uniforme. Une unité est choisie aléatoirement dans le voisinage N sur laquelle on choisit (aléatoirement aussi) un composant du vecteur B_s et on affecte sa valeur au composant correspondant de l'unité centrale U_s (sur laquelle on applique le croisement).

d- La mutation

Une position aléatoire l est choisie le long du vecteur B_s, puis on ajoute au composant $B_{s,l}$ correspondant une valeur m choisie aléatoirement dans l'intervalle $[-\mu_s, \mu_s]$ la valeur de

μ_s doit être telle que m soit inférieur a δ et en plus, dépendre de la texture locale. On utilise donc cette formule :

$$\mu_s = \varepsilon\delta \left\{ 1 - \frac{\left| k_l (x_{W_s}) \right|}{1 + w^2} \right\} \qquad \text{(III-9)}$$

Le premier terme $\varepsilon\delta$ réduit μ_s par rapport à δ en mettant ε petit l'expérience montre qu'une valeur de 0.02 est tout à fait satisfaisante. Le second terme fait dépendre l'amplitude de la mutation de la texture locale en introduisant le terme $|k_l|$. Si celui-ci est petit, cela veut dire qu'une bonne moitié des cliques de type l n'a pas le même signe que l'autre moitié ce qui induit une confusion, dans ce cas le taux de mutation doit croître pour sortir de cette état de confusion, par contre si $|k_l|$ est grand cela veut dire que les cliques de type i ont le même signe dans le voisinage du pixel s (ce qui est normal pour une même texture). Et cela veut dire que nous sommes sur le « bon » pixel et que pour y rester, il faut réduire le taux de mutation.

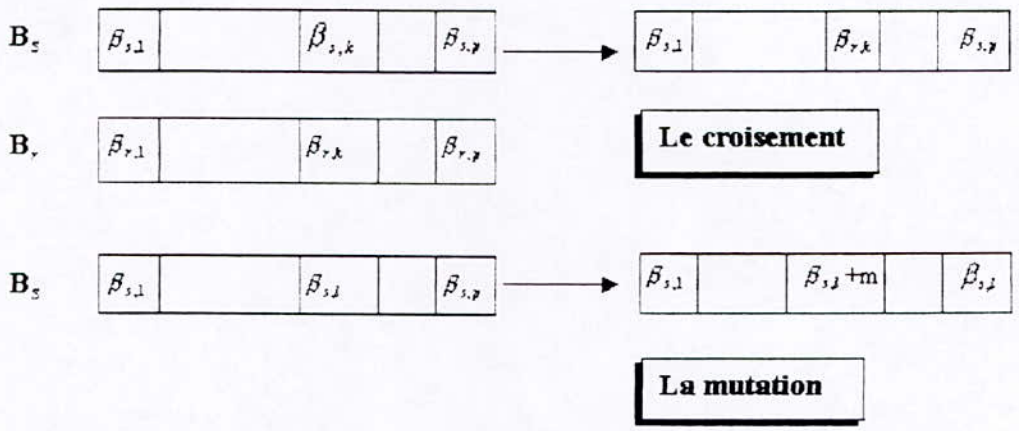


Figure III.3: Effet des opérateurs de croisement et de mutation sur les vecteurs de paramètres *B*.

III.4- Un algorithme supervisé pour la segmentation des images texturées et non texturées

III.4.1- Description

Cet algorithme a été conçu dans le cadre de ce projet principalement pour palier au fait que les méthodes de segmentation de textures sont souvent inadaptées aux images non texturées. Par exemple, la méthode de relaxation précédentes par algorithmes génétiques donne des résultats médiocres sur des images non texturées. Cela est dû à la nature des paramètres B qui peuvent seulement discriminer entre les textures. Quant aux images normales, les B seront presque les mêmes pour toute l'image même si cette dernière contient plusieurs régions différentes.

L'idée est d'utiliser un mélange entre des paramètres de textures et des paramètres statistiques, les premiers seront ceux du modèle de *Markov* on utilisera les $k_i(x)$ plutôt que les

$\beta_i(x)$ car les $k_i(x)$ sont calculés une fois seulement sur l'image entière, alors que les $\beta_i(x)$ sont inconnus et doivent être préalablement estimés. Ce qui induit un surcoût en terme de précision et de temps de calcul.

Le vecteur $K(x)$ est défini comme suit :

$$K(x) = (k_1(x),\ldots,k_p(x)) \tag{III-10}$$

$$k_i(x) = \sum_{c \in C_i} \Delta_c(x) \tag{III-11}$$

Les Δ_c sont calculés à partir de la relation (III-3).

Pour chaque pixel ce vecteur peut être calculé sur une fenêtre $w \times w$ autour de ce pixel les éléments du vecteur peuvent être positifs où négatifs leurs valeurs sont limitées telles que :

$$-w^2 < k_i(x) = \sum_{c \in C_i} \Delta_c(x) < w^2 \tag{III-12}$$

Maintenant, on introduit la mesure des statistiques du premier et du deuxième ordre (cf. §I.1.5) à l'intérieur de la fenêtre $w \times w$ autour de chaque pixel. Soit le pixel s de position (i, j) :

$$m_1(s_{i,j}) = \frac{1}{w^2} \sum_{k=i-\frac{w}{2}}^{i+\frac{w}{2}} \sum_{l=j-\frac{w}{2}}^{j+\frac{w}{2}} s_{k,l} \tag{III-13}$$

$$\sigma(s_{i,j}) = \sqrt{\frac{1}{w^2} \sum_{k=i-\frac{w}{2}}^{i+\frac{w}{2}} \sum_{l=j-\frac{w}{2}}^{j+\frac{w}{2}} \left(s_{k,l} - m_1(s_{i,j})\right)^2} \tag{III-14}$$

Ces éléments vont constituer avec les $K_i(x_s)$ un nouveau vecteur V_s caractérisant chaque pixel s est et qui est défini comme suit :

$$V(s) \begin{pmatrix} k_1(x_s) \\ k_2(x_s) \\ \cdot \\ \cdot \\ \cdot \\ k_p(x_s) \\ a\, m_1(s) \\ b\, \sigma(s) \end{pmatrix} \tag{III-15}$$

Les constantes a et b sont utilisées pour assurer l'homogénéité entre les éléments du vecteur qui n'ont pas la même nature, la condition à satisfaire est la suivante :

$$\left(Max(\alpha\, m_1(s)) - Min(\alpha\, m_1(s))\right) = \left(Max(k_i(x_s)) - Min(k_i(x_s))\right) \tag{III-16}$$

D'après (III-12) :

$$\alpha\left(Max(m_1(s)) - Min(m_1(s)\right) = \frac{1}{2}\omega^2$$

$$\alpha = \frac{1}{2\left(Max(m_1(s)) - Min(m_1(s)\right)}\omega^2 \qquad \text{(III-17)}$$

On applique le même développement pour b et on trouve :

$$b = \frac{1}{2\left(Max(\sigma(s)) - Min(\sigma(s)\right)}\omega^2 \qquad \text{(III-18)}$$

III.4.2- Prise d'échantillons

Dans une première étape des échantillons sont prélevés sur chaque région dans l'image sur lesquelles. On calcule le vecteur $V(s)$. Les échantillons doivent être pris à des endroits représentatifs pour chaque région afin que la segmentation soit précise.
On constitue donc un ensemble de vecteurs modèles $V_l\big|_{l=1,n}$ où n est le nombre de régions dans l'image, et à chaque vecteur V_l est associé un label l. La segmentation consiste à attribuer à chaque pixel s dans l'image le label L_i correspondant à sa région

III.4.3- Phase de segmentation

Après la prise des échantillons, on calcule le vecteur $V(s)$ pour chaque pixel s de l'image et on trouve parmi les vecteurs modèles, celui qui est le plus proche de $V(s)$ au sens de la distance euclidienne. C'est à dire trouver le label l qui minimise :

$$Dist(V(s),V_l) = \|V(s) - V_l\|$$

$$Dist(V(s),V_l) = \sqrt{\sum_{i=1}^{p}\left(k_i(x_s) - k_{i,l}\right)^2 + \left(m_1(s) - m_{1,l}\right)^2 + \left(\sigma(s) - \sigma_l\right)^2} \qquad \text{(III-19)}$$

A la fin on aura une image dans laquelle chaque pixel aura un label et appartiendra donc à une région bien précise.

III.5- Conclusion

Dans ce chapitre, on a présenté deux méthodes de segmentation d'images basées sur la modélisation par les champs de *Markov*. Tout au long du chapitre, on n'a utilisé que le modèle d'*Ising* d'ordre 2, et ce pour sa simplicité. Cependant, il existe d'autres modèles plus compliqués et qui peuvent mieux segmenter certaines images qui ont des motifs de textures plus étalés spatialement –plus proche des textures réelles- le modèle du cinquième ordre [3] avec 24 voisins et 12 type de cliques est particulièrement efficaces, mais il faudrait, dans ce cas, faire des compromis en matière de temps de calcul.

L'algorithme exposé dans §III.4 nécessite sans doute quelques améliorations, la fixation empirique de certains paramètres- bien qu'efficaces- ne permettent pas une généralisation de cette approche. Un développement théorique plus poussé s'impose, surtout pour déterminer les relations effectives qui peuvent exister entre les paramètres markoviens et les grandeurs statistiques comme la moyenne ou la variance. L'utilisation de statistiques d'ordre supérieur et aussi à étudier.

Les résultats d'implantation pour la relaxation sélectionniste et l'algorithme supervisé sont exposés dans le chapitre V.

Chapitre IV

Segmentation par Algorithmes Génétiques et Attributs Fractals

IV.1- Introduction

Il existe deux approches pour la segmentation, l'approche régions et l'approche contours. La première consiste à subdiviser l'image en différentes régions contenant des caractéristiques homogènes. La deuxième est la fixation des limites qui entourent les différents objets dans l'image. Cette méthode s'inscrit dans ce deuxième cas c'est à dire la détection de contours.

Jusqu'ici, plusieurs méthodes de détection de contours ont été développées, telles que l'operateur de *Sobel*, l'opérateur de *Dériche* et l'opérateur de *Roberts*, [1], [9], [10], [11]. Mais ils ne sont pas en mesure de détecter les contours des images texturées. Les formes locales des textures sont reconnues comme des contours quand on se base sur les pixels. Par conséquent on extrait les caractéristiques de la texture dans une petite région en utilisant quelques méthodes de mesure des textures est accomplir la détection du contour suivant la variance des caractéristiques dans le voisinage de la région.

Cet algorithme [4], peut être résumé comme suit :

Après la division de l'image en plusieurs petites régions (par fenêtrage) de même taille, on calcule les paramètres du modèle AR [5], la dimension fractale, [1] la moyenne et la variance de chaque petite région et on construit ce qu'on appelle : « vecteur caractéristique » pour ces régions. Après ça, on calcule la variance du vecteur caractéristique dans une région et les huit autres régions qui l'entourent. Cela étant fait, on procède à la détection grossière des contours dans la région qui a la plus grande valeur de variance et les régions candidates sont obtenues. Après on cherche graduellement dans ces régions candidates les régions optimales par la répétition de l'information du traitement génétique et l'arrangement des régions, est accompli en partant de l'idée que le plus court individu dans les régions candidates est l'individu optimal.

IV.2- Caractérisation de la texture

Dans ce travail, le modèle AR bidimensionnel pour une texture aléatoire $\{x_{ij}, i = 1,2,\cdots,I; j = 1,2,\cdots,J\}$ est donné par :

$$x_{ij} = \sum_{p=0}^{P}\sum_{q=0}^{Q} a_{pq} x_{i-p,j-q} + n_{ij} \qquad \text{(IV.1)}$$

avec $a_{00} = 0$, et n_{ij} est un bruit blanc gaussien de moyenne nulle et de variance σ_n^2.

Ce modèle est appelé modèle causal. Il est représenté sur la figure IV.1. P et Q désignent l'ordre du modèle. Les coefficients $\{a_{pq}, p = 0,1,\cdots,P; q = 0,1,\cdots,Q\}$ et σ_n^2 sont estimés par la méthode des moindres carrés. Ces paramètres sont utilisés pour caractériser la texture. Dans notre cas, P=Q=1.

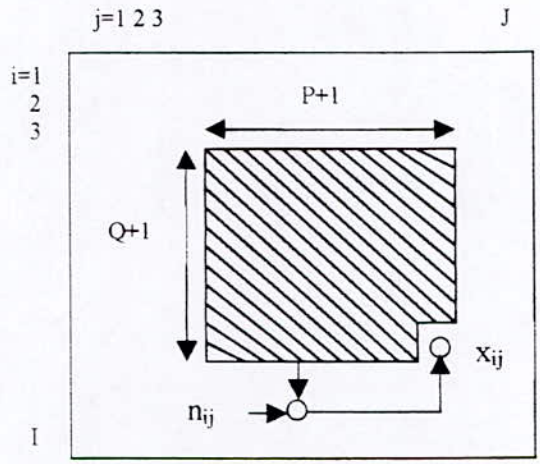


Figure IV.1 : Modèle AR bidimensionnel

Pour les données $\{x_{ij}, i = 0,1,2,\cdots,2^n; j = 0,1,2,\cdots,2^n\}$ contenues dans une petite région de $N \times N$ pixels, (représentée sur la figure IV.1), la dimension fractale est obtenue par la méthode suivante :

- Considérons les données $\{x_{ij}, i = 0,r,2r,\cdots,2^n; j = 0,r,2r,\cdots,2^n\}$ qui sont définies pour chaque r, où $\{r = 1,2,\cdots,2^{m-1}; m \le n\}$.
- On prend la valeur de 4 pixels, x_1, x_2, x_3, x_4, sur 4 points adjacents $p_1(kr, lr)$, $p_2((k+1)r,lr)$, $p_3(kr,(l+1)r)$ et $p_4((k+1)r, (l+1)r)$.
- On définit les points P_1, P_2, P_3, P_4 dans l'espace tridimensionnel (i,j,x) montré sur la figure IV.2.
- Les aires $s_1(k,l)$, $s_2(k,l)$ des deux triangles $\Delta P_1P_2P_3$ et $\Delta P_2P_3P_4$ sont calculés respectivement où $k = 0,1,2,\cdots,\frac{2^{n-1}}{r} - 1$, $l = 0,1,2,\cdots,\frac{2^{n-1}}{r} - 1$. Ces aires sont calculées pour tous les k et l. La surface totale est égale à leur somme :

$$S(r) = \sum_k \sum_l \{s_1(k,l) + s_2(k,l)\} \quad \text{(IV.2)}$$

- La surface N(r) d'une petite région est définit comme :

$$N(r) = \frac{S(r)}{r^2} \quad \text{(IV.3)}$$

- En utilisant N(r) obtenue pour différentes valeurs de r, on peut avoir l'équation suivante :

$$\log(N(r)) = a + b\log(r) \quad \text{(IV.4)}$$

Ainsi, on définit la dimension fractale $D = -b$.

En plus on calcule la moyenne μ et la variance σ^2 (cf. §I.1.5) de chaque région, et on obtient le vecteur caractéristique qui est constitué par $\{a_{ij}\}$, σ_n^2, D, μ et σ^2.

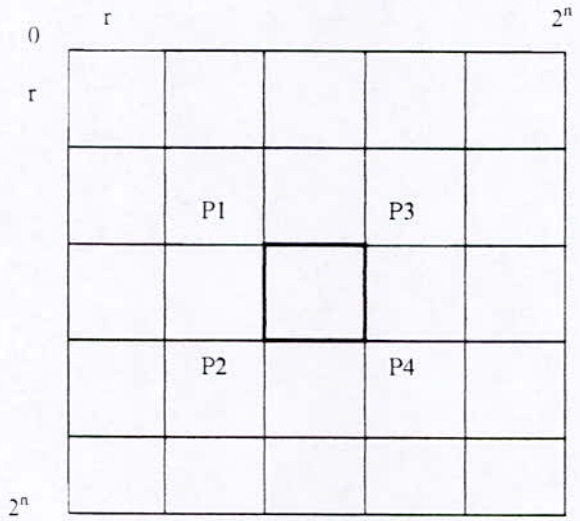


Figure IV.2 : La fenêtre pour l'obtention de la dimension fractale

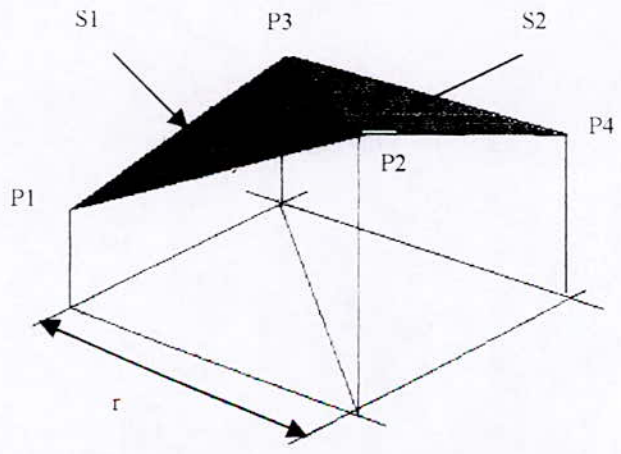
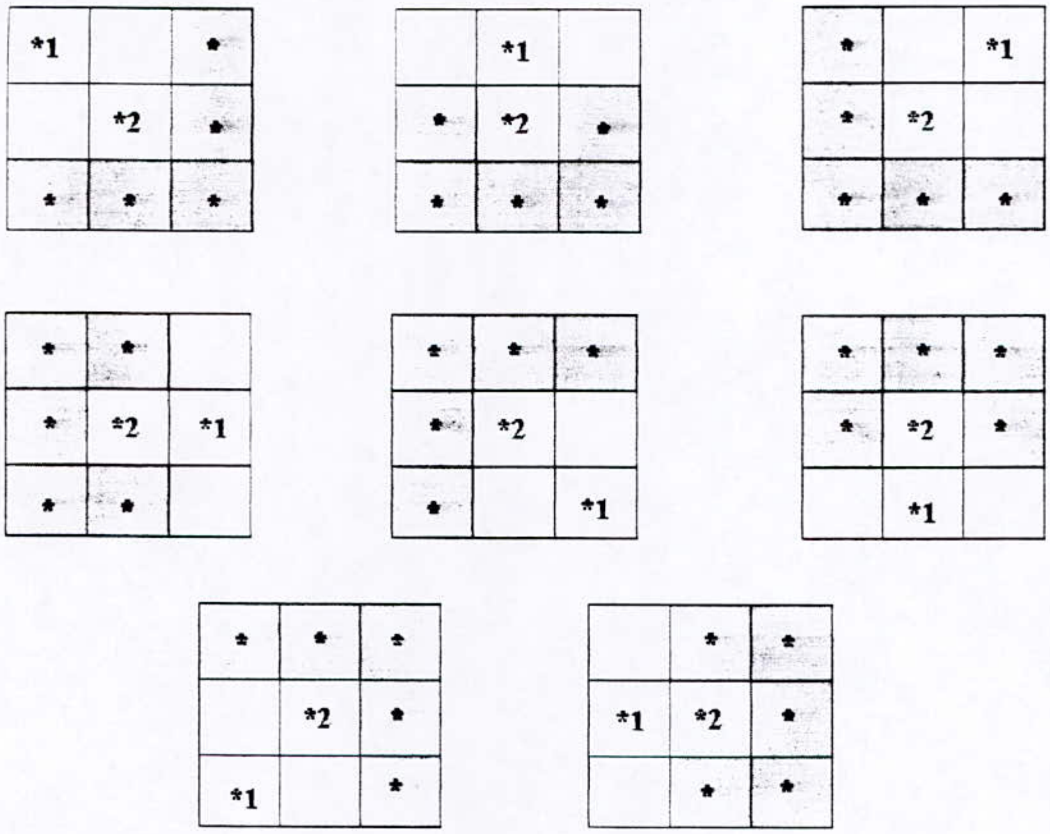


Figure IV.3 : La méthode pour obtenir la dimension fractale

IV.3- La détection des régions contours candidates

Dans cette méthode l'algorithme de détection de contours est constitué de deux parties. L'élection des régions contours candidates et leur arrangement. Le processus de l'élection des régions candidates est le suivant :



*1 & *2 : Points cherchés

* : Points de recherche

Figure IV.4 : Méthode de recherche du point suivant

1. On divise l'image originale en plusieurs régions rectangulaires de même taille $\{r_{ij}, i = 0,1,2,\cdots,I; j = 0,1,2,\cdots,J\}$. Et on extrait le vecteur caractéristique z_{ij} dans chaque petite région.
2. On calcule la variance de ce vecteur au centre et dans les 8 régions voisines donnée par :

$$v_{ij} = \frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} z(i+k, j+l) - \bar{z}_{ij} \qquad \text{(IV.5)}$$

où $\bar{z}_{ij}$ est le vecteur caractéristique moyen dans le centre et les 8 voisins.

3. On sélectionne la fenêtre qui inclus la région qui a une plus grande valeur que *s*, où *s* est variable. On obtient ainsi l'ensemble $\Re$ des régions qui ont une valeur plus grande que *s*. Dans cette recherche, chaque fenêtre consiste en $K \times L$ petites régions et les fenêtres voisines sont recouvertes par $\frac{K}{2} \times \frac{L}{2}$ mutuellement. K et L sont variables aussi.

4. Dans chaque fenêtre, on choisit de façon aléatoire le point de départ $\{r_m \in \Re, m = 1,2,\cdots,M\}$, et on commence la recherche de la région candidate. Dans ce cas, le second point est la région qui a la valeur juste en dessous du point de départ.

On choisit les points de recherche qui suivent comme le montre la figure IV.4.
On répète l'étape 5 jusqu'à ce que la longueur de la séquence des régions atteigne x (x variable) ou on arrive à la fin de la fenêtre. Et on répète le processus depuis le début jusqu'à M, où M est le nombre de r_m dans la fenêtre de recherche. Ensuite la même procédure est appliquée à la fenêtre suivante. Et quand on termine toutes les fenêtres, on obtient la séquence de M contours candidats dans chaque fenêtre. Tous les ensembles des séquences correspondent aux régions candidates. Après on effectue l'arrangement de ces régions en utilisant les algorithmes génétiques

IV.4-L'algorithme génétique

IV.4.1- Le Codage

Pour appliquer les algorithmes génétiques sur n'importe quel problème pratique, la représentation structurelle des solutions suivant les contraintes du problème est nécessaire.

Dans cette méthode un individu est représenté comme une combinaison de la séquence des contours obtenue précédemment. On choisit aléatoirement une séquence dans chaque petite fenêtre et on constitue l'individu. Ensuite, on attribue le label 1 aux petites régions constituant la séquence choisie, et le label 2 pour des régions autres mais qui constituent, elles aussi, une séquence. On donne le label 0 pour les régions qui restent. Ainsi, toutes les régions qui ne contiennent pas de contours garderont le label 0 jusqu'à la fin de l'algorithme. De cette manière, on peut obtenir la séquence bidimensionnelle qui inclue l'individu « g » (Voir figure IV.5). Cette séquence bidimensionnelle, montre l'état des régions contours dans une image. Ces régions correspondent à un individu.

0	0	0	0	0	1	1	2
0	0	0	0	2	1	2	0
0	0	0	2	1	2	0	0
0	0	0	1	2	0	0	0
0	0	2	1	0	0	0	0
0	2	1	2	0	0	0	0
2	1	2	0	0	0	0	0
1	2	0	0	0	0	0	0

Petite région

1	1	1	1	1	1	1	1

Individu ***g***

Figure IV.5 : Un exemple d'individu

IV.4.2- L'opérateur de croisement

Nous utiliserons, dans cette méthode, le croisement uniforme. Il est effectué par la génération d'un masque bidimensionnel. Puis échanger les régions suivant ce masque. C'est à dire, quand on rencontre un 0 on prend une région du premier parent, et une région de l'autre parent si on rencontre 1 (figure IV.6).

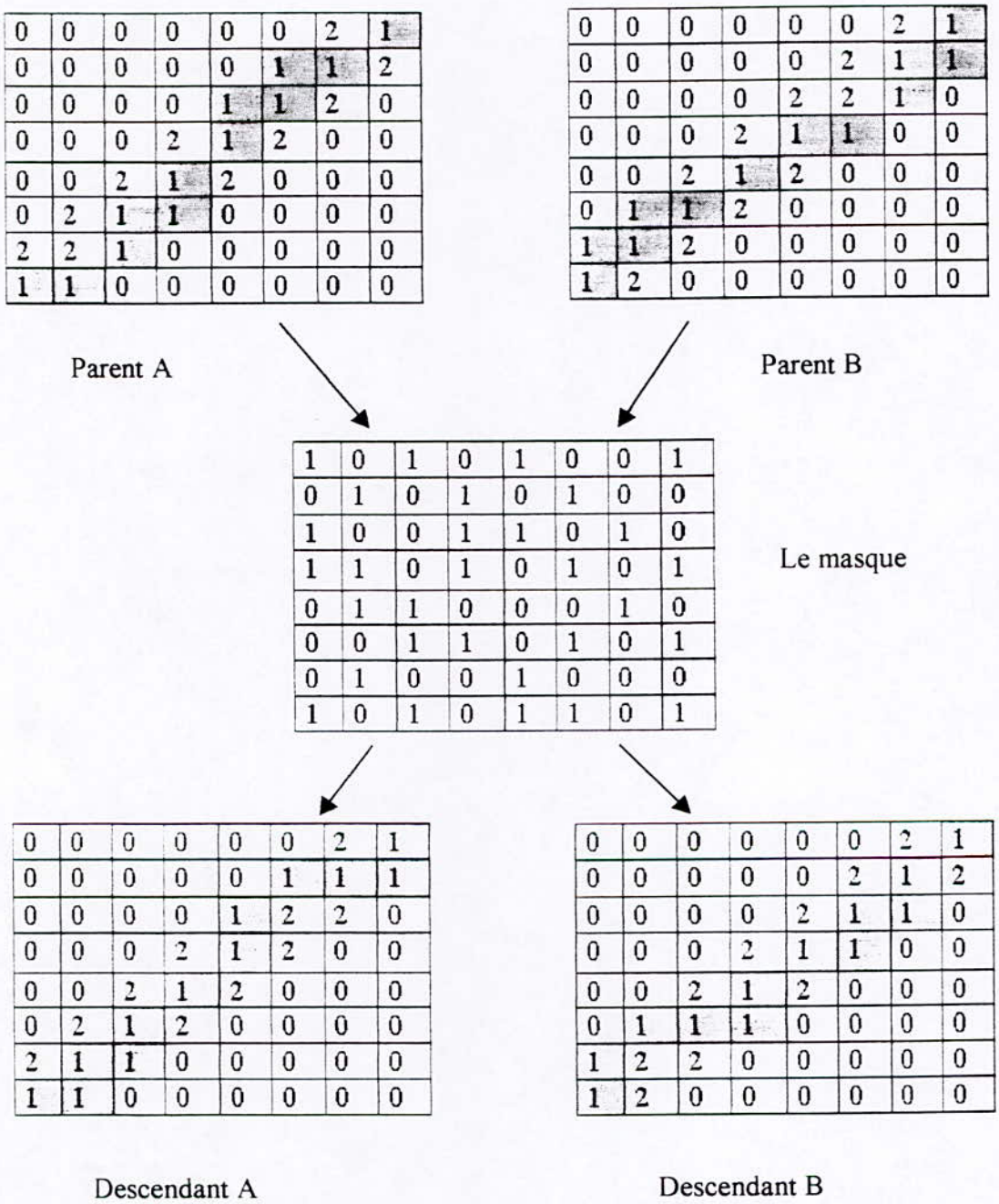
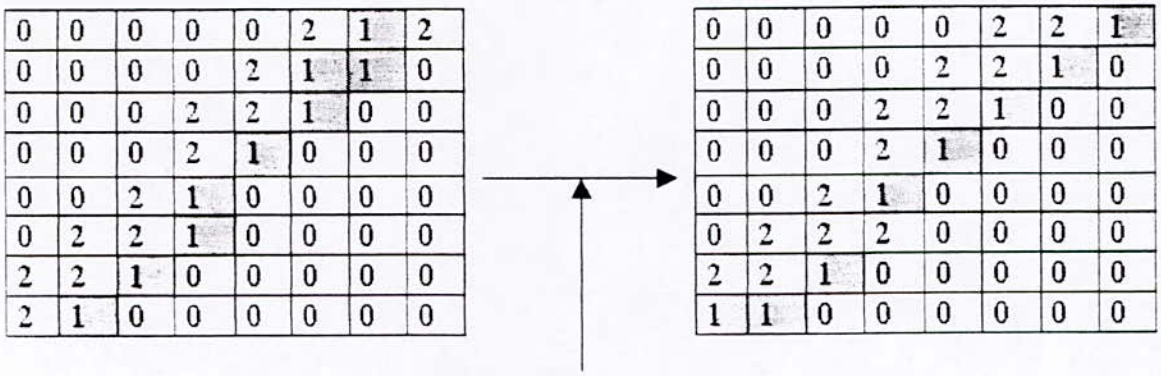


Figure IV.6 : Un exemple de l'opérateur de croisement

IV.4.3- L'opérateur de mutation

On utilise ici la mutation par bit. C'est à dire, on choisit le bit (ou la petite région) sur lequel on va effectuer la mutation avec une probabilité uniforme en changeant le 1 par 2 et l'inverse. (figure IV.7).



Points de mutation : (0,6) ;(0,7) ;(1,5) ;(5,3) ;(7,0)

Figure IV.7 : Un exemple de l'opération de mutation

IV.4.4- La fonction d'adaptation

La fonction d'adaptation utilisée est donnée par la formule suivante :

$$eval(g) = V - \sum_{m=1}^{M} v_m \qquad \text{(IV.6)}$$

où V est la somme totale des v_m, et v_n (n=1,2,...,M) n'est pas utilisé si le nombre des petites régions contours dans le voisinage pour chaque v_m sur l'individu est supérieur ou inférieur à 2. Cette contrainte résulte de l'idée que le plus court individu, sur lequel r_m est attaché, dénote la région contour la plus optimale. Par conséquent, plus la valeur du terme représentant la somme dans la fonction d'adaptation est petite, plus l'individu est robuste et sa probabilité de survivre à la génération suivante augmente.

IV.5- Optimisation des régions candidates par algorithme génétique

Après la détection de contours grossière déjà mentionnée, les algorithmes génétiques sont appliqués pour arranger les régions candidates selon le processus suivant :

1. Initialiser la première population aléatoire. Comme on l'a déjà vu, un individu est généré par la répétition du choix de la séquence des régions contours candidates dans chaque fenêtre qui inclue r_m. La taille de la population est $3 \times p$.
2. On calcule la valeur d'adaptation de chaque individu, et on ordonne les individus suivant leurs longueurs.

3. On copie les 10 plus longs. On choisit ensuite les p-plus longs individus et on effectue le croisement entre eux avec une probabilité uniforme. On aura ainsi $2 \times p$ individus. On procède alors à la mutation sur les mêmes p individus par la méthode décrite précédemment, et on garde les p nouveaux individus. Arrivé à ce stade et en faisant la somme on aura ainsi une population de $3 \times p + 10$ individus.
4. On calcule une nouvelle fois la valeur d'adaptation de chaque individu et on les ordonne suivant leurs longueurs.
5. On choisit les p-plus longs et on construit la nouvelle population. Chaque nouvelle génération aura ainsi une génération de p individus exceptée la première.

Le processus est répété de l'étape (2) à (5) jusqu'à ce que le critère d'arrêt soit satisfait.

IV.6- Conclusion

Dans ce chapitre, nous avons présenté une méthode de segmentation (approche contours) d'images texturées qui utilise un algorithme génétique de type combinatoire. Les attributs utilisés dans cette méthode sont la dimension fractale, la variance (cf. §I.1.5), et le modèle AR (*autorégressif)* à deux dimensions.

Il est très important pour une texture d'extraire ses caractéristiques de façon précise, et plusieurs méthodes de mesure ont été développées. Le modèle autorégressif (AR) peut exprimer les caractéristiques dynamiques de l'image dans le voisinage des pixels, c'est pour ça qu'il est souvent utilisé dans la compression des données, la synthèse et le traitement pour la restauration d'images.

La dimension fractale est capable de représenter les caractéristiques de la texture dans un voisinage plus large que le modèle AR. Dans la méthode présentée, on a utilisé ces deux grandeurs pour caractériser les textures et ensuite optimiser la détection des contours par les algorithmes génétiques. Dans ce cas, l'optimisation est faite localement dans des régions candidates qui contiennent les contours pour que l'espace de recherche soit plus restreint et donc permettre une convergence rapide pour les régions optimales.

A cause du modèle AR on n'a pas pu tester cette méthode car l'estimation des paramètres a_{ij}, requiert la solution d'un système d'équations normales par application de l'algorithme de *Levinson* sur une matrice de type Toeplitz bloc Toeplitz.

Chapitre *V*

Résultats et interprétations

V.1- Présentation du Logiciel

Nous avons choisi d'implanter les algorithmes sous forme de logiciel sous Windows, cela permet une bonne visualisation des données et par conséquente une meilleure interprétation des résultats. En plus de cela on peut jouer facilement sur les paramètres et tester les limites de chaque algorithme.

Le langage de programmation choisi est le langage C ++ [24], avec le compilateur Visual C++ 5.0 [25], [26]. Ce choix peut être justifié par :

- Le temps de calcul étant un facteur important - *les AG sont des algorithmes lents*- l'utilisation d'un langage comme MATLAB serait inadéquate, le C++ est reconnu comme un langage rapide.
- La structure orientée - objet de C++ facilite la programmation d'un grand nombre d'algorithmes avec un minimum de codes.
- Le C++ est le langage le plus utilisé au niveau des laboratoires de recherche
- La convivialité et les outils fournis par Visual C++ rendent plus facile la création de fenêtres, de boites de dialogue, de menus …etc.

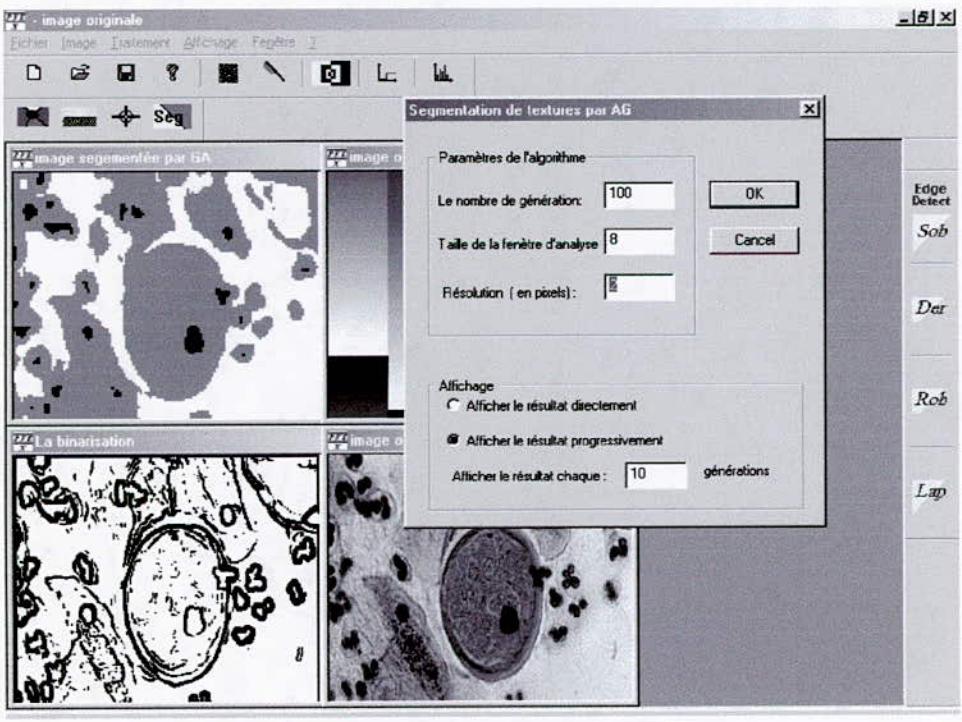


Figure V.1 : Aspect du logiciel PFE

V.2- Aspects de programmation

La programmation des différents algorithmes et opérateurs s'est faite dans un environnement orienté objet. Toutes les fonctions ont étés définie sous forme de plusieurs classes–objet. La technique d'héritage permet à une classe d'avoir accès à toutes les fonctions et données d'une autre classe mère. Dans le cas du filtrage, par exemple, nous avons programmé l'algorithme de filtrage indépendamment du filtre. Cela permet d'implanter autant de filtres qu'on veut, juste en programmant sa réponse implulsionnelle,

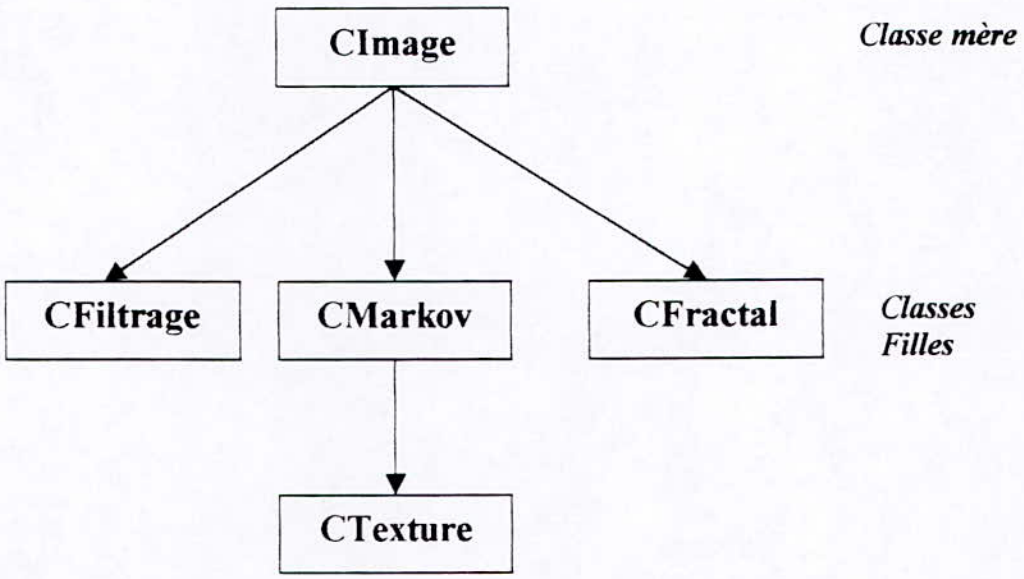


Figure V.2 : Hiérarchie des différentes classes programmées.

La classe de base est CImage elle contient les informations sur l'image comme la largeur, la longueur, la moyenne et la variance ainsi que des opérateurs généraux très utilisés comme l'histogramme. De cette classe, sont dérivées trois autres classes :

CFiltrage encapsule la fonction de filtrage linéaire et permet de mettre au point autant les filtres de lissage que les filtres détecteurs de contours comme le filtre de *Dériche*.

CMarkov contient les différentes fonctions nécessaires à l'algorithme de relaxation sélectionniste (chapitre III) avec sa partie champs de *Markov (calcul de probabilité de vraisemblance* ...) et sa partie algorithme génétique (*selection, croisement, mutation*...).

CFractal contient la fonction de calcul de dimension fractale mais l'algorithme de segmentation n'a pas pu être programmé.

V.3- Les résultats

Pour tester l'efficacité des méthodes de segmentation, on se base sur les critères suivants :

- L'aptitude à segmenter des images très entrelacées.
- La capacité de segmenter un grand nombre de textures.
- La rapidité.
- Le taux d'erreurs entre les textures.

V.3.1- Méthode de la relaxation sélectionniste

Les résultats qu'on va présenter dans ce paragraphe montre l'efficacité de la méthode décrite dans le chapitre III, pour la segmentation d'images hautement texturées.

Dans la figure V.3 on a généré artificiellement deux textures avec le modèle de *Markov*. La limite entre les deux textures a la forme d'une sinusoïde de façon à avoir un entrelacement entre les deux régions. Dans la figure V.4, ainsi que dans la figure V.5 on augmente l'entrelacement entre les deux textures – en augmentant la fréquence de la sinusoïde- pour tester les limites de cette méthode. On constate une bonne segmentation dans les trois cas où les régions sont parfaitement séparées.

Dans la figure V.5, on a augmenté le nombre de textures présentes dans l'image. On observe une robustesse vis à vis du nombre de textures : les quatre régions sont bien séparées.

En ce qui concerne le temps de calcul, une évaluation objective nous oblige à prendre en considération plusieurs paramètres, car la rapidité d'exécution dépend du nombre de générations nécessaire à la convergence de l'algorithme génétique. Hors ce dernier n'est pas le même pour toutes les images. Néanmoins on a constater qu'en moyenne une bonne convergence est obtenue après 100 générations. En plus de ça, la taille de la fenêtre d'analyse w (cf. III.2). Le temps de calcul est proportionnel à w^2. De bons résultats sont obtenus avec des fenêtres de 8×8. Sans oublier la résolution d'affichage w_a qui n'intervient pas dans la partie théorique, mais qui est plutôt un paramètre pratique : dans l'implantation on a attribué un label L_s non pas à un pixel mais à une fenêtre de pixels de taille $w_a \times w_a$, sans altérer considérablement l'aspect visuel. Et comme pour w, la relation avec le temps de calcul est quadratique. Cette astuce permet donc d'améliorer le temps de calcul.

On a effectuer la comparaison avec un autre algorithme de segmentation de textures basé sur la modélisation par les MRF, appelé algorithme EM (*Expectation Maximization*) Ce dernier fût implanté et testé dans un travail de magister (par M. *Boudaieb*) [23][22].

Image	Temps de calcul en s
à 2 régions	34
à 3 régions	118

Tableau 1 : Temps de calcul pour la méthode EM

Nombre de régions	Résolution d'affichage w_a	Nombre de générations	Temps de calcul en s
2	2*2	100	41
2	4*4	200	25
4	2*2	100	47
4	4*4	100	12

Tableau 2 : Temps de calcul pour la méthode de la relaxation sélectionniste

A la lumière des résultats des deux tableaux –et bien que la comparaison soit difficile vue la nature différente des deux algorithmes- on peut conclure :

- L'algorithme EM est plus rapide pour les images avec un nombre petit de régions
- Pour les images avec plusieurs régions la relaxation est plus efficace car le temps de calcul ne dépend pas du nombre de région.
- L'algorithme EM nécessite le nombre de régions comme donnée fournie par l'utilisateur alors que l'algorithme RS ne nécessite aucune information a priori sur l'image. Cela est en partie grâce à l'algorithme génétique. Cette remarque est particulièrement importante pour les systèmes de traitement automatique d'images, où l'opérateur humain intervient le moins possible.

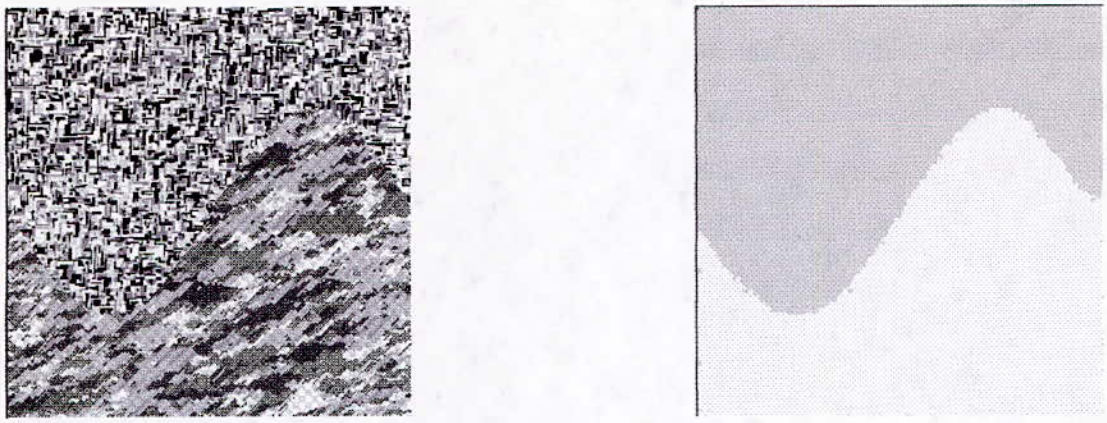


Figure V.3 : à gauche deux textures avec :
B1=(1.2,-0.3,1.3 ,-0.9) en haut.
B2=(-1.2,1.3,-1.3,0.9) en bas.
A droite, l'image segmentée par SR.

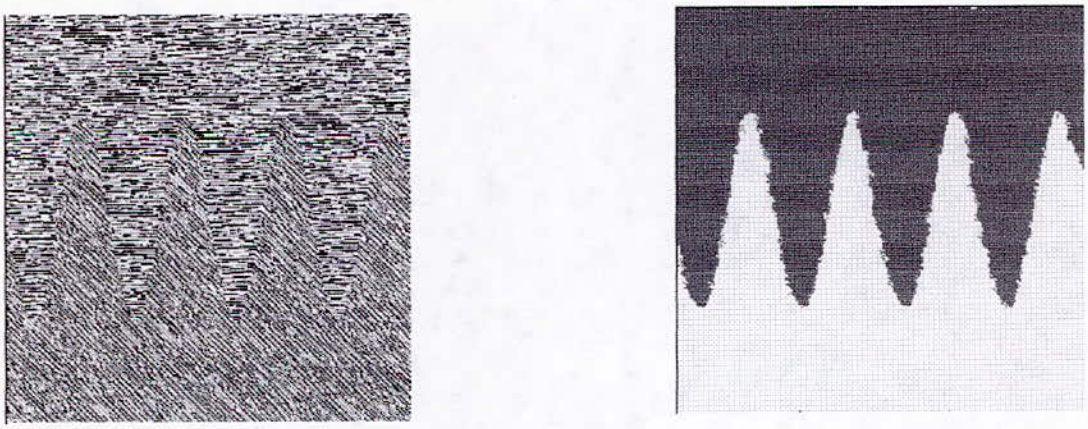


Figure V.4 : à droite deux textures avec :
B1=(-0.9,-1.1,0.9 ,-1) en haut
B2=(-0.9,-1.4,-0.85,+1.1) en bas
A gauche l'image segmentée SR.

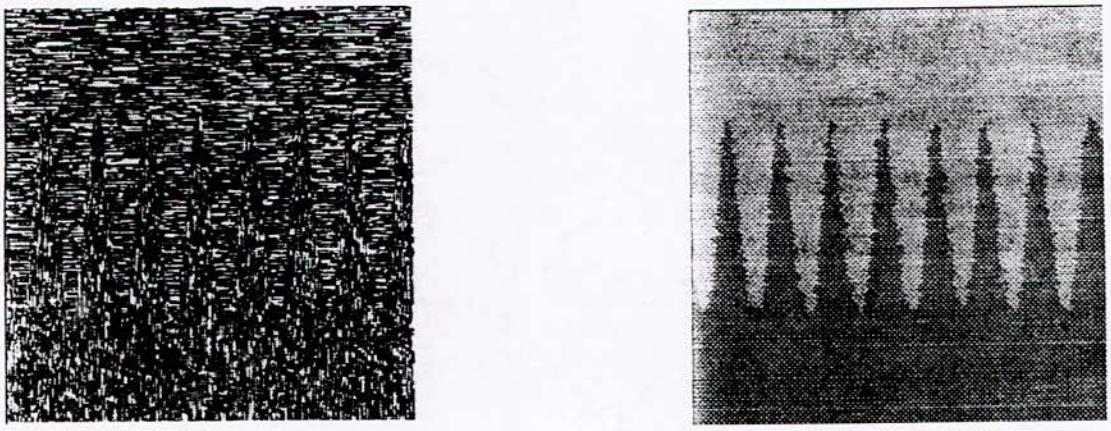


Figure V.5 : A gauche deux textures avec :
B1=(-1,-1.2,1 ,-1.2) en haut
B2=(1,-1.2,-1,-1.2) en bas
A droite l'image segmentée

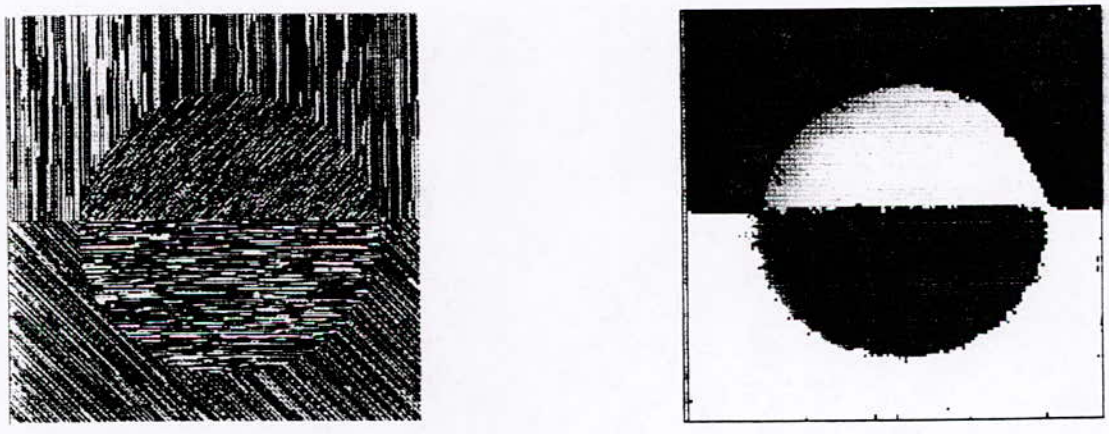


Figure V.6 : A gauche image à 4 textures
A droite l'image segmentée

V.3.2- L'algorithme supervisé de segmentation

En voyant la figure V.7 on constate que la séparation entre les régions est plus nette qu'avec l'algorithme SR (figure V.6). Et cela même pour un grand nombre de textures (figureV.8). Après ça, la détection du contour se fait par l'opérateur de *Dériche* (cf. § I.2.3).

L'avantage de cet algorithme est qu'il n'est pas destiné aux seules images texturées, mais il peut tout aussi bien segmenter avec une bonne précision des images réelles, comme le montre la figure V.9. qui montre la segmentation d'une tumeur cancéreuse (en blanc) au niveau du sein.

L'intérêt réside dans le dépistage automatique du cancer et ça peut aussi fournir une aide au diagnostique et une évaluation objective de l'ampleur de la tumeur (par calcul automatique de sa surface).

La figure V.10 montre une image radar, de la région de *LAGHOUAT*, à l'état brut (bruitée et peu contrastée) à laquelle, on a appliqué l'algorithme de segmentation après égalisation d'histogramme. Les contrastes géographiques apparaissent clairement après segmentation.

On a comparé cet algorithme avec un autre algorithme supervisé du même type, très utilisé, qui utilise l'optimisation par le *recuit simulé* [22]. En ce qui concerne le temps de calcul, et contrairement aux autres méthodes cette dernière présente un temps de calcul presque constant qu'elle que soit le nombre de régions et il est de l'ordre de 26 secondes en général. Il est beaucoup plus rapide que l'algorithme par *recuits simulés* qui prend un temps variable selon le type d'image et qui varie de 2 minutes à plusieurs dizaine de minutes.

Le seul désavantage de cette méthode est qu'elle travaille en mode supervisé c'est à dire qu'il y a intervention de l'opérateur humain lors de la phase de prises d'échantillons (cf. § III.4.1.a), en effet dans notre logiciel la prise d'échantillons se fait par une clique de souris sur les régions que nous désirons segmenter.

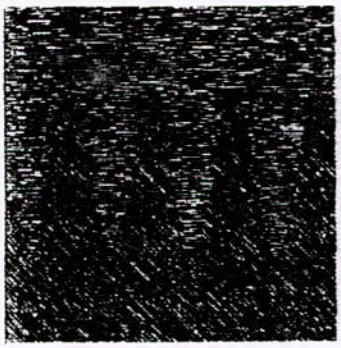
Test de reconnaissance

La segmentation n'étant pas un but en soit, nous avons appliqué un algorithme simple de reconnaissance [27] sur les contours obtenus après segmentation d'images texturées. L'algorithme utilisé effectue un suivi de contour (aminci et fermé) mesure les changements de direction dans le contour. Au tour de chaque pixel il y a 8 voisins suivant les directions horizontales, verticales et diagonales. Chaque direction est codée suivant un codage de *Freeman* (sur 3 bits) [1].

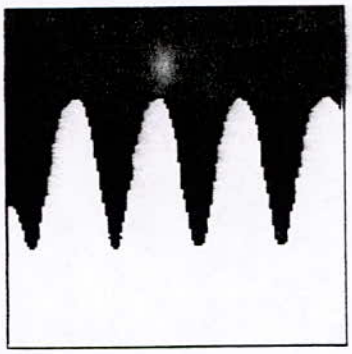
Les formes géométriques les plus simples peuvent être différenciées par la chaîne de codes qui caractérise leurs contours. Ainsi, pour un rectangle on aura plus de directions verticales et horizontales, alors qu'un losange présente plus de directions diagonales lors du suivi de son contour.

La figure V.11 présente une image texturée de forme rectangulaire. Après segmentation, et détection de contours, on applique un suivi de contour. Le programme peut, alors reconnaître cette forme (parmi d'autres formes simples), comme étant un rectangle. De même pour la figure V.12 qui est reconnue comme étant un cercle.

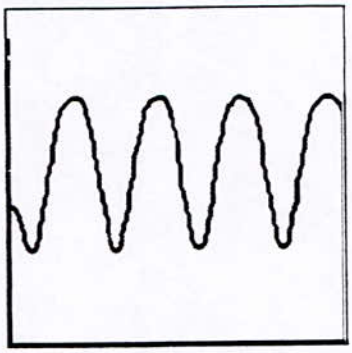
Cependant cette reconnaissance ne marche que sur les formes basiques (rectangle, triangle, cercle, losange) et des contours fermés, binarisé et surtout aminci, c'est à dire dans l'épaisseur est de 1 pixel. Pour pouvoir reconnaître des formes plus complexe, il faudrait construire une base de donnée plus large, chose qui sort de notre sujet d'étude, qui est la segmentation.



- a -

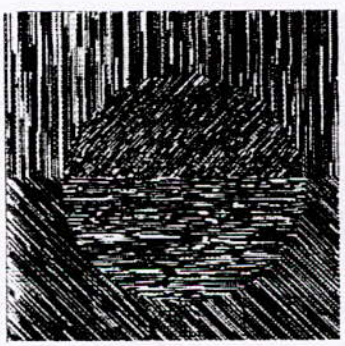


- b -

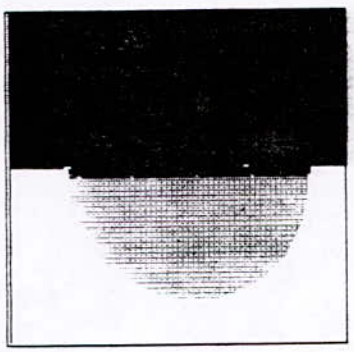


- c -

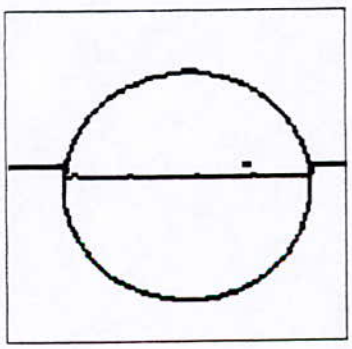
Figure V.7:
-a: Image texturée originale.
-b: Image segmentée.
-c: Image contour (après filtrage de Dériche).



- a -

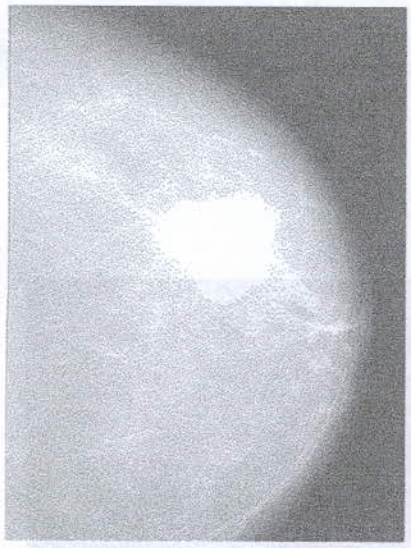


- b -

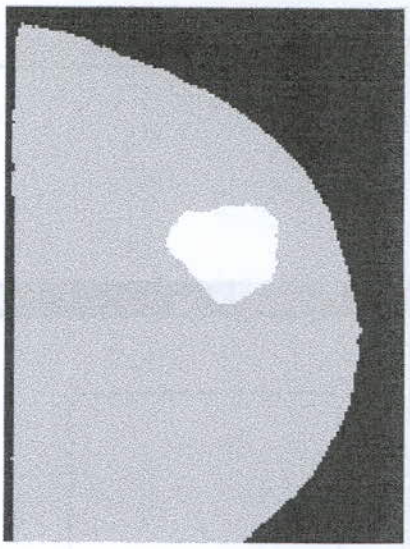


- c -

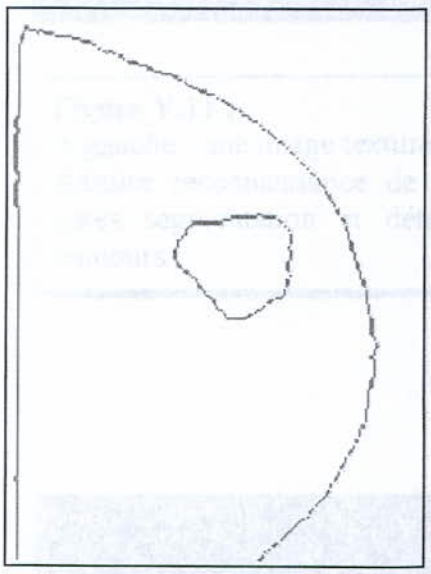
Figure V.8 :
-a: Image texturée originale.
-b: Image segmentée.
-c: Image contour (après filtrage de Dériche) .



- a -



- b -



- c -

Figure V.9 : Segmentation d'une image radiologique La partie blanche représente une tumeur cancéreuse au niveau du sein.
a- Image originale
b- Image segmentée par l'algorithme supervisé
c- Détection des contours par *Dériche.*

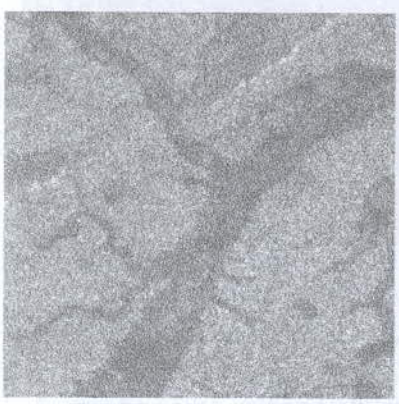


Figure V.10 :
A gauche : image radar.
A droite : image segmentée après rehaussement.

Conclusion générale

Au début, le but de ce travail était de faire de la segmentation d'images par les algorithmes génétiques. Nous nous sommes très vite rendus compte qu'il n'existait pas de méthodes de segmentation par algorithmes génétiques, mais que les AG étaient utilisés sur des modèles différents sous des formes très variées. Une présentation généralisée était impossible. On a, donc, fait le choix de deux méthodes différentes : une approche régions et une approche contours, d'où l'indépendance des chapitres III et IV.

Nous avons présenté deux méthodes dans lesquelles les AG contribuent à rendre la segmentation d'images de textures plus efficace, plus rapide et surtout avec le minimum d'interventions de l'opérateur humain. C'est à dire plus automatique. Nous avons testé, en particulier, la première méthode qui est la relaxation sélectionniste qui utilise la modélisation par les champs de *Markov* (MRF). Cette modélisation est l'une des plus utilisées en traitement d'images et surtout en segmentation de textures. Son principal problème est l'estimation des paramètres du modèle. La plupart des méthodes classiques utilisées mettent un temps relativement long pour la convergence et surtout requièrent des informations a priori sur l'image comme le nombre de régions. Dans cette méthode, l'estimation des paramètres ainsi que la segmentation se font par un processus d'évolution géré par un algorithme génétique. Les bons résultats obtenus sur des images texturées confirment l'efficacité de cette méthode. En plus, elle ne possède pas les désavantages des autres méthodes cités ci-dessus.

En marge du chapitre III, nous avons proposé un algorithme empirique de segmentation supervisée, basé sur une combinaison des paramètres Markoviens et statistiques, qui donne de bons résultats sur des images texturées et non texturées ainsi qu'un temps de calcul satisfaisant par rapport à d'autres algorithmes supervisés (comme les recuits simulés).

En ce qui concerne les perspectives des AG en traitement d'images, à part la segmentation ou la reconnaissance de formes, il serait intéressant de noter qu'ils sont déjà utilisés pour le design des filtres 2D [18], [19], des bancs de filtres pour le codage ainsi que le design de codebooks pour la quantification vectorielle [20] utilisée en compression d'images notamment les images animées.

Mais les applications les plus prometteuses sont sûrement celles qui concernent les sciences cognitives. Celles-ci ont, justement, pour finalité d'imiter les capacités de reconnaissance, d'intelligence et d'adaptation, humaines. L'utilisation de métaphore biologique comme les AG, les réseaux de neurones ou bien une combinaison des deux approches, ne peut que se développer.

Annexes

Annexe 1

Approximation de la fonction de partition $Z_W(B)$

Cette annexe est consacrée à la démonstration [3] de la formule utilisée dans (§ III.1.2). Pour une fenêtre W et un ensemble de paramètres de textures $B=(\beta_1,...,\beta_p)$ on va montrer comment obtenir la formule approximative de la fonction de partition $Z_W(B)$. Pour la simplicité des calculs on va mettre $W=S$ (c'est à dire que la fenêtre d'analyse contient l'ensemble des sites). La fonction de partition à approximer est la suivante :

$$Z(B)=\sum_{x\in\Omega}\exp\{-E(x;B)\} \tag{1}$$

où :

$\Omega=G^n$ est l'ensemble de toutes les configurations possibles dans S.
n étant le nombre de sites en S et $G=\{0,...,g-1\}$ est l'ensemble des niveaux de gris.
$C=\bigcup_{i=1}^{p}C_i$ est l'ensemble des cliques dans S, C_i est l'ensemble des cliques de type i.

Le nombre de cliques à l'intérieur de S est égal à n pour chaque type de clique.
En faisant l'approximation de chaque terme de $Z(B)$ par un développement au second ordre on aura :

$$\begin{aligned}\tilde{Z}(B)&=\sum_{x\in\Omega}1-\sum_{x\in\Omega}E(x;B)+\tfrac{1}{2}\sum_{x\in\Omega}\exp E(x;B)^2\\ \tilde{Z}(B)&=Z_0(B)-Z_1(B)+\tfrac{1}{2}Z_2(B)\end{aligned} \tag{2}$$

Le problème maintenant est de réarranger les termes Z_1 et Z_2 en fonction des β_i et S (Z_0 étant constant et égal à g^n). Pour y parvenir on utilise les propriétés suivantes :

Pour toute clique $c\in C$ et pour chaques deux cliques $c_1\neq c_2$ on peut montrer que :

$$\sum_{x\in\Omega}\Delta_c(x)=(g-2)g^{n-1} \tag{3}$$

$$\sum_{x\in\Omega}\left[\Delta_c(x)\right]^2=g^n \tag{4}$$

$$\sum_{x\in\Omega}\Delta_{c_1}(x)\Delta_{c_2}(x)=(g-2)^2g^{n-1} \tag{5}$$

En utilisant les définitions de $Z_1(B)$, $E(x;B)$ et $k_i(x)$ on obtient :

$$\begin{aligned}Z_1(B)&=\sum_{i=1}^{p}\beta_i\sum_{c\in C_i}\sum_{x\in\Omega}\Delta_c(x)\\ &=n(g-2)g^{n-1}\sum_{i=1}^{p}\beta_i\end{aligned} \tag{6}$$

Pour calculer $Z_2(B)$ on fait comme pour $Z_1(B)$ et on obtient :

$$Z_2(B) = \sum_{i=1}^{p}\sum_{j=1}^{p}\beta_i\beta_j \underbrace{\sum_{c_1\in C_i}\sum_{c_2\in C_j}\sum_{x\in\Omega}\Delta_{c_1}(x)\Delta_{c_2}(x)}_{\omega_{ij}}$$

Les coefficients ω_{ij} sont calculés en distinguant pour chaque c_1 les cliques c_2 qui sont égales à c_1 de ceux qui y diffèrent, et en utilisant (4) et (5).

Dans l'expression des ω_{ij}, il y aura seulement une clique pour laquelle $c_2 = c_1$ et n-1 cliques pour lesquelles $c_2 \neq c_1$. Dans l'expression de ω_{ij}, $j \neq i$ toutes les cliques c_2 sont différentes de c_1 dès qu'elles appartiennent à différents types de cliques.

$$\begin{aligned}
\omega_{ij} &= \sum_{c_1\in C}\left\{\sum_{x\in\Omega}\Delta^2{}_{c_1}(x) + \sum_{\substack{c_2\in C_i \\ c_2\neq c_1}}\sum_{x\in\Omega}\Delta_{c_1}(x)\Delta_{c_2}(x)\right\} \\
&= ng^{n-2}(g^2 + (n-1)(g-2)^2) \\
\omega_{ij} &= \sum_{c_1\in C_i}\sum_{\substack{c_2\in C_i \\ c_2\neq c_1}}\sum_{x\in\Omega}\Delta_{c_1}(x)\Delta_{c_2}(x) \quad (\textit{pour } j\neq i) \\
&= n^2(g-2)^2 g^{n-2}
\end{aligned}$$

Avec ces coefficients la formule finale de $Z_2(B)$, est la suivante :

$$Z_2(B) = 4n(g-1)g^{n-2}\sum_{i=1}^{p}\beta_i^2 + n^2(g-2)^2 g^{n-2}\left(\sum_{i=1}^{p}\beta_i\right)^2 \qquad (7)$$

Annexe 2

Démonstration et implantation du filtre de *Dériche*

1- Approche par filtrage optimal

Dans cette approche, le contour est modélisé par un échelon d'amplitude U_0 noyé dans un bruit blanc. *Canny* a élaboré une approche qui consiste à trouver le filtre optimal de réponse impulsionnelle h(x) satisfaisant les trois contraintes suivantes pour un échelon d'entrée :

- une bonne détection
- une bonne localisation
- une faible multiplicité des maxima dus au bruit

Soit A(x) un signal monodimensionnel représentant un saut d'amplitude U_0, noyé dans un bruit blanc stationnaire N(x) de moyenne nulle et de densité spectrale de puissance (d.s.p) N_0^2 :

$$A(x) = U_0 U(x) + N(x) \tag{1}$$

où

$$U(x) = \begin{cases} 1 & , x \geq 0 \\ 0 & \textit{ailleurs} \end{cases}$$

le signal de sortie est donné par l'expression suivante :

$$C(x) = A * h(x) = \int_{-\infty}^{+\infty} A(t)h(t-x)dt \tag{2}$$

Le problème posé est de trouver h(x) tel que C(x) soit maximum au point x=0 en respectant les trois contraintes précédentes.

Le critère de bonne détection s'exprime à l'aide du rapport signal sur bruit RSB défini comme le rapport du maximum de la réponse due au signal seul sur la racine carrée de la puissance du bruit en sortie :

$$RSB = \frac{U_0 \int_{0}^{-\infty} h(x-t)dt}{\left(E\left\{ \left| \int_{-\infty}^{+\infty} N(t)h(x-t)dt \right|^2 \right\} \right)^{\frac{1}{2}}} = \frac{U_0 \int_{0}^{-\infty} h(x-t)dt}{N_0 \left(\int_{-\infty}^{-\infty} h^2(t)dt \right)^{\frac{1}{2}}} \tag{3}$$

Le filtre recherché est un dérivateur, afin d'obtenir une réponse nulle pour un signal d'entrée constant. Ensuite, toujours en se plaçant en x=0, on obtient :

$$RSB = \frac{U_0 \int_{-\infty}^{0} h(t)dt}{N_0 \left(\int_{-\infty}^{+\infty} h^2(t)dt \right)^{\frac{1}{2}}} = \frac{U_0}{N_0} \Sigma \tag{4}$$

où $\Sigma = \dfrac{\int_{-\infty}^{0} h(t)dt}{\left(\int_{-\infty}^{+\infty} h^2(t)dt\right)^{\frac{1}{2}}}$.

La bonne localisation est mesurée par l'inverse de la variance de la distance entre le maximum de la réponse et la position réelle de la transition (il faudra donc le rendre maximum). Pour le calculer, on cherche à exprimer $E\{x_0^2\}$, x_0 étant la position calculée de la transition. Cette position x_0 correspond au maximum du signal de sortie, donc un passage par zéro de sa dérivée première puisque le filtre étudié est un dérivateur. On a donc :

$$\left[\frac{\partial C}{\partial x}\right]_{X_0} = \left[\frac{\partial}{\partial x} A * h(x)\right]_{X_0} = \left[A * h'(x)\right]_{X_0} = 0 \tag{5}$$

d'où :

$$\begin{aligned}\left[\frac{\partial C}{\partial x}\right]_{X_0} &= U_0\left[U * h'(x)\right]_{X_0} + \left[N * h'(x)\right]_{X_0} \\ &= U_0 \int_{-\infty}^{-\infty} U(x_0 - t)h'(t)dt + \left[N * h'(x)\right]_{X_0} \\ &= U_0 \int_{-\infty}^{X_0} h'(t)dt + \left[N * h'(x)\right]_{X_0} \\ &= U_0 h(x_0) + \left[N * h'(x)\right]_{X_0}\end{aligned} \tag{6}$$

Ensuite, on exprime le développement limité de la fonction h(x) impaire et continue au voisinage de x=0 :

$$h(x_0) \approx h(0) + x_0 h'(0) = x h'(0) \tag{7}$$

car h(0)=0. Il s'ensuit que :

$$\left[\frac{\partial C}{\partial x}\right]_{X_0=0} \approx U_0 x_0 h'(0) + N * h'(0) = 0$$

donc :

$$E\left\{\left|U_0 x_0 h'(0)\right|^2\right\} = E\left\{\left|N * h'(0)\right|^2\right\}$$

on aura alors :

$$U_0^2 h'^2(0) E\{x_0^2\} = N_0^2\left(\int_{-\infty}^{+\infty} h'^2(t)dt\right) \tag{8}$$

ce qui donne :

$$E\{x_0^2\} \approx \frac{N_0^2\left(\int_{-\infty}^{+\infty} h'^2(t)dt\right)}{U_0^2 h'^2(0)} \tag{9}$$

Le critère de localisation est la racine carrée de l'inverse de l'expression ci-dessus :

$$\frac{U_0}{N_0}\Lambda = \left(\frac{U_0^2 h'^2(0)}{N_0^2 \left(\int_{-\infty}^{+\infty} h'^2(t)dt \right)} \right)^{\frac{1}{2}} = \frac{U_0}{N_0} \frac{|h'(0)|}{\left(\int_{-\infty}^{+\infty} h'^2(t)dt \right)^{\frac{1}{2}}} \tag{10}$$

Le produit $\Sigma\Lambda$ est un critère qui combine une bonne détection et une bonne localisation. Il ne dépend pas de l'amplitude U_0 de l'échelon, ni du facteur d'échelle κ. Pour démontrer cette dernière propriété, on pose : $h_\kappa(x) = h\left(\frac{x}{\kappa}\right)$ et on obtient :

$$\Sigma_\kappa = \sqrt{\kappa}\Sigma \qquad \text{et} \qquad \Lambda_\kappa = \frac{1}{\sqrt{\kappa}}\Lambda$$

d'où : $$\Sigma_\kappa \Lambda_\kappa = \Sigma\Lambda$$

Pour l'optimisation on procède à la maximisation du produit $\Sigma\Lambda$ en utilisant une troisième contrainte qui est la minimisation de la densité d_0 de passages par zéro de la réponse due au bruit. Pour un processus gaussien B(x) de moyenne nulle, on a :

$$d_0 = \frac{1}{\pi} \sqrt{\left(\frac{-R_{BB}''(0)}{R_{BB}(0)} \right)} \tag{11}$$

où $R_{BB}(0)$ est la fonction d'autocorrélation du signal. Posons $B(x) = N\,h(x)$, où N(x) est un bruit blanc gaussien, on a :

$$R_{BB}(0) = N_0^2 \int_{-\infty}^{+\infty} h^2(t)dt$$

Compte tenu de la relation $R_{B'B'}(\tau) = -R_{BB}''(\tau)$ on peut écrire :

$$R_{BB}''(0) = -N_0^2 \int_{-\infty}^{+\infty} h'^2(t)dt \tag{12}$$

On s'intéresse aux transitions du signal d'entrée qui correspondent aux extremums du signal de sortie du filtre dérivateur étudié. Ces extremums sont aussi les passages par zéro de la dérivée seconde du signal d'entrée, donc de la dérivée du signal de sortie. On exploite alors la formule (11) en remplaçant B(x) par B'(x) :

$$d_0 = \frac{1}{\pi} \left(\frac{\int_{-\infty}^{+\infty} h''^2(t)dt}{\int_{-\infty}^{+\infty} h'^2(t)dt} \right)^{\frac{1}{2}} \tag{13}$$

et on pose : $x_{moy} = \frac{1}{d_0}$

En arrivant à ce point de calcul, *Déricbe*, à l'inverse de *Canny*, a choisi de prendre un filtre à réponse impulsionnelle infinie (RII). Ensuite, il faut trouver la fonction h(x) qui minimise la fonctionnelle :

$$\Psi(x,h,h',h'') = h^2 + \lambda_1 h'^2 + \lambda_2 h''^2 + \lambda_3 h \tag{14}$$

La solution optimale h(x) satisfaisant les conditions de continuité et de dérivabilité est obtenue par calcul variationnel comme solution de l'équation d'*Euler*, correspondant à une fonctionnelle du deuxième ordre :

$$\Psi_h - \frac{\partial}{\partial x}\Psi_{h'} + \frac{\partial^2}{\partial x^2}\Psi_{h''} = 0 \tag{15}$$

où $\Psi_h = \frac{\partial \Psi}{\partial h}$.

La fonction h(x) est alors la solution de l'équation différentielle suivante :

$$2h(x) - 2\lambda_1 h''(x) + 2\lambda_2 h''''(x) + \lambda_3 = 0 \tag{16}$$

qui admet comme solution générale :

$$h(x) = a_1 e^{\alpha x}\sin\omega x + a_2 e^{\alpha x}\cos\omega x + a_3 e^{-\alpha x}\sin\omega x + a_4 e^{-\alpha x}\cos\omega x \tag{17}$$

avec :

$$\lambda_2 - \frac{\lambda_1^2}{4} > 0 \;;\quad \alpha^2 - \omega^2 = \frac{\lambda_1}{2\lambda_2} \;;\quad 4\alpha^2\omega^2 = \frac{\lambda_1^2 - 4\lambda_2}{4\lambda_2^2}.$$

Les conditions aux limites imposées sont :

$$h(0) = 0;\quad h(+\infty) = 0;\quad h'(0) = 0;\quad h'(+\infty) = 0$$

La solution devient donc :

$$h(x) = ce^{-\alpha|x|}\sin\omega x \tag{18}$$

qui est appelée l'opérateur optimal de *Déricbe*.

2- Implantation récursive des opérateurs de Deriche

Ces filtres possèdent une réponse impulsionnelle infinie dont l'expression analytique permet une implantation récursive exacte. Cette implantation est du type parallèle.

2.1- Opérateur monodimensionnel

Dérivation

Une mise en forme du filtre optimal de dérivation $h(x) = cxe^{-\alpha|x|}$ peut être obtenue en employant la transformée en Z.
Soit $h[i]$ la séquence discrète résultant de l'échantillonnage de $h(x)$ et $H[z]$ sa transformée en Z :

$$H(z) = \sum_{i=-\infty}^{-\infty} h[i]z^{-i} \tag{19}$$

La séquence $z[i]$ est mise sous la forme d'une séquence causale $h_+[i]$ et d'une séquence non causale $h_-[i]$ telle que $h[i]=h_-[i]+h_+[i]$ avec :

$$h_-[i]=\begin{cases} c\,ie^{-\alpha i} & ,i\geq 0 \\ 0 & ,i<0 \end{cases}$$

et

$$h_+[i]=\begin{cases} 0 & ,i\geq 0 \\ c\,ie^{\alpha i} & ,i<0 \end{cases}$$

La transformée $H(z)$ a pour expression :

$$H(z)=H_-(z^{-1})+H_+(z) \qquad (20)$$

avec :

$$H_-(z)=\frac{ce^{-\alpha}z^{-1}}{1-2e^{-\alpha}z^{-1}+e^{-2\alpha}z^{-2}}$$

et

$$H_+(z)=-\frac{ce^{-\alpha}z}{1-2e^{-\alpha}z+e^{-2\alpha}z}$$

$H_-(z^{-1})$ (respectivement $H_+(z)$) converge pour $\left|e^{-\alpha}z^{-1}\right|<1$ (respectivement $\left|e^{-\alpha}z\right|<1$. Comme toutes les singularités de $H_-(z)$ (resp. $H_-(z^{-1})$) sont à l'intérieur (resp. extérieur) du cercle unité pour α réel positif, ces deux transformées en Z correspondent à deux fonctions de transfert de filtres récursifs stables et du second ordre. Le premier $H_-(z^{-1})$ opérant de gauche à droite et le second $H_+(z)$ de droite à gauche.

Le coefficient c est pour obtenir un maximum d'amplitude de 1 en réponse à un échelon unité, ce qui donne :

$$B[i]=\sum_{m=-\infty}^{\infty}U[i-m]h[m]=1 \qquad \text{pour i=1}$$

$$B[0]=-c\sum_{m=0}^{\infty}me^{-\alpha m}=1\Rightarrow c=-\frac{\left(1-e^{-\alpha}\right)^2}{e^{-\alpha}}$$

On peut remarquer que cette valeur de la constante de normalisation est différente de celle calculée pour un filtre continu. On obtient alors les équations aux différences suivantes :

$$B_1[i]=ce^{-\alpha}A[i-1]+2e^{-\alpha}B_1[i-1]-e^{-2\alpha}B_1[i-2] \qquad \text{pour i=1,...,M}$$

$$B_1[i]=-ce^{-\alpha}A[i+1]+2e^{-\alpha}B_2[i+1]-e^{-2\alpha}B_2[i+2] \qquad \text{pour i=M,...,1}$$

où M est la longueur de la séquence d'entrée A[i]. La séquence de sortie est alors donnée par :

$$B[i]=B_1[i]+B_2[i] \qquad (21)$$

L'intérêt de cette mise en œuvre récursive est le faible nombre d'opérations ; 5 multiplication et 5 additions pour l'opérateur dérivée première. Le nombre d'opérations est indépendant de la valeur de la résolution α utilisée pour détecter les contours. En effet, la forme du filtre liée à α peut varier mais le nombre d'opérations par point reste identique. A titre de comparaison, une implantation RIF de l'opérateur h[n] à l'aide de 2N+1 coefficients non nuls, requiert

2N+1 operations par point de sortie. Un gain de (2N+1)/5 en résulte de l'utilisation récursive qui présente en plus l'avantage de ne pas introduire de déformation par troncature de la réponse impulsionnelle.

Lissage

En suivant la même démarche que dans le paragraphe précédent, le filtre de lissage $f(x) = b(\alpha|x|+1)e^{-\alpha|x|}$ peut être implanté de manière récursive. En prenant la transformée en Z du filtre numérique correspondant et en le décomposant en une partie causale et une partie anti-causale on aura :

$$F(z) = F_+(z^{-1}) + F_-(z)$$

avec :

$$F_+(z^{-1}) = b\frac{e^{-\alpha}(\alpha-1)z^{-1}}{1-2e^{-\alpha}z^{-1}+e^{-2\alpha}z^{-2}}$$

et

$$F_-(z) = b\frac{e^{-\alpha}(\alpha+1)z - e^{-2\alpha}z^2}{1-2e^{-\alpha}z+e^{-2\alpha}z}$$

Les domaines de convergence sont les mêmes que précédemment. Alors les équations aux différences auront pour expression :

$$B_1[i] = bA[i] + be^{-\alpha}(\alpha-1)A[i-1] + 2be^{-\alpha}B_1[i-1] - e^{-2\alpha}B_1[i-2]$$

pour i=1,...,M.

$$B_2[i] = be^{-\alpha}A[i+1] - be^{-2\alpha}(\alpha-1)A[i+2] + 2be^{-\alpha}B_2[i+1] - e^{-2\alpha}B_2[i+2]$$

pour i=M,...,1.

et $$B[i] = B_1[i] + B_2[i]$$

Pour réaliser un lissage il faut donc 8 multiplications et 7 additions, indépendamment du facteur α. La constante b est calculée pour obtenir une réponse d'amplitude 1 à une entrée constante unitaire :

$$b = \frac{(1-e^{-\alpha})^2}{1+2\alpha e^{-\alpha}-e^{-2\alpha}} \qquad (22)$$

Cette valeur est différente de celle trouvée en continu.

2.2- Implantation de l'opérateur gradient bidimensionnel de Deriche

En transposant les relations :

$$h_x(x,y) = -b\alpha^2 x e^{-\alpha|x|}b(\alpha|y|+1)e^{-\alpha|y|} \qquad (23)$$

et

$$h_y(x,y) = -b\alpha^2 x e^{-\alpha|y|}b(\alpha|x|+1)e^{-\alpha|x|} \qquad (24)$$

l'image $B_i[i,j]$ de la dérivée directionnelle suivant x est calculée en appliquant la relation :

$$B_i[i,j] = [A * f[j]] * h[i] \qquad (25)$$

De même : $$B_j[i,j] = [A * f[i]] * h[j] \qquad (26)$$

Annexe 3

Eléments de la théorie fractale et multifractale

1- Définition

Soit μ une mesure de probabilité Borélienne sur [0,1], c'est à dire vérifiant les deux propriété suivantes :

- $\mu(\varnothing) = 0$
- $\mu\left(\bigcup_k E_k\right) = \sum_k \mu(E_k)$ si E_k sont des sous-ensembles disjoints de [0,1].

Le support de la mesure μ est l'ensemble des points sur lesquels la mesure est définie. Soit v_n une suite croissante d'entiers positifs. On définit une suite d'intervalles $I_{i,n}$ de [0,1] :

$$I_{i,n} = \left[\frac{i}{v_n}, \frac{i+1}{v_n}\right], \quad i=0,\ldots,v_n-1. \tag{1}$$

2- Dimension fractale

On définit usuellement la dimension fractale (ou dimension de *Hausdorff*) d'un ensemble F (notée D) de la façon suivante :

$$\forall \delta > 0, H_s^\delta(F) = \inf\left\{\sum_{i=1}^{+\infty} |U_i|^s\right\} \tag{2}$$

$\{U_i\}$ est une δ-couverture de F où $|U|$ est le diamètre de U et une δ-couverture est un recouverement dénombrable par des ensembles de diamètre inférieur ou égal à δ. Alors

$$D = \inf\left\{s; \lim_{\delta \to 0} H_s^\delta = 0\right\} = \sup\left\{s; \lim_{\delta \to 0} H_s^\delta = +\infty\right\} \tag{3}$$

Cette définition permet de mesurer finement la dimension d'un ensemble et de mettre en évidence pour certains types d'ensembles des dimensions fractionnaires (i.e. non entières).

3- Dimension fractale généralisée

Si on veut caractériser non seulement la géométrie d'un ensemble, mais la répartition d'une mesure (par exemple, les niveaux de gris) définie sur celui-ci, on considère les quanttés suivantes :

$$\tau_n(q) = -\frac{1}{\ln v_n} \ln\left(\sum_{\substack{0 \le i \le v_n \\ \mu(I_{i,n})=0}} \mu(I_{i,n})^q\right) \tag{4}$$

On dit que μ possède un comportement multifractal si $\lim_{n \to +\infty} \tau_n(q) = \tau(q)$ existe pour q appartenant à un intervalle non vide de $\Re$. $\tau(q)$ caractérise le comportement global de la mesure quand la taille des intervalles tend vers zéro, et est relié à une notion de dimension généralisée. En effet, si l'on définit :

$$\begin{cases} D_q = \dfrac{1}{q-1}\tau(q), q \neq 1 \\ D_1 = \lim\limits_{q \to +\infty}\left(\dfrac{1}{q-1}\tau(q)\right) \end{cases} \quad (5)$$

alors D_0 est la dimension fractale du support de μ, D_1 est la dimension d'information, D_2 est la dimension de corrélation.

4- Spectre multifractale

On peut effectuer une autre description multifractale :
Soit $I_n(x)$ l'intervalle $I_{i,n}$ contenant x. On définit :

$$E_\alpha = \left\{ x \in [0,1[, \lim_{n \to +\infty}\left(\frac{\ln \mu(I_n(x))}{\ln v_n}\right) = -\alpha \right\} \quad (6)$$

Les indices α caractérisent l'invariance d'echelle locale de la mesure ; si α existe au point x, alors on a $\mu(I_n(x)) \approx v_n^{-\alpha(x)}$ quand $n \to +\infty$, v_n étant la taille linéaire de l'intervalle autour de x sur lequel on évalue μ. α est appelé exposant de *Hölder* au point x. E_α peut alors être interprété comme le sous-ensemble des points ayant même comportement d'echelle, décrit par α. Pour obtenir une description multifractale de μ, on calcule d'abord l'ensemble des exposants α possibles, puis on évalue la « taille » du sous-ensemble E_α de $[0,1[$ associé à α, en calculant sa dimension de *Hausdorff*, notée $f(\alpha)$. Cette description en terme de $(\alpha, f(\alpha))$ est ainsi à la fois locale (via α) et globale (via $f(\alpha)$). Elle est désignée sous le nom de spectre multifractal de μ.

5- Lien entre spectre multifractal et dimension fractale généralisée

Un problème central de la théorie multifractale est de relier les deux descriptions $(\alpha, f(\alpha))$ et $(q, \tau(q))$. Ceci a d'importantes applications. En effet, $\tau(q)$ est généralement beaucoup plus facile à calculer sur des données expérimentales que $(\alpha, f(\alpha))$: $\tau(q)$ est obtenu par un moyennage sur un grand nombre d'intervalles suivi par un passage à la limite. α est plus sensible au bruit, puisqu'il est calculé indépendamment en chaque point. En ce qui concerne $f(\alpha)$, son calcul implique l'évaluation d'une dimension de *Hausdorff*, tâche généralement difficile. Sous des hypothèses très générales il est prouvé que :

$$f(\alpha) \leq \inf_q \{q\alpha - \tau(q)\} \quad (7)$$

Pour certaines classes spéciales de mesure, qui incluent les mesures multinômiales, on a une égalité $f(\alpha) = \inf_q \{q\alpha - \tau(q)\}$. C'est à dire que la dimension de *Hausdorff* de E_α est obtenue en effectuant une transformée de *Legendre* de $\tau(q)$. Dans le cas de mesures multinômiales, $D(\alpha)$ a une forme en cloche. Cette forme est aussi observée pour un grand nombre de phénomènes naturels. Cependant, elle n'est en aucun cas une propriété générale des spectres multifractals, puisqu'il est démontré que toute fonction réglée peut être le spectre d'une mesure multifractale.

6- Lacunarité

La lacunarité est un paramètre fractal du second ordre qui permet de mesurer la géométrie d'une région **R** relativement au reste de l'image **S** :

$$\Lambda = \int_S \left(\frac{\iint_{B(s,r)} i_R(r,\theta)\,dr\,d\theta}{m} - 1 \right)^2 ds \qquad (8)$$

où :

- B(s,r) est le disque de centre s et de rayon r,
- m est le nombre moyen de points de **R** dans un disque de rayon r,
- i_R est la fonction indicatrice sur **R**.

Bibliographie

[1] J. P. Coquerez et S. Philipp, *Analyse d'images : Filtrage et Segmentation.* Edition MASSON,Paris,1995.

[2] D. E. Goldberg, *Algorithmes Génétiques.* Edition Addison-Wesley, 1989.

[3] P. Andrey and P. Taroux, "Unsupervised Segmentation of Markov Random Field Modeled Textured Images Using Selectionist Relaxation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, pp 252-262. 1998.

[4] M. Yoshimura et S. OE, "Edge Detection of Texture Image using Genetic Algorithms," *SICE'97.* Juillet 29-31, Tokushima.

[5] H. Youlal, M. Janati-Idrissi, et M.Najim, *Modélisation Paramétrique en traitement d'images.* Edition MASSON. 1994.

[6] Petrowski, "Une Introduction à l'Optimisation par Algorithmes Génétiques," http://www.eark.polytechnique.fr/EC/Welcome.html

[7] M. Gen et R. Cheng, *Genetic Algorithms and Engineering Design.* Edition John Wiley & Sons, INC, 1997.

[8] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs.* 3^{e} édition,. Edition Springer, 1995.

[9] M. Kunt, *Traitement numérique des images.* Edition DIFFUSION, 1985.

[10] Toumazet, *Traitement de l'image sur micro ordinateur*. Edition SYBEX, 1987.

[11] R. C. Gonzales, *Digital image processsing*. Addison-Wesley Pub., 1992.

[12] K. F. Man, K. S. Tang and S. Kwong, "Genetic Algorithms : Concept and Applications," *IEEE Trans on Industrial Electronics*, vol. 43, pp. 519-533, 1996.

[13] Abdel Salam, W. El-Haweet and E. Matter, "A Survey on Genetic Algorithms Concept and Operators," *Computer & Automatic Control Dept. University of Alexandria.*

[14] H. Derin and H. Elliott, "Modeling and Segmentation of Noisy and Textured Image Using Gibbs Random Fields," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 38, pp. 39-55, 1987.

[15] B. S. Manjunath and R. Chellappa, "Unsupervised Texture Segmentation Using Markov Random Field Models," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 478-482, 1991.

[16] S. German and D. German, "Stochastic Relaxation, Gibbs Distribution and the Bayesian Restoration of Images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721-741, 1984.

[17] B. Bhanu, S. Lee and J. Ming, "Adaptive Image Segmentation Using a Genetic Algorithm," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 25 pp 1543-1567. 1995

[18] K. Dalibassis, P. E. Undrill and G. G. Cameron, "Designing texture filters with genetic algorithms: an application to medical images," Signal Processing, 57, 1,19-33. 1997.

[19] K. Dalibassis, P. E. Undrill and G. G. Cameron, "Using genetic algorithm to design 2D filters for texture interpretation and image restoration in the presence of noise," *IEE Proc. Colloquium on pattern Recognition*, Digest 1997/018, IEE Press, London, pp 4/1–4/6.

[20] D .R Bull and D. W. Redmill, "Optimization of image coding and architecture," *IEEE Trans. on Industrial Electronics*, vol. 43, pp 549-558. 1996.

[21] S. Chendra and M. Bhandarkar, "Image segmentation using evolutionary computation," *IEEE Trans. on Evolutionary Computation*, vol. 3. pp 1-21. 1999.

[22] A. Boudaieb, "Etude des Champs Aléatoires de Markov et leur Application à la Segmentation d'images S.A.R," Thèse de magister, Ecole Nationale Polytechnique.

[23] A. Boudaieb, A.Zerguerras, "Modélisation Markovienne et Segmentation d'images S.A.R.," A.J.O.T, série B, vol 14, no 1, 1999.

[24] C. Delannoy, *Programmer en Langage C++*. Edition CHIHAB-EYROLLES, 1995.

[25] D. Chapman, *Visual C++ 6*. Edition S & SM, 1998.

[26] C. Walnum et P. Robichaux, *Using MFC and ATL*. Special edition. Edition *QUE*, 1997.

[27] P. Fabre, *Exercices de reconnaissance des formes par ordinateur*. Edition MASSON, 1989.