ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

PROJET D'APPLICATION PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN TECHNOLOGIE DES SYSTÈMES
M.ING.

PAR

HUGUES BÉLANGER

RÉSEAU DE KOHONEN POUR LA DÉTECTION DES CONTOURS D'OBJETS DANS UNE IMAGE À NIVEAUX DE GRIS

MONTRÉAL, LE 28 FÉVRIER 1998.

CE PROJET D'APPLICATION A ÉTÉ ÉVALUÉ
PAR UN JURY COMPOSÉ DE:

- Mme Rita Noumeir, Ph.D., professeur-tuteur et professeur
  au département de génie électrique à l'École de technologie supérieure

- M. Richard Lepage, Ph.D., professeur-cotuteur et professeur
  au département de génie de la production automatisée à l'École de technologie supérieure (auparavant, professeur au département de génie électrique)

- M. Jacques De Guise, Ph.D., professeur
  au département de génie de la production automatisée à l'École de technologie supérieure

- M. Langis Gagnon, Ph.D., chercheur
  Centre de recherche informatique de Montréal (CRIM)

IL A FAIT L'OBJET D'UNE PRÉSENTATION DEVANT CE JURY ET UN PUBLIC
LE 21 DÉCEMBRE 1998
À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

Réseau de Kohonen pour la détection des contours d'objets dans une image à niveaux de gris

Hugues Bélanger

(Sommaire)

La détection du contour d'un objet spécifique se trouvant dans une image est une opération importante de plusieurs systèmes de vision. Des solutions proposées auparavant consistaient à utiliser un détecteur d'arêtes suivie d'une méthode de regroupement. On a reproché à ces méthodes d'être trop rigides et sensibles au bruit. D'autres approches plus récentes ne regroupent pas les arêtes au sens classique du terme, c'est-à-dire en ordonnant une série d'arêtes détectées dans l'image. Plutôt, ces approches vont représenter un ensemble d'arêtes de façon optimale en minimisant une fonction d'énergie. Les contours actifs ou "snake" et les cartes de Kohonen en sont des exemples.

Des travaux, ayant pour but de ramener ces deux algorithmes et quelques autres à un cadre de travail commun, ont été effectués avec succès. Ce projet tire avantage de ce nouveau cadre afin de modifier l'algorithme de Kohonen de deux façon. La première consiste à changer le type d'entrée de simple position des arêtes dans l'image à une position pondérée par le gradient de ces arêtes. La deuxième consiste à créer une structure d'accueil pour un retour d'information du niveau supérieur d'un système de vision.

L'algorithme ainsi modifié a été comparé à l'algorithme classique et au *snake*. Il semble que cette modification permet de capturer de l'information à propos du contour qui serait auparavant passée inaperçue avec des méthodes comme le *snake* ou l'algorithme de Kohonen dans sa forme classique. Un test de robustesse en présence de bruit a révélé que le nouvel algorithme dépendait fortement des opérations de pré-traitement pour lui fournir des gradients pertinents au contour de l'objet.

Le *snake* se comporte mieux en présence de bruit. Par contre, une hypothèse menant au développement des contours actifs demande que le contour initial se trouve près du contour que l'on cherche. Ceci n'est pas nécessaire avec l'algorithme de Kohonen à cause du traitement global de l'ensemble des arêtes.

Il est impératif de poursuivre plus avant les recherches à l'intérieur de ce cadre de travail. Le traitement global effectué par l'algorithme de Kohonen en fait une approche plus robuste lorsque le contour initial est éloigné du contour de l'objet et donc, lorsque

peu d'information est disponible quant à la position approximative du contour. Le *snake* à l'avantage d'être plus précis en présence de bruit. Une combinaison de ces approches ou une nouvelle approche regroupant ces avantages est souhaitable.

Kohonen neural network for the detection of objects contours in a graylevel image

Hugues Bélanger

(Abstract)

The contour detection of the objects found in an image is an important module of a vision system. Previous solutions consisted in using an edge detection filter followed by an edge linking operation. These methodes had the inconvenient of being too rigid and sensitive to noise. More recently, other methods have been used that do not link edges in the classical sense that is to say by forming an ordered list of edges from an unordered list. Instead, these approaches represent a set of edges optimally by minimising an energy function. Active contours or "snakes" and Kohonen maps are prime examples.

Previous work was successful at expressing these two algorithms within a common framework. This project took advantage of this common framework to modify the Kohonen algorithm in two ways. The first modification consisted in changing the type of input which was the position (or coodinates) of the object edges in the image to a position of edges weighted by the gradient magnitude. The second one consisted in creating a structure capable of receiving feedback from a vision system's higher processing levels.

The modified algorithm is then compared to the classical Kohonen maps and to the *snake*. The modified algorithm allowed capturing information about the contour that could not be detected using other methods like the *snake* or the classical maps. The robustness of the algorithm in the presence of noise revealed that the new algorithm strongly depended on the pre-processing operations to provide edges and gradients that are pertinent to the object's contour.

The *snake* behaves better in the presence of noise. However, one hypothesis leading to the development of the active contours assumes that the initial contour is located near the final contour. This hypothesis does not apply to the Kohonen maps because of the global processing of the whole set of edges.

It is imperative to pursue research within this framework. The global processing of the Kohonen maps gives it robustness when the intial contour is far from the contour of the object i.e. when little information is available about the approximate position of the contour. The *snake* is more precise in noisy images. A combination of these approaches would make a better contour detection algorithm.

Table des matières

Page

Table des matières (suite)

Liste des figures

Page

Liste des figures (suite)

Page

REMERCIEMENTS

Je voudrais d’abord exprimer ma profonde reconnaissance à mes directeurs de projet, Rita Noumeir, Ph.D. et Richard Lepage, Ph.D., pour la confiance qu'ils m'ont accordée en me proposant ce projet et pour leur soutien tout au long de son développement. Je veux aussi remercier Jacques de Guise, Ph.D. et Langis Gagnon, Ph.D. respectivement du département de la production automatisée de l'ÉTS et du Centre de recherche informatique de Montréal pour leurs remarques constructives qu'ils m'ont prodiguées lors de la revue de mon travail.

Ma reconnaissance va également à mes amis et collègues qui m’ont fourni remarques et suggestions pour ce travail et m'ont offert leur support lors de la présentation et de la rédaction de ce projet. Je remercie notamment mon employeur, Neoglyphics Media Corporation, de m'avoir alloué du temps pour travailler à la complétion de ce projet.

Enfin, je désire remercier chaleureusement les membres de ma famille pour le soutien et l’encouragement qu’ils m’ont apportés tout au long de ce travail.

INTRODUCTION

La détection du contour d'un objet bien déterminé dans une image joue un rôle décisif dans bien des cas en traitement d'image et en vision artificielle. Plusieurs tentatives ont été faites pour accomplir cette tâche et améliorer la qualité de la segmentation. Une solution usuelle consiste à utiliser une détection locale à l'aide de détecteurs d'arêtes comme Sobel (1970), Marr-Hildreth (1980) ou Canny (1986) suivi d'une méthode de regroupement. Le regroupement des arêtes "est le procédé par lequel une liste ordonnée d'arêtes est formée à partir d'une liste non-ordonnée" (Jain et al., 1995, p. 143). Ce type d'approche est cependant sévèrement affecté par le bruit se trouvant dans l'image. Plus récemment, d'autres classes d'algorithmes ont été proposées et qui sont basées sur une détection globale du contour de l'objet à l'aide d'un modèle paramétrique déformable comme celui du contour actif (aussi connu sous le nom de "snake") introduit par Kass et al. (1987).

Dans l'approche des contours actifs, le contour de l'objet est un modèle physique décrit par une suite d'unités, ou noeuds, soumises simultanément à des forces externes qui représentent les caractéristiques de l'image et à des forces internes d'élasticité et de rigidité qui assurent une certaine cohérence durant l'évolution du modèle initial. Une grande rigidité empêche le contour de faire des angles et des coins alors que l'élasticité permet au contour de s'allonger et de se rétrécir. L'évolution de ce modèle permettra au contour de trouver un état d'équilibre en minimisant toutes les énergies. Le développement théorique de cette approche est décrit dans le premier chapitre.

Une autre technique peut aussi être utilisée dans le même contexte de détection du contour d'un objet. Cette approche découle des cartes de caractéristiques auto-organisées ("Self-Organizing Feature Map" ou SOFM), ou modèle neuronique auto-organisateur, proposées par Kohonen (1982a, 1982b, 1991). La détection de contours utilise une carte unidimensionnelle. Le contour de l'objet est approximé comme dans l'approche précédente par une série de points représentés par les poids des neurones se déplaçant dans un espace à 2 dimensions, attirés par les forces de l'image représentées par les arêtes. La théorie reliée à cette approche est développée dans le second chapitre.

Une interprétation souple de la définition d'un algorithme de regroupement d'arêtes est nécessaire afin de classifier le *snake* et l'algorithme de Kohonen parmi ceux-ci. En effet, ces deux algorithmes n'ordonnent pas, à proprement parler, une liste non-ordonnée d'arêtes. Ils optimisent plutôt un chemin représentant l'ensemble des arêtes sans nécessairement que les points du contour trouvé ne correspondent exactement aux arêtes que l'on veut représenter. Dès lors, on est en droit de se demander si ces algorithmes ne partageraient pas une forme commune. Des recherches entreprises par Abrantes et al. (1995, 1996) ont permis de répondre à la question. À strictement parler, la réponse dans le cas général est non. Cependant, après certaines manipulations, il est possible de concevoir ces deux algorithmes comme étant des cas particuliers d'un seul et même cadre général exprimant une approche unifiée. Cette approche permettrait de construire de nouveaux algorithmes en ayant en tête les propriétés de ceux qui existent déjà. L'unification des algorithmes est présentée dans le chapitre 3.

Marr et Nishihara (1978) préconisaient l'approche des systèmes de vision en tant que procédés autonomes du bas vers le haut et que, jusqu'au niveau du croquis 2½D, aucune information de niveau supérieur ne devait être prise en ligne de compte : les calculs ne s'effectuant qu'à partir de l'information se trouvant dans l'image. Kass et al. (1988) soutenaient plutôt l'approche visant à procurer aux modules de traitement de haut niveau un ensemble de solutions alternatives au lieu de lui présenter une solution qui pourrait être prématurée. À défaut d'avoir à portée de la main un tel module de haut

niveau pouvant interagir avec les niveaux plus bas, ils ont développé une approche interactive afin d'explorer d'autres possibilités d'organisation du contour. D'une façon similaire, l'algorithme de Kohonen a été modifié par l'auteur en supposant que le module de haut niveau pourrait retourner de l'information et déterminer, avec un degré de certitude acceptable, les arêtes appartenant au contour réel de l'objet. Cette modification à l'algorithme de Kohonen est présentée dans le chapitre 4.

L'algorithme de Kohonen tel qu'implanté par Abrantes et Marques (1995, 1996) prend, comme données d'entrée, les vecteurs de position des arêtes préalablement déterminées à l'aide d'un opérateur de détection d'arêtes. Le cadre commun développé par ces auteurs a permis de considérer l'algorithme d'une façon plus globale et de modifier l'algorithme de façon à pondérer les vecteurs de position à l'entrée de l'algorithme avec le gradient de l'image calculés avec le filtre de Sobel. La ligne de pensée ayant mené à cette modification est similaire à celle présentée dans le paragraphe précédent. Cette approche voulait que la solution au problème de la détection du contour ne devait pas être précipitée et figée aux niveaux inférieurs mais plutôt impliquer les modules de traitement de plus haut niveau. La modification proposée à l'algorithme remplace une information rigide que sont les arêtes par une information plus riche que sont les gradients de l'image. Aucune décision n'a alors été prise à ce niveau quant à l'emplacement des arêtes et donc, des points du contour. Ce nouvel algorithme considère non seulement la position des points de l'image mais aussi leur *pertinence* telle que suggérée par la magnitude du gradient qui leur est associé. Cette autre modification est aussi présentée au chapitre 4.

Dans ce projet, les propriétés conférées à l'algorithme de Kohonen par ses paramètres ont d'abord été établies. Les paramètres manipulés lors de ces simulations sont le nombre d'unités de sorties du réseau, la fonction d'apprentissage et le voisinage topologique. Les effets de ces paramètres ont été évalués en fonction de la qualité globale du contour, et particulièrement des coins, ainsi que le temps et le nombre d'itérations nécessaire à la convergence. Les propriétés topologiques de l'algorithme de Kohonen sont un élément central de son succès. La capacité des cartes de Kohonen à capturer la

topologie globale de la forme d'un objet dans une image a été explorée en initialisant l'algorithme avec un contour possédant une topologie différente de celui de l'objet en question. Le contour initial possède, à la différence du contour de l'objet, un noeud ou chevauchement. Cette propriété a aussi été testée avec le *snake* pour fin de comparaison.

L'algorithme de Kohonen a ensuite été modifié de façon à prendre en ligne de compte la magnitude des gradients de l'image. Le nouvel algorithme a été appliqué à la détection d'une image synthétique dont une partie du contour a été volontairement altérée de façon à générer des gradients plus petits lorsque l'image est convoluée avec le filtre de Sobel. Ces gradients plus petits ne sont pas identifiés comme étant des arêtes lorsqu'ils sont seuillés et binarisés. Le contour trouvé a ensuite été comparé à ceux trouvés par l'algorithme de Kohonen classique et le *snake*.

Un critère de sélection plus souple des pixels potentiellement candidats à appartenir au contour de l'objet implique une plus grande chance de faire des erreurs c'est-à-dire, de considérer un point de l'image ne faisant pas partie du contour réel de l'objet. Afin de tester cette hypothèse, l'image utilisée auparavant a été bruitée. Les algorithmes de Kohonen classiques, de Kohonen modifié et du *snake* ont été comparés quant à la qualité des contours trouvés. À ce point, des conclusions ont été tirées afin de rendre l'algorithme plus robuste lors de la détection de contour d'un objet se trouvant dans une image bruitée.

La seconde modification dont il est question, nommément celle permettant de recevoir un retour d'information d'un niveau de traitement plus haut, a été implantée et testée sur la même image synthétique. Le but est de démontrer la faisabilité d'une telle implantation ainsi que les résultats que l'on est en droit de s'attendre de cette interaction entre le niveau intermédiaire et le haut niveau.

Finalement, les trois algorithmes ont été appliqués à des images réelles. En l'occurence, une image tomographique était le centre de cette investigation. Il existe des mesures quantitatives de qualité du contour. Cependant, celles-ci impliquent que le

contour soit édité manuellement. La nature même de l'image, qui présente un contour flou, rend cette opération hasardeuse en l'absence d'un expert dans l'interprétation de ce type d'images. Conséquemment et à défaut d'avoir accès à un tel groupe d'experts, des critères qualitatifs ont été utilisés pour évaluer les performances respectives des algorithmes.

En résumé, la démarche consiste à tester l'algorithme classique et la version modifiée sur des image synthétiques procurant des conditions expérimentales contrôlées. Certaines propriétés sont aussi comparées avec l'algorithme du snake afin de vérifier le type d'avantage que procure l'approche de Kohonen. Enfin, l'application à une image tomographique des algorithmes permet de constater la performance relative de chacun des algorithmes. Le but de l'approche est de colliger de l'information quant au comportement et aux propriétés de l'algorithme de Kohonen classique, de sa version modifiée et de leur performance vis-à-vis du *snake*. À la fin de ce rapport, le lecteur devrait avoir une bonne idée du comportement de l'algorithme de Kohonen lorsqu'il est appliqué à la détection de contours ainsi que comprendre les possibilités et les limites de l'algorithme.

CHAPITRE 1.

MODÈLE DU CONTOUR ACTIF

Une technique prometteuse dans le domaine de la segmentation des images bidimensionnelles (détection du contour plus précisément) et considérablement étudiée durant les cinq dernières années, est celle du contour actif proposée par Kass et al (1988). Le contour est représenté comme un modèle élastique déformable contrôlé par une contrainte de continuité dont les mouvements de glissement se produisant lors de la déformation lui ont valu le nom de *snake* (ou serpent). La segmentation est réalisée à travers un processus de minimisation d'une énergie basé sur deux forces de contrainte: force externe, qui dépend des propriétés du contour à détecter et qui est responsable de mettre le contour proche d'un minimum local en terme d'énergie, et une force interne responsable de la courbure et de la continuité du modèle en question. L'algorithme nous permet d'introduire d'autres contraintes de haut niveau en définissant des nouvelles énergies.

1.1 Formulation

Puisque les contours actifs appartiennent à la famille des courbes continues et dérivables, on peut toujours les modéliser par une forme paramétrique normalisée comme:

$$\begin{aligned} \Omega = [0,1] &\mapsto \Re^2 \\ s &\mapsto v(s) = [x(s), y(s)]^T \end{aligned} \qquad \textbf{(1.1)}$$

où

$[\ ]^T$ indique le transposé du vecteur

s est l'abscisse curviligne ou le paramètre sur la courbe $\in$ au domaine spatial Ω

$v(s)$ est le vecteur de position du point de contour de coordonnées $x(s)$ et $y(s)$

$v(1)$ et $v(0)$ sont les vecteurs de position des extrémités du contour

$v(1) = v(0)$ pour un contour fermé

L'énergie totale du contour qu'on cherche à minimiser est alors représentée par la fonction suivante (Kass et al. 1988):

$$E^*_{snake} = \int_0^1 E_{snake}(v(s))ds = \int_0^1 [E_{interne}(v(s)) + E_{externe}(v(s)) + E_{contrainte}(v(s))]\, ds \qquad \textbf{(1.2)}$$

$E_{interne}$ représente l'énergie interne due à la rigidité et l'élasticité du contour, elle est basée sur la topographie courante du contour et est fonction de la forme et de la courbure (contraintes dans l'algorithme du segmentation). $E_{externe}$ représente l'énergie externe du système due aux gradients de l'image et $E_{contrainte}$, l'énergie des autres contraintes de haut niveau ("high-level constraints") jugées pertinentes pour augmenter la précision durant la segmentation. Le contour actif localise et segmente les régions d'intérêt dans l'image en minimisant simultanément tous les termes d'énergies sur tout le modèle. Par simplicité, et pour les besoins de ce projet, l'énergie de contrainte ne sera pas approfondie plus avant dans les sections qui suivent et donc, $E_{contrainte} = 0$.

1.2 Énergie interne

Dans le but d'introduire notre connaissance *a priori* sur la forme et la courbure du contour de l'objet dans l'algorithme de segmentation, une énergie interne est incluse dans la formulation du problème. L'énergie interne sert à maintenir une certaine topologie cohérente du contour, en empêchant des noeuds individuels sur le contour de se ballader trop loin de leurs noeuds voisins. Le but est de limiter l'influence des effets externes sur la déformation du contour.

L'énergie interne modélise entre autre la tension, et elle peut être définie par (Kass et al. 1988):

$$E_{\text{int}erne}(v(s)) = \alpha(s)\left|v_s(s)\right|^2 + \beta(s)\left|v_{ss}(s)\right|^2 \qquad \textbf{(1.3)}$$

$v_s(s)$ est la dérivée première de $v(s)$ par rapport à s, tandis que $v_{ss}(s)$ est la dérivée seconde.

On peut voir que cette énergie se compose de deux termes, un terme du premier ordre ($v_s(s)$) contrôlé par $a(s)$ qui représente l'élasticité du contour, et un autre terme du second ordre ($v_{ss}(s)$) contrôlé par $b(s)$ qui représente la rigidité du contour. Le choix des fonctions $a(s)$ et $b(s)$ impose les caractéristiques du contour durant sa déformation.

La propriété d'élasticité peut être illustrée en examinant le terme de la dérivée première, $|v_s(s)|^2$, qui n'est autre que le module au carré du vecteur tangent à la courbe :

$$\left|v_s(s)\right|^2 = \left(\frac{dx(s)}{ds}\right)^2 + \left(\frac{dy(s)}{ds}\right)^2 \qquad \textbf{(1.4)}$$

D'autre part la longueur d'une courbe définie paramétriquement peut être exprimée par :

$$L = \int_0^1 \sqrt{\left(\frac{dx(s)}{ds}\right)^2 + \left(\frac{dy(s)}{ds}\right)^2}\; ds \qquad \textbf{(1.5)}$$

Ces deux expressions (1.4) et (1.5) montrent que la longueur du contour n'est qu'une simple intégration du module de la dérivée première le long de la courbe. La minimisation de la dérivée première résulte en une minimisation de la longueur globale du contour, ce qui reflète une certaine élasticité entre les différents noeuds du contour, i.e. les noeuds sont attirés par eux-mêmes. Puisque le poids relatif de ce terme est contrôlé par *a(s)*, plus ce paramètre est grand, plus l'élasticité est grande, et plus la tendance du contour à se contracter est grande.

De même, la rigidité peut être illustrée par la dérivée seconde, $|v_{ss}(s)|^2$, qui n'est autre que le taux du changement de la valeur de la tangente à la courbe. Minimiser ce module revient à réduire la possibilité d'un changement brusque en n'importe quel noeud. Plusieurs autres caractéristiques d'un contour actif sont encore évidentes à partir des définitions mentionnées ci-dessus, par exemple dans le cas d'un contour où il n'y a pas de forces externes avec $a(s) > 0$, le contour tente de former un cercle et il tend vers un point (cercle de rayon nul) avec le temps; en plus, une valeur positive de $b(s) > 0$ empêche le contour d'avoir des discontinuités locales, i.e. il ne peut pas former des coins aigus.

1.3 Énergie externe

L'énergie externe dépend des caractéristiques de l'image et du bon fonctionnement de l'algorithme de segmentation. C'est la force qui dirige le contour vers la position désirée dans l'image. On peut regrouper à l'intérieur de cette énergie plusieurs termes qu'on juge nécessaire pour une bonne segmentation spécifique. Elle contient essentiellement un terme associé aux changements abruptes de l'image, les arêtes, en plus des autres termes optionnels relatifs à chaque cas. Une des forces les plus utilisées est celle relative au gradient de l'image définie par :

$$E_{edge} = -\left|\nabla I(x,y)\right|^2 \qquad \textbf{(1.6)}$$

où I(x,y) est l'illuminance de l'image en question, ∇ est le gradient. Le signe négatif indique que les gradients plus grands minimiseront cette énergie et attireront le *snake*.

1.4 Déformation du contour actif

La déformation du contour actif sous les différentes contraintes dans un processus de minimisation de l'énergie suit les lois de la mécanique classique. En posant *v(x,y)*, la forme paramétrique du contour, comme étant la coordonnée généralisée utilisée pour la résolution des équations du mouvements donnée par l'équation de Lagrange. Le problème consiste alors à minimiser la fonction J suivante :

$$J = \int_T L(v(s,t))\, ds \qquad \textbf{(1.7)}$$

où $L(v(s,t))$ est le Lagrangien du contour

Le Lagrangien L est définit par :

$$L(v) = K(v) - U(v) \qquad \textbf{(1.8)}$$

$K(v)$ est l'énergie cinétique du contour due à son mouvement
$U(v)$ est l'énergie potentielle du contour due à sa position

L'énergie cinétique est donnée par l'équation suivante :

$$K(v) = \frac{1}{2} \int_\Omega m(s) \left| v_t \right|^2 ds \qquad \textbf{(1.9)}$$

$m(s)$ est la densité linéique

$v_t = \dfrac{\partial v}{\partial t}$: dérivée partielle de $v(s,t)$ par rapport à t

L'énergie potentielle n'est autre que E_{snake} donnée par l'équation (1.2) et qui dépend de l'énergie interne du modèle physique (élasticité et rigidité) et de l'énergie potentielle externe associée à l'image :

$$U(v) = E_{snake} = \int_{\Omega} E(v(s))\, ds = \int_{\Omega} \left[E_{interne}(v(s)) + E_{externe}(v(s))\right] ds \qquad \textbf{(1.10)}$$

Le problème de minimisation posée par l'expression 1.7 peut être reformulée comme suit:

$$J = \frac{1}{2} \int_{T} \int_{\Omega} \left[m(s)\left|v_t\right|^2 - E_{interne}(v(s)) - E_{externe}(v(s))\right] ds\, dt \qquad \textbf{(1.11)}$$

La solution au problème physique de minimisation posé par l'équation (1.7) se résume à trouver un chemin possédant un état stationnaire. Selon le calcul des variations, ce minimum doit toujours satisfaire l'équation d'Euler-Lagrange:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial n_t}\right) - \frac{\partial L}{\partial n} = 0 \qquad \textbf{(1.12)}$$

pour un système conservatif.

En substituant K et U de l'équation (1.8) par leur expression correspondante des équations (1.9) et (1.10) respectivement, le Lagrangien s'exprimera donc comme suit:

$$\begin{aligned} L &= \frac{1}{2} \int_{\Omega} \left[m(s)\left|v_t\right|^2 - w_1(s)\left|v_s(s)\right|^2 - w_2(s)\left|v_{ss}(s)\right|^2 - E_{externe}(v(s))\right] ds \\ L &= L(s, t, x(s,t), y(s,t), x_t, y_t, x_s, y_s, x_{ss}, y_{ss}) \end{aligned} \qquad \textbf{(1.13)}$$

Dans ce cas les équations d'Euler-Lagrange relatives au mouvement du contour deviennent:

$$\begin{aligned}
&\frac{\partial}{\partial x}(L) - \frac{\partial}{\partial t}\left(\frac{\partial}{\partial x_t}(L)\right) - \frac{\partial}{\partial s}\left(\frac{\partial}{\partial x_s}(L)\right) + \frac{\partial^2}{\partial^2 s}\left(\frac{\partial}{\partial x_{ss}}(L)\right) = -\frac{\partial}{\partial x_t}(D(v_t)) \\
&\frac{\partial}{\partial y}(L) - \frac{\partial}{\partial t}\left(\frac{\partial}{\partial y_t}(L)\right) - \frac{\partial}{\partial s}\left(\frac{\partial}{\partial y_s}(L)\right) + \frac{\partial^2}{\partial^2 s}\left(\frac{\partial}{\partial y_{ss}}(L)\right) = -\frac{\partial}{\partial y_t}(D(v_t))
\end{aligned} \qquad \textbf{(1.14)}$$

où $D(v_t) = \frac{1}{2}\int_{\Omega} g(s)\left|v_t\right|^2 ds$ est l'énergie de dissipation et $g(s)$ est le facteur de viscosité.

En remplaçant la valeur du Lagrangien de l'équation (1.13) dans l'équation (1.14), et après quelques manipulation, on obtient:

$$\begin{aligned}
&m(s)x_{tt} + g(s)x_t - \frac{\partial}{\partial s}(w_1(s)x_s) + \frac{\partial^2}{\partial s^2}(w_2(s)x_{ss}) = -\frac{1}{2}\frac{\partial}{\partial x}(E_{externe}(v)) \\
&m(s)y_{tt} + g(s)y_t - \frac{\partial}{\partial s}(w_1(s)y_s) + \frac{\partial^2}{\partial s^2}(w_2(s)y_{ss}) = -\frac{1}{2}\frac{\partial}{\partial y}(E_{externe}(v))
\end{aligned} \qquad \textbf{(1.15)}$$

Si on suppose que la densité linéique *m(s)* est nulle (Cohen, 1991), $g(s)$ est constant et :

$$\begin{aligned}
a(s) &= \frac{1}{g}w_1(s) \\
b(s) &= \frac{1}{g}w_2(s) \\
E_{externe} &= \frac{1}{2g}E_{externe}
\end{aligned} \qquad \textbf{(1.16)}$$

Les équations (1.15) et (1.16) donnent:

$$\frac{\partial v(s,t)}{\partial t} - \frac{\partial}{\partial s}(a(s)\frac{\partial v(s,t)}{\partial s}) + \frac{\partial^2}{\partial s^2}(b(s)\frac{\partial^2 v(s,t)}{\partial s^2}) + \nabla E_{externe}(v(s,t)) = 0 \qquad \textbf{(1.17)}$$

+ les conditions initiales et les conditions aux limites

L'évolution du contour consiste au déplacement des points du contour de façon itérative de telle sorte que les forces interne et externe du modèle physique soit équilibrées. La solution finale est obtenue lorsque $\frac{\partial v(s,t)}{\partial t}$ approche de zéro. Plusieurs méthodes numériques ont été utilisées pour la résolution de cette équation. Cohen (1991) utilise l'approximation par différences finies ("FDM"). Il s'agira essentiellement de faire évoluer le contour élastique à un temps donné t_0 en se basant sur la position du *snake* à un temps antérieur $t=t_0-\Delta t$. L'approximation par les différences finies pour les dérivées partielles prend la forme suivante (avec h = pas entre les noeuds du contour et Δt = pas temporel):

$$\left.\frac{\partial v}{\partial t}\right|_{i,t} \approx \frac{v(s_i,t)-v(s_i,t-\Delta t)}{\Delta t} = v_i(t)-v_i(t-1) \quad pour\ \Delta t = 1$$
$$\left.\frac{\partial v}{\partial s}\right|_{i,t} \approx \frac{v(s_i+h,t)-v(s_i,t)}{h} = \frac{v_{i+1}(t)-v_i(t)}{h} \qquad \textbf{(1.18)}$$
$$\left.\frac{\partial^2 v}{\partial s^2}\right|_{i,t} \approx \frac{v(s_i+h,t)-2v(s_i,t)+v(s_i-h,t)}{h^2} = \frac{v_{i+1}(t)-2v_i(t)+v_{i-1}(t)}{h^2}$$

(*voir figure*1.1)

i = 0,1, ..., N-1 *N* = nombre des noeuds

Si on substitue les termes de l'équation (1.17) par leurs valeurs respectives des équations (1.18), on aura:

$$\frac{v_i(t)-v_i(t-\Delta t)}{\Delta t}+\frac{a_i}{h}(v_i(t)-v_{i-1}(t))-\frac{a_{i-1}}{h}(v_{i+1}(t)-v_i(t))+\frac{b_{i-1}}{h^2}(v_{i-2}(t)-2v_{i-1}(t)+v_i(t))$$
$$-2\frac{b_i}{h^2}(v_{i-1}(t)-2v_i(t)+v_{i+1}(t))+\frac{b_{i+1}}{h^2}(v_i(t)-2v_{i+1}(t)+v_{i+2}(t))+f(v_i(t))=0 \qquad \textbf{(1.19)}$$

avec $f(v(s,t))=f(x(s,t),y(s,t))=\nabla E_{externe}(v(s,t))$

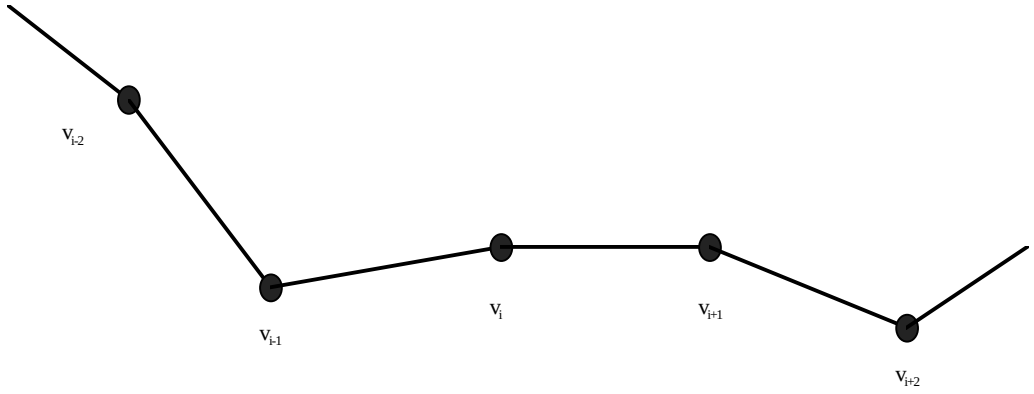


Figure 1.1: Position des noeuds sur le contour

Si on suppose que la force externe est constante durant un pas de temps, l'équation précédente devient :

$$(I+t\,A)V^t=V^{t-1}+\Delta tF(V^{t-1}) \qquad \textbf{(1.20)}$$

où I = matrice identité de dimension N et

$$\mathbf{A}=\begin{bmatrix} c_0 & b_0 & a_0 & & & & a_{N-2} & b_{N-1} \\ b_0 & c_1 & b_1 & a_1 & & & & a_{N-1} \\ a_0 & b_1 & c_2 & b_2 & a_2 & & & \\ & a_1 & b_2 & c_3 & b_3 & a_3 & & \\ & & & & \cdots\cdots & & & \\ & & & a_{N-5} & b_{N-4} & c_{N-3} & b_{N-3} & a_{N-3} \\ a_{N-2} & & & & a_{N-4} & b_{N-3} & c_{N-2} & b_{N-2} \\ b_{N-2} & a_{N-1} & & & & a_{N-3} & b_{N-2} & c_{N-1} \end{bmatrix} \qquad \textbf{(1.21)}$$

A est une matrice symétrique pentadiagonale avec les éléments:

$$a_i = b_{i+1}$$

$$b_i = -2\,b_i - 2b_{i+1} - a_{i+1}$$

$$c_i = b_{i+1} + 4b_i + b_{i+1} + a_i + a_{i+1}$$

De l'équation (1.25), sachant que

$$V = [v_0 , v_1 , \ldots , v_{N-1}]^T$$

$$F(V) = [f(v_0), f(v_1), \ldots , f(v_{N-1})]^T$$

nous obtenons que

$$V^n = (I+\Delta t A)^{-1}(V^{t-1}+\Delta t F(V^{t-1})) \qquad (\mathbf{1.22})$$

Cependant, il est plus facile et surtout plus rapide de résoudre l'équation (1.20) par la décomposition en LU de $(I + \Delta t A)$ (Cohen, 1991). La décomposition en LU consiste à factoriser une matrice A par un produit de deux matrice triangulaires: l'une inférieure (ou "Lower") et l'autre supérieure (ou "Upper"). Cette technique de l'algèbre linéaire permet d'éviter un grand nombre de substitutions lors du calcul de la matrice inverse.

Les deux matrices $(I - \Delta t A)$ et $(I + \Delta t A)^{-1}$ dérivent du modèle des forces internes, et imposent une certaine régularisation sur le contour à chaque itération. Ces deux matrices jouent le rôle d'un filtre passe-bas, et l'algorithme se résume par un cycle répétitif:

mouvement ⇨ régularisation ⇨ mouvement ⇨ régularisation .....

Une des propriétés des *snakes* est la continuité, c'est-à-dire qu'ils peuvent remplir les vides entre les différentes arêtes là où il y a un manque d'informations dû au bruit ou à la qualité de l'image en question.

CHAPITRE 2.

CARTES DE KOHONEN (SOFM)

Des modèles d'auto-organisation, inspirés par l'organisation corticale des vertébrés, ont été proposés dès les années 70, notamment par von der Malsburg (1973), von der Malsburg et Willshaw (1977), Willshaw et von der Malsburg (1976, 1979), puis par Kohonen (1982). On entend par "auto-organisation" un processus par lequel un système trouve, sans apprentissage supervisé, une solution optimale à un problème. D'un point de vue pratique dans les domaines d'application, ces modèles d'auto-organisation ont pour objectif de représenter des données complexes, souvent bruitées (caractérisées par de nombreuses mesures, c'est-à-dire appartenant généralement à un espace de grande dimension) dans un espace discret dont la topologie est limitée à 1, 2 voire 3 dimensions. Globalement, il s'agit donc de modèles de Quantification Vectorielle (QV) qui seront dotés de propriétés topologiques particulières. Dans ce chapitre, nous présentons le principe des cartes auto-organisées de Kohonen afin de bien montrer le lien avec la méthode des contours actifs.

2.1 Cartes auto-organisées de Kohonen

Ce modèle a été présenté par Kohonen (1982) , et malgré son intérêt, il est resté assez longtemps ignoré. Dans les années 1990, Kohonen a proposé diverses variantes pour la classification dont les algorithmes de Quantification Vectorielle à apprentissage. Ce modèle appartient à la classe des réseaux de neurones non-supervisés c'est-à-dire qu'aucune intervention humaine n'est requise et que peu d'informations sont nécessaires relativement aux caractéristiques des données d'entrée.

Les cartes de Kohonen, comme les autres réseaux auto-organisés, ne font appel à aucun retour d'information de l'environnement. Un tel réseau se chargera donc lui-même de découvrir les relations qui existent entre les données d'entrée. De façon générale, un réseau de neurones peut être décrit par 3 caractéristiques soient: une architecture, une règle d'apprentissage et une règle de transmission.

2.1.1 L'architecture

L'architecture des cartes de Kohonen est constituée de 2 couches de neurones. La couche d'entrée comporte un nombre *p* d'unités d'entrées, où *p* est la dimensionnalité de l'espace de départ devant être transformé en un espace de représentation (voir figure 2.1).

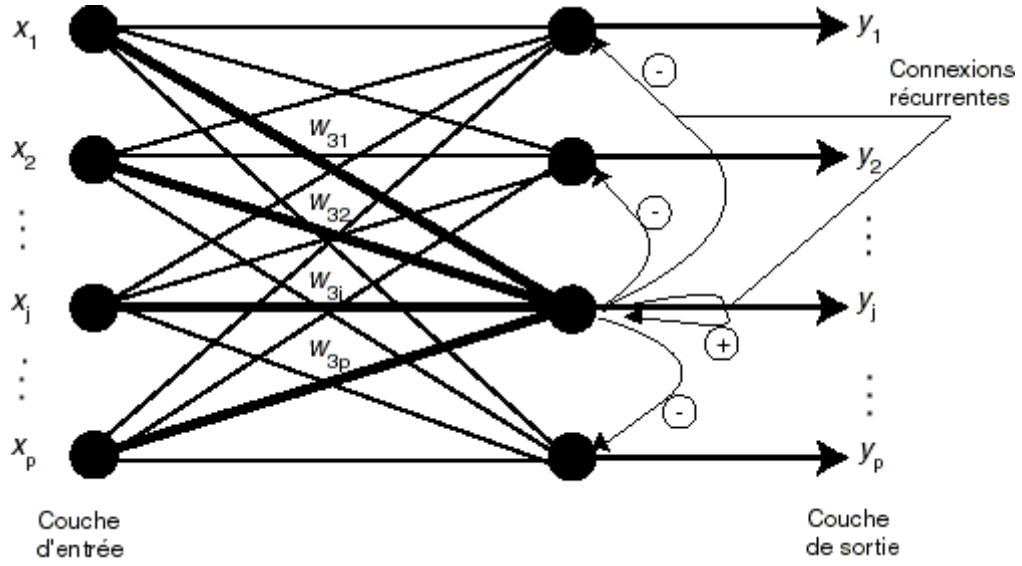


Figure 2.1: Architecture du réseau de Kohonen montrant la relation entre les vecteurs d'entrée ***x***, les poids des connexions ***w***, les connexions récurrentes d'excitation (+) et d'inhibition (-), et les sorties ***y***.

Par exemple, une carte de Kohonen destinée à représenter un groupe d'élèves selon leur poids, leur taille et leur note au dernier examen de géographie transformerait un ensemble de départ à 3 dimensions et posséderait donc, 3 unités d'entrées. Dans le modèle de base, la couche de sortie contient *N* neurones dont le nombre et la topologie

sont fixés dès le départ. Le nombre de neurones détermine le niveau de détail ou l'échelle désirée pour le modèle résultant. L'échelle influencera l'exactitude ou la capacité de généralisation du modèle. Les deux caractéristiques étant contradictoires, la propriété voulue doit être déterminée par le concepteur. Certaines heuristiques existent mais l'expérience compte pour beaucoup dans la conception de réseaux de neurones en général. La topologie de cette couche est déterminée par une fonction de voisinage. Les unités de sortie sont disposées de telle sorte que chacune d'entre elle possède des voisines définies et conserve les mêmes unités voisines pendant l'apprentissage et après l'apprentissage. Plusieurs configurations sont possibles. Pour un réseau unidimensionel, chaque unité possèderait deux unités voisines sauf pour celles se trouvant aux extrémités. Celles-ci n'auraient qu'une seule unité voisine. Un réseau bidimensionel peut adopter une configuration "carrée" où chaque unité possèderait 4 unités voisines par exemple.

En général, ces neurones sont disposés en un arrangement en 1 ou 2 dimensions. Chacune des *N* unités d'entrée est reliée à chacun des neurones de sortie par *p* connexions modifiables. Le réseau contient donc *Nxp* connexions au total. Enfin, chaque neurone possède des connexions latérales récurrentes, qui viendront modifier le patron d'excitation de façon à créer la forme d'un "chapeau mexicain". La règle de transmission, détaillée à la section 2.1.2, décrit le processus de génération du signal de sortie et ainsi que le rôle des connexions latérales. Ce chapeau mexicain est caractérisé par 3 zones distinctes tel qu'indiqué à la figure 2.2:

1. Une région restreinte d'excitation.
2. Un région de pénombre ou d'action inhibitoire.
3. Un région d'excitation plus faible; cette zone possède une action négligeable et est habituellement ignorée.

Ce type d'interaction correspond approximativement aux relations entre les micro-colonnes corticales.

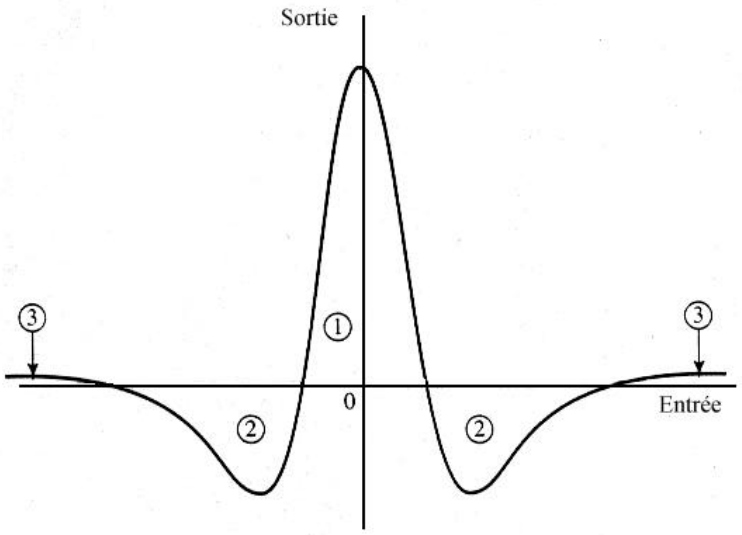


Figure 2.2 Noyau de convolution « chapeau mexicain »

2.1.2 Règle de transmission

Soit $x = [\, x_1, x_2, \ldots, x_p]$, le vecteur d'entrée appliqué au réseau où p est le nombre d'unités d'entrée. Posons maintenant $w_j = [\, w_{j1}, w_{j2}, \ldots, w_{jp}\,]$, le vecteur des poids synaptiques relatif au neurone de sortie j. Soit $c_j = [\, c_{j-k}, \ldots, c_{j-1}, c_j, c_{j+1}, \ldots, c_{j+k}\,]$ le vecteur poids des connexions récurrentes relatif au neurone j, où K détermine le rayon d'action de l'interaction latérale. Posons enfin $y = [\, y_1, y_2, \ldots, y_N]$ le vecteur de sortie du réseau où N est le nombre d'unités de sortie du réseau. La sortie du réseau peut être exprimée sous la forme générale suivante (Haykin, 1994):

$$y_j = j\left(\sum_{i=1}^{p} w_{ji}\, x_i + \sum_{l=-k}^{k} c_{jl}\, y_{j+l}\right) \quad \text{où } j = 1, 2, \cdots, N \qquad \textbf{(2.1)}$$

où $j(.)$ est une fonction non-linéaire (limite la valeur de y_j et assure que $y_j \geq 0$). La première somme exprime la réponse pondérée des signaux d'entrée sur un neurone j. La seconde somme exprime la réponse de contre-réaction interne générée par les connexions latérales. La solution de cette équation non-linéaire est trouvée itérativement en utilisant une technique de relaxation avec des équations aux différences (Haykin, 1994):

$$y_j(n+1)=j\left(\sum_{i=1}^{p} w_{ji}\,x_i+b\sum_{l=-k}^{k} c_{jl}\,y_{j+l}(n)\right) j=1,\ 2,\ \cdots,\ N \qquad (\mathbf{2.2})$$

où *n* est le temps discret et b est le taux de convergence de ce processus. Cette fonction génère une "bulle" d'activation de la forme du chapeau mexicain présenté à la figure 2.2 autour de l'unité où le produit scalaire $w_{ji}x_i$ est le plus grand, c'est-à-dire où l'activation initiale de l'unité de sortie y_j est maximale. Cette unité est aussi appelée "unité gagnante" ou "neurone gagnant".

En idéalisant la sortie, et pour les fins des simulations, il est possible d'imposer une sortie de la forme suivante:

$$y_j=\begin{cases} a, & \textit{neurone } j \textit{ à l'intérieur de la bulle} \\ 0, & \textit{neurone } j \textit{ à l'extérieur de la bulle} \end{cases} \qquad (\mathbf{2.3})$$

Il est donc possible d'exploiter la formation d'une bulle d'activation de façon à prendre un raccourci. Notons ici que les régions 2 et 3 du chapeau mexicain de la figure (2.2) sont négligées. La notion de voisinage topologique est ici introduite pour redéfinir la façon dont les simulations se sont effectuées. Le voisinage topologique consiste en les voisins de l'unité gagnante i(**x**) et est dénoté L_i(n). Cette fonction détermine l'étendue et l'amplitude de la bulle d'activation selon que l'on l'ajuste pour simuler une contre-réaction latérale positive plus grande (grand voisinage), ou une contre-réaction latérale négative plus grande (petit voisinage).

2.1.3 Règle d'apprentissage

Le mode d'apprentissage des réseaux auto-organisés constitue le principal volet d'innovation de ceux-ci puisque leur architecture représente plus ou moins des combinaisons d'autres types de réseau comme les mémoires associatives, le perceptron, le perceptron multicouche . Il consiste principalement à trouver le neurone gagnant *i* à l'aide d'une mesure de similarité et de mettre à jour les poids des neurones concernés c'est-à-dire ceux qui se trouvent à l'intérieur du voisinage topologique L_i(n). L'idée d'utiliser des connexions latérales inhibitoires pour induire une compétition du type "le plus fort emporte tout" a été proposée pour la première fois par Rosenblatt (1958). La règle

d'apprentissage compétitive présentée ici a quant à elle été introduite originellement par Grossberg (1969).

Une étape de l'apprentissage s'effectue en choisissant aléatoirement un vecteur de l'ensemble d'apprentissage *x*. Ce vecteur est ensuite comparé à chacun des vecteurs poids. Le neurone gagnant *i(x)* est ensuite identifié à l'aide d'une mesure de similarité habituellement définie comme étant la distance euclidienne entre les deux vecteurs.

$$i(x) = \arg\min \left\| x - w_j \right\| \quad j = 1, 2, \cdots, N \qquad (\mathbf{2.4})$$

où $\|.\|$ est la norme euclidienne telle que définie par,

$$\|x\| = \sqrt{\sum_{j=1}^{p} x_i^2} \qquad (\mathbf{2.5})$$

Une fois trouvé le vecteur de poids gagnant, le réseau s'ajuste de manière à mieux représenter ce vecteur d'apprentissage. Le résultat final peut être visualisé par un filet s'ajustant au nuage des données d'entrée de façon à ce que chaque vecteur poids approxime la densité de probabilité d'une région de l'ensemble d'apprentissage. La règle d'apprentissage pour un neurone *j* appartenant au voisinage topologique variable du neurone gagnant *i(x)* décrit une rotation des vecteurs poids vers le vecteurs d'entrée et s'énonce comme suit:

$$w_j(n+1) = \begin{cases} w_j(n) + h(n)\left[x - w_j(n)\right] & ssi\ j \in \Lambda_{i(x)}(n) \\ w_j(n) & ssi\ j \notin \Lambda_{i(x)}(n) \end{cases} \qquad (\mathbf{2.6})$$

où $\Lambda_{i(x)}(n)$ dénote le voisinage topologique autour du neurone gagnant au temps discret n, et $h(n)$ est la fonction de voisinage qui contient une fonction de taux d'apprentissage $a(n)$ décroissante en fonction du temps, elle détermine aussi le taux d'adaptation des neurones situés à l'intérieur de $\Lambda_{i(x)}(n)$. Cette fonction se veut un moyen plus simple de simuler les connexions latérales entre neurones. Dans le cas d'une fonction de voisinage gaussienne autour du neurone gagnant *i(x)*, $h(n)$ s'exprime comme suit:

$$h\,(n) = a\,(n) \cdot \exp\left(-\frac{\left\| r_j - r_{i(x)} \right\|^2}{2s\,(n)^2} \right) \qquad j \in \Lambda_{i(x)}(n) \qquad (\mathbf{2.7})$$

où $\left\| r_j - r_{i(x)} \right\|$ est la distance entre le neurone *j* et celui gagnant *i(x)*.

Plusieurs fonctions peuvent être utilisées pour le voisinage en autant qu'elles respectent la condition de non-croissance. Une fonction constante autour du neurone gagnant peut s'avérer moins coûteuse au plan des calculs. Un noyau gaussien est cependant souvent utilisé à cause de la rapidité de convergence qu'il procure. *La convergence est établie quand la carte résultante ne change plus de façon significative.*

Algorithme d'apprentissage :

1. *Initialisation*: choix aléatoire des vecteurs poids initiaux $w_j(0) \quad j = 1,2,\cdots,N$ où *N* est le nombre des neurones dans la structure et est égal à la dimension des vecteurs d'apprentissage. La seule restriction est que les poids soient différents. Si cette restriction n'est pas respectée, chacun des vecteurs initiaux seraient à une distance égale de tous les vecteurs d'apprentissage empêchant du coup toute possibilité de déplacement des vecteurs de la carte. Il est préférable d'avoir des petites valeurs pour les poids.
2. *Présentation*: tirage aléatoire d'une entrée *x* dans la base d'apprentissage afin de le présenter au réseau.
3. *Similarité*: recherche du neurone gagnant *i(x)* en utilisant le critère de la distance euclidienne minimale (neurone plus proche, équation (2.3)).
4. *Modification des poids*: ajuster les poids des neurones appartenant au voisinage topologique $\Lambda_{i(x)}(n)$ suivant le pas d'adaptation $h(n)$ (équation (2.5)). $\Lambda_{i(x)}(n)$ et $h(n)$ varie dynamiquement durant l'apprentissage et ceci pour un meilleur résultat.
5. *Continuation*: retour à l'étape 2 si le changement des poids demeure non-négligeable.

2.2 L'initialisation

Il est possible d'utiliser trois types d'initialisation des neurones: l'initialisation aléatoire, l'initialisation à partir d'un échantillon et l'initialisation linéaire. L'initialisation aléatoire attribue à chaque neurone des poids au hasard. L'initialisation à partir d'un échantillon de l'ensemble d'apprentissage utilise des éléments de l'ensemble d'apprentissage pour déterminer les poids. Ceci a l'avantage que les vecteurs poids des neurones (qui représentent les coordonnées de ces neurones dans l'espace des caractéristiques) pointe déjà sur les mêmes régions que l'ensemble d'apprentissage. Quant à l'initialisation linéaire, elle tire profit de l'analyse en composantes principales pour l'analyse des données. Les vecteurs poids des neurones sont donc initialisés de telles sorte qu'ils se trouvent dans le même plan que celui formé par les deux vecteurs propres se trouvant dans les directions de plus grandes inerties.

2.3 Propriétés des SOM

Une fois que l'algorithme a convergé, le SOM ("Self-Organizing Maps" ou Cartes de Kohonen) démontre des caractéristiques statistiques importantes. Afin de démontrer ces propriétés, considérons un ensemble d'entrée continu X dont la topologie est décrite par une relation métrique des vecteurs $x \hat{I} X$. L'ensemble A est un ensemble discret de sorties dont la topologie a été déterminée par l'apprentissage des vecteurs poids. Dénotons aussi Φ, une transformation non-linéaire que nous appellerons *carte de caractéristiques* qui cartographie l'espace d'entrée X sur l'espace de sortie A tel que,

$$\Phi : X \Rightarrow A \qquad \textbf{(2.8)}$$

Étant donné un vecteur d'entrée x, le SOM procède d'abord à la détermination du neurone gagnant $i(x)$ dans l'espace de sortie A selon une règle de transformation donnée par la carte des caractéristiques Φ. Le poids w_i associé au neurone gagnant $i(x)$ est alors considéré comme un *pointeur* de ce neurone vers l'espace X. Ces opérations sont représentées par la figure 2.3 ci-dessous. Cette propriété est utilisée dans la détection de contours d'objets en ce que le réseau transforme un ensemble d'arêtes provenant d'un

ensemble continu (ou plutôt presque continu puisque l'image est numérisée) en un ensemble discret d'unités de contour. Le poids de ces unités sont des approximations des positions des arêtes de l'image.

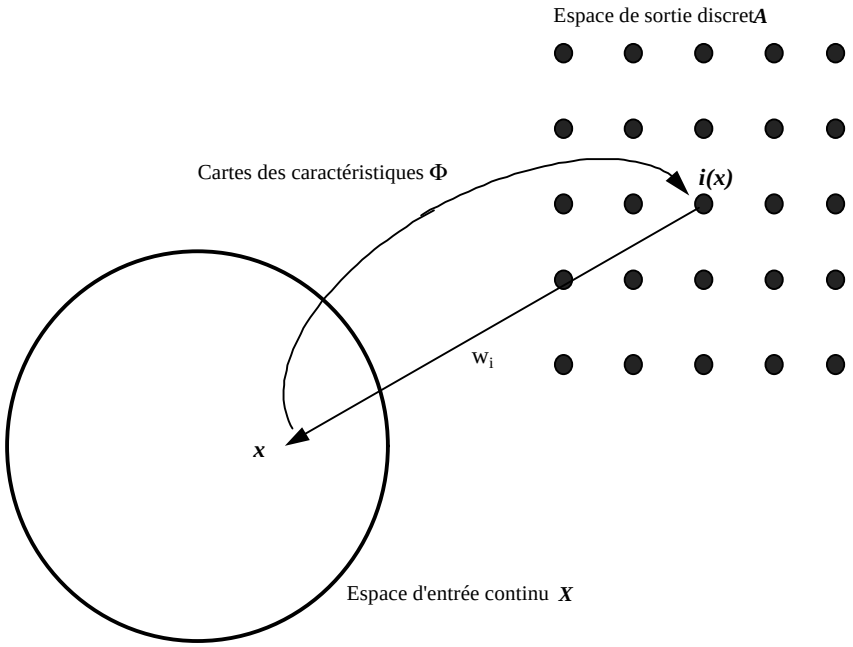


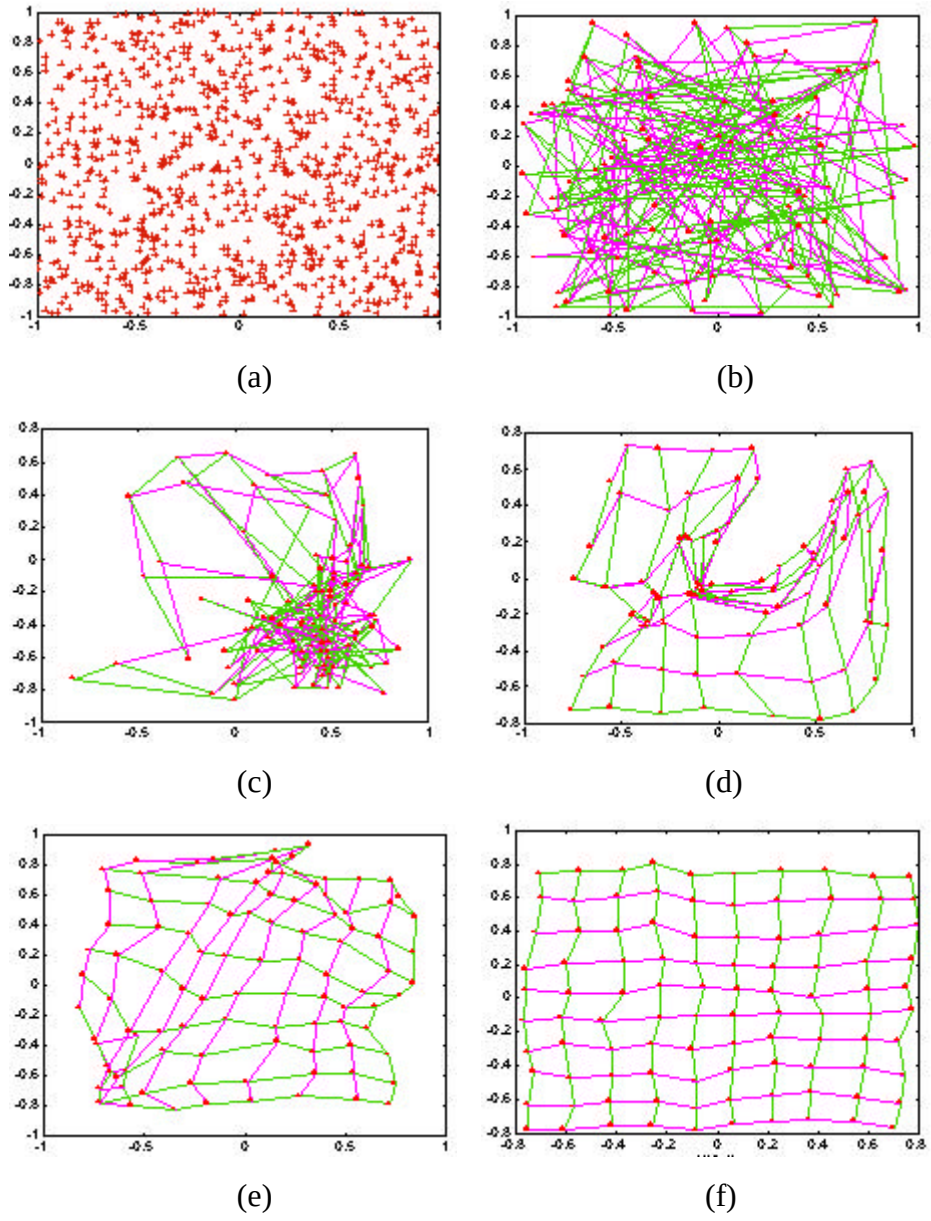
Figure 2.3 Relation entre la carte des caractéristiques Φ et le vecteur poids w_i du neurone gagnant $i(x)$.

2.3.1 Approximation de l'espace d'entrée

Le SOM Φ, représenté par les vecteurs poids $w_j \mid j = 1, 2,..., N$, constitue une bonne approximation de l'espace de départ X. Le but principal d'une telle transformation est de faire l'approximation d'un grand ensemble de vecteurs $x \hat{I} X$ par une série plus petite de prototypes $w_j \hat{I} A$. La base théorique de cette idée repose sur la théorie de la quantification vectorielle ("*vector quantization theory*") dont les points centraux sont la réduction de la dimensionnalité et la compression des données.

Afin d'illustrer cette idée, l'exemple qui suit applique les cartes de Kohonen à un ensemble de vecteurs dont nous connaissons *a priori* la distribution. Ceci permettra de mieux visualiser la capacité de représentation lors de l'application des cartes de Kohonen sur des distributions plus complexes. Soit X la portion de $\hat{A}^2$ définie par : $x = [\ -1 < x_1 < +1\ ;\ -$

$1< x_2 <+1]$. Étant donné la nature bidimensionnelle du probleme, c'est-à-dire une distribution dans $\hat{A}^2$, il est approprié de représenter ces points sur une carte 2D de Kohonen. Cette carte représentera ultimement l'approximation de l'ensemble X par un autre plus petit A qui contient les neurones de coordonnées w_j. On choisit dans cet exemple un réseau de 10 neurones par 10 neurones et un ensemble d'apprentissage X de 1000 vecteurs de position distribués uniformément dans le sous-espace X. Une distribution uniforme devrait générer une carte dont les unités sont aussi uniformément distribuées dans le sous-espace. En effet, le résultat de l'apprentissage de la carte de Kohonen est un ensemble *représentatif* de l'ensemble d'apprentissage en l'occurence, des 1000 vecteurs uniformément distribués. La figure 2.5 montre l'évolution du réseau au cours de l'apprentissage dans le cas d'une distribution de points x répartis uniformément sur tout le domaine. À la convergence, les neurones représentent des zones R_i assez équiprobables, régulièrement réparties et de même surface (en moyenne), donc chaque neurone a la même probabilité d'être sélectionné. Ces zones R_i sont formées autour de chacune des unités w_j du réseau et sont caractérisées par ce que chacun des points de cette zone sont plus près de l'unité w_j et seulement de cette unité. Les R_i sont donc des ensembles mutuellement exclusifs dont l'union comprend tous les points de l'espace X tel un casse-tête. Ces régions sont aussi appelées "cellules de Voronoï" et seront traitée plus avant à la section 4.2 du chapitre suivant.



(a) (b)

(c) (d)

(e) (f)

Figure 2.5: (a) distribution uniformes de 1000 vecteurs de position; (b) poids initiaux des neurones (10x10); état du réseau après (b) 50 , (c) 250 , (d) 1000, (e) 10000 itérations respectivement.

2.3.2 Appariement de la densité d'une distribution

La carte des caractéristiques Φ reflète les variations dans la distribution des vecteurs de l'ensemble d'apprentissage. La densité de la répartition des neurones dans l'espace de sortie reflète les variations statistiques de l'espace d'entrée: les régions de l'espace d'entrée desquelles les vecteurs d'entrée sont tirés avec une grande probabilité sont

cartographiées sur de plus grandes régions dans l'espace de sortie. Ceci permet en général d'avoir une meilleure résolution que pour les régions où la probabilité de tirer un vecteur est plus faible. Autrement dit, les zones de représentation R_i sont grandes dans les régions à forte densité de probabilité et inversement. De façon générale, l'algorithme SOM tend à sur-représenter les régions de faibles probabilité et à sous-représenter les régions de grande probabilité. Un moyen d'améliorer l'appariement des densités est d'ajouter une heuristique appelée conscience à l'algorithme d'apprentissage (DeSieno, 1988). Cette heuristique biaise le processus de compétition en effectuant une "redistribution" des probabilités que chacun des neurones gagne la compétition. Les neurones tiennent un décompte du nombre de fois qu'ils ont gagné et se retirent de la course (par sentiment de culpabilité!) lorsque celui-ci devient trop grand.

2.3.3 Conservation de l'ordre topologique

La propriété de l'ordre topologique est une conséquence directe de l'équation 2.5 de mise à jour des vecteurs poids qui force, entre autre, le neurone gagnant $i(x)$ et tous les neurones appartenant à son voisinage topologique $L_{i(x)}$ de se déplacer vers le vecteur d'entrée x. La proximité topologique des neurones est conservée après convergence, c'est-à-dire que les vecteurs semblables de l'ensemble de départ X sont cartographiés sur des neurones se trouvant près les uns des autres. On peut donc visualiser que le SOM Φ , avec une topologie de 1D ou 2D déjà définie dans l'espace A, ressemble à un contour élastique ("elastic net"). Cette propriété est fondamentale à l'utilisation des SOM pour la détection de contours.

CHAPITRE 3.

APPROCHE COMMUNE

Jusqu'à maintenant, et d'après les deux chapitres précédents, aucune similitude entre les deux algorithmes (contours actifs et Kohonen 1D) n'est saillante. D'un côté, les contours actifs évoluent d'une manière globale au sens où tous les points discrets (noeuds) du contour se déplacent en même temps. De l'autre côté, la modification des poids (coordonnées des neurones dans le plan caractéristique) de la carte de Kohonen se fait d'une manière stochastique, c'est-à-dire que l'état du réseau change à chaque présentation d'un vecteur de l'ensemble d'apprentissage. L'algorithme des contours actifs impose une régularisation au niveau des noeuds à la fin de chaque itération, tandis que celui de Kohonen ne le fait pas. De plus, les deux algorithmes génèrent un contour final dont les unités discrètes ne se trouvent pas nécessairement sur une arête (voir figure 3.1) mais effectuent l'approximation de l'ensemble de ces arêtes d'une façon analogue au concept de la moyenne en statistique: la moyenne représente un échantillon sans toutefois avoir nécessairement la valeur d'un élément de l'échantillon. Les contours actifs et les cartes auto-organisées utilisent l'information se trouvant dans l'image et "interpolent" la localisation du contour réel de l'objet. Exprimer ces deux algorithmes à l'aide d'une approche commune servira à mieux les comparer.

Dans la section présente, il est bon de noter dès le départ que la segmentation est bidimensionnelle et que le contour que l'on cherche est fermé. Ceci implique donc une carte de Kohonen unidimensionnelle avec un voisinage périodique, où le premier et le dernier neurones sont voisins.

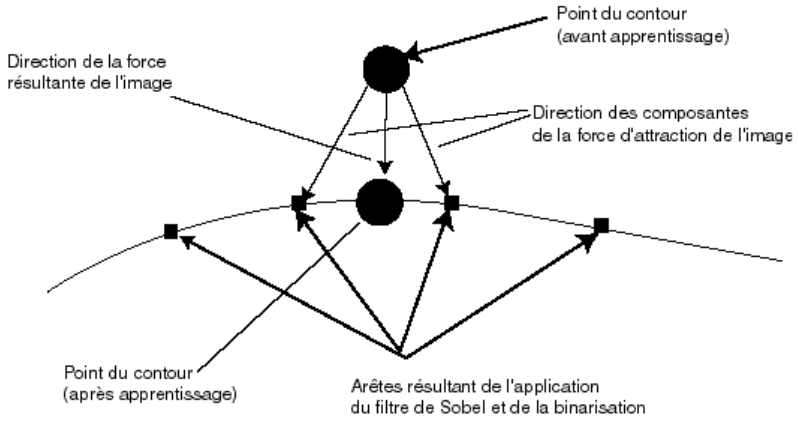


Figure 3.1. Exemple d'une suite d'arêtes formant un contour.

3.1 Notation

Avant de commencer la reformulation des deux algorithmes, il est nécessaire de trouver une notation commune. Puisqu'on travaille avec des images bidimensionnelles, les unités - i.e. les noeuds dans le cas des contours actifs et neurone dans le cas de la carte de Kohonen - sont des points en 2D. Chacun des points du contour est identifié par un vecteur $z=[x, y]^T$.

Soit: $z^t=\left[z_0^t, z_1^t, \cdots, z_{N-1}^t\right]^T$ le vecteur des points de contour à la t ième itération.

$$\text{où } z=\begin{cases} v(s)=\left(x(s), y(s)\right) \textit{contour actif} & \text{"}\textit{snake}\text{"} \\ w_j=(x_j, y_j) \ \textit{Kohonen} & (\textit{neurone } j) \end{cases}$$

3.2 Algorithmes modifiés

3.2.1 Algorithme de Kohonen

Dans la phase d'apprentissage, chaque vecteur d'entrée est présenté au réseau qui sélectionne le neurone le plus proche par une mesure de similarité euclidienne. L'ensemble d'apprentissage est généré par le résultat d'un seuillage de l'image filtrée avec un filtre de Sobel. Le résultat est ensuite seuillé pour générer un ensemble d'arêtes (voir figure 3.2). Ces arêtes serviront, pour l'algorithme de Kohonen, de points de référence indiquant où se trouve le contour de l'objet de l'image. Ce sont ces arêtes que l'algorithme doit relier afin de ne former qu'un seul contour.

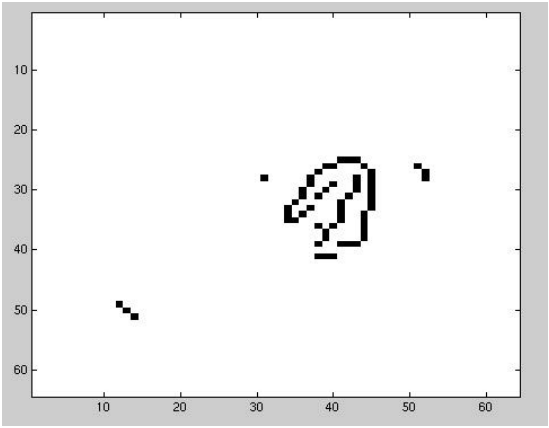


Figure 3.2. Résultat du filtrage et seuillage avec un filtre de Sobel pour une image tomographique.

Les vecteurs résultants représentent donc les endroits de l'image où le gradient est maximal.

Définissons $P = [\ p_1,p_2,...,p_M\ ]^T$ comme étant l'ensemble d'apprentissage contenant M éléments. Soit $z_j^t = [x_j^t, y_j^t]$ la position du $j^{\text{ième}}$ neurone du contour à la $t^{\text{ième}}$ itération. Les neurones sont généralement identifiés tel que $j = 0,1, ... , N-1$ où N est le nombre de neurones de la carte. Rappelons que b est le rayon du voisinage topologique $L(t)$ (voir figure 3.3).

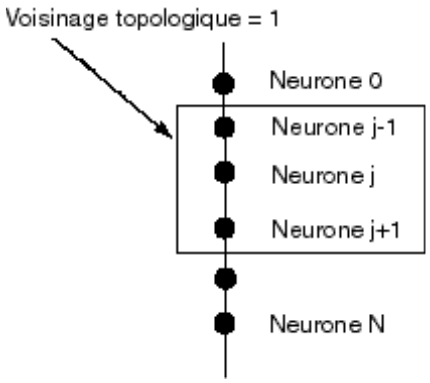


Figure 3.3: Voisinage topologique d'une carte de Kohonen unidimensionnelle.

Dans le cas où les poids sont normalisés, i.e. que leur module est unitaire, le neurone gagnant z_j est celui possédant la plus courte distance euclidienne avec le vecteur d'entrée p_i ($i = 1...M$) tel que:

$$n = \arg\min_j \left\| p_i - z_j^t \right\| \qquad j = 0,1,\cdots,N-1 \qquad \textbf{(3.1)}$$

le réseau est mis à jour suivant la règle:

$$z_{(k+n)}^t = z_{(k+n)}^{t-1} + g\ (p_i - z_{(k+n)}^{t-1}) \qquad n = -b,\cdots,0,\cdots,+b \qquad \textbf{(3.2)}$$

où g est la constante d'apprentissage.

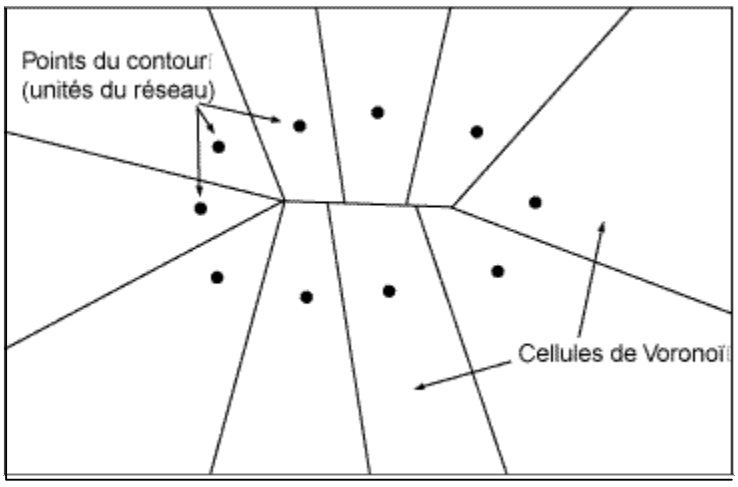


Figure 3.4: Partition de l'espace d'entrée - cellules de Voronoï

Notons V_j^t, l'ensemble de tous les vecteurs de positions de l'image qu'il est possible de cartographier au neurone j à la t ième itération. La cartographie s'effectuant selon la règle du plus proche voisin, l'ensemble V_j^t comprendra donc tous les points de l'image qui sont les plus près du neurone j à la t ième itération. Cet ensemble est souvent appelée cellule de *Voronoï* (figure 3.4). Les vecteurs d'entrée de l'image ne constituent donc qu'un sous-ensemble de chacune de ces cellules. L'union de toutes les cellules de Voronoï forme le plan caractéristique de l'image. Cette répartition de l'espace d'entrée change à chaque passage d'un vecteur p_i. Étant donné que le rayon du voisinage est égal à b, chaque neurone j appartient à $2b+1$ cellules de Voronoï, V_{j+k} ($k = -b,...,+b$), pour n'importe quelle itération. Si on suppose que ces cellules sont fixes durant un passage complet de l'ensemble P, le neurone j se déplacera successivement vers tous les points p_i cartographiés à l'intérieur de ces $2b+1$ cellules. En généralisant ce cas sur tous les neurones du réseau, la mise à jour peut être faite une seule fois après chaque passage de l'ensemble d'apprentissage - les Anglo-Saxons appellent ce mode de mise à jour des poids le "batch mode". La règle de mise à jour peut ensuite être mise sous la forme suivante (Abrantes et al 1996):

$$z_k^t = z_k^{t-1} + g \sum_{n=k-b}^{k+b} \sum_{i \in V_{(k)_N}^{t-1}} (p_i - z_k^{t-1}) \qquad k = 0, \cdots, N-1 \qquad \textbf{(3.3)}$$

où $V^t{}_{(k)N}$ est la région de l'espace "cartographiée" au (k modulo N)ième neurone à la t ième itération. Dénotons maintenant par m_n^t la masse totale des vecteurs d'apprentissage se trouvant dans la cellule V_n^t. Puisque ces vecteurs sont le résultat de la binarisation des gradients, chacun de ceux-ci possède un poids égal à 1. La masse totale des vecteurs d'une cellule, m_n^t, est donc simplement, dans ce cas particulier, le nombre de vecteurs d'entrée se trouvant à l'intérieur de la cellule V_n^t. Dénotons aussi par x^{V_n} le centroïde de ces vecteurs d'apprentissage pour chacune de ces cellules, les centroïdes seront calculés comme suit:

$$x_n^t = \frac{1}{m_n^t} \sum_{i \in V_n^t} p_i \qquad \textbf{(3.4)}$$

L'équation 3.3 devient alors:

$$z_k^t = z_k^{t-1} + g \sum_{n=k-b}^{k+b} m_n^{t-1}(x_n^{t-1} - z_k^{t-1}) \qquad k = 0, \cdots, N-1 \qquad \textbf{(3.5)}$$

On peut dès lors observer que la force de l'image agissant sur chacune des unités dépend non seulement du centroïde de sa cellule de Voronoï mais aussi des centroïdes voisins. En décomposant chacune des forces, on obtient (Abrantes et Marques 1996):

$$z_k^t = z_k^{t-1} + g \sum_{n=k-b}^{n=k+b} m_n^V (z_n - z_k) + g \sum_{n=k-b}^{k+b} m_n^V (x_n^V - z_n) \qquad \textbf{(3.6)}$$

$$z_k^{t+1} = z_k^t - g \left(\sum_{\substack{n=k-b \\ n \neq k}}^{n=k+b} m_n^V z_n + z_k \sum_{\substack{n=k-b \\ n \neq k}}^{n=k+b} m_n^V \right) + g \sum_{n=k-b}^{n=k+b} m_n^V (x_n^V - z_n) \qquad \textbf{(3.7)}$$

Le terme entre parenthèses ne dépend que de la forme du contour et peut dès lors être interprété comme une force interne. Le dernier terme dépend des différences entre les centroïdes et les unités du réseau et peut être interprété comme un force de l'image. La récursion de l'équation 3.7 peut être exprimée sous forme matricielle comme suit:

$$z^{t+1} = (I - g\ A^t) z^t + g\ B^t (x^{V_t} - z^t) \qquad \textbf{(3.8)}$$

où la matrice *A* est définie de la façon suivante,

$$a_{ij} = \begin{cases} \sum_{\substack{k=j-b \\ k \neq j}}^{k=j+b} m_k^V & i = j \\ -m_j^V & 0 \leq |i-j| \leq b \\ 0, & |i-j| > b \end{cases} \qquad \textbf{(3.9)}$$

et la matrice B,

$$b_{ij} = \begin{cases} m_j^V & |i-j| \le b \\ 0, & autrement \end{cases} \qquad \textbf{(3.10)}$$

3.2.2 Algorithme du contour actif ou *snake*

Dans le cas des contours actifs, la règle d'évolution, ou mise à jour, est donnée par l'équation 1.17:

$$\begin{aligned} z^t &= Tz^{t-1} + g\ f_{image}^{t-1}(z^{t-1}) \\ T &= (I - t\ A) \end{aligned} \qquad \textbf{(3.11)}$$

où F_{image} est le vecteur des forces de l'image appliqué aux noeuds du contour.

$$f_{image}(x, y) = -\nabla P_{image}(x, y) \qquad \textbf{(3.12)}$$

$P_{image}(x,y)$ est un potentiel de l'image possédant un minimum sur le contour de l'objet but. Plusieurs modèles de potentiel peuvent être utilisés selon l'application en question. Afin d'amener l'équation 3.11 à une forme semblable à celle de l'équation 3.8, la fonction P_{image} sera prise comme une image contenant des fonctions impulsionnelles (réponses d'un filtre d'égalisation) localisées aux arêtes détectés par un autre algorithme.

$$P_{image}(x, y) = -E(x, y) * K(x, y) \qquad \textbf{(3.13)}$$

où $K(x,y)$ est la réponse impulsionnelle d'un filtre d'égalisation autour de (x,y).

Sachant que

$$E(x, y) = \sum_k d\,(x - p_{x_k}\ ,\ y - p_{y_k})$$

$p_k = [\ p_{xk}\ ,\ p_{yk}\ ]^T$ est la position de la $k^{\text{ième}}$ arête.

l'équation (3.12) devient:

$$f_{image}(x, y) = E(x, y) * \nabla K(x, y) \qquad \textbf{(3.14)}$$

Donc en remplaçant pour l'expression de E(x,y),

$$f_{image}(x,y) = \sum_k \nabla K(x - p_{x_k}, y - p_{y_k}) \qquad \textbf{(3.15)}$$

Vu que $\nabla K(z)$ tend vers zéro quand $\|z\|$ augmente, seulement les arêtes appartenant au voisinage d'un noeud du *snake* vont l'attirer. La forme de ce voisinage dépend de la réponse impulsionnelle du filtre *K*, un choix typique du filtre est une fonction gaussienne 2D. Si on choisit une forme parabolique tronquée de rayon D, la fonction *K* s'exprime comme suit:

$$K(x,y) = \begin{cases} \frac{1}{2}\left[D^2 - (x^2 + y^2)\right] & si \;\; x^2 + y^2 \le D \\ 0 & si\, non \end{cases} \qquad \textbf{(3.16)}$$

$$f_{image}(x,y) = \sum_{(x_k, y_k) \in R_{x,y}} \begin{bmatrix} p_{x_k} - x \\ p_{y_k} - y \end{bmatrix} \qquad \textbf{(3.17)}$$

La sommation se fait sur toutes les arêtes à l'intérieur du cercle avoisinant chaque noeud (voir la figure 3.5)

$$R_{x,y} = \left\{(u,v);\, (u-x)^2 + (v-y)^2 \le D^2\right\} \qquad \textbf{(3.18)}$$

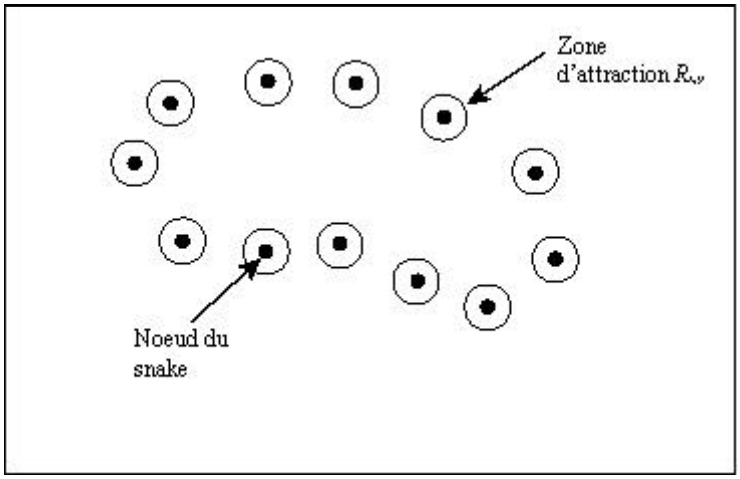


Figure3.5: Cercle $R_{x,y}$ autour de chaque noeud

En utilisant la notation vectorielle telle qu'établie dans la section 3.1, on obtient que:

$$f_{image}^{t}(z)=\sum_{z\in R_{x,y}}\left(p^{t}-z^{t}\right) \qquad \textbf{(3.19)}$$

et

$$z^{t+1}=(I-gA)z^{t}+g\ f_{image}^{t} \qquad \textbf{(3.20)}$$

Deux différences doivent être notées à ce point-ci. D'abord, la matrice *A*, telle que définie à l'équation 3.8 dépend du temps et doit donc être calculée à chaque époque. La matrice *A* de l'équation 3.20 est déterminée par les paramètres de rigidité et de flexibilité associés au modèle physique du contour actif. De plus, la matrice *B* de l'équation 3.8 n'est pas diagonale. Elle ne le serait que pour le cas particulier où le voisinage serait égal à 0.

CHAPITRE 4.

MODIFICATIONS À L'ALGORITHME DE KOHONEN

Maintenant muni d'une nouvelle perspective sur l'algorithme de Kohonen, il est plus facile de le modifier. Dans le présent chapitre, deux types de modifications seront présentées. Ces modifications ont été apportées par l'auteur afin d'explorer et d'étendre les capacités de l'algorithme de Kohonen dans sa version classique. En premier lieu, un changement dans la façon de calculer les centres de masse est susceptible d'améliorer la performance de l'algorithme dans certaines situations où l'information permettant de reconstituer les contours n'est pas complète. L'autre modification consiste à tracer la voie vers une intégration de l'opération de niveau intermédiaire qu'effectue l'algorithme de Kohonen à l'intérieur d'un système de vision complet. Un tel système ajouterait un niveau de traitement de plus haut niveau qui ne sera pas implanté et sur lequel certaines hypothèses seront émises.

4.1 Calcul des centres de masse

Présenté sous la forme de l'équation 3.7, il est maintenant clair que l'algorithme de Kohonen pourrait tirer un plus grand avantage de l'information se trouvant dans l'image. Rappelons d'abord que, pour appliquer l'algorithme de détection de contour, une convolution est appliquée à l'image d'intérêt à l'aide d'un filtre de Sobel. Les gradients ainsi trouvé sont binarisés de telle sorte que seulement les gradients maxima sont représentés dans l'ensemble des vecteurs d'apprentissage.

L'apprentissage du contour initial s'effectue en découpant le plan image en cellules de Voronoï puis en déplaçant chaque neurone vers le centre de masse des vecteurs d'apprentissage se trouvant dans cette cellule.

Le centre de masse est ici calculé à partir de données binaires valant 1 si une arête est présente et 0 autrement. Ce processus a donc deux implications directes pouvant affecter la performance de l'algorithme. D'abord, certains gradients de faible amplitude sont ignorés du fait qu'il ne sont pas sélectionnés lors de la binarisation de l'image. Ensuite, chacun des gradients sélectionnés possède le même poids dans la détermination du centre de masse de la cellule de Voronoï.

La principale modification proposée ici découle du fait que la notion de centre de masse est ici prise en son sens le plus strict et étendu en conséquence avec plus de subtilité. Supposons maintenant que les gradients générés par l'application du filtre de Sobel sont tous conservés tels quels au lieu d'être binarisés. Reprenons l'équation 3.4 pour le calcul des centres de masse de chacune des cellules:

$$x_n^t = \frac{1}{m_n^t} \sum_{i \in V_n^t} p_i$$

La somme s'effectue sur toute les positions p_i des vecteurs d'apprentissage. Pour appliquer la nouvelle définition du centre de masse, chaque position sera ajustée par un facteur de pondération égal au module du gradient calculé à cette position par le filtre de Sobel. L'équation résultante s'exprime donc comme suit:

$$x_n^t = \frac{1}{m_n^t} \sum_{i \in V_n^t} M_i p_i \qquad \textbf{(4.1)}$$

où M_I est le module du gradient calculé par l'opérateur de Sobel tel que:

$$M_i = \sqrt{s_x^2 + s_y^2} \qquad \textbf{(4.2)}$$

Comme tous les opérateurs de gradient, s_x et s_x peuvent être implantés en utilisant des masques de convolutions:

$$s_x = \begin{vmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{vmatrix} \quad et \quad s_y = \begin{vmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{vmatrix} \qquad \textbf{(4.3)}$$

Pour des raisons pratiques, il est possible d'appliquer une opération de seuillage sur l'image convoluée. Cette opération aurait pour but d'éliminer du calcul du centre de masse une grande quantité de vecteurs d'entrée de faible gradient dont la contribution serait substantielle mais dont la signification ne justifierait pas un tel biais du résultat. En effet, l'automatisation du procédé étant limitée, un utilisateur de ce système devra déterminer une région d'intérêt qui limitera l'action de l'algorithme aux gradient se trouvant à l'intérieur de cette région. Cette région étant arbitraire, elle pourrait inclure une grande quantité de gradients faibles pouvant affecter le contour final d'une façon tout aussi arbitraire. En appliquant seulement l'opération de seuillage, et non celle de binarisation, seuls les gradients les plus significatifs sont gardés ainsi que leur poids respectif.

4.2 Intégration de l'algorithme à un système de vision

Puisque le module de haut niveau ne sera pas implanté, l'interaction proposée ici du module de niveau intermédiaire avec un module de haut niveau impose qu'une hypothèse soit faite en ce qui concerne les capacités du module de plus haut niveau. Nous supposerons qu'il est possible d'évaluer la probabilité qu'une arête particulière appartienne au contour de l'objet. Les critères et les règles possibles ne sont pas explicitées ici et pour les fins d'expérimentation, un utilisateur génèrera un retour d'information artificiellement.

L'algorithme spécifiant l'interaction entre les modules de haut niveau et ceux de niveau intermédiaire peut être décrit comme suit:

1. Une région d'intérêt ainsi qu'un contour initial sont sélectionnéss par l'utilisateur.
2. L'apprentissage du réseau est effectué selon la modification apportée à la section 5.1.

3. Une fois l'apprentissage terminé, un premier contour est généré.
4. Le contour trouvé est traité à un niveau supérieur afin de trouver des arêtes ayant une grande possibilité de faire partie du contour de l'objet recherché. Ces arêtes sont identifiées par leur position dans le plan bidimensionnel de l'image et retournées au niveau plus bas de traitement.
5. Les unités du contour existant étant les plus rapprochées des arêtes retournées par le niveau supérieur sont identifiées. Les unités ainsi identifiées prennent ensuite la même position que ces arêtes.
6. L'apprentissage recommence avec les paramètres suivants: le contour initial est maintenant le contour final de l'étape précédente à la seule différence que les arêtes retournées par le module de haut niveau ont remplacé les unités du contour se trouvant les plus près. Ces unités ne sont plus mises à jour lors de la nouvelle session d'apprentissage.
7. Il est possible de retourner à l'étape 3 et de répéter jusqu'à ce que le critère d'arrêt soit satisfait. Un exemple de critère d'arrêt pourrait être l'absence de nouvelles arêtes possédant une probabilité suffisante d'appartenir au contour.

Afin de maintenir en place les unités jugées comme appartenant au contour, il est nécessaire d'apporter une modification à l'implantation de l'algorithme. Les matrice A et B (respectivement définies aux équations 3.9 et 3.10), semblent offrir la meilleure possibilité de modification qui n'affectera pas substantiellement la continuité du contour. Rappelons d'abord que la matrice A est ici considérée comme un matrice de régularisation et la matrice B, comme une matrice d'adaptation ou de déplacement des poids. Le comportement désiré pour la matrice est tout simplement de continuer à régulariser les unités du contour avoisinant un point fixe sans affecter ce point fixe. Pour un point fixe, la solution sera:

$$\left(I_i - gA_i^t\right) z_{i^t}^t = z_i^t \quad \textit{pour un i donné} \qquad \textbf{(4.4)}$$

Donc,

$$I_i - gA_i^t = 1 \qquad \textbf{(4.5)}$$

et

$$gA_i^t = 0 \qquad \textbf{(4.6)}$$

Au niveau de l'implantation, la fonction γ prend la forme d'une constante à chaque pas d'itération plutôt que celle d'un vecteur. Il est donc plus facile de mettre à zéro toutes les rangées de A ayant un indice *i* désiré.

La matrice B étant une matrice de changement des poids, l'effet désiré est bien sûr de garder les points fixes à la même place donc, d'annuler tout déplacement. Ceci résulte en la contrainte suivante:

$$g\,B_i^t\left(x_i^t - z_i^t\right) = 0 \;\; \textbf{\textit{pour un i donné}} \qquad \textbf{(4.7)}$$

La solution est triviale. La $i^{\text{ème}}$ ligne de la matrice B doit être égale à zéro pour chaque indice *i* désiré. Concrètement, les matrice A et B seront définies de la façon suivante.

$$a_{ij} = \begin{cases} \sum_{\substack{k=j-b \\ k\neq j}}^{k=j+b} m_k^V & i = j \\ -m_j^V & 0 \le |i-j| \le b \\ 0, & |i-j| > b \\ 0, & si\ i = indice\ d'un\ point\ fixe \end{cases} \qquad \textbf{(4.8)}$$

Explicitement et dans le cas où β=1, cette matrice aurait habituellement la forme suivante:

$$\mathbf{A} = \begin{bmatrix} m_M^V + m_2^V & -m_2^V & 0 & \dots & -m_M^V \\ -m_1^V & m_1^V + m_3^V & -m_3^V & \dots & 0 \\ 0 & -m_2^V & m_2^V + m_4^V & -m_4^V & \dots \\ \dots & \dots & \dots & \dots & \dots \\ -m_1^V & \dots & \dots & -m_4^V & m_1^V + m_3^V \end{bmatrix} \qquad \textbf{(4.9)}$$

Supposons que l'indice du point fixe i=2. La matrice A, après retour d'information, s'exprimerait alors comme suit:

$$\mathbf{A} = \begin{bmatrix} m_M^V + m_2^V & -m_2^V & 0 & \dots & -m_M^V \\ 0 & 0 & 0 & \dots & 0 \\ 0 & -m_2^V & m_2^V + m_4^V & -m_4^V & \dots \\ \dots & \dots & \dots & \dots & \dots \\ -m_1^V & \dots & \dots & -m_4^V & m_1^V + m_3^V \end{bmatrix} \qquad \textbf{(4.10)}$$

Dans cette matrice, tous les éléments a_{ij} dont les indices i=2 correspondent aux indices des points fixes sont annulés. Les neurones avoisinants ne sont pas affectés directement par ce changement. Ils seront affectés lors du processus d'apprentissage par le fait que la cellule de Voronoï du point fixe sera toujours calculée à partir du même point et l'arrangement des neurones ne suivra pas la même dynamique d'évolution.

La même stratégie est implanté pour la matrice B qui s'exprime comme suit:

$$b_{ij} = \begin{cases} m_j^V & |i-j| \le b \\ 0, & \textit{autrement} \end{cases} \qquad \textbf{(4.11)}$$

Explicitement et dans le cas où β=1, cette matrice aurait habituellement la forme suivante:

$$\mathbf{B} = \begin{bmatrix} m_1^V & m_2^V & \dots & \dots & \dots & m_M^V \\ m_1^V & m_2^V & m_3^V & \dots & \dots & 0 \\ 0 & m_2^V & m_3^V & m_1^V & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & 0 \\ m_1^V & \dots & \dots & \dots & m_{M-1}^V & m_M^V \end{bmatrix} \qquad \textbf{(4.12)}$$

Supposons encore que l'indice du point fixe i=2. La matrice B, après retour d'information, s'exprimerait alors comme suit:

$$\mathbf{B} = \begin{bmatrix} m_1^V & m_2^V & \dots & \dots & \dots & m_M^V \\ 0 & 0 & 0 & \dots & \dots & 0 \\ 0 & m_2^V & m_3^V & m_1^V & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & 0 \\ m_1^V & & \dots & \dots & m_{M-1}^V & m_M^V \end{bmatrix} \qquad \textbf{(4.13)}$$

Dans cette matrice, tous les éléments b_{ij} dont les indices i=2 correspondent aux indices des points fixes sont annulés de la même façon que précédemment.

Le prochain chapitre démontrera les effets des changements apportés à l'algorithme.

CHAPITRE 5.

DISCUSSION ET INTERPRÉTATION DES RÉSULTATS

L'étude de l'algorithme de Kohonen présentée ici se divise en six (6) parties. La première consiste à explorer le comportement de l'algorithme de base en manipulant les paramètres de voisinage topologique (β), le nombre de neurones et le type d'objet qui doit être détecté. D'une façon générale dans cette étude, les paramètres des algorithmes testés ont été choisis après quelques essais de façon à obtenir un contour de bonne qualité à l'intérieur de délais raisonnables. Un changement dans ces paramètres peut influencer le temps de calcul, la qualité du contour et la convergence des algorithmes. L'utilisation d'images synthétiques (carré, croix) permet de contrôler les paramètres de l'image. Deux raisons ont guidé le choix des images. La première voulait que les algorithmes soient testés sur des images dont on connaît bien le contour et dont les gradients se trouvent à l'endroit de ce contour. La seconde permettait de tester certains paramètres de l'objet: coins (carré), coins concaves et convexes (croix) ainsi qu'une zone de gradients plus faibles (croix). Ceci permet d'exhiber les forces et les faiblesses de l'algorithme de façon à proposer certaines améliorations.

La deuxième partie consiste à comparer l'aptitude de l'algorithme de Kohonen et du *snake* à capturer la topologie globale du contour. Les différences à ce niveau sont notées. L'algorithme de Kohonen modifié de façon à prendre les gradients de l'image comme source additionnelle d'information est ensuite implanté et comparé avec l'algorithme classique et le *snake* dans la troisième partie. La quatrième partie consiste à évaluer la

robustesse de l'algorithme ainsi modifié lorsqu'il est appliqué à une image bruitée. Le mécanisme par lequel la détection du contour pourra interagir avec un module de traitement de plus haut niveau est implanté dans la cinquième partie. Enfin, l'algorithme de Kohonen dans ses versions classique et modifiée ainsi que le *snake* ont été appliquées à une image tomographique de coeur humain.

5.1 Résultats de détection de contour simple

Le premier résultat est généré à partir d'un problème simple. Il s'agit ici de trouver le contour d'un carré se trouvant dans une image binaire. L'algorithme utilisé ici sera l'algorithme de Kohonen classique. Les exemples présentés dans cette section posent les bases du fonctionnement du réseau et offrent des résultats élémentaires. Le contour initial a été choisi assez près du contour cherché. Le paramètre d'apprentissage γ, influençant la grandeur des déplacements des unités du réseau, a été fixée à 0,003 après quelques essais. Ce paramètre s'est avéré assez efficace dans les cas qui nous concernent au sens où il procure une certaine justesse du contour trouvé tout en limitant le temps de calcul dans un délai raisonable. Un voisinage de 1 permet une grande flexibilité dans la recherche du contour. Étant donné que le contour initial se trouve près du contour recherché, la capacité d'organisation globale que procure un grand voisinage ne sera pas exploitée. Le critère d'arrêt utilisé ici est celui que la somme des déplacements de tous les neurones soit inférieur à 0,1 pixel. Un déplacement est calculé comme étant la distance en pixel entre les positions d'un neurone avant et après la mise à jour du contour. Le résultat de la simulation est montré à la figure 5.1.

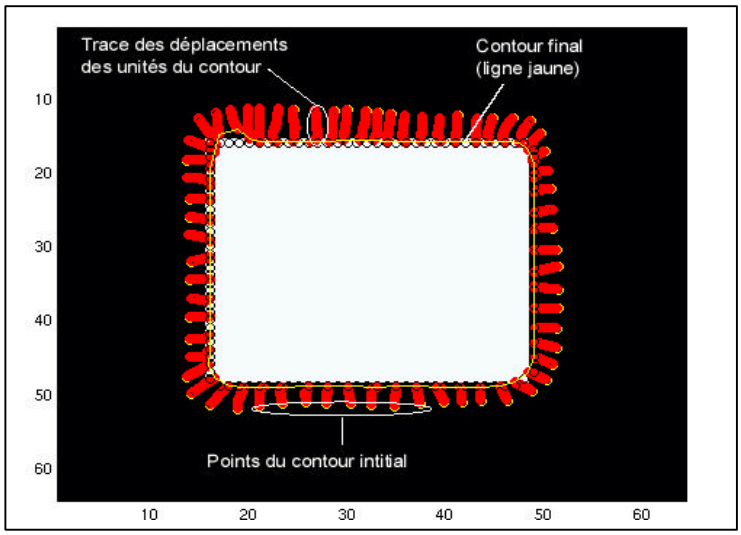


Figure 5.1: Simulation de détection de contour avec 64 neurones, γ= 0,003, β=1 et le nombre d'itérations est de 244.

On peut remarquer que le contour final épouse bien la forme recherchée d'une façon générale. Ce résultat n'a cependant été atteint qu'après 244 itérations. La succession des mises à jour du contour comporte beaucoup d'étapes qui auraient pu être évitées. Cette observation tient du fait que les points de déplacement du contour sont tellement près les uns des autres qu'ils forment un ligne presque continue entre le contour initial et le contour final. Certains pas d'adaptation peuvent être remplacés par des pas plus grands au début et diminuant rapidement par la suite. Ceci est possible en remplaçant la constante d'apprentissage γ par une fonction qui diminue en fonction du temps. Cette fonction peut être composée par exemple de la somme d'un terme exponentiel diminuant plus ou moins rapidement et d'un terme constant exprimant le régime permanent. La valeur du régime permanent doit être suffisamment petite puisqu'une valeur trop grande pourrait empêcher la convergence.

Le paramètre d'apprentissage fixe est maintenant remplacé par le paramètre fonctionnel $g = 0{,}5e^{-t} + 0{,}003$. L'évolution de ce paramètre d'apprentissage est illustré à la figure 5.2. La première itération prendra un paramètre d'apprentissage d'environ 0,19. Cette valeur est réduite de plus de la moitié à la 2ème itération seulement et le régime

permanent de 0,003 est atteint aux environ de la 7ème itération. La rapiditié de convergence est réglée à l'aide du paramètre multipliant l'exposant -*t* (la valeur 1 est utilisée comme paramètre ici).

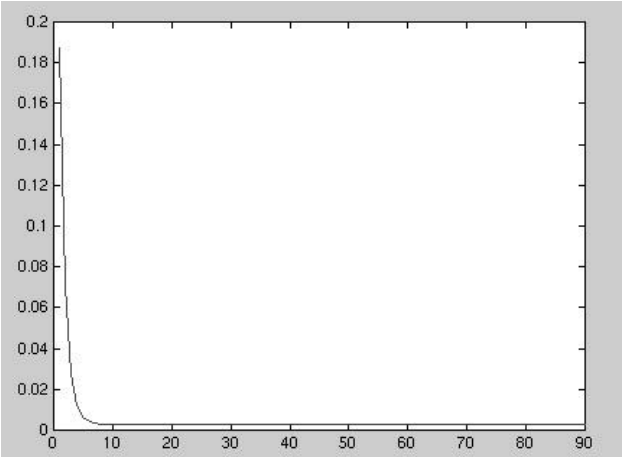


Figure 5.2: Fonction d'apprentissage $g = 0{,}5e^{-t} + 0{,}003$

Le résultat obtenu à l'aide de cette fonction est illustré à la figure 5.3.

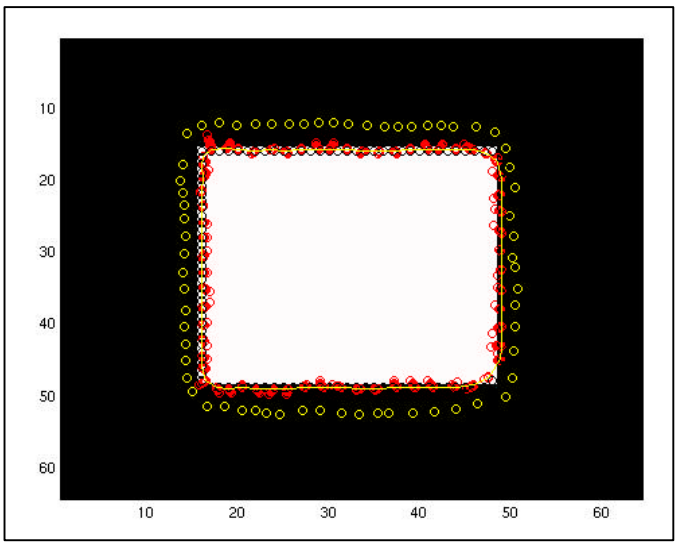


Figure 5.3: Simulation de détection de contour avec 64 neurones, $g = 0{,}5e^{-t} + 0{,}003$, β=1 et le nombre d'itérations est de 90.

Le nombre de neurones, le paramètre de voisinage ainsi que le contour initial sont les mêmes qu'à la figure 5.1. La modification du paramètre d'apprentissage a donc permis de réduire de plus de la moitié le nombre d'itérations nécessaires avant la convergence de l'algorithme. Par inspection visuelle, on note que le résultat est essentiellement le même qu'à la figure 5.1 sauf pour le coin supérieur gauche où le contour s'est mieux fixé au contour du carré.

Intuitivement, on pourrait imaginer que si l'on accroît le nombre d'unités du réseau, l'agrandissement du voisinage ne devrait que très peu affecter la capacité de l'algorithme à épouser des formes discontinues telles que les coins du carré. En augmentant le nombre de neurones à 170 (près de 3 fois celui des exemples précédents) et en fixant le voisinage à β=4, on obtient un résultat tel qu'illustré à la figure 5.4.

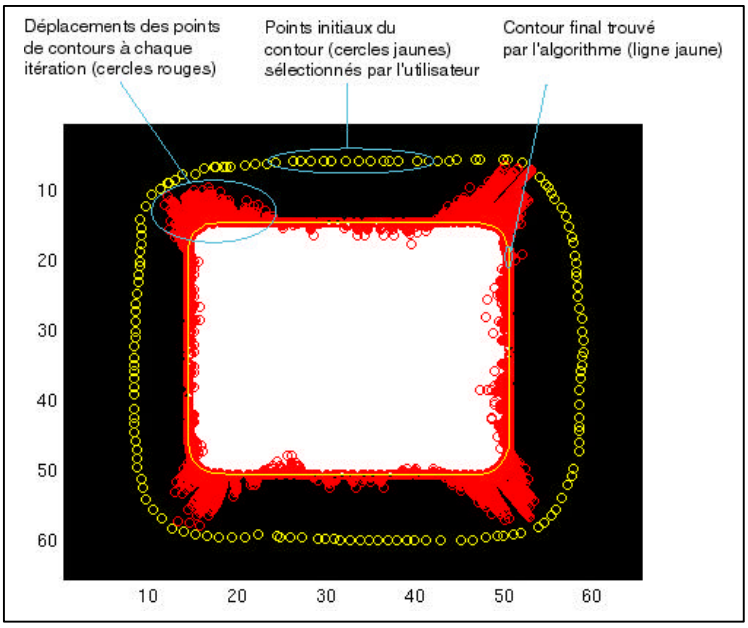


Figure 5.4: Simulation de détection de contour avec 149 neurones, $g = 0{,}03e^{-(0{,}9t)} + 0{,}003$, β=4 et le nombre d'itérations est de 359.

Le processus de convergence est ici plus complexe car il implique plus d'unités. Cela explique en partie pourquoi le nombre d'itérations est plus élevé que pour l'essai présenté à la figure 5.3. Il est cependant moins élevé que pour l'essai de la figure 5.1 ce

qui donne un indice quant à l'efficacité du paramètre d'apprentissage. On remarque que les coins sont bien représentés malgré un voisinage de 4. Il est cependant possible de distinguer un arrondissement plus marqué des contours trouvés en ces endroits. L'aspect de la régularisation inclus dans la matrice *A* (équation 3.9) est senti subtilement dans cet essai. Le voisinage plus grand impose une cohérence plus grande de chacunes des unités avec leur voisines mais un nombre plus grand de neurones tend à minimiser cet effet. Le voisinage topologique joue un rôle important dans la régularisation du contour. Afin d'illustrer l'effet du voisinage dans la capture des traits globaux de la forme recherchée, la capacité de l'algorithme est évaluée dans la prochaine section.

5.2 Capture de la topologie globale du contour

Cette évaluation consiste essentiellement à laisser l'algorithme trouver le contour du carré avec, comme contour initial, une boucle en forme de "8". La difficulté réside dans le fait que les deux contours possèdent une topologie différente. Le carré ne possède qu'une seule région intérieure. Par contre, le contour initial en possède deux.

Pour ce faire, un voisinage de 1 est d'abord imposé au réseau. Il s'agit encore une fois d'un voisinage offrant beaucoup de flexibilité. Le résultat est présenté à la figure 5.5.

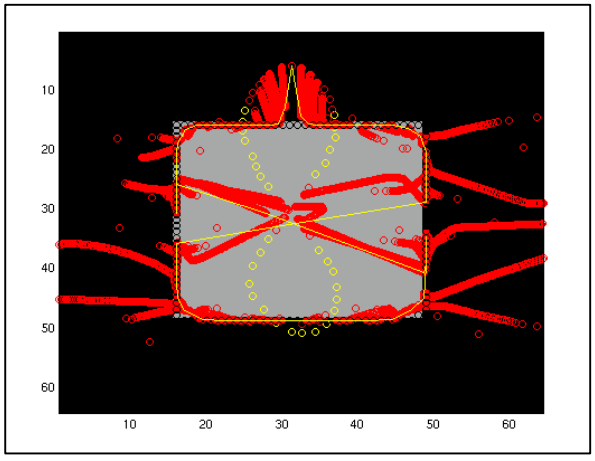


Figure 5.5: Simulation de détection de contour avec 53 neurones, $g = 0{,}5e^{-t} + 0{,}003$, β=1 et le nombre d'itérations est de 630. Le contour initial est en forme de 8.

On peut constater que le réseau a, globalement, réussi à capturer la forme générale de la figure. Cependant, il n'a pas réussi à défaire le 8 pour épouser parfaitement le contour. En effet, par la trace des déplacement, on peut constater que certains points se retrouvent au centre du carré, là où il n'y a pas d'arêtes. Cette solution au problème de minimisation d'énergie est valide au sens où elle constitue un minimum local. Cependant, l'utilisation de l'algorithme dans le but de trouver un contour impose une contrainte de "continuité" puisque le contour trouvé doit correspondre au contour perçu par un être humain. En l'occurence dans le cas présent, le contour devrait évidemment représenter un carré.

Une autre tentative a été effectuée dans le but de résoudre le problème en augmentant le voisinage à 4 tout en conservant les autres conditions constantes (contour initial, nombre de neurones et paramètre d'apprentissage). Le résultat est illustré à la figure 5.6. Cette fois-ci, on peut noter que le réseau a formé un contour plus plausible. Le fait que la topologie du contour initial, notamment sa boucle, soit défaite lors de l'apprentissage est particulièrement intéressant. Le voisinage topologique semble avoir l'effet de guider immédiatement l'évolution des unités de contour vers un minimum local différent de celui trouvé en utilisant un voisinage topologique β=1. Notons aussi la réduction drastique du nombre d'itérations.

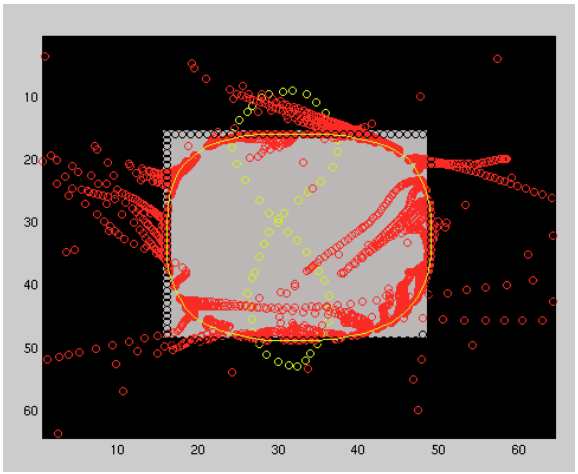
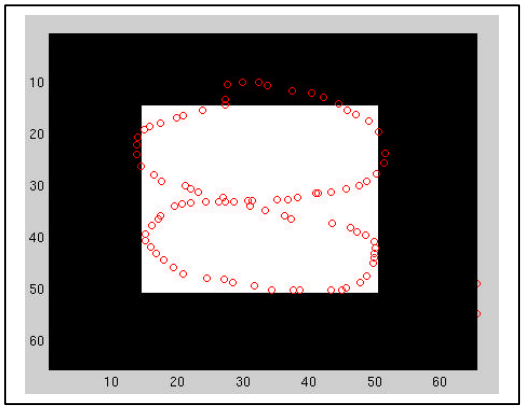


Figure 5.6: Simulation de détection de contour avec 52 neurones, $g = 0{,}5e^{-t} + 0{,}003$, β=4 et le nombre d'itérations est de 144. Le contour initial est en forme de 8.

Quelque tests ont cependant démontré que cette solution, nommément celle d'utiliser un voisinage topologique β=4, ne trouve pas le contour à tout coup. Cependant en augmentant le rayon du voisinage topologique dans une proportion plus grande, par rapport au nombre d'unités du contour, au début de l'apprentissage donnne un résultat beaucoup plus stable. Le rapport rayon/(no. d'unités) était de 1/13 (ou 4/52) dans l'exemple précédent. En augmentant ce rapport à près de 1/4 (20/86 exactement), l'organisation globale du contour s'en voit améliorée et ce, même avec un contour initial plus difficile à lisser (voir figure 5.7).



Figurze 5.7: Contour initial en forme de 8 comme point de départ à la détection du contour du carré.

Il est possible de noter qu'à l'image de la figure 5.6, les coins du carré sont arrondis. Ceci est dû au relativement grand voisinage topologique qui a été utilisé. L'algorithme n'est cependant pas contraint à une valeur fixe de contour pendant tout l'apprentissage: il est possible de varier le voisinage en cours d'apprentissage. Afin de tirer avantage à la fois des capacités d'organisation globale et de détection des détails plus fins, le rayon du voisinage topologique a été diminué graduellement lors de l'apprentissage. La diminution du rayon s'est faite d'après la valeur de la variation totale des déplacements des unités du contour (en nombre de pixels).

La séquence d'attribution du voisinage topologique (β) en fonction du déplacement total (Variation) se détaille comme suit:

1- β=20 si Variation > 20

2- β=4 si 3 < Variation <20

3- β=3 si 1 < Variation < 3

4- β=2 si 0,8 < Variation < 1

5- β=1 si 0,4 < Variation < 0,8

Le résultat obtenu en utilisant ces paramètres est présenté à la figure 5.8.

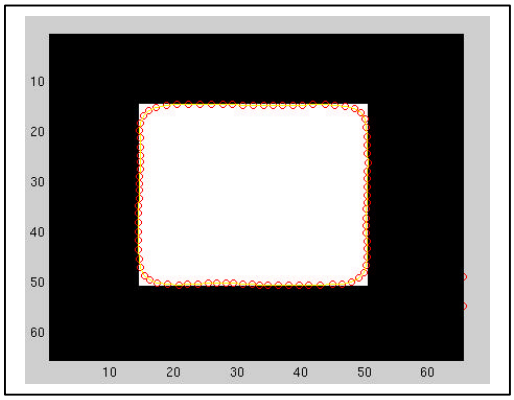


Figure 5.8: Simulation de détection de contour avec 86 neurones, $g = 0{,}5e^{-t} + 0{,}003$, β varie entre 1 et 20. Le nombre d'itérations est de 195.

La capacité du réseau de Kohonen à défaire la topologie du contour initial est une caractéristique émanant de l'algorithme et démontre une différence fondamentale avec l'algorithme du *snake*. La figure 5.9 montre le résultat d'une simulation avec l'algorithme du *snake* sur un contour initial en forme de 8.

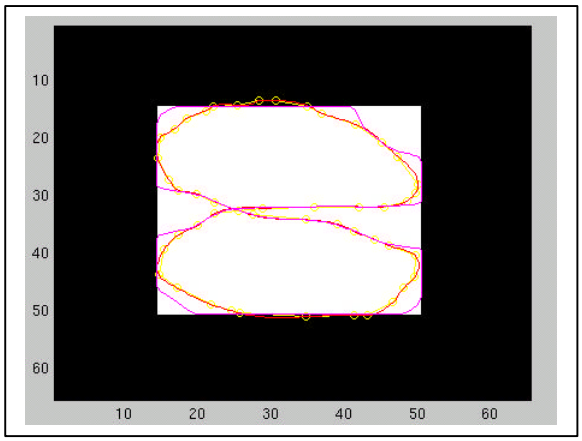


Figure 5.9: Contour final (en mauve) à partir d'un contour initial (en jaune) en 8 avec le *snake*.

Afin de défaire le croisement du contour initial avec l'algorithme de Kohonen, la grandeur du voisinage topologique a été augmentée. Cette forme de régulation, transposée au *snake*, se traduit par une plus grande rigidité du modèle physique. L'implantation du *snake* utilisé ici a un paramètre de rigidité déterminé uniformément pour tout le contour. En augmentant le paramètre de rigidité du contour à une valeur maximale permettant une convergence ne suffit pas pour éliminer le croisement du contour initial sur lui-même. Au-delà d'une valeur-seuil de rigidité, le modèle physique se comporte comme une barre très rigide et n'accepte que des mouvements restreints. La différence entre les algorithmes se situe au niveau de la région d'influence du mécanisme de régulation. L'algorithme de Kohonen permet une régulation globale permettant de trouver une solution d'énergie minimale à l'intérieur de chacune des cellules de Voronoï et pour l'ensemble des cellules de Voronoï. Aussi, la matrice de régulation est recalculée après chacune des itérations dans la recherche d'une solution optimale, ce qui n'est pas le cas pour le *snake*. Le *snake* par contre possède une constante de rigidité déterminant le comportement dynamique du contour entier en plus de force agissant dans un rayon restreint autour de chacun des points du contour. Une attraction locale d'un point du contour n'aura donc que peu d'effet sur la rigidité puisque le contour est ainsi difficilement modifiable.

5.3 Algorithme modifié: comparaison avec l'algorithme classique

Le prochain exemple sert à mettre en évidence l'utilité de modifier l'algorithme de Kohonen de telle sorte que les gradients continus et non-binarisés soient considérés. L'image sur laquelle l'algorithme sera appliqué consiste en une croix dont le niveau de gris est maximal (i.e. blanc) partout à l'exception du coin concave supérieur gauche (voir figure 5.10). Les niveaux de gris des pixels de ce coin ont été intentionnellement fixés de façon à créer une progression plus douce du noir au blanc. Posons une valeur de noir égale à 0 et une valeur de blanc égale à 1. Pour ces valeurs, la progression comporte trois lignes de 2 pixels de large possédant des valeurs respectives de 0,25, 0,50 et 0,75.

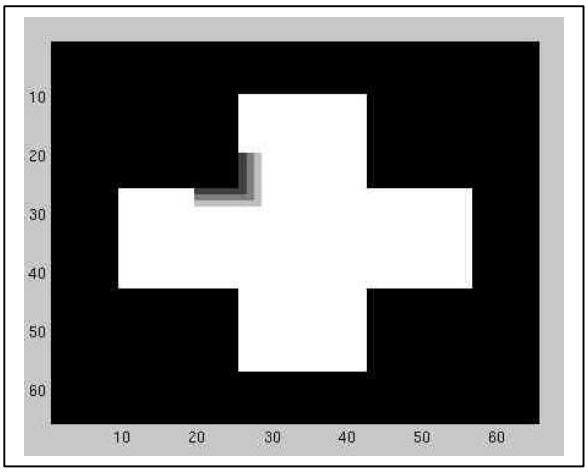


Figure 5.10: Image initiale dont le coin concave supérieur gauche a été modifié.

L'hypothèse ayant servi à la construction de cette image se résume comme suit. Le coin comporte des pixels voisins exhibant une augmentation plus lente des niveaux de gris que ceux se trouvant ailleurs sur le contour de la croix. Il ne fait aucun doute que cette information est cruciale à la détection complète de la croix. L'application d'un filtre de Sobel et la binarisation des gradients ainsi trouvés ne génèreront pas d'arêtes puisque ceux-ci sont trop faibles. Cette région ne procurera donc aucune information à l'algorithme classique permettant au contour de se fixer correctement au coin. En revanche, l'algorithme modifié tiendra compte de ces gradients continus par la nature même de sa définition.

La différence entre l'information traitée dans chacun des cas est visible dès lors qu'on jette un coup d'oeil sur l'image-test après traitement avec un filtre de Sobel. Ces images sont présentées à la figure 5.11 ci-dessous.

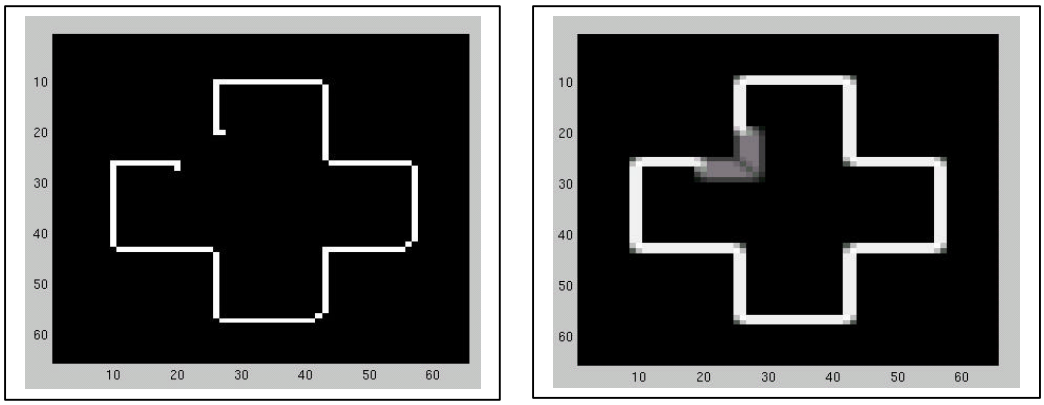


Figure: 5.11: Images de la croix traitée avec un filtre de Sobel. En a) les gradients ont été seuillés et binarisés tandis qu'en b) tous les gradients non-nuls ont été gardés.

Les traitement différents accordés à une image résultent entre autre en un nombre différents de vecteurs d'apprentissage. La figure 5.11a compte 169 vecteurs d'apprentissage tandis que la figure 5.11b en compte 433. Ceci devrait normalement impliquer que l'algorithme classique soit plus rapide que la version modifiée. Ceci n'est pas vrai dans tous les cas puisque la rapidité de l'algorithme dépend aussi du paramètre d'apprentissage. L'explication de ce comportement sera élaboré dans la discussion en relation avec le tableau 5.1.

Les résultats montrés aux figures 5.12 à 5.14 ont tous été obtenus à l'aide du même contour initial contenant 100 unités. Le seul paramètre qui a été manipulé est celui du voisinage topologique qui varie de $\beta=1$ à $\beta=3$. Les images montrées en partie a) de chacune des figures ont été générées à partir de l'algorithme classique. Les images

montrées en partie b) de chacune des figures ont quant à elles été générées à partir de l'algorithme modifié.

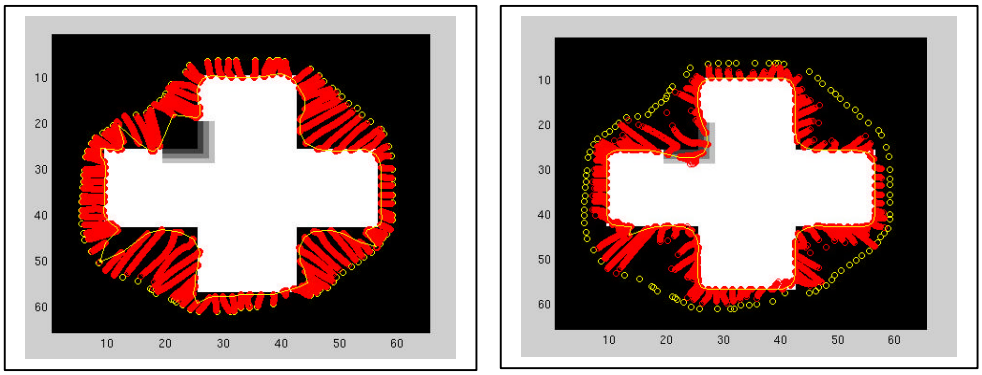


Figure: 5.12: Détection de contour à l'aide des paramètres suivants: β=1, $g = 0{,}003e^{(-i)} + 0{,}0003$ avec 100 unités dans le contour en utilisant a) l'algorithme de Kohonen classique et b) l'algorithme de Kohonen modifié.

En observant les images de la figure 5.12 a) et b) respectivement, il est possible de constater que le nombre de pas nécessaires à la détection du contour est de beaucoup plus grand pour la figure a) que pour la figure b). En effet, la trace laissée par les points intermédiaires se trouvant entre le contour initial (à l'extérieur) et le contour final (près du contour de la croix elle-même) est plus dense. De plus, la convergence de l'algorithme classique (figure 5.12a) s'est effectuée en 1638 itérations alors que celle de l'algorithme modifié s'est effectuée en 569 itérations. L'explication de ce comportement nécessite le rappel de l'équation 3.5:

$$z_k^t = z_k^{t-1} + g \sum_{n=k-b}^{k+b} m_n^{t-1}(x_n^{t-1} - z_k^{t-1}) \quad k = 0, \cdots, N-1$$

D'après cette équation, le changement dans la valeur du vecteur $\mathbf{z}^{t-1}$ consiste en l'expression:

$$\Delta z = g \sum_{n=k-b}^{k+b} m_n^{t-1}(x_n^{t-1} - z_k^{t-1}) \quad k = 0, \cdots, N-1$$

Trois facteurs influencent ici la grandeur du pas d'adaptation. Le premier est bien sûr la valeur de γ au temps = t-1. Le second est la valeur de μ représentant la "masse" totale des vecteurs d'apprentissage contenus dans une cellule de Voronoï. Le dernier facteur consiste enfin en la différence entre la position du centre de masse de chacune des cellules de Voronoï et la position de chacune des unités du réseau.

Dans ce cas-ci, le seul facteur qui diffère est la masse de chacune des cellules de Voronoï. La différence fondamentale entre les deux algorithmes consiste au nombre de vecteurs d'apprentissage utilisés. L'algorithme classique contient un nombre plus faible de vecteurs d'apprentisage étant donné que seules les arêtes possédant une valeur maximale sont retenues. L'algorithme modifié conserve toutes les valeurs de gradient (sauf les valeurs nulles). La somme des gradients contenus à l'intérieur d'une cellule de Voronoï est donc plus grande dans le cas de l'algorithme modifié que dans le cas de l'algorithme classique.

Cependant cette explication s'applique strictement si le nombre d'unités du réseau est le même. Dans le cas inverse où le nombre de vecteurs d'apprentissage est gardé constant et le nombre d'unités du réseau est changé sera posé plus loin afin de compléter cette illustration.

La qualité des contours trouvés diffère aussi pour les deux algorithmes. Le contour trouvé par l'algorithme classique exhibe des angles qui dévient du contour de l'objet (voir figure 5.13a). Par contre, l'algorithme modifié n'affiche qu'un seul de ces angles qui est, en comparaison, assez mineur (voir figure 5.13b).

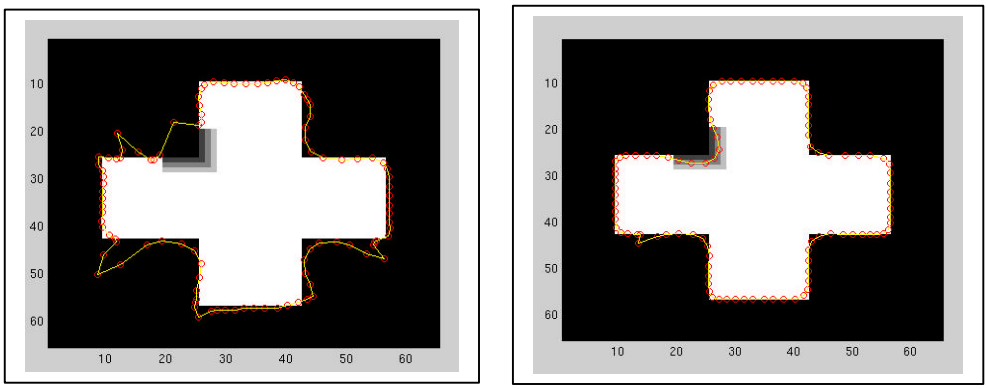


Figure: 5.13: Détection de contour à l'aide des paramètres suivants: β=1, $g = 0{,}003e^{(-i)} + 0{,}0003$ avec 100 unités dans le contour en utilisant a) l'algorithme de Kohonen classique et b) l'algorithme de Kohonen modifié.

Ceci vient encore de la différence dans le nombre de vecteurs d'apprentissage utilisés par les deux algorithmes. Dans la figure 5.13a, il est possible de constater que le neurone situé dans le coin inférieur gauche n'a subit que très peu d'adaptation. Certains autres neurones formant les angles indésirés ont soit subit moins de pas d'adaptation que ceux "collant" adéquatement au contour ou ces pas sont à chaque fois plus petit. Pendant l'apprentissage, les cellules de Voronoï sont recalculées après chaque pas. Il est possible que dès le départ ou au cours de l'évolution, certaines unités du réseau tombent à l'intérieur d'une cellule ne contenant que peu ou pas de vecteurs d'apprentissage. Si la cellule de Voronoï contient peu de vecteurs d'apprentissage, la masse de cette cellule sera faible et donc le pas d'apprentissage sera aussi faible. Si la masse de la cellule est nulle, l'unité du réseau ne subira aucun apprentissage. Un voisinage de β=1 contribue à l'adaptation de ces unités mais n'est pas toujours suffisant. En augmentant le nombre de vecteurs d'apprentissage, les chances (puisqu'on parle ici d'un processus stochastique) qu'une unité du réseau soit contenue dans une cellule de masse non-nulle sont plus grandes d'où un meilleur contour pour l'algorithme modifié.

Une autre différence s'affiche en ce qu'elle est le résultat de la motivation derrière les modifications apportées à l'algorithme. Le coin intérieur qui a été altéré par l'adoucissement de la transition du noir au blanc est mieux représenté par l'algorithme modifié (figure 5.13b) que par l'algorithme classique (figure 5.13a). Ceci vient confirmer l'hypothèse émise dans la section 5 quant à l'amélioration possible de l'algorithme parce que plus de pixels devraient être retenus pour l'apprentissage du contour. L'algorithme classique n'a pas détecté du tout ce coin. Il n'a d'ailleurs pas détecté d'autres parties de l'objet. À cet effet, d'autres démonstrations plus convaincantes seront montrées un peu plus loin. L'algorithme modifié a pour sa part généré un contour qui épouse bien la forme de ce coin. Cependant, le contour final tend même à se fixer un peu à l'intérieur du contour réel. Ceci tient à deux facteurs combinés. Premièrement, la définition même du centre de masse ξ calculé de la façon suivante:

$$x = \frac{\sum_{i=1}^{N} m_i \times d_i}{N}$$

où m_i est la masse (représentant la norme des gradients),

d_i est la distance à partir d'un point de référence et

N est le nombre total de vecteurs d'apprentissage.

D'après cette expression, non seulement la norme des gradients mais aussi leur position affectent le centre de masse.

Deuxièmement, à cause de l'opération appliquant le filtre de Sobel, les gradients ne se trouvent pas exactement à l'endroit du contour sur l'image mais un peu à côté. De plus, l'image a été modifiée artificiellement créant ainsi des gradients à un peu à l'intérieur de l'endroit où se serait normalement trouvé le contour de l'objet. Ceci implique qu'un groupe de gradients peut influencer le déplacement du contour un peu à côté du contour réel dans ces conditions.

Les images présentées à la figure 5.14 ont été générées à partir des mêmes conditions que précédemment sauf pour le voisinage topologique qui est maintenant β=2.

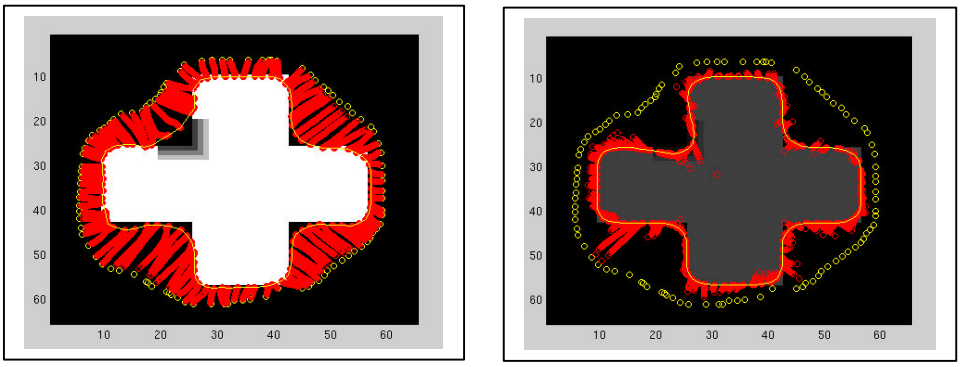


Figure: 5.14: Détection de contour à l'aide des paramètres suivants: β=2, $g = 0{,}003e^{(-i)} + 0{,}0003$ avec 100 unités dans le contour en utilisant a) l'algorithme de Kohonen classique et b) l'algorithme de Kohonen modifié.

Les images 5.14a et 5.14b montrent maintenant plus clairement l'effet des modifications de l'algorithme sur la détection du coin auparavant indétectable avec l'algorithme classique. Les images des figures 5.14a et 5.14b montrent un contour final qui est, de façon globale, plus près du contour réel que celui trouvé précédemment alors que le rayon était β=1. L'algorithme classique ne peut toujours pas s'agripper au coin qui a été altéré alors que l'algorithme modifié continue de bien performer à ce niveau. De plus, de la même façon que l'augmentation du rayon permet une plus grande cohérence globale du contour pour l'algorithme classique, elle génère un contour plus lisse en éliminant la bosse indésirable. De ce même fait, les angles du contour trouvé tendent à s'arrondir. Le contour offre un plus grande rigidité du fait qu'un plus grand nombre de voisins doivent s'adapter dépendant du pas d'adaptation d'un neurone particulier. La figure 5.15 montre le contour final en l'absence des points montrant l'évolution du contour.

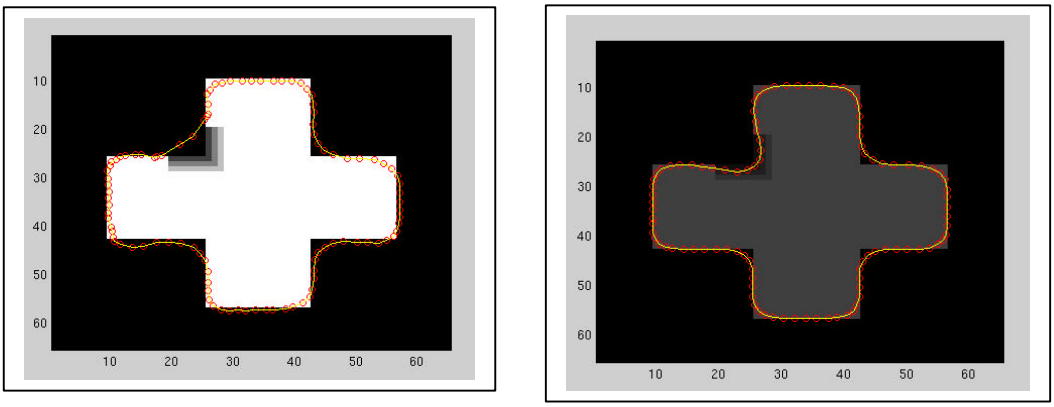


Figure: 5.15: Détection de contour à l'aide des paramètres suivants: β=2, $g = 0{,}003e^{(-i)} + 0{,}0003$ avec 100 unités dans le contour en utilisant a) l'algorithme de Kohonen classique et b) l'algorithme de Kohonen modifié.

Cet effet est plus clairement visible lorsqu'on agrandit encore le voisinage topologique pour l'amener à β=3. Dans l'exemple qui suit, tous les paramètres ont encore été gardés constants sauf le rayon du voisinage topologique. Les résultats sont présentés à la figure 5.16.

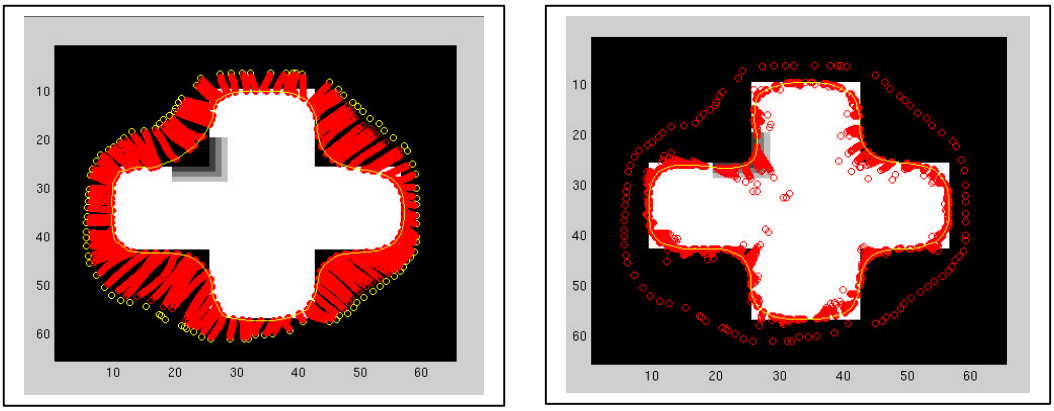


Figure: 5.16: Détection de contour à l'aide des paramètres suivants: β=3, $g = 0{,}003e^{(-i)} + 0{,}0003$ avec 100 unités dans le contour en utilisant a) l'algorithme de Kohonen classique et b) l'algorithme de Kohonen modifié.

L'algorithme a convergé en 929 itérations pour l'algorithme classique tandis qu'il n'a fallu que 223 itérations à l'algorithme modifié. Les images de la figure 5.17 montrent les détails du contours plus clairement.

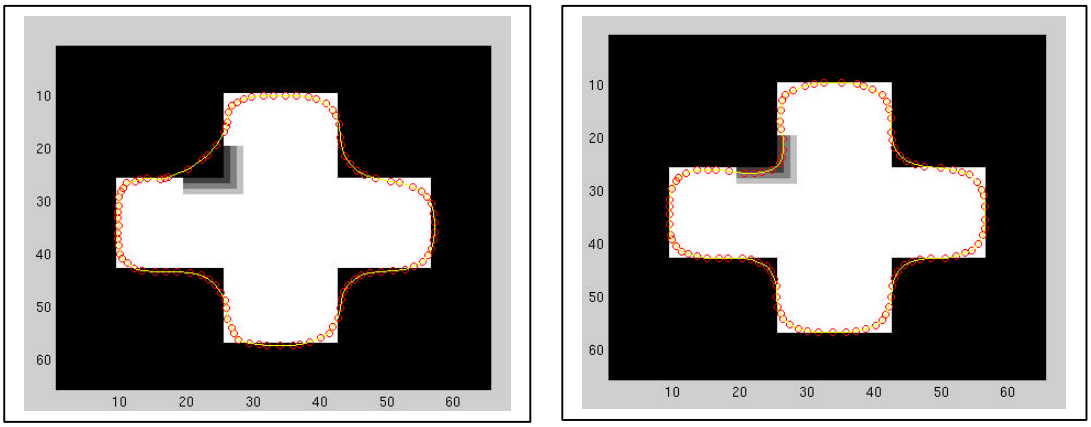


Figure: 5.17: Détection de contour à l'aide des paramètres suivants: β=3, $g = 0{,}003e^{(-i)} + 0{,}0003$ avec 100 unités dans le contour en utilisant a) l'algorithme de Kohonen classique et b) l'algorithme de Kohonen modifié.

Deux observations peuvent être faites en comparant les images de la figure 5.17 avec celles de la figure 5.13. D'abord cette comparaison montre plus clairement l'arrondissement du contour trouvé en ce qui concerne les angles convexes et concaves. Le coin altéré s'avère aussi plus près du contour réel de l'image. Il s'agit cependant d'une illusion. En regardant la figure 5.11b, il est facile de constater que les gradients de l'image à cet endroit se trouvent clairement à l'intérieur du reste du contour. Le contour trouvé n'est donc pas exact de la même façon que les autres coins ont été un peu arrondis.

La performance en terme de nombre d'itérations, comme en terme de temps pris par chacun des algorithmes pour converger, varie en fonction du rayon du voisinage β tel qu'illustré par le tableau 5.1. D'abord, le nombre d'itérations est, indépendemment de β, plus élevé pour l'algorithme modifié que pour l'algorithme classique. Par contre, le temps pour chacune des itérations est constamment supérieur pour l'algorithme modifié que pour l'algorithme classique. Ce dernier résultat ne surprend guère puisque, tous paramètres étant égaux, l'algorithme modifié utilise un plus grand nombre de vecteurs d'apprentissage

(433 pour l'algorithme modifié et 169 pour l'algorithme classique). L'algorithme modifié prend donc moins de temps au total dû au nombre d'itérations nécessaires pour la convergence. Ceci se remarque d'ailleurs dans toutes les images montrant l'évolution de chacune des unités du contour (figures 5.12 a et b, 5.14 a et b et 5.16 a et b). La différence dans le nombre respectif d'itérations dépend du paramètre d'apprentissage γ qui avantage l'algorithme modifié lorsque ce paramètre est le même pour la simulation à l'aide de chacun des algorithmes.

	Algorithme classique		Algorithme modifié	
	No. d'itérations	Temps/ itération	No. d'itérations	Temps/ itération
β=1	1638	0,86s	495	1,15s
β=2	1209	0,86s	252	1,16s
β=3	929	0,88s	223	1,17s

Tableau 5.1: Comparaison du nombre d'itérations et du temps pour chacune des Itérations pour l'algorithme classique et l'algorithme modifié en tenant fonction du voisinage β.

L'évolution de la performance, telle que décrite au tableau 5.1, porte à croire qu'agrandir le rayon du voisinage topologique réduit le nombre d'itérations. Or ce n'est pas vrai. Des tests ont été effectués avec des rayons de 10 et 20 respectivement. En comparaison avec le nombre d'unités (100 neurones), ces voisinages sont très grands et n'ont pas permis pas à l'algorithme de converger. Il semble donc que les très petits voisinages prennent plus de temps à cause de la grande flexibilité du contour. Agrandir un peu le voisinage résulte en une structure légèrement plus rigide limitant le nombre de configurations possibles. Agrandir le voisinage au-delà d'une certaine valeur crée un contour trop rigide. Dans ce dernier cas, supposons qu'un neurones se déplace. Ses 20 ou 40 voisins (2 fois le rayon pour le nombre total de voisins) devront aussi se déplacer. La somme cumulée de ces déplacements, additionné pour chacun des déplacements de tous

les neurones, résulte en une variation trop grande pour satisfaire le critère qui, rappelons-le, veut que la somme de tous les déplacements de toutes les unités soient plus petits que 0,1.

L'algorithme du *snake* appliqué à la même image permet de distinguer certaines nuances entre le *snake* et l'algorithme de Kohonen modifié. Des tests ont été effectués en modifiant systématiquement les constantes d'élasticité et de rigidité. Le contour initial a aussi été choisi plus ou moins loin du contour désiré. Les meilleurs résultats obtenus sont présentés ci-dessous. D'abord, en plaçant le contour initial assez près du contour escompté, le *snake* réussi à trouver assez bien le contour désiré. Le résultat est comparable à l'algorithme de Kohonen classique tel que montré à la figure 5.15a. Le contour trouvé avec l'algorithme du *snake* est présenté à la figure 5.18.

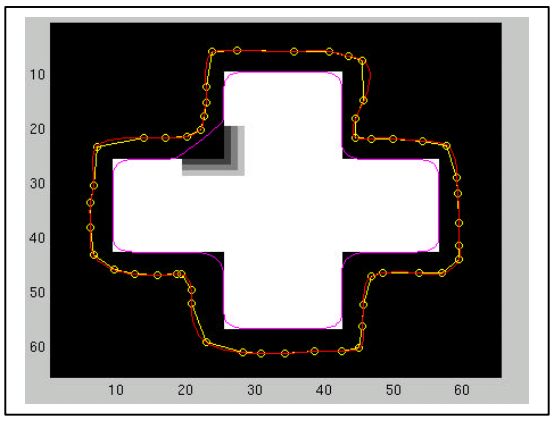


Figure 5.18: Simulation de détection de contour avec l'algorithme du *snake*.

Les paramètres utilisés sont les suivants:

Nombre de points du contour = 400

Coefficient d'élasticité = 3

Coefficient de la rigidité = 3

Pas temporel = 0,01

Nombre d'itérations = 8000

Coefficient de la force = 0,8

Deux points principaux sont à noter. D'abord, les coins sont comparables à ceux trouvés auparavant avec l'algorithme de Kohonen classique. Ensuite, l'intersection de la branche supérieure avec la branche gauche - là où se trouvent les gradients plus faibles - sont également passés inapperçus aux yeux du *snake*. Cette région avait cependant été détectée par l'algorithme de Kohonen modifié dont le résultat est montré en 5.15b. Étant donné que les paramètres du *snake* s'appliquent uniformément sur tout le contour, ceci impose de sacrifier l'une ou l'autre des caractéristiques de la précision du détail et de l'exactitude du contour global.

Le *snake* a ensuite été testé à partir d'un contour initial éloigné du contour désiré d'une façon similaire à ceux employés dans les simulations précédentes avec l'algorithme de Kohonen. Les paramètres utilisés sont les mêmes que ceux ayant produit le résultat de la figure 5.18. Le contour résultant est présenté à la figure 5.19.

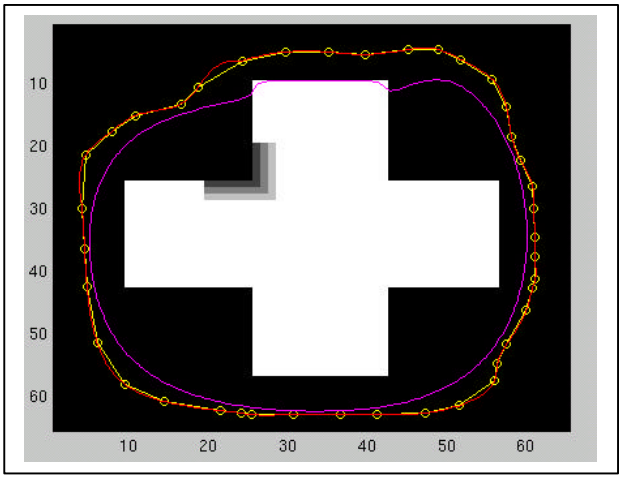


Figure 5.19: Simulation de détection de contour avec l'algorithme du *snake* avec un contour initial éloigné.

Le résultat montre bien la difficulté qu'éprouve le *snake* à se coller au contour de la croix. En effet, lorsque la distance est grande, la force que les gradients exercent sur le contour actif est très faible. Dans le cas présent, elle est insuffisante pour déplacer le contour dans sa direction. Ceci démontre la différence principale entre l'algorithme de Kohonen et le modèle des contours actifs. La "région d'action" du *snake* est locale.

La force de l'image exercée sur le contour n'est active qu'à l'intérieure d'un court rayon et elle est négligeable à mesure que l'on s'éloigne du point de l'image générant la force. En contrepartie, l'algorithme de Kohonen considère d'emblée l'ensemble de l'image en la découpant en cellules de Voronoï. Un jeu d'optimisation se produit ensuite afin de minimiser l'énergie à l'intérieur de chacune de ces cellules et l'énergie globale de l'ensemble des cellules.

5.4 Robustesse en présence de bruit

Dans le traitement des images réelles, la présence de bruit fait couramment obstacle à une détection précise des contours. La robustesse au bruit est donc un critère important pour l'évaluation d'un algorithme. En considérant le fait que le bruit peut être traité au niveau du pré-traitement de l'image et qu'il n'incombe pas seulement à l'algorithme de détection de contour de pallier à un manque ou à une mauvaise qualité d'information, le comportement de l'algorithme a été observé en présence de bruit.

L'image initiale de la croix a été reprise et bruitée artificiellement. Pour ce faire, une distribution aléatoire de valeurs situées entre 0,0 et 0,25 a été générée pour la surface de l'image (65x65 pixels) et ajoutée à celle-ci. Le résultat est montré à la figure 5.20 ci-dessous.

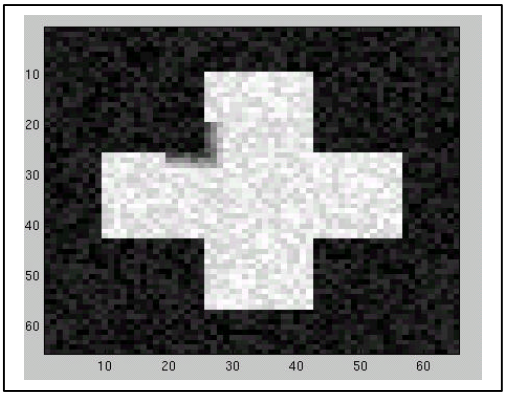


Figure 5.20: Croix bruitée avec des valeurs aléatoires situées entre 0 et 0,25 générées par une distribution normale.

Il est clair que l'ajout du bruit changera l'allure de l'image traitée avec le filtre de Sobel. L'image des gradients est montrée à la figure 5.21 ci-dessous. Cette image comprend tous les gradients générés par le filtre de Sobel.

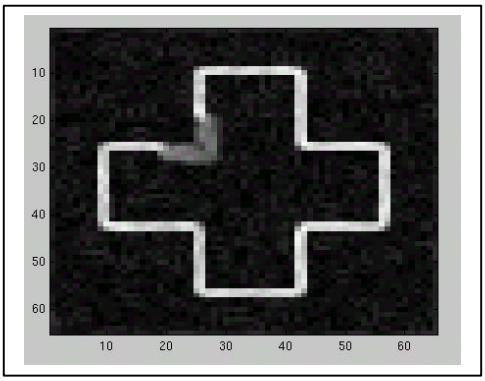


Figure 5.21: Résultat de l'application du filtre de Sobel sur l'image d'intérêt.

La détection de contour a été effectuée à partir des mêmes paramètres qu'à la section 5.2 et avec un voisinage β=3 dans le but de faciliter la comparaison. Les résultats sont présentés à la figure 5.22a pour l'algorithme classique et à la figure 5.22b pour l'algorithme modifié.

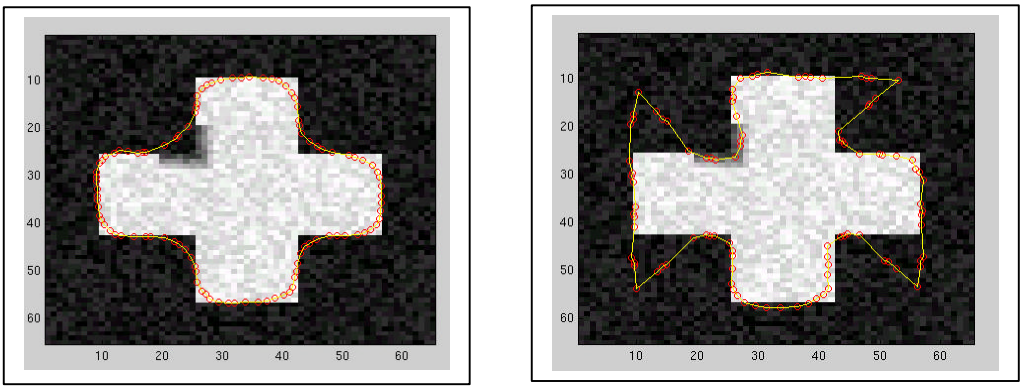


Figure: 5.22: Détection de contour d'une image bruitée à l'aide des paramètres suivants: β=3, $g = 0{,}003e^{(-i)} + 0{,}0003$ avec 100 unités dans le contour en utilisant a) l'algorithme de Kohonen classique et b) l'algorithme de Kohonen modifié.

Étant donné que tous les gradients sont pris en considération dans l'algorithme modifié, il est important de noter que, dû au bruit, des gradients se trouvent maintenant dispersé dans l'image. Ces gradients auront plus ou moins d'importance dépendant de l'ampleur locale du bruit. Aussi, lors de la sélection de la région d'intérêt, toute l'image a été selectionné et donc, tous les gradients se trouvant dans l'image. Le nombre de vecteurs d'apprentissage est de 168 pour l'algorithme classique (1 vecteur de moins que précédemment) et de 3906 pour l'algorithme modifié (un peu plus de 9 fois plus que pour l'image sans bruit).

Il est possible de noter, en comparant les résultats obtenus, que l'algorithme classique n'a pas été substantiellement affecté par le bruit. Le résultat est sensiblement le même que celui montré à la figure 5.17, c'est-à-dire l'image sans bruit avec des paramètres similaires. Le bruit introduit dans l'image n'a pas créer de gradients suffisamment grands pour qu'ils soient retenus lors des opérations de seuillage et binarisation des gradients. Les gradients retenus après seuillage et binarisation sont montrés à la figure 5.23.

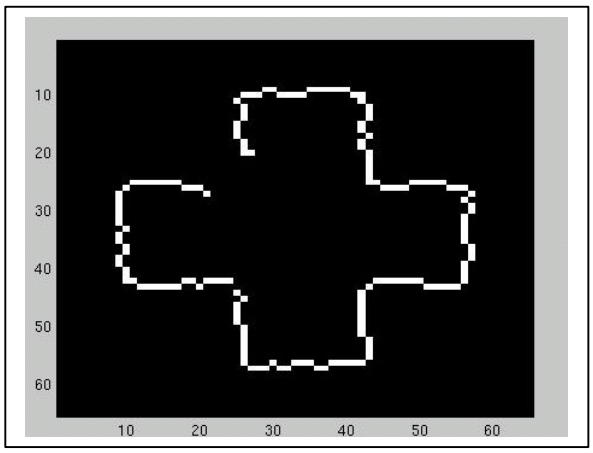


Figure 5.23: Seuillage et binarisation des gradients de l'image bruitée.

La forme globale a été conservée quoique les lignes droites sont devenues brisées. Il est à noter aussi que le coin situé à l'intersection de la branche du haut et de celle de droite n'apparaît toujours pas sur l'image des gradients. En contrepartie, les gradients

conservés pour appliquer l'algorithme modifié sont plus nombreux et répartis aléatoirement sur la surface de l'image. La notion de centre de gravité comme point d'attraction des unités du contour de l'algorithme modifié deviennent, de ce coup, moins significatifs de la direction à prendre pour le contour puisqu'ils contiennent maintenant des gradients générés par le bruit. Deux solutions sont possibles ici: réduire la région d'intérêt afin d'éliminer les gradients se trouvant à l'extérieur de la croix ou bien seuiller les gradients de telle sorte que les gradients négligeables soient exclus dans le calcul des centres de gravité.

La première solution, celle de choisir une région d'intérêt plus près de l'objet est illustrée à la figure 5.24. Notons d'emblée que l'on perd ici l'avantage de l'algorithme modifié puisqu'on élimine manuellement une partie substantielle des gradients dans une seule région de l'espace. La fine ligne blanche entourant la croix représente les limites de la région d'intérêt. Les gradients qui seront conservés se trouvent à l'intérieur de cette région.

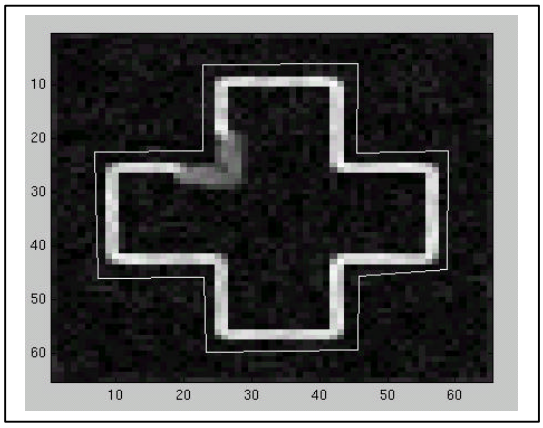


Figure 5.24: Région d'intérêt choisie autour de la croix.

Le contour trouvé par l'algorithme modifié à partir de cette région d'intérêt plus petite est présenté à la figure 5.25.

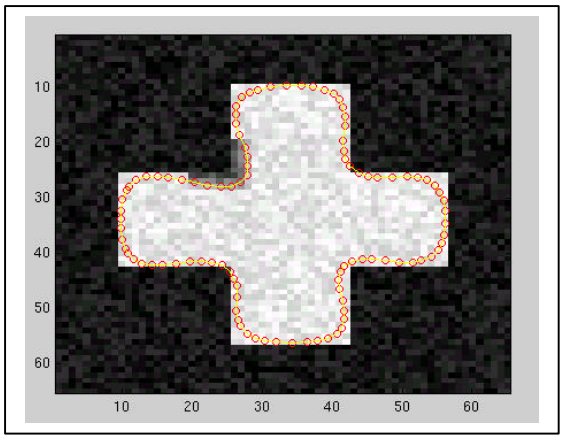


Figure: 5.25: Détection de contour d'une image bruitée à l'aide des paramètres suivants: β=3, $g = 0{,}003e^{(-i)} + 0{,}0003$ avec 100 unités dans le contour. Ce contour a été obtenu à partir des vecteurs d'apprentissage contenu dans la région telle que montrée à la figure 5.24.

En comparant la figure 5.25 avec la figure 5.14, il est possible de constater que le contour trouvé se trouve légèrement à l'intérieur du contour réel de l'objet. Ceci est dû au fait que l'intérieur de la croix contient maintenant un certain nombre de petits gradients dont l'action combinée de leurs centres de masse attire le contour vers l'intérieur. Ce comportement est minime étant donné que le niveau de bruit est relativement bas. Un niveau de bruit plus élevé produirait des gradients plus grands à l'intérieur de l'objet. De ce fait, les centres de masse des cellules de Voronoï se déplaceraient vers l'intérieur de l'objet entraînant ainsi le contour avec eux. La prochaine solution apparaît en ce regard comme étant plus robuste puisqu'elle élimine les gradients plus faibles d'une façon uniforme sur la surface de l'image plûtot qu'à l'extérieur du contour fermé seulement.

La valeur maximale du gradient de l'image traitée avec le filtre de Sobel a été établie à 4,8. Pour la simulation qui suit, une valeur de seuil de 2,0 a été choisie de façon à éliminer les plus petits gradients tout en gardant un nombre significatif de plus grands gradients. Les gradients qui ont été conservés sont montrés à la figure 5.26.

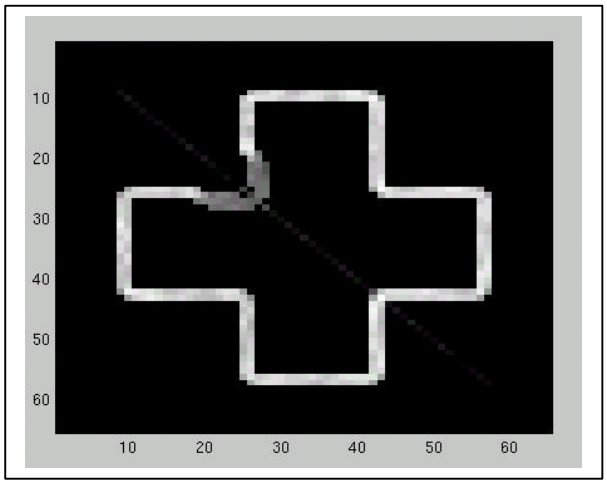


Figure 5.26: Gradients générés à partir de l'image bruitée. Le gradient maximal est de 4,83 et la valeur de coupure pour le seuillage est de 2,00.

Le gradient de l'image non-bruitée (figure 5.11b) est approximativement retrouvée en seuillant le résultat du filtrage de l'image au filtre de Sobel. La détection du contour de cette image s'est avérée plus efficace avec seulement 185 itérations (au lieu de 223 auparavant) et un nombre réduit de vecteurs d'apprentissage de 348 (au lieu de 433). Le contour final est montré ci-dessous à la figure 5.27.

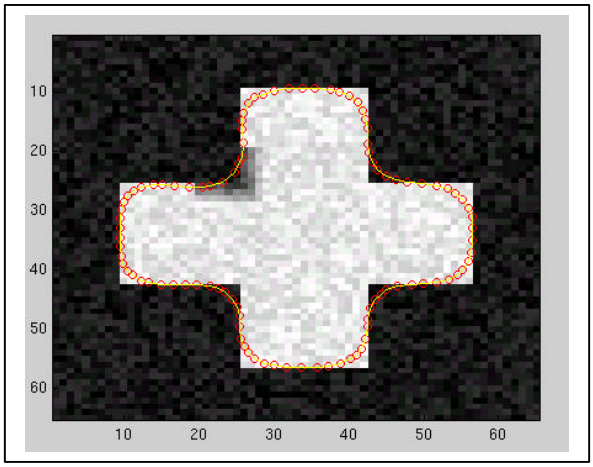


Figure 5.27: Détection de contour de l'image bruitée de la figure 5.20 à l'aide des paramètres suivants: β=3, $g = 0{,}003e^{(-i)} + 0{,}0003$ avec 100 unités dans le contour.

En comparant encore avec la figure 5.17, il est possible de constater que les contours trouvés sont très similaires. Une exception demeure toutefois dans le coin de l'intersection de la branche supérieure et de la branche de gauche. Le coin du contour trouvé à la figure 5.27 est moins incurvé que le même coin à la figure 5.17. Ceci est possiblement dû à la position et au nombre de gradients qui ont été conservés dans cette région.

En appliquant le *snake* à la même image bruitée (figure 5.28), on constate que l'algorithme performe bien malgré la présence du bruit. Ceci s'explique par le fait que le *snake* est moins dépendant des arêtes trouvées par un autre algorithme mais utilise les forces de l'image.

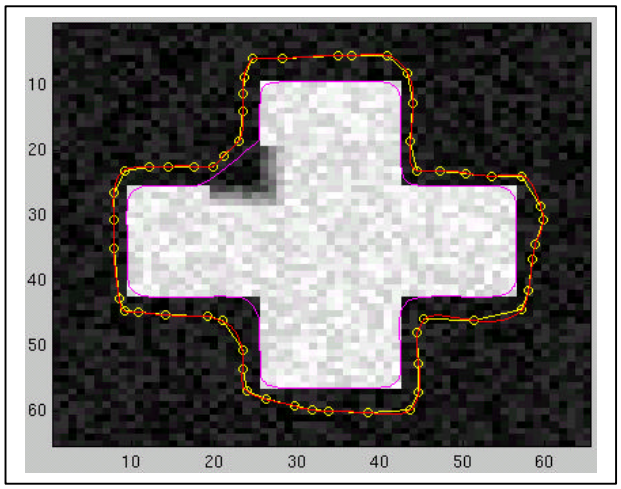


Figure 5.28: Résultat du *snake* pour la détection du contour d'un objet d'une image bruitée. Le contour initial est en rouge et le contour final en mauve.

En résumé, il a été démontré plusieurs possibilités de détection de contours à l'aide du réseau de Kohonen et sa version modifiée. La version modifiée remédie au manque d'information rencontrée par l'algorithme classique en considérant les gradients plutôt que leur comparses binarisés. La performance s'en trouve potentiellement diminuée étant donné le nombre accrû de vecteurs d'apprentissage traités par l'algorithme. Ce problème peut être partiellement résolu en effectuant seulement une opération de seuillage sur les

gradients de l'image qui diminue de ce fait le nombre de vecteurs d'apprentissage. Cette solution semble aussi contourner assez bien le problème de bruit dans l'image à niveau de gris. Le rayon du voisinage topologique joue aussi un rôle dans la vitesse et la précision du voisinage. Il semble exister un rayon - entre 1 et le nombre total d'unités du contour - qui génèrera une performance optimale en terme de vitesse. En ce qui concerne la précision du contour, un petit voisinage représente mieux les angles droits. À la lumière de ces résultats, il est maintenant possible d'améliorer la performance de l'algorithme en terme de rapidité et en terme de précision en combinant les paramètres ci-haut de la façon suivante:

1. Utiliser la version modifiée.
2. Seuiller l'image gradient de façon à ne garder que les gradients significatifs.
3. Utiliser un rayon de voisinage variable. Ceci permettra une organisation globale au début, puis un plus grand raffinement dans les régions qui demandent plus de précision.

5.5 Apprentissage dans le cas où certains points sont déterminés avec certitude

Jusqu'à maintenant, les propriétés de l'algorithme ont été explorées sans sa relation au reste du système de vision. Un système de vision conventionnel comprendrait un module de traitement de haut niveau pouvant effectuer un retour d'information vers les modules de plus bas niveau. Tel que discuté à la section 4.2, l'hypothèse que le module de haut niveau puisse associer une probabilité à chacune des unités du contour est le fondement de la présente exploration. Si cette hypothèse s'avérait être une voie praticable, il serait alors simple de modifier l'algorithme de telle sorte qu'il puisse recevoir cette sorte d'information et d'interagir avec le module de haut niveau. D'une façon similaire, il serait possible de connaître des points du contour dès avant l'application de l'algorithme. La modification considérée ici s'appliquerait de la même façon dans ce cas.

L'image de la croix est encore utilisée ici à titre d'exemple. Un voisinage de 3 est utilisé à la figure 5.29. Un tel voisinage produit des contours arrondis. L'apprentissage

est effectué comme à l'habitude jusqu'à ce que le critère d'arrêt soit atteint. Ensuite, tous les angles se trouvant à droite sur la croix sont choisis comme points appartenant au contour de la croix (6 en tout). Ces points sont, par le fait même, fixés. Notons aussi que les points choisis comme appartenant au contour réel n'appartiennent pas au contour trouvé après un premier apprentissage. L'hypothèse sous-tendant ce choix est qu'il est possible de faire une telle inférence. Le résultat est montré à la figure 5.29 ci-dessous.

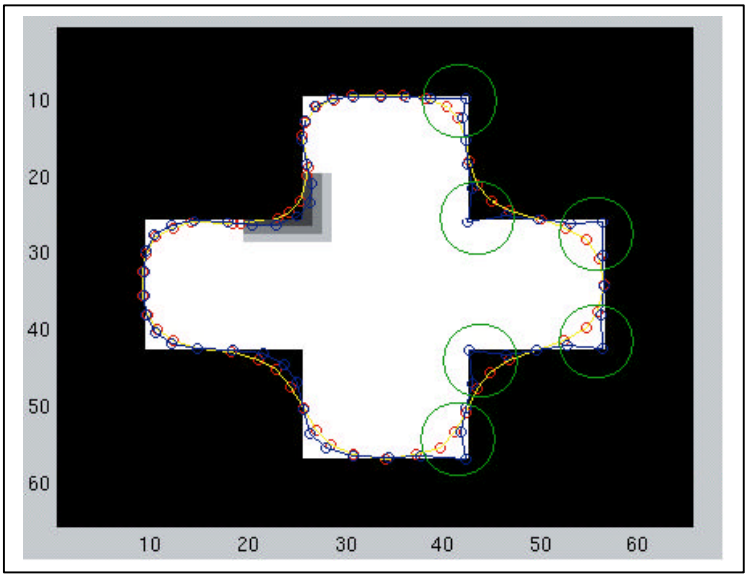


Figure 5.29: Contour trouvé en utilisant le retour d'information. Le contour trouvé avant feedback est en jaune. Le contour final est indiqué en bleu. Les régions soumises au retour d'information sont notées par des cercles verts.

Les coins sont maintenant bien représentés dans la mesure où ils ont été sélectionnés correctement. Il est possible de voir l'amélioration de la détection de ces coins aux endroits indiqués par des cercles verts. Le contour trouvé avant l'interaction est indiqué en jaune tandis que le résultat final est indiqué en bleu. Le reste du contour est constitué de segments de droites seulement et ceci ne constitue pas une grande difficulté pour l'algorithme de les représenter adéquatement.

5.6 Application à une image tomographique

Enfin l'algorithme a été appliqué à un cas réel d'image tomographique. La figure 5.30 montre une image d'une coupe du corps humain au niveau du coeur. Il s'agit d'une image acquise en médecine nucléaire à l'Hôpital Sacré-Coeur de Montréal. L'image consiste en une coupe d'une reconstruction tomograhique cardiaque à l'aide de l'agent radioactif mibi. Les niveaux de gris représentent l'aspect fonctionnel des organes. La couleur est plus pâle là où il y a le plus d'activité.

La détection du contour du coeur est utile et nécessaire afin de déterminer son volume et donc, la fraction d'éjection. Les contours trouvés permettront aussi de modéliser l'organe à l'aide d'ellipse dans le but d'en étudier le mouvement.

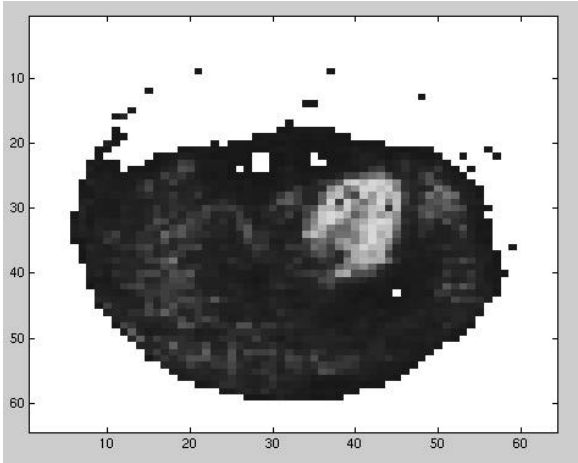
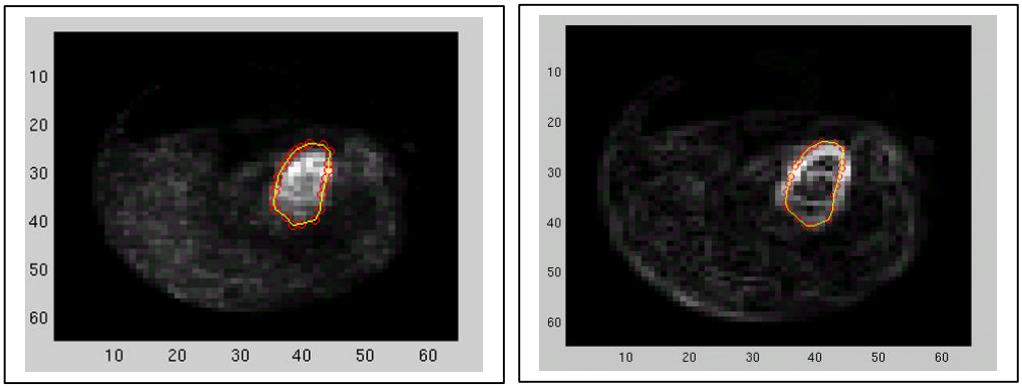


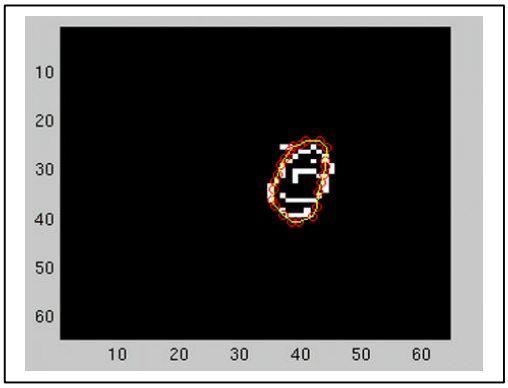
Figure 5.30: Image tomographique du coeur.

Trois algorithmes sont comparés dans la détection du coeur dans l'image 5.30: l'algorithme de Kohonen classique, Kohonen modifié et le *snake*. Les deux versions de l'algorithme de Kohonen ont été testées en utilisant les mêmes paramètres d'apprentissage ($\gamma = 0{,}003\ e^{(-i * 0{,}9)} + 0{,}0003$, rayon du voisinage $\beta=2$ et le critère d'arrêt = 0,1). Le *snake* a été utilisé avec les paramètres qui semblaient donner les meilleurs résultats et avec un contour initial ressemblant à celui utilisé pour les algorithmes de Kohonen. La validité des contours ainsi trouvés sera ensuite évaluée en les comparants avec des tranches en coupe de l'image convoluée avec un filtre de Sobel. Mais en premier lieu, les résultats des contours trouvés sont présentés directement sur l'image tomographique et l'image

convoluée au filtre de Sobel. Ces résultats pour l'algorithme de Kohonen classique sont présentés à la figure 5.31.



a) b)



c)

Figure 5.31: Résultat de l'application de l'algorithme de Kohonen classique à la détection du contour du coeur d'une image tomographique. Les images représentent respectivement a) le contour final sur l'image à niveau de gris, b) le contour final sur l'image convoluée avec le filtre de Sobel et c) le contour sur l'image convoluée avec le filtre de Sobel et binarisée.

Le résultat de la convolution de l'image avec le filtre de Sobel sera pris comme norme pour évaluer le succès des algorithmes pour détecter les contours. Rappelons que l'algorithme de Kohonen classique est utilisé avec une image binarisée. Afin de comparer le résultat de l'algorithme de Kohonen avec son ensemble d'apprentissage, la figure 5.31c

illustre le résultat de la détection du contour sur l'image binarisé. Le résultat de la détection de contour avec l'algorithme de Kohonen modifié est présenté à la figure 5.32.

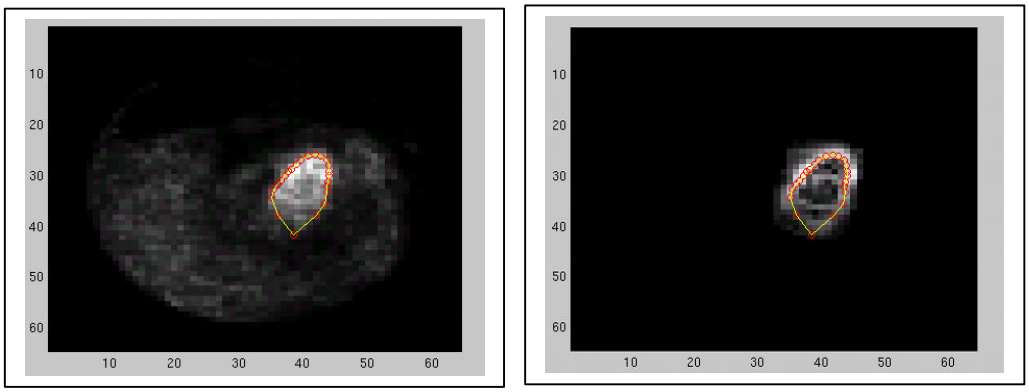


Figure 5.32: Résultat de l'application de l'algorithme de Kohonen modifié à la détection du contour du coeur d'une image tomographique. Les images représentent respectivement a) le contour final sur l'image à niveau de gris, b) le contour final sur l'image convoluée avec le filtre de Sobel.

Le contour de la même image a aussi été détecté en utilisant l'algorithme du *snake* dont le résultat est présenté à la figure 5.33.

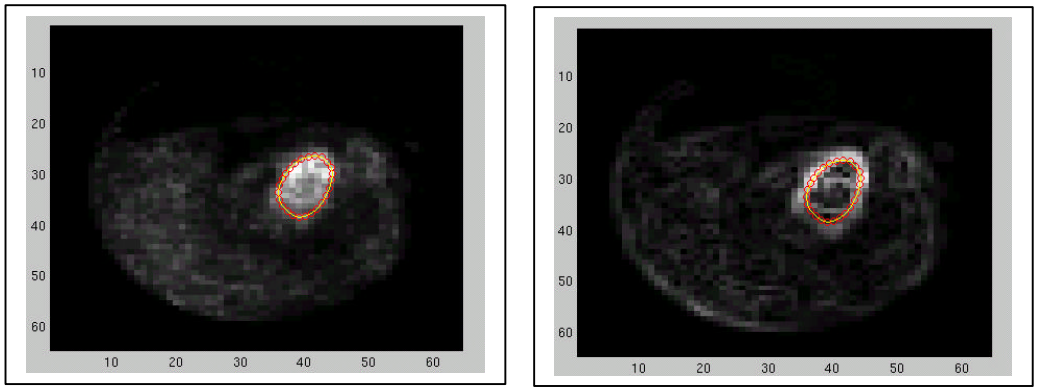
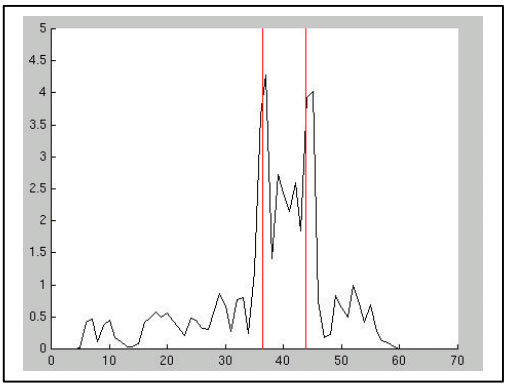
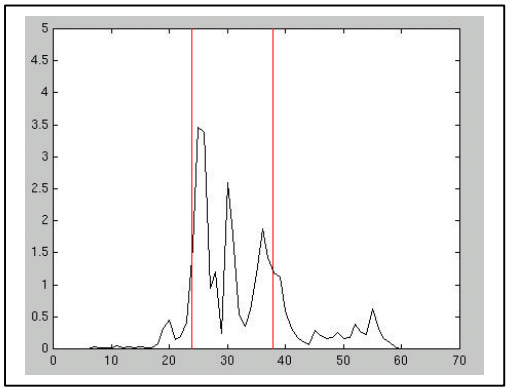


Figure 5.33: Résultat de l'application de l'algorithme du contour actif (*snake*) à la détection du contour du coeur d'une image tomographique. Les images représentent respectivement a) le contour final sur l'image à niveau de gris, b) le contour final sur l'image convoluée avec le filtre de Sobel.

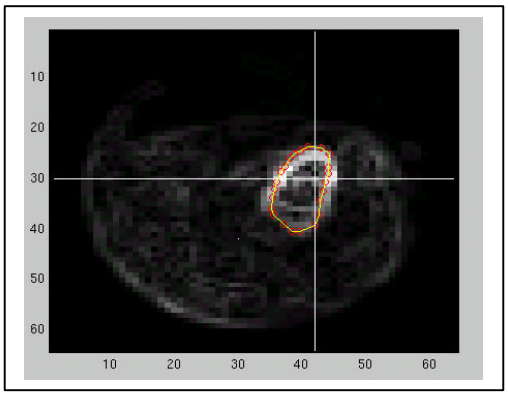
Afin d'évaluer la position du contour final, deux "tranches" de l'image convoluée avec le filtre de Sobel ont été extraites de l'image et présentées en coupe. La vue en coupe s'impose ici pour évaluer le contour trouvé d'un objet qui ne possède pas un contour clair et facilement identifiable par un individu non-spécialiste de l'imagerie médicale. Il serait possible d'y appliquer des mesures plus objectives de qualité du contour trouvé tels que mentionnés dans Abrantes (1996). Cependant, ceci impliquerait la détermination du contour réel tel que perçu par plusieurs spécialistes, ce qui dépasse le cadre du présent projet. La position du contour trouvé par chacun des algorithmes est indiqué par des barres verticales rouges. Les tranches ont été extraites à la 30ème ligne (tranche horizontale) et à la 42ème colonne (tranche verticale). Les positions de chacune de ces tranches sont montrées aux figures 5.34c, 5.35c et 5.36c pour les algorithmes de Kohonen classique, Kohonen modifié et pour le *snake* respectivement.



a) contour à x=36,45 et 43,88

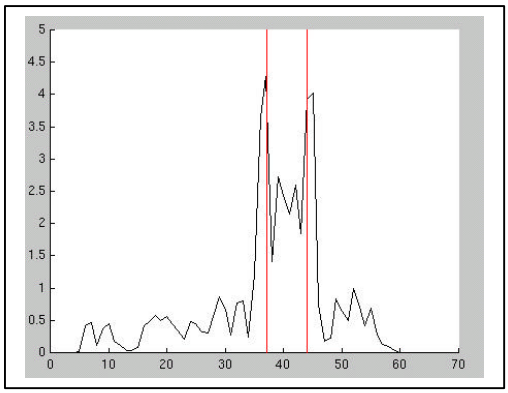


b) contour à y=23,81 et 37,74

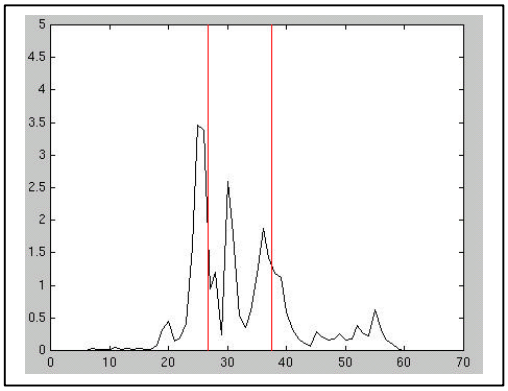


c)

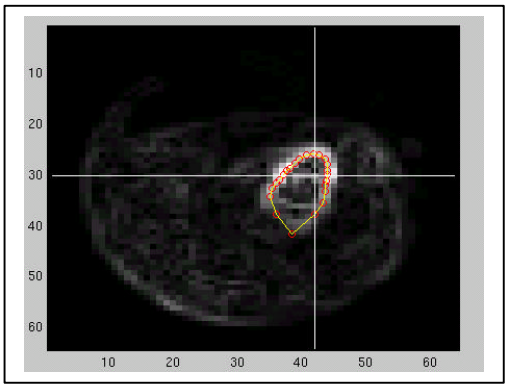
Figure 5.34: Résultat de la détection du contour du coeur dans une image tomographique avec l'algorithme de Kohonen classique dont a) montre la coupe horizontale à la 30ème ligne, b) montre la coupe verticale à la 42ème colonne et c) montre les positions de ces tranches, le contour final trouvé superposés à l'image convoluée avec le filtre de Sobel.



a) contour à x=37,08 et 43,99

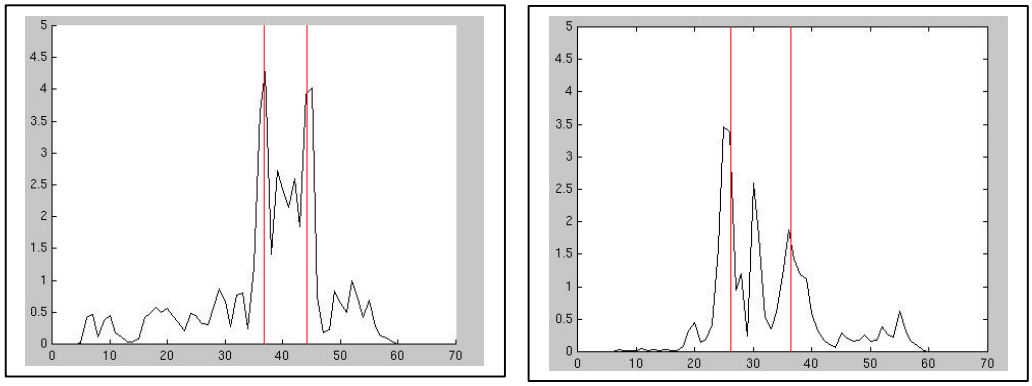


b) contour à y=26,62 et 37,54



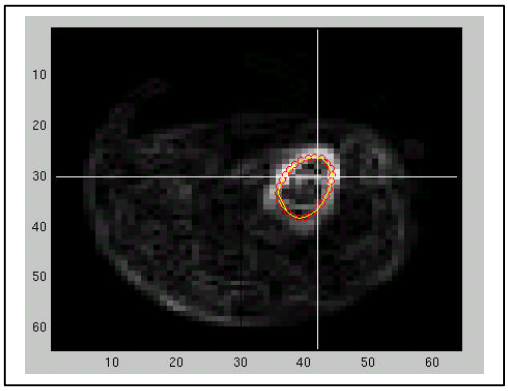
c)

Figure 5.35: Résultat de la détection du contour du coeur dans une image tomographique avec l'algorithme de Kohonen modifié dont a) montre la coupe horizontale à la 30ème ligne, b) montre la coupe verticale à la 42ème colonne et c) montre les positions de ces tranches, le contour final trouvé superposés à l'image convoluée avec le filtre de Sobel.



a) contour à x=36,77 et 44,27 ,22 et 36,41

b) contour à y=26



c)

Figure 5.36: Résultat de la détection du contour du coeur dans une image tomographique avec l'algorithme des contours actifs (*snake*) dont a) montre la coupe horizontale à la 30ème ligne, b) montre la coupe verticale à la 42ème colonne et c) montre les positions de ces tranches, le contour final trouvé superposés à l'image convoluée avec le filtre de Sobel.

Le tableau 5.2 contient le résumé des coordonnées du point où le contour croise la 30ème ligne ou la 42ème colonne. Comparons maintenant les images présentées aux figures 5.34 (a et b), 5.35 (a et b) et 5.36 (a et b). Dans l'ensemble, tous les algorithmes représentent assez bien le contour en se basant sur les gradients générés par le filtre de Sobel et des résultats similaires ont été obtenus pour toutes les coupes horizontales (figures 5.34a, 5.35a et 5.36a). L'algorithme du *snake* représente le mieux le contour de

l'objet, dans ce cas, puisqu'il positionne son contour exactement aux sommets des gradients de l'image. Ce comportement est exactement l'effet recherché puisque le contour se positionne dans un minimum local. Ici, les sommets sont représentés comme des pics mais l'équation (1.6) spécifie qu'il s'agit d'un minimum à cause du signe négatif. Aussi, la notion même du Lagrangien se veut une recherche d'un "chemin" de moindre énergie.

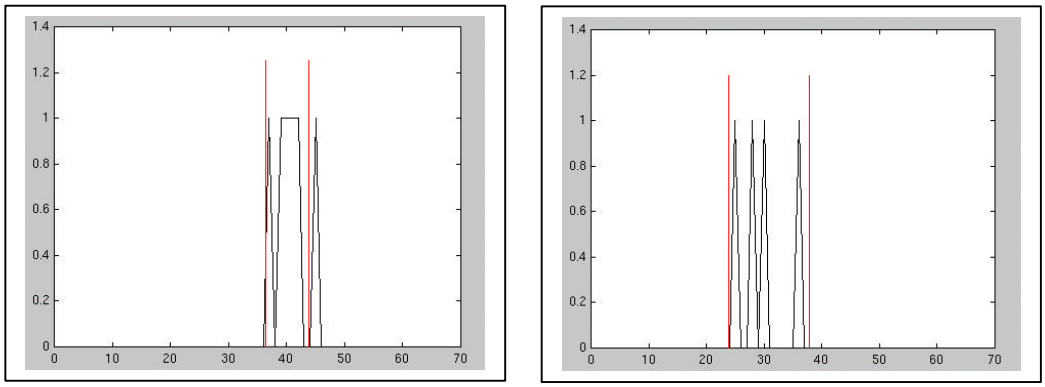
	Valeurs de x lorsque y=30		Valeurs de y lorsque x=42	
Kohonen classique	36,45	43,88	23,81	37,74
Kohonen modifié	37,08	43,99	26,62	37,54
Snake	36,77	44,27	26,22	36,41

Tableau 5.2: Coordonnées des contours trouvés par chacun des algorithmes lorsque y=30 et lorsque x=42.

À la figure 5.35b, l'algorithme de Kohonen modifié fixe son contour un peu à la droite des deux pics de gradients principaux. La position du contour à la gauche de l'image est un peu à l'intérieur signifiant que le contour à probablement été attiré par le pic des gradients du centre. La position du contour à la droite de l'image est quant à elle un peu à l'extérieur du pic de droite. Si l'on retient l'hypothèse que le pic central a attiré le contour de gauche, et se rappelant que les cellules de Voronoï sont mutuellement exclusives, le pic de droite est laissé seul pour attirer le contour qui ira se fixer près de son centre de gravité. Ceci met en lumière un problème potentiel de l'algorithme de Kohonen qui est que les cellules de Voronoï sont choisies selon une règle du plus proche voisin et que, dépendant du contour initial, ces voisins ne sont pas toujours correctement distribués entre les cellules de Voronoï. Ces cellules évoluent plutôt de façon stochastique avec des conditions initiales fixées par l'utilisateur.

La figure 5.34b montre quant à elle des contours qui passent à l'extérieur des pics de gradients principaux, c'est-à-dire à la gauche du pic de gauche et à la droite du pic de

droite. L'algorithme de Kohonen classique prend des données binaires. Les figures 5.37 a et b montrent les arêtes détectées après la binarisation ainsi que l'emplacement des contours. En comparant la figure 5.32b à la figure 5.34b, on remarque que même avec les données binaires, le contour se retrouve un peu à l'extérieur du contour de l'objet tel que suggéré par l'image des gradients.



a) contour à x=36,45 et 43,88 b) contour à y=23,81 et 37,74

Figure 5.37: Résultat de la détection du contour du coeur dans une image tomographique avec l'algorithme de Kohonen classique dont a) montre la coupe horizontale à la 30ème ligne et b) montre la coupe verticale à la 42ème colonne de l'image convoluée avec un filtre de Sobel puis binarisée avec un seuil de 0,13.

Ceci peut être expliqué du fait que les points voisins de ceux représentés ici ont été attirés plus fortement de ce coté des gradients. À cause de la méthode de régulation des voisins topologiques, le contour à cet endroit particulier a aussi été attiré dans cette direction. Il semble donc que le *snake* performe mieux dans ce cas où le contour de l'objet est plus flou. L'algorithme de Kohonen modifié donne de bons résultats mais ceux-ci ne peuvent être garantis étant donné le processus de répartition des cellules de Voronoï qui ne peut être contrôlé. L'algorithme de Kohonen classique donne des résultats acceptables mais dépend fortement des arêtes trouvées lors de la binarisation de l'image filtrée.

CONCLUSION ET RECOMMANDATIONS

L'algorithme de Kohonen à été présenté ici comme technique de regroupement des arêtes. Quoique développé sur des bases différentes, il a été demontré que cet algorithme peut être présenté sous une forme commune à celle du *snake*. C'est à l'intérieur de ce cadre commun que l'algorithme a par la suite pu être modifié et les résultats interprétés.

En premier lieu, le fonctionnement de l'algorithme de Kohonen, tel que présenté dans sa version classique, a été démontré sur un contour simple que procure un carré. Ensuite, la division de l'image entière en cellules de Voronoï a suggéré des capacités globales de capture de l'ordre topologique des arêtes de l'image. L'utilisation du filtre de Sobel, combinée avec la définition de l'algorithme dans sa version déterminée par l'approche commune, a par la suite ouvert la voie à une modification de l'algorithme de façon à prendre en compte les gradients continus générés par le filtre de Sobel. L'aspect de capture globale de la topologie a aussi été mis en relief en le comparant avec l'algorithme du *snake*. Par la suite, l'algorithme modifié a été utilisé avec une image bruitée afin d'évaluer sa robustesse en présence de bruit. La nouvelle définition de l'algorithme de Kohonen que procure le cadre de travail commun a aussi permis de démontrer comment l'algorithme pourrait être modifié afin de recevoir un retour d'information d'un module de niveau supérieur d'un système de vision complet. Finalement, les algorithmes de Kohonen classique, Kohonen modifié et du *snake* ont été appliqués à la détection d'un contour dans une image tomographique et comparés en visualisant des tranches de l'image ainsi que les contours trouvés.

L'algorithme de Kohonen classique prend, comme données d'entrée, des vecteurs de positions des arêtes préalablement identifiées à l'aide d'un détecteur d'arêtes. Abrantes et Marques (1996) ont utilisé l'algorithme de Canny (1986). Quoique ce dernier soit un algorithme plus performant pour la détection d'arêtes (Haralick et Shapiro, 1992), l'algorithme de Sobel a été utilisé ici à cause de la possibilité intéressante d'utiliser éventuellement un ensemble de gradients continus. Les effets du paramètre d'apprentissage γ et du voisinage topologique β ont aussi été démontrés. Le paramètre d'apprentissage contrôle la grandeur des déplacements des unités du contour. Étant donné que le critère d'arrêt de l'algorithme dépend des déplacements combinés de toutes les unités, diminuer ce paramètre équivaut en quelque sorte à "refroidir" le système. Le voisinage topologique assure la cohérence des unités sur un rayon égal à β. Un grand rayon permet une cohérence globale du contour trouvé tandis qu'un petit rayon permet au contour de s'adapter aux détails du contour de l'objet d'intérêt.

Afin de pousser plus avant la notion de cohérence globale du contour en relation avec le voisinage topologique, un contour initial radicalement différend du contour de l'objet a été testé. Le contour initial en forme de "8" a été imposé à l'algorithme dans le but de trouver la forme d'un carré. En ajustant le voisinage topologique correctement, il a été possible de défaire le noeud du contour initial et de capturer la forme du carré adéquatement. Ceci n'a pas été possible avec le *snake*. Ces simulations ont permis de mettre en relief les effets des zones d'attractions respectives pour chacun des algorithmes: cellules de Voronoï pour Kohonen et une Gaussienne tronquée pour le *snake*. Les cellules de Voronoï favorisent l'organisation globale du contour tandis que le *snake* est attiré par des forces dont les effets ne sont sentis que sur de courtes distances.

Certaines variations de l'éclairage auquel est soumis un objet, de l'activité ou de la densité de certains organes dans le cas d'images biomédicales (rayons X, MRI, images tomographiques, etc.) résultent en un manque d'informations nécessaires à reconstituer les contours. Au moins deux stratégies sont alors envisageables. La première consiste à utiliser un détecteur d'arêtes plus performant. Cette approche a été prise par Abrantes et

Marques (1996). La seconde consiste a utiliser un opérateur de gradient, l'opérateur de Sobel en l'occurence, et d'utiliser les gradients continus au-delà d'un seuil donné. C'est l'approche qui a été développée et testée dans ce projet. Selon cette approche, les gradients ainsi obtenus sont considérés comme des poids affectant la force d'attraction des unités d'un contour soumis à l'algorithme de Kohonen. Des gradients plus petits auront ainsi un effet d'attraction réduit vis-à-vis des points de l'image possédant un gradient plus grand. L'hypothèse sous-jacente est qu'une grande quantité de points de l'image se trouvant à proximité les uns des autres appartiennent au contour de l'objet. Dans l'approche de Kohonen ainsi modifiée, la somme pondérée de ces gradients a résulté en la détection d'une partie de contour auparavant indétectable par l'algorithme de Kohonen classique utilisant les arêtes générées par une simple binarisation de l'image des gradients ou avec l'algorithme du *snake*. Cette approche relaxe les contraintes d'appartenance de pixels de l'image au contour et propose un algorithme de regroupement d'arêtes permettant de détecter un contour de façon plus nuancée.

Mais cette relaxation des contraintes pourrait aussi créer des inconvénients. En effet, le bruit présent dans l'image peut aussi générer des gradients n'ayant aucune signification quant à leur appartenance au contour de l'objet. En d'autre termes, les pixels affectés par le bruit génèrent aussi des arêtes-bruit. Un seuillage adéquat a permis d'obtenir un contour correct. Cependant, ceci démontre la dépendence de cette méthode sur les opérations de plus bas niveau que sont le filtrage de l'image et la détection des arêtes. Le regroupement des arêtes avec l'algorithme de Kohonen modifié dépend donc fortement de l'exactitude des arêtes détectées, l'exactitude des arêtes se définissant comme étant la représentation correcte du contour réel de l'objet par les arêtes détectées.

Le regroupement des arêtes n'est habituellement pas une fin en soi mais s'insère au niveau intermédiaire d'un système de vision à base de connaissances dans lequel on retrouve aussi un module de haut niveau doté d'un mécanisme d'inférence dont le rôle est de représenter et d'interpréter des objets ou parties d'objets. Pour ce faire, il est impératif de modifier l'algorithme afin qu'il puisse interagir avec ce module de haut niveau. Ce module de haut niveau n'a pas été développé lors de ce projet mais une hypothèse a été

posée quant à l'information utile que ce module pourrait générer et qui pourrait être utilisée par l'algorithme de regroupement d'arêtes afin d'améliorer la qualité du contour trouvé. Cette hypothèses est que le module de haut niveau soit capable d'évaluer la probabilité que certaines arêtes appartiennent vraiment au contour de l'objet. Si certaines arêtes possèdent un niveau de certitude assez élevé, une unité du contour du réseau de Kohonen y est spécifiquement attribuée et l'apprentissage reprend sans que ces unités ne soient mises à jour. Cette autre modification de l'algorithme a permis de démontrer qu'un tel retour d'information pouvait améliorer le contour initial trouvé par l'algorithme.

Enfin, l'algorithme de Kohonen modifié (dans le sens du calcul des centres de masse) a été comparé à l'algorithme de Kohonen classique ainsi qu'à l'algorithme du *snake* par l'application à une image tomographique réelle. L'image était différente des autres images-tests en ce que son contour n'était pas aussi clairement défini que le carré ou la croix synthétiques utilisés auparavant. En se basant sur une vue en coupe de l'image de gradients produite par la convolution avec le filtre de Sobel, l'emplacement des contours a été examiné. Le *snake* semblait donner les meilleurs résultats dans ce cas quoique les autres algorithmes produisaient aussi des contours satisfaisants.

En résumé, l'algorithme de Kohonen modifié a réussi à compenser la perte d'information causée par l'algorithme de détection des arêtes, c'est-à-dire le filtrage de l'image avec le filtre de Sobel et la binarisation. L'algorithme performait d'une façon comparable à des algorithmes existants comme le *snake* et l'algorithme de Kohonen en ce qui concerne le contour final trouvé. La capacité de l'algorithme de capturer la forme globale du contour a aussi été démontrée, ce qui lui donne un avantage sur le *snake*. Cependant, l'algorithme dépend fortement du résultat de la détection des arêtes et donc de l'efficacité du pré-traitement à éliminer les arêtes générées par le bruit. L'utilité du cadre de travail utilisé ici ne fait aucun doute. Dans ce même cadre, il serait possible de tirer avantage des différents algorithmes qu'il représente et de développer un algorithme tirant avantage de ceux présentés ici. Par exemple, en superposant les cellules de Voronoï utilisées par Kohonen et les zones d'attractions du *snake*, on peut éliminer le chevauchement de ces zones d'attractions et potentiellement limiter les mouvements de

l'algorithme pour ainsi éviter que le contour *snake* ne s'effondre sur lui-même. Ceci pourrait être réalisé en effectuant une transition d'un mode d'évolution à un autre ou bien en combinant les deux en un seul mode d'évolution.

RÉFÉRENCES

Abrantes, A. J. et Marques, J. S. (1995). Unified Approach to Snakes, Elastic Nets, and Kohonen Maps. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing* - Proceedings Special Sessions Proceedings of the 1995 International Conference on Acoustics, Speech, and Signal Processing. 9-12 mai 1995, Detroit, MI, USA, pp. 3427-3430.

Abrantes, A. J. et Marques, J. S. (1996). A Class of Constrained Clustering Algorithms for Object Boundary Extraction. *IEEE Transactions on Image Processing*, Vol. 5, no.11.

Canny, J. F. (1986). A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intellingence*, 8(6): 679-698.

Cohen, L.D. (1991). Note on active contour models and balloons. *Computer Vision, Graphics, and Image Processing: Image Understanding* (CVGIP:IU), Vol. 53 no. 2 pp.211-218, Mars 1991.

DeSieno, D., (1988). Adding a Conscience to Competitive Learning, *Proceedings of the Second Annual IEEE International Conference on Neural Networks*, Volume 1.

Haralick, R. M. et Shapiro, L. G. (1992). Computer and Robot Vision. Reading, Massachusetts: Addison-Wesley.

Haykin, S. (1994) *Self-Organizing Systems II: Competitive Learning*, in *Neural Networks: A Comprehensive Foundation*, pp: 397-443. Toronto: MacMillan Publishing Co.

Jain, R., Kasturi, R. et Schunck, B. (1995). *Machine Vision*, McGraw-Hill.

Kass, M., Witkin, A., Terzopoulos, D. (1987). Snakes: Active Contour Models. *First International conference on Computer Vision*, Londres.

Kass, M., Witkins, A. et Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, vol.1, pp. 259-268.

Kohonen, T. (1991). *Self-Organizing Maps: Optimization Approaches*, in *Artificial Neural Networks*, pp: 981-990. North-Holland: Elsevier Science Publishers B.V.

Kohonen, T. (1982a). Clustering, taxonomy, and topological maps of patterns. *Proceedings of the 6th International Conference on Pattern Recognition*.

Kohonen, T. (1982b). The self-organizing map. *Proceedings IEEE*, vol. 78, no.9, pp. 1464-1480.

Malsburg, C. von der (1973). Self-organization of Orientation Sensitive Cells in the Striate Cortex. *Kybernetik*: 14, pp. 85-100.

Malsburg, C. von der, Willshaw, D.J. (1977). How to label nerve cels so that they can interconnect in an ordered fashion. *Proceedings of the National Academy of Science* USA, no. 74, 5176-5178.

Marr, D. et Hildreth, E. (1980). Theory of edge detection. *Proceedings of the Royal Society of London*, vol B207, pp. 187-217.

Marr, D. et Nishihara, H. K. (1978). Visual Information Processing: Artificial Intelligence and the Sensorium of Sight. *Technology Review*, vol. 81(1).

Rosenblatt, F. (1958). The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*, no.65, pp.386-408.

Sobel, I. (1970). *Camera models and machine perception.* Stanford AI Memo 121, Mai 1970.

Willshaw, D.J., Malsburg, C. von der (1976). How patterned neural connections can be set up by self-organization. *Proceedings of the Royal Society*, B194, pp. 431-445.

Willshaw, D.J., Malsburg, C. von der (1979). A marker induction mechanism for the establishment of ordered neural mappings: its application to the retino-tectal problem. *Phil. Trans. R. Soc. London*, B287, pp. 203-243.