Étude d'un détecteur de contours basé sur la logique floue

Frédéric Mayer

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL RHÔNE-ALPES

CENTRE D'ENSEIGNEMENT DE GRENOBLE

MEMOIRE

présenté par **Frédéric Mayer**

en vue d'obtenir

LE DIPLÔME D'INGENIEUR C.N.A.M.

en INFORMATIQUE

Etude d'un détecteur de contours basé sur la logique floue

Soutenu le 1er Juillet 2010

JURY

Présidente : M[r] Eric Gressier-Soudan

Membres : M[me] Catherine Garbay
M[r] Jean-Pierre Giraudin
M[r] André Plisson
M[r] Lionel Reveret
M[r] Horia Nicolae Teodorescu
M[r] Mathias Voisin-Fradin

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL RHÔNE-ALPES

CENTRE D'ENSEIGNEMENT DE GRENOBLE

MEMOIRE

présenté par **Frédéric Mayer**

en vue d'obtenir

LE DIPLÔME D'INGENIEUR C.N.A.M.

en INFORMATIQUE

Etude d'un détecteur de contours basé sur la logique floue

Soutenu le 1er Juillet 2010

Les travaux relatifs à ce mémoire ont été effectués dans le laboratoire du CERFS à Iasi en Roumanie sous la direction de *Mr le professeur, Docteur et Docteur Honoris Causa H. N. Teodorescu.*

Remerciements

Je voudrais particulièrement remercier le Professeur Teodorescu qui m'a accueilli dans son équipe du laboratoire CERF à Iasi en 2004 ainsi que Madame Garbay de l'équipe SIC pour ses conseils.

Je me souviendrai sans conteste des efforts constants de Monsieur Giraudin pour garder éveillée la flamme durant mes années d'inactivité.

Merci à Oana, Ciprian, Lucian, et Sidor, les membres du laboratoire, ils m'ont permis d'avancer plus vite par les échanges fructueux et leur collaboration.

Enfin merci aux membres de ma famille et à Antoine qui m'ont soutenu sans faille du premier au dernier jour dans cette longue aventure qu'ont été ces années d'étude au CUEFA conclues par ce stage en Roumanie.

Académie Roumaine
Conservatoire National des Arts et Métiers
2003-2010

Table des matières

Table des figures et des tables

Introduction

Dans ce chapitre, après une présentation générale du contexte dans lequel s'est déroulé mon stage de cycle ingénieur du Conservatoire National des Arts et Métiers, on définira la problématique propre au stage.

1.1. L'Académie Roumaine et l'Institut d'Informatique Théorique à Iasi

En Roumanie, la ville de Iasi est la capitale de la région de la Moldavie [IASI]. Elle est située à la frontière nord est du pays. Au XIX^{ème} siècle, c'est ici qu'a été fondée la première université du pays. Iasi a donc toujours défendu la réputation d'un grand centre universitaire et culturel. D'illustres écrivains et musiciens y vécurent comme Eminescu ou Enescu. Les liens avec la France restent forts. D'ailleurs, le centre culturel français programme régulièrement des manifestations scientifiques avec différentes branches de la faculté en collaboration avec une association locale de professeurs francophones. Le jumelage avec la ville de Poitiers est aussi très actif et permet de nombreux échanges universitaires dans le cadre du programme européen Erasmus.

L'académie Roumaine de Iasi y a été fondée en 1948 [ACAD]. Son but est de coordonner les activités de recherches et d'offrir un lieu d'échanges entre des domaines scientifiques différents à travers l'organisation de congrès, conférences, ou tables rondes. La plus importante de ces manifestation permet à chacune des entités de présenter les résultats de l'année passée : Certaines recherches trouvent alors une applicabilité dans un domaine complètement différent de celui où a été menée la première recherche.

L'Académie est composée aujourd'hui de 14 unités de recherche. Elle est dirigée par un collège de 19 membres permanents et des correspondants honoraires venus d'autres universités ainsi que des pays voisins : la Moldavie et l'Ukraine. Elle édite périodiquement trois revues dont une nommée "Sisteme fuzzy si Inteligenta Artificiala " soit "Systèmes flous et Intelligence artificielle". Au niveau national, elle est en relation avec les filiales des autres villes universitaires du pays.

Les activités de recherche de l'Institut d'Informatique Théorique [ICS] sont dédiées à la recherche dans les domaines suivants :

- le développement de méthodes et d'algorithmes pour le traitement de l'image. Les applications se font dans les domaines de la médecine, la métallographie, la reconnaissance de caractères et l'analyse d'images satellitaires ;

- le traitement du signal avec des systèmes flous et des réseaux de neurones appliqués à la médecine, l'analyse sémantique du langage ;
- le développement et l'analyse de stabilité d'ensembles de contrôle/commande par des systèmes flous pouvant être associés à des réseaux de neurones ;
- le traitement automatique de la synthèse vocale appliqué spécifiquement à la langue roumaine ;
- l'analyse et la génération automatique de langage naturel. Le développement de bases de données linguistiques et de lexiques pour la langue roumaine en SGML ;
- la théorie de la programmation pour les systèmes distribués ;
- les modèles biophysiques des neurones néocorticaux isolés et des réseaux de neurones.

1.2. Le centre de recherche CERF

Au sein de l'Institut d'informatique théorique, le centre de recherche CERF [CERF] est dédié aux recherches fondamentales et appliquées dans les domaines des systèmes à logique floue, des réseaux de neurones, des systèmes dynamiques et de la théorie du chaos, des systèmes basés sur la connaissance et leurs applications dans l'ingénierie biomédicale et en médecine. L'acronyme CERF en est un résumé. Il a été fondé en 1990. Il a obtenu en 2004 la distinction de "centre de recherche d'excellence" par l'équivalent du CNRS Roumain en reconnaissance de la valeur des travaux qui y ont été réalisés.

Le centre est dirigé par son fondateur, le Professeur Teodorescu. Connu comme un homme de science d’envergure nationale, membre de l'Académie Roumaine depuis 1993, il est aussi reconnu comme un spécialiste au niveau mondial de par ses nombreuses contributions à la théorie des systèmes flous et neuro-flous ou au domaine de la dynamique du chaos non linéaire et leurs applications. A l'étranger, il a été en charge de l'enseignement dans les universités de Lausanne, Izuka (Japon) et de "South florida" (Floride).

Les résultats des recherches menées au sein du CERF ont fait l'objet de publications dans des revues éditées par l'Académie Roumaine, des bulletins scientifiques des universités, des revues spécialisées de diffusion nationale. A un niveau international, certains articles ont été publiés dans des revues de l'IEEE, des travaux ont été présentés lors de conférences internationales et plus de 25 articles ont été publiés dans des volumes des conférences internationales prestigieuses.

Parmi les sujets de recherches ayant abouti à des publications, on peut citer :

- l'étude d'une base de connaissances basée sur des paramètres non-linéaires des tremblements parkinsoniens. L'évaluation des résultats du traitement de la maladie de Parkinson. Développement d'un système adaptatif de simulation des tremblements parkinsoniens ;
- l'étude d'un système d'aide à la décision dans le choix des prothèses cardiaques ;
- la caractérisation et la prédiction de séries de données génomiques à l'aide d'un système neuro-flou ;
- la modélisation de processus biologiques et de systèmes cellulaires avec un système chaotique adaptatif ;
- le développement d'un module embarqué sur une sonde martienne pour la mesure de l'accélération lors de la rentrée de la sonde dans l'atmosphère ;
- le développement de systèmes de réalité virtuelle augmentée pour le diagnostic et le traitement médical.

Ces recherches se font aussi en collaboration avec d'autres universités comme celles de Toulouse, au sein de programmes européens.

Tous les deux ans, les années paires, le CERF organise la conférence ECIT : conférence européenne sur les systèmes et les technologies intelligents. Durant trois jours, une centaine de scientifiques présentent les évolutions de leurs recherches dans les domaines connexes aux domaines de recherche du CERF.

Le CERF participe aussi aux journées académiques de Iasi : ce moment de rencontre avec des scientifiques de tous les domaines permet d'échanger des idées, de confronter les méthodes et les résultats. Ces rencontres sont très constructives : dans certains cas, une application d'une technique ou d'un principe employé dans un domaine de recherche peut être envisagée dans un autre domaine en apparence très éloigné.

1.3. Le stage

Le stage s'est donc déroulé au sein de l'équipe du centre de recherche CERF durant l'année 2004 (Nov. 03 - Nov. 04).

La problématique principale a été d'étudier la possibilité d'utiliser une architecture basée sur la logique floue pour filtrer les contours présents sur une image. L'architecture du filtre m'a été proposée par le Professeur Teodorescu sous la forme d'un algorithme en pseudo code. Après avoir vérifié que cette méthode pouvait donner des résultats intéressants par l'implémentation d'un prototype, nous avons voulu d'une part, quantifier la validité de ces résultats et les comparer avec ceux obtenus par d'autres algorithmes plus classiques et, d'autre part, améliorer les résultats du filtre en effectuant des réglages fins sur l'ensemble des paramètres qui le caractérisent.

Ce document présente les différents aspects de cette étude : dans un premier temps, nous établirons un panorama comparatif des solutions permettant d'extraire les contours présents sur une image. Les chapitres suivants présenteront les éléments techniques utilisés comme la logique floue et les algorithmes génétiques. Puis, nous verrons comment nous avons pu quantifier la qualité de la détection des contours. Ensuite, nous présenterons l'algorithme basé sur la logique floue et les détails des méthodes qui nous ont permis d'affiner son paramétrage. Enfin, nous présenterons les résultats que nous avons obtenus pratiquement et les détails de l'implémentation.

2. Etat de l'art

Dans un premier temps, ce chapitre présentera les différentes notions auxquelles nous ferons référence et introduira certaines notations. Puis, il présentera et comparera certains algorithmes de détection de contours existants. Enfin, nous présenterons les bases de la théorie de la logique floue et des algorithmes génétiques.

2.1. Notion de contour

Un contour idéal est le contour dit *en escalier* [MARTIN *et al.* 2000]. Il représente un changement brusque du niveau de luminosité entre deux points adjacents. Par exemple, dans la figure 2.1, un contour *en escalier* va être détecté entre les points des quatrièmes et cinquièmes colonnes des deux premières rangées. Ce genre de contour est considéré comme idéal parce qu'en réalité, dans une image réelle, les changements d'intensité sont répartis sur une plage couvrant plusieurs points et sont perturbés par du bruit. De plus, la numérisation et des pertes dues à la compression engendrent des imprécisions supplémentaires sur le contenu même de l'image. Ces erreurs cumulées engendrent des incertitudes lors de la détection de contours et il est donc peu probable que les contours puissent être détectés à leur emplacement réel.

On appelle contour *en rampe* ou *progressif*, une variation de luminosité répartie sur plusieurs pixels. On considère habituellement le milieu de la rampe reliant les zones de basse et haute luminosité comme le véritable emplacement du contour. Néanmoins, certains détecteurs feront des suppositions basées sur des calculs dont les résultats placeront les contours ailleurs. La figure 2.1 exemplifie ces types de contours.

contour *en escalier* contour *en rampe*

Figure 2.1 : Exemples de contours

Puisque les contours sont définis comme une variation de l'intensité de niveau de gris répartie sur plusieurs pixels contigus, la plupart des algorithmes de détection sont basés sur un principe

commun : la détection du plus fort gradient. Ils utilisent le calcul de la dérivée première ou seconde de la fonction $f(i,j)$.

La qualité de la détection des contours d'une image en trois dimensions ou en couleur présente des difficultés supplémentaires. Elle est dépendante du point de vue : selon la géométrie de la scène, la répartition de l'éclairage, l'opacité de certains objets certains contours sont apparents ou cachés, mis en valeur ou masqués par manque de contraste par rapport à l'arrière plan. Dans des images en couleurs, les différences de couleurs peuvent marquer des zones différentes appartenant à un même objet. De même, si une ligne est présente sur un fond de couleur uniforme avec un contraste marqué, alors on détectera deux contours, de chaque côté de la ligne.

Une autre difficulté existe dans le cadre des images radiographiques : Ces images en deux dimensions représentent la projection sur un plan d'un espace en trois dimensions. Certaines arrêtes s'en trouvent masquées. Des intersections existent sur l'image mais elles ne signifient pas que les éléments de l'image sont réellement concomitants. Si on doit interpréter le contenu de l'image, il faut alors en tenir compte.

2.2. Définitions et notations

On dénote :

- *I* une image ;
- *i* et *j*, les coordonnées d'un point de l'image selon les axes *x* et *y* ;
- *M* et *N*, les dimensions de l'image selon les axes *x* et *y* ;
- $f(i,j)$, l'intensité de la luminosité en un point en fonction des coordonnées *i* et *j* d'un point ;
- le nombre de points de contour de l'image : *nbEdgP* ;
- le nombre de points de l'image ne faisant pas partie du contour : $\overline{nbEdgP}$. On les appellera aussi points de fond de l'image ou d'arrière plan.

On a la relation : $\overline{nbEdgP} + nbEdgP = M \cdot N$ (Eq. 2.1)

On présume que tous les algorithmes présentés donnent en résultat une image représentant les contours en blanc sur un fond noir.

Une opération mathématique nommée convolution est fréquemment utilisée dans le traitement des images. Si *m* est une matrice de convolution de taille 3x3 alors la valeur résultante $I_c(i, j)$ vaudra :

$$I_c(i, j) = \sum_{u=-1}^{u=1} \sum_{v=-1}^{v=1} m(u+1, v+1) \cdot I(i, j) \qquad \text{(Eq. 2.2)}$$

Cette opération est utilisée pour filtrer les images : suivant les valeurs des matrices de convolution, on peut filtrer le bruit, rendre l'image plus floue, ou au contraire accentuer les contours, augmenter les contrastes.

Après ces définitions, on aborde maintenant quelques points généraux concernant le traitement des images.

2.2.1. Conversion des images

Dans une image en niveaux de gris, le niveau de luminosité est codé pour chacun des points. La fonction $f(i, j)$ renvoie cette information pour un point de coordonnées *i* et *j*. La conversion d'une image en couleurs en une image en niveau de gris est possible par l'utilisation de règles qui permettent d'évaluer la luminosité à partir des informations du système de codage original : CYG, RGB.

2.2.2. Représentation des gradients de luminosité

Le gradient de luminosité en un point est la différence entre la luminosité en ce point et celle de ses voisins, directement adjacents ou non.

La sensibilité de l'œil humain et celle des périphériques d'impression sur l'ensemble de toutes les valeurs de luminosité possibles ne sont pas égales. La visibilité des contours étant liée à la fois à l'amplitude des gradients et au positionnement du contour sur l'échelle des niveaux de gris, une image imprimée ne permet pas toujours un rendu équivalent de tous les contours. En général, dans ce rapport, une représentation numérique comme celle de la table 2.1 sera adoptée. Elle permet une meilleure appréhension des petites variations car elle les représente toutes d'une manière homogène.

Coordonnées des pixels	1	2	3	4	5	6	7
1	5	7	6	4	153	148	149
2	3	6	9	10	145	150	136
3	5	7	30	60	90	120	150

Table 2.1 : Exemples de contours.

Cependant l'utilisation d'une telle représentation est irréaliste pour des images entières. Nous n'utiliserons donc cette méthode que pour la représentation des détails [BELLET *et al.* 1998]. On peut utiliser une autre méthode pour représenter les variations de luminosité dans l'image en dessinant le graphique de la fonction $f(i, j)$ en une ou deux dimensions comme dans la table 2.1.

2.3. Passage en revue et comparaison des algorithmes existants

Ce chapitre décrit et compare des approches différentes. Il existe une très grande diversité d'algorithmes qui apportent des réponses aux points faibles des algorithmes précédemment inventés. Cet inventaire n'est donc pas exhaustif. Nous avons fait le choix de restreindre l'étendue de notre comparaison à sept d'entre eux parce qu'ils forment un échantillon représentatif de ceux qui sont le plus utilisés (Sobel, Canny), et d'approches théoriques différentes : analyse de la dérivée première ou seconde, calcul statistique, flou ou basé sur les réseaux de neurones.

Les filtres d'extraction des contours sont tous basés sur le traitement du calcul du gradient entre des points adjacents, mais leur complexité est variable : si Sobel et Laplace ne calculent qu'une indication de la qualité du contour pour chacun des points, des étapes supplémentaires sont implémentées dans l'algorithme de Canny ou le CFED qui permettent une analyse plus fine.

Chacune de ces méthodes sera donc ici présentée puis comparée aux autres.

2.3.1. Sobel

L'opérateur de Sobel est un algorithme de détection de contours simple utilisant le calcul de la première dérivée de la fonction "niveau de gris" des points d'une image. Il recherche des maximums des variations de niveau de gris [BOW 2002, EFFORD 2000]. L'opérateur est basé sur l'utilisation de deux masques de convolution de 3*3 pixels qui sont appliqués à l'image originale pour produire une carte des gradients de niveau de gris. Les zones les plus claires sont représentatives de secteurs

où l'intensité de niveau de gris varie rapidement dans un petit intervalle. Elles représentent donc des zones de contours sur l'image originale.

Avec un masque de 3*3 pixels centré sur le point de coordonnées (*i, j)*, le gradient selon l'axe *y* est :

$$\partial f(x,y)/\partial y = \left[f(i,j+1) - f(i,i) + f(i,i) - f(i,j-1)\right]/2 = \left[f(i,j+1) - f(i,j-1)\right]/2 \qquad \text{(Eq. 2.3)}$$

Si on ignore le facteur ½, le masque de convolution devient : $\begin{vmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & +1 & 0 \end{vmatrix}$

Des masques similaires peuvent être calculés selon ces autres axes : *x*, *x=y* et *x=-y* qui donnent respectivement :

$$\begin{vmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{vmatrix}, \begin{vmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & +1 \end{vmatrix}, \begin{vmatrix} 0 & 0 & +1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{vmatrix}$$

Si ces masques sont associés, on obtient les masques de Prewitt :

$$\begin{vmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{vmatrix}, \begin{vmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{vmatrix}$$

Pour améliorer leurs résultats, l'idée de Sobel est de donner plus d'importance aux variations sur les axes x et y en leur associant un poids plus grand. Les deux masques sont alors utilisés pour établir des cartes de gradients selon les axes horizontaux et verticaux.

$$\text{Sob}_{\text{horizon}} = \begin{vmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{vmatrix} \cdot I \text{ et } \text{Sob}_{\text{vertical}} = \begin{vmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{vmatrix} \cdot I \qquad \text{(Eq. 2.4)}$$

Finalement, le degré de contour en un point est calculé comme ceci :

$$\text{Edge}(i,j) = \sqrt{\text{Sob}_{\text{horizon}} + \text{Sob}_{\text{vertical}}} \qquad \text{(Eq. 2.5)}$$

Cette information peut aussi être utilisée pour générer une carte de la direction des contours en chaque point de l'image. En prenant soin de conserver les signes de valeurs de $\mathrm{Sob}_{\mathrm{horizontal}}$ et de $\mathrm{Sob}_{\mathrm{vertical}}$, on calcule alors la direction de la tangente de la courbe de contour en un point :

$$\theta = \arctan(\mathrm{Sob}_{\mathrm{vertical}} / \mathrm{Sob}_{\mathrm{horizon}}). \qquad \text{(Eq. 2.6)}$$

Cependant, ceci ne peut pas être calculé si $\mathrm{Sob}_{\mathrm{horizon}}$ est nul. Dans ce cas, la valeur de $\mathrm{Sob}_{\mathrm{vertical}}$ doit être considérée : si elle est nulle aussi, alors θ sera égal à 0, sinon il sera égal à 90 degrés.

2.3.2. Laplace

Cet algorithme de détection de contours est basé sur l'étude de la dérivée seconde de la fonction d'intensité des niveaux de gris [EFFORD 2000, BOW 2002] :

$$\partial^2 \mathrm{f}(x,y) = \partial^2 \mathrm{f}(x,y)/\partial x^2 + \partial^2 \mathrm{f}(x,y)/\partial y^2 \qquad \text{(Eq. 2.7)}$$

En rappelant que la dérivée seconde en un point peut être estimée comme la différence entre les dérivées aux deux points adjacents, on peut alors la calculer ainsi selon l'axe *y* :

$$\partial^2 \mathrm{f}(x,y)/\partial y^2 = \left[(\mathrm{f(i,j+1)}-\mathrm{f(i,j)})/\alpha - (\mathrm{f(i,j)}-\mathrm{f(i,j-1)})/\alpha\right]/\alpha = \left[\mathrm{f(i,j+1)}+2\cdot\mathrm{f(i,j)}-\mathrm{f(i,j-1)}\right]/\alpha^2 \qquad \text{(Eq. 2.8)}$$

Si on ignore le facteur α^2, le masque est calculé comme la somme des masques horizontaux et verticaux, soit :

$$\begin{vmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{vmatrix} + \begin{vmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{vmatrix} = \begin{vmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{vmatrix}$$

Si on considère aussi les axes *x=y* and *x=-y*, on obtient le masque de Laplace : $\begin{vmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{vmatrix}$

L'utilisation du masque de Laplace peut être étendue à un autre usage que la détection de contours : En ajoutant un masque dit "identité" au masque de Laplace et en appliquant cette convolution à l'image originale, on va en accentuer les contours qui deviendront donc plus apparents

que sur l'image originale. C'est une méthode couramment employée pour améliorer une image un peu floue.

2.3.3. Canny : le calcul mathématique.

L'équipe de John Canny [CANNY 1986, MARTIN *et al.* 2000] a commencé par donner une définition plus précise de ce qu'est un contour dans une image et donc ce que doit faire un algorithme de détection de contours :

- L'algorithme de détection de contours ne doit détecter que les contours. Il doit tous les détecter et n'en oublier aucun ;
- La distance entre le contour détecté et sa position réelle devra être aussi faible que possible ;
- Il ne doit pas détecter plusieurs contours, là où il n'y en a qu'un.

Mathématiquement, cette définition se traduit par trois équations :

- Une détection exacte : $\Sigma = \frac{\int_0^{\infty} f(x)\,dx}{\sqrt{\int_{-\infty}^{\infty} f^{2}(x)\,dx}}$ (Eq. 2.9)
- Une localisation parfaire : $\Lambda = \frac{\left| f^{'}(x)\,dx \right|}{\sqrt{\int_{-\infty}^{\infty} f^{'2}(x)\,dx}}$ (Eq. 2.10)
- L'unicité de la réponse : $\frac{\left| f^{'}(0) \right|}{\sqrt{\int_{-\infty}^{\infty} f^{''2}(x)\,dx}} = k \cdot \frac{\int_{-\infty}^{0} f(x)\,dx}{\sqrt{\int_{-\infty}^{\infty} f^{2}(x)\,dx}}$ (Eq. 2.11)

La minimisation de ces critères dans le contexte du modèle proposé débouche sur une équation différentielle dont la solution est de la forme [CANNY 1986] :

$$h(x) = a_1 \cdot e^{x/\sigma}.\cos(w \cdot x) + a_2 \cdot e^{x/\sigma} \cdot \sin(w \cdot x) + a_3 \cdot e^{-x/\sigma} \cdot \cos(w \cdot x) + a_4 \cdot e^{-x/\sigma} \cdot \sin(w \cdot x) + C \quad \text{(Eq. 2.12)}$$

L'implémentation décompose le processus de détection en six étapes principales [CANNY 1986]:

1. Dans un premier temps, il filtre le bruit présent sur l'image originale en appliquant un filtre Gaussien standard. Plus la largeur de ce filtre est grande, plus la sensibilité au bruit du détecteur de contours est réduite, mais, en contrepartie, la localisation des contours sera alors moins précise.

2. L'étape suivante consiste à calculer un degré de contour par rapport aux directions horizontales et verticales en se basant sur l'algorithme de Sobel. Une approximation du degré de contour est alors calculée à partir de la formule suivante :

$$|G| = |\text{Sob}_{\text{horizon}}| + |\text{Sob}_{\text{vertical}}| \quad \text{(Eq. 2.13)}$$

On peut alors calculer la direction des contours.

3. Lorsque la direction des contours est connue, il s'agit de corréler l'indication de direction du contour avec une direction cohérente par rapport aux points voisins. On considère alors une fenêtre de 5x5 pixels :

x x x x x
x x x x x
x x a x x
x x x x x
x x x x x

Si on observe la configuration autour du point a, on remarque qu'il ne peut y avoir que 4 directions de contours dans son voisinage immédiat.

- 0 degrés, si le contour est horizontal ;
- 45 degrés, si le contour suit l'axe $x=y$;
- 90 degrés, si le contour est vertical ;
- 135 degrés, si le contour suit l'axe $x=-y$.

4. Donc, suivant la valeur calculée de l'orientation du contour, on simplifiera les calculs en lui associant la valeur possible la plus proche. De plus, connaissant les valeurs des tangentes des angles médians (22,5 ou 67,5 degrés), on n'aura pas besoin de calculer le résultat de la fonction *arctan()* qui est relativement coûteuse en ressource. Par exemple, si on calcule que l'angle du contour en un point est 6 degrés, alors on prendra en compte que l'angle du contour est égal à zéro en ce point.

5. Lorsque la direction des contours est connue, on peut alors appliquer un algorithme de suppression des points qui ne sont pas des maximums. Le but est de tracer un contour plus fin en privilégiant les points du contour qui sont dans la direction du tracé du contour.

6. Finalement, les stries sont éliminées en utilisant un filtre avec seuillage par hystérésis. Les stries sont un effet secondaire d'un filtre à seuil unique : elles sont dues à la fluctuation des résultats autour de la valeur de seuil. Si un seul seuil S1 est utilisé pour le filtrage de l'image, et que les points du contour ont une valeur moyenne proche de S1, alors, à cause du bruit, certaines instances des points du contour auront une valeur inférieure au seuil. La ligne de contour ne sera alors pas continue. Pour éviter ceci, le seuillage par hystérésis utilise deux seuils : un haut et un bas. Tout point ayant une valeur supérieure à S1 est considéré comme faisant partie du contour. Par la suite, tous ses points voisins dont la valeur est supérieure à S2 sont aussi considérés comme points de contour.

2.3.4. Deriche : l'optimisation mathématique.

R. Deriche a cherché à implémenter un filtre dont les temps de calcul seraient proportionnels à la taille de l'image, quel que soit le type d'opérateur utilisé (en comparaison aux techniques de l'époque qui pouvaient utiliser des opérateurs de transformation de l'image avant d'effectuer la détection de contours afin d'en améliorer les résultats). Il s'est basé sur les résultats de Canny et l'utilisation d'un filtre gaussien et sur une optimisation proposée par Shen et Casta en 1987 [DERICHE 1990].

Il propose un filtre dont la dérivée est la solution exacte de l'équation de Canny étendue aux filtres à supports infinis. Son expression mathématique est la suivante :

$$h(x) = k \cdot (\alpha \cdot |x| + 1) \cdot e^{-\alpha \cdot |x|} \text{ où } \alpha = \frac{\sqrt{\Pi}}{\sigma} \text{ dans l'équation 2.12} \qquad \text{(Eq. 2.14)}$$

$$\text{et } k = \frac{\left(1 - e^{-\alpha}\right)}{\left(1 + 2 \cdot \alpha \cdot e^{-\alpha} - e^{-2\alpha}\right)} \qquad \text{(Eq. 2.15)}$$

$$\text{On dénote aussi : } h'(x) = -k' \cdot x \cdot e^{-\alpha \cdot |x|} \text{ et } k' = \frac{\left(1 - e^{-\alpha}\right)^2}{e^{-\alpha}} \qquad \text{(Eq. 2.16)}$$

2.3.5. Le calcul par réseau de neurones : Bezdek

Le filtre de Bedzek est basé sur l'utilisation d'un jeu de test pour l'apprentissage d'un réseau de neurones [BEZDEK *et al.* 1994]. En utilisant une fenêtre carrée de 3 points sur 3 points, Bedzek a utilisé l'opérateur de Sobel pour évaluer toutes les combinaisons possibles soit environ 2^9 possibilités pour une image binaire. Ce jeu de test a été soumis à un réseau de neurones lors de la

phase d'apprentissage. L'appartenance à un contour est calculée comme une mesure floue comprise entre 0 et 1. Le réseau de neurones comprenant une ou deux couches cachées est entraîné pour calculer une mesure d'appartenance à la classe contour pour chaque fenêtre.

Cette approche démontre la puissance des réseaux de neurones : elle fournit un excellent détecteur de contours avec des résultats équivalents au filtre de Sobel sur des images binaires, mais produit des meilleurs résultats sur des images en niveaux de gris avec un faible contraste.

Cependant, cette méthode présente une limite liée au concept même : le filtre de Sobel étant utilisé pour l'apprentissage, les résultats seront au mieux équivalents. Le but de ce filtre n'est pas d'obtenir un progrès de la qualité de la détection mais une diminution de la durée du calcul.

2.3.6. Le calcul par logique floue : CFED

La logique floue peut aussi être utilisée pour la détection de contours. L'algorithme "*Competitive Fuzzy Edge Detector* " est appelé CFED soit "détecteur de contours par concurrence floue". Premièrement, il classe les points en deux catégories : "contour" et "non-contour" [LIANG *et al.* 2003]. Afin d'affiner le tracé des contours, il applique ensuite des règles de compétition entre les points selon leur classification.

Ce filtre agit donc en deux étapes successives. Durant la première, la classification est basée sur la logique floue. Six classes sont définies : les contours horizontaux et verticaux, les contours obliques suivant les axes *x=y* et *x=-y*, une classe "texture homogène" ou "fond de l'image", une classe "texture mouchetée". Les fonctions d'entrée de l'étape de fuzzification sont soit des fonctions Gaussiennes ($K(u) = \frac{1}{\sqrt{2\Pi}} \cdot e^{-\frac{1}{2} \cdot u^2}$) soit des fonctions d'Epanechnikov étendues ($K(u) = \frac{3}{4} \cdot \left(1 - u^2\right)$). La valeur du sous-ensemble flou qui est attribuée à chacun des points dépend des valeurs normalisées des vecteurs calculés pour chacun de points comme suit.

Pour chacun des points, on obtient le vecteur *(d(1), d(2), d(3), d(4))* en calculant la différence de luminosité avec chacun des points directement adjacents suivant les quatre directions principales.

$$d(1) = |f(a_{1,2}) - f(a_{2,2})| + |f(a_{3,2}) - f(a_{2,2})|, \ldots, d(4) = |f(a_{1,1}) - f(a_{2,2})| + |f(a_{3,3}) - f(a_{2,2})| \quad \text{(Eq. 2.17)}$$

Chaque vecteur est évalué en calculant son appartenance à chacune des six classes floues. La valeur maximum détermine la classe à laquelle le point appartient.

Durant la seconde étape, dénommée phase de concurrence les points de contour situés dans un voisinage proche sont évalués comparativement. Si un point de contour est perdant, il est considéré alors comme un point du fond de l'image. Au final, on obtient une ligne noire sur un fond blanc.

Ce processus met aussi en évidence les mouchetures comme des points ou des couples de points de contour isolés. Un algorithme spécifique est employé par la suite pour les éliminer.

L'algorithme est décrit ci-dessous :

Classification Fuzzy

Paramétrage des fonctions d'appartenance des points aux classes floues.

Ouverture de l'image.

Pour chaque point de l'image :

* calculer les variations de niveau de gris dans les quatre directions principales ;
* construire le vecteur caractéristique de chaque point ;
* pour chaque classe, calculer la valeur floue correspondant au vecteur caractéristique ;
* choisir la valeur floue ayant la valeur maximum comme représente de l'appartenance du point à cette classe.

Phase de concurrence

* Si (classe contour) alors appliquer l'algorithme de compétition ;
* Si (classe "fond de l'image") alors marquer un point noir ;
* Si (classe "moucheture") alors marquer un point blanc.

Suppression des mouchetures isolées

Pour chaque point de l'image :

* Si (un point ou un couple de point est isolé) alors marquer un point noir.

2.3.7. La méthode des contours actifs dite des serpents

Cette méthode considère qu'un contour est la ligne de délimitation entre deux zones de l'image d'énergies différentes. Elle autorise l'utilisateur à guider l'outil de détection de contours en lui proposant des débuts de solutions ou des améliorations par rapport aux solutions déjà trouvées. Par la suite, ce modèle cherche à se rapprocher au plus près d'un contour dont on lui a indiqué la

proximité. Cette interaction est justifiée parce que souvent l'interprétation des images n'est pas un processus lié uniquement à l'observation intrinsèque des pixels. L'expert apporte lui aussi une information de plus haut niveau aidant à la compréhension globale de l'image et déterminante pour une détection des contours utile dans son domaine d'expertise.

La ligne de contours est un comparée à un serpent. Il est ouvert ou fermé. La recherche du meilleur contour s'apparente à la minimisation des contraintes sur ses faces interne et externe. La somme de ces contraintes sur l'ensemble de l'image est le calcul de l'énergie de l'image. On peut l'exprimer ainsi [KASS *et al.* 1987] :

$$E_{totale} = \int_0^1 \left[E_{\text{int}\,erne}(v(s)) + E_{image}(v(s)) + E_{externe}(v(s)) \right] ds \qquad \text{(Eq. 2.18)}$$

La méthode utilisée pour le calcul de chacun des trois membres va influencer les caractéristiques du contour. Par exemple, si on définit

$$E_{\text{int}\,erne} = \tfrac{1}{2} \cdot \left(\alpha(s) \cdot \left(\frac{dv}{ds} \right)^2 + \beta(s) \cdot \left(\frac{d^2 v}{ds^2} \right)^2 \right), \qquad \text{(Eq. 2.19)}$$

alors le terme du premier ordre influencera la tension et celui du second ordre la courbure ou l'élasticité du contour.

En basant le terme $E_{externe}$ sur une fonction des gradients locaux du contour, la minimisation de l'énergie permet d'interpoler des portions de contours manquants sur l'image originale.

De 1987 à 1998, cette approche a été enrichie par les apports de Xu et Prince [XU *et al.* 1987]. Ils proposent une évaluation de l'énergie externe par un champ de vecteurs indépendant de la position du contour (de son évaluation à chaque itération de l'algorithme). Ce champ est aussi beaucoup plus large que dans la proposition de Kass, ce qui donne moins d'importance aux minimums locaux.

2.3.8. Comparaison théorique des algorithmes

Les opérateurs de Sobel et de Laplace donnent généralement de bons résultats même si ils manquent de précision et de continuité dans le tracé des contours [SALOTTI *et al.* 1996, MARTIN *et al.* 2000). L'opérateur de Sobel n'est pas isotropique. Il est sensible à l'orientation du contour en un point. Cette propriété est d'ailleurs utilisée pour produire une information sur la direction des contours. Il est très efficace pour détecter les lignes horizontales et verticales mais peut oublier certains coins ou des contours arrondis ou en diagonale. Il est aussi très sensible au bruit.

L'opérateur de Laplace présente certains désavantages : il ne donne aucune indication sur la direction du contour en un point et amplifie le bruit, polluant l'image avec des mouchetures. Si les calculs de dérivées premières (simple différence) sont sensibles au bruit, ceux de dérivées secondes le sont encore plus. De petites erreurs dans les données de l'image originale peuvent produire un résultat avec de grandes erreurs. Une atténuation du bruit devrait être envisagée avant la détection des contours.

L'opérateur de Laplace étant basé sur la détection du croisement de l'axe $y=0$ de la dérivée seconde, il produit un tracé de contour fin et le positionne à l'endroit de la plus grande variation de luminosité. Mais dans le cas où la variation de luminosité n'est pas uniforme, l'opérateur de Laplace peut produire un double tracé. Si on utilise un filtre Gaussien en pré traitement avant la détection de contours, on donne un peu de flou et on diminue la sensibilité au bruit. Mais on réduit aussi la sensibilité du filtre : certains petits détails peuvent devenir insignifiants et l'emplacement réel des contours peut varier. La qualité de la détection est d'autant plus réduite en utilisant un tel filtre que l'image originale est claire avec des contours francs, bien marqués.

Habituellement, une étape additionnelle est effectuée après l'utilisation d'un de ces algorithmes : On utilise un filtre avec seuil pour ne garder que les points qui donnent une indication maximum de contour. Plus le seuil du filtre est bas, plus des points sont sélectionnés et les résultats, pollués par le bruit, mettront en évidence des caractéristiques inexistantes de l'image. Inversement, les lignes les plus subtiles et les détails seront oubliés si ce seuil a une valeur trop grande.

La complexité des algorithmes doit aussi être prise en compte si on veut les comparer. Deux classes existent : les opérateurs de Sobel ou Laplace sont les plus simples puisque qu'ils permettent d'obtenir un résultat après respectivement une convolution simple ou double d'un masque sur

chacun des points de l'image. L'approche neuronale est difficilement comparable : si le calcul est très rapide, la phase d'apprentissage doit aussi être prise en considération et elle relativement longue [HARVEY *et al.* 1997].

L'opérateur de Canny détecte presque tous les contours mais il a tendance à ne pas voir certaines jonctions en Y. Le tracé du contour est fin et n'est pas très sensible au bruit. Selon Liang [LIANG *et al.* 2003], l'algorithme CFED est beaucoup plus rapide que celui de Canny. Toutes ces méthodes sauf CFED doivent être utilisées en combinaison avec un filtrage postérieur à la détection du contour. Cependant, les paramètres de ce dernier filtre doivent être adaptés à la qualité de l'image originale et aux besoins de l'application à laquelle la détection de contour est destinée.

Les détecteurs à contours actifs ne sont pas sensibles au bruit, mais ils présentent quelques faiblesses : les paramétrages initiaux, valeurs très sensibles, doivent être initialisés manuellement pour optimiser le résultat, la convergence est très dépendante du placement initial du contour (fausses détections sur minimums locaux) et ils ne peuvent pas détecter les concavités (dans leur version initiale). Si ceux basés sur les champs de vecteur gradient (dits GVF) permettent de palier à ce dernier problème, ils sont très gourmands en temps de calcul. Par contre, ils permettent de détecter les contours d’objets alors que l’objet à détecter n’était pas entièrement contenu dans le contour initial [XU *et al.* 1997]. Parce qu’elles nécessitent des informations sur les zones à analyser lors de leur initialisation, on observe que ces deux méthodes sont utilisées en complément d’autres techniques de détection de contours. Elles n’ont pas le même but que le filtre pseudo-statique et nous ne les comparerons pas aux autres méthodes de filtrage dans la suite de cette étude.

2.4. Logique floue

Ce chapitre est une présentation des principes qui régissent la logique floue. Par la suite, nous utiliserons ces notions pour expliquer le fonctionnement du filtre pseudo-statistique et définir les méthodes qui nous permettront d'optimiser le paramétrage de ce filtre.

2.4.1. Historique et impact

L'idée de base de la structure de la logique floue, les "sous-ensembles flous" a été proposée par le professeur en automatisme L. A. Zadeh dans [ZADEH 1965]. Etant confronté aux limites des modèles classiques à équation différentielle, il chercha à modéliser ces phénomènes en introduisant l'idée d'incertitude. Il construisit alors une théorie rigoureuse pour traiter de ce qui est subjectif, incertain. Si la logique est basée sur les prédicats VRAI ou FAUX, la logique floue étend ce concept en créant des valeurs possibles entre le VRAI et le FAUX, des probablement, des "presque" qui s'expriment sous la forme de valeurs numériques comprises dans l'intervalle $[0,1]$.

Les principaux domaines d'applications étaient alors ceux de l'automatisme classique, soit l'asservissement ou la régulation de processus industriels. Progressivement des améliorations sensibles ont été ensuite réalisées, entre autres au Japon grâce aux professeurs M. Mamdani et M. Sugeno dans les années 1970 et 1980 [SUGENO 1985].

On peut prendre comme exemple un processus de commande de température d'un liquide. Classiquement, on va décider de le refroidir ou de le réchauffer en calculant à partir de l'information numérique donnée par la sonde de température quels moyens doivent être mis en œuvre pour atteindre la température désirée. Avec la logique floue, on va traduire l'information numérique en utilisant la sémantique habituelle de l'expert maîtrisant habituellement ce processus. En utilisant cette même sémantique, on va formaliser les règles qu'utilise cet expert pour contrôler le système. Ces règles sont alors utilisées par un moteur d'inférence pour indiquer quels sont les moyens à mettre en œuvre pour agir sur le système en fonction des variables d'entrée. La dernière phase consiste alors à traduire ces consignes en valeurs numériques correspondant à une action réelle sur le système.

Ce fonctionnement a l'avantage d'être plus proche de la démarche cognitive du cerveau humain que la démarche de traitement purement numérique mise en œuvre par des calculateurs aussi puissants soient-ils. Les avantages de logique floue sont principalement :

- sur un plan économique : la simplification des phases de modélisation et d'identification ;
- sur un plan organisationnel : une mise en forme et une mémorisation du savoir sous forme d'une base de connaissances synthétisant une expérience humaine à travers les règles d'inférence ;
- sur un plan technique : la robustesse, la facilité d'évolution et de modification, les performances dues entre autres à l'utilisation de processeurs spécialisés.

Les applications basées sur la logique floue sont aujourd'hui relativement nombreuses même si le grand public n'a pas forcément conscience de leur présence :

- appareils électroménagers (lave-linge, aspirateurs, etc.) ;
- systèmes audio-visuels (appareils de photos autofocus, caméscope à stabilisateur d'images, photocopieurs, etc.) ;
- systèmes automobiles embarqués (ABS, suspension, climatisation, etc.) ;
- systèmes autonomes mobiles ;
- systèmes de transport (train, métro, ascenseur, etc.) ;
- systèmes de conditionnement d'ambiance ;
- systèmes de décision, diagnostic, reconnaissance ;
- systèmes de contrôle/commande dans la plupart des domaines industriels de production, transformation, traitement de produits et déchets.

2.4.2. Principes de fonctionnement

La logique floue est donc une extension de la logique prenant en compte une incertitude sur les variables d'entrée du système. Elle sert à représenter des connaissances incertaines et imprécises. Cette représentation est alors utilisée par la commande floue qui sert à prendre une décision même si l'on ne peut estimer les entrées/sorties qu'à partir de prédicats vagues ou lorsque ces entrée/sorties sont entachées d'erreurs que l'on ne peut évaluer que grossièrement.

On constitue une chaîne de commande et de contrôle complète composée d'un module d'évaluation des variables d'entrée, d'un ensemble de règles définissant le système, d'un moteur d'inférence et d'un module de conversion des résultats en commande.

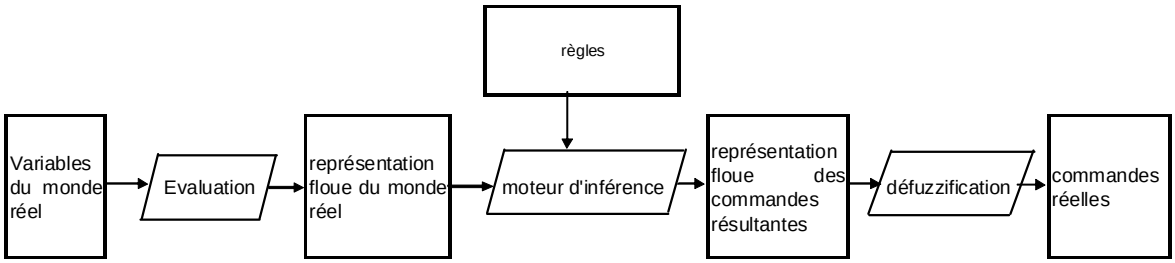


Figure 2.2 : Principe de calcul en logique floue

Les opérations d'évaluation (ou fuzzification) et de défuzzification permettent de réaliser une conversion entre des valeurs quantitatives et les sous-ensembles flous. Elles sont le moteur de l'interface entre une information numérique et une information symbolique (linguistique et qualitative) de la réalité.

2.4.3. Evaluation des entrées- Fuzzification

Le but de cette étape est de traduire une mesure numérique un terme proche du langage des spécialistes. On associe à chaque valeur, une ou plusieurs qualité(s) et on associe à chacune un degré de confiance. Par exemple, on peut dire d'un liquide à 32 degrés Celsius qu'il n'est pas chaud, qu'il est plutôt tiède voire même un peu froid. Le module de fuzzification pourrait donc associer cette valeur de température aux classes d'appartenance "tiède" et "froid" en attribuant un degrés de confiance supérieur à la qualité "tiède" par rapport à la qualité "froid". Les classes d'appartenance de ce domaine sont "chaud", "tiède" et "froid". Elles sont représentées dans la figure 2.3. Le degré d'appartenance étant traduit dans le langage commun par l'utilisation d'adverbes comme "très", "plutôt" ou "pas du tout".

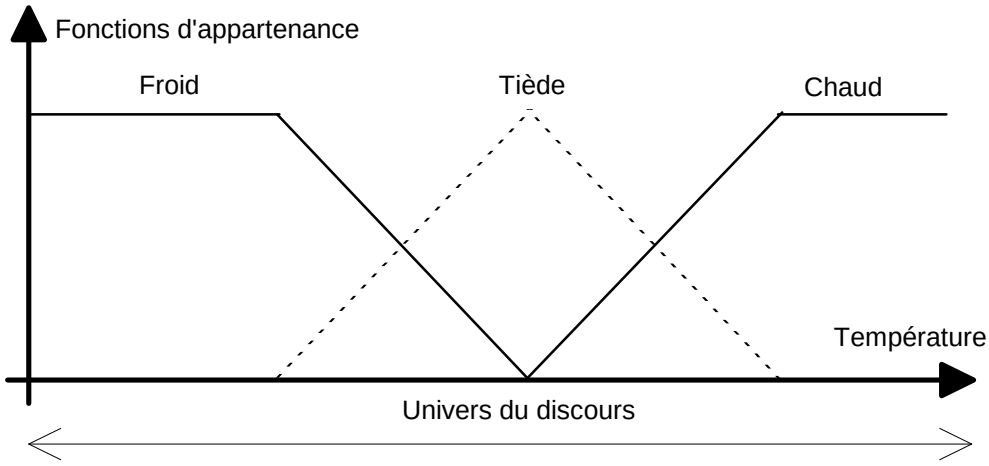


Figure 2.3 : Fonction de conversion du discours en paramètres flous.

La première étape de la définition d'un module de fuzzification consiste à répertorier la sémantique du domaine et à choisir un nombre de termes limité caractérisant chacun des aspects de la problématique. On étudie alors l'univers du discours qui correspond au domaine de fonctionnement du processus. On ne cherchera pas forcément à obtenir un inventaire exhaustif de toutes les nuances linguistiquement possibles pour caractériser un aspect du processus. On cherchera plutôt à obtenir une couverture complète du domaine de fonctionnement par trois, cinq ou sept prédicats bien choisis. Chacun des prédicats est représentatif de d'un sous-ensemble flou. L'union de tous les sous-ensembles flous couvrant tout l'univers du discours.

Les fonctions d'appartenance permettent alors de traduire la mesure quantitative en une approximation de l'appartenance à un sous-ensemble flou. Cette mesure n'est pas une évaluation statistique, une représentation de la qualité d'un individu par rapport à un grand nombre d'échantillons de la population totale. Cette mesure est une représentation empirique de cette caractéristique dans le cadre de cet opérateur d'évaluation. Elle est sujette à une certaine subjectivité.

Les fonctions d'appartenance sont habituellement soit triangulaires (figure 2.4), soit trapézoïdales, mais on peut aussi utiliser des fonctions trigonométriques ou en forme de cloche.

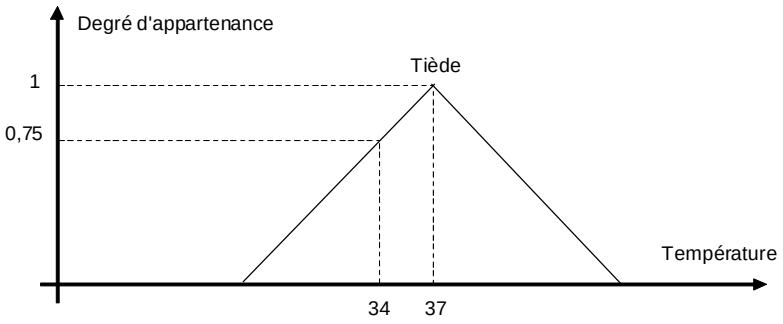


Figure 2.4 : Conversion à l'aide de fonction d'appartenance.

L'utilisation des quelques règles suivantes permettent d'assurer une meilleure efficacité du filtre d'entrée. Il faut :

- que l'intersection entre deux prédicats successifs soit non nulle, que la totalité des sous-ensembles flous couvre l'étendue du domaine de langage. Cela permet de pouvoir exercer une pondération sur la commande et éviter des zones .de non-intervention du régulateur, les zones d'instabilité du réglage ;
- que l'intersection entre deux prédicats successifs ne soit pas trop importante pour éviter les imprécisions ;
- que les fonctions d'appartenance de deux prédicats voisins ne soient pas égales à 1 en même temps.

On a alors moyen d'attribuer à chaque valeur réelle de chaque entrée un sous-ensemble flou qui sera utilisé en entrée du moteur d'inférence.

2.4.4. Règles et moteur d'inférence

En fonction des valeurs des paramètres en entrée du système, le moteur d'inférence déduit de l'ensemble des règles logiques les consignes à appliquer sur le système pour le rapprocher de l'état désiré. Il prend en compte d'une part la base de règles et d'autre part les sous-ensembles flous représentant l'état du système. La base de règles est une représentation de la logique qui régit le fonctionnement du processus exprimée dans les termes du spécialiste. Par exemple, "si le liquide est froid ajouter du liquide chaud". En général, les règles ne définissent les commandes que dans des termes qualitatifs et non pas quantitatifs.

La définition de ces règles passe forcément par la spécification de l'ensemble des commandes possibles. Comme pour l'étape de fuzzification, on définit des ensembles flous couvrant l'univers du discours du domaine de la commande. Ensuite, à l'écoute du spécialiste, on peut définir les règles : elles définissent la relation entre les causes et les commandes.

Si on a n ensembles flous, comprenant chacun m_i termes, on peut calculer le nombre $N_{\max}$ de règles maximum : $N_{\max}=\prod_{i=1}^{n} m_i$

La complexité du calcul du moteur d'inférence augmente très vite par rapport au nombre des prédicats. Pour limiter ceci, certaines règles peuvent ne pas être définies si elles représentent des cas impossibles. D'autres peuvent être regroupées si elles sont similaires.

Les déductions floues sont évaluées par le moteur d'inférence qui calcule un degré d'appartenance à un ou plusieurs élément(s) d'une classe de commandes. L'élément de chaque classe de commandes est défini par la règle. Le degré d'appartenance à cette classe de commandes est calculé suivant la conjonction des opérateurs d'implication et "ET", "OU", "NOT" qui forment la règle. Plusieurs approches différentes ont été proposées pour le calcul de la conjonction de plusieurs règles :

- Méthode d'inférence max-min ;
- Méthode d'inférence max-prod ;
- Méthode d'inférence somme-prod.

2.4.5. Evaluation des sorties- Défuzzification

Les méthodes d'inférences fournissent une fonction d'appartenance résultante pour la variable de sortie. Il s'agit donc d'une information floue. Il convient de la transformer en une grandeur de commande précise. C'est le but de l'étape de défuzzification. Les méthodes les plus couramment utilisées sont citées ci-dessous :

- La méthode du centre de gravité ;
- La méthode du maximum ;
- La méthode des surfaces ;
- La méthode des hauteurs ;
- La méthode de Sugeno.

2.5. Algorithme Génétique

2.5.1. Historique

La notion moderne de génétique est née des travaux de Charles Darwin et Johann Mendel. En effet, en 1860 Charles Darwin publie le livre « L'origine des espèces au moyen de la sélection naturelle ou la lutte pour l'existence dans la nature ». Darwin y présente la vie sur Terre comme une évolution continue où les espèces qui ont survécu sont celles qui ont su s'adapter aux changements environnementaux qui leur ont été imposés. Pour lui, les systèmes naturels ne sont pas figés. Ils sont capables d'adaptation en particulier lors du processus de reproduction. C'est la théorie de l'évolution des espèces.

De son côté, Johann Mendel, publia en 1865 un article présentant les bases théoriques de la génétique et de l'hérédité.

Durant le XXème siècle, la génétique s'est progressivement imposée comme une discipline scientifique à part entière du domaine de la biologie. Il s'est agit, par exemple, de démontrer le bien fondé de la théorie des mutations génétiques. En 1943, Les chercheurs Salvator Luria et Max Delbrück ont démontré que la probabilité de transmission de la faculté d'adaptation des cellules à un changement de milieu est supérieure au taux naturel de mutation confirmant ainsi certaines hypothèses de Charles Darwin.

Les découvertes de l'ADN, de la division cellulaire, des maladies génétiques ont été primordiales. Depuis les années 80, elles ont été suivies par la mise au point de processus et d'outils dont les applications les plus connues sont dans le champ de la médecine et de la biotechnologie (Organismes Génétiquement Modifiés, par exemple).

Dans le domaine informatique, le traitement de l'information s'est fait dans un premier temps de manière figée. Durant la phase de conception, en partant de l'analyse des conditions d'exploitation, on a résolu des problématiques en mettant en place des systèmes peu adaptables à des changements de conditions d'environnement, à des variables nouvelles ou évolutives.

Dans les années 60, la recherche en informatique s'est penchée sur ces limitations en proposant des algorithmes évolutifs [FOGEL *et al.* 1966]. L'approche consiste à simuler un système par des

automates à état finis et à en améliorer la qualité de réponse par la sélection des parents les plus efficaces.

Les algorithmes génétiques sont décrits en 1975 par John Holland [HOLLAND 1975]. Ils ont été popularisés par John Goldberg en 1989 [GOLBERG *et al.* 1989]. Nous en détaillons les principes dans la section ci-dessous.

Leur vulgarisation se fera durant les deux décennies suivantes et la réflexion amène aujourd'hui vers des nouvelles solutions comme la programmation génétique. Les domaines d'application sont très divers : industrie, économie.

2.5.2. Principes de fonctionnement

Le but d'un algorithme génétique est d'affiner une solution à un problème complexe sans connaître en détail le processus modélisé pour obtenir un résultat.

Durant la phase d'apprentissage, les paramètres du problème sont codés sous forme binaire. Ce codage est rassemblé dans une seule chaîne s'apparentant à un chromosome. Chaque individu représente une solution et est défini par un chromosome. On créé alors une population d'individus à tester et pour chacun d'entre eux on évalue la distance de la solution exacte au problème, c'est-à-dire la pertinence de la solution au problème qu'ils apportent. Pour faire évoluer la population, on applique les opérateurs génétiques de sélection, reproduction, croisement et mutation à la population.

Voici le principe de fonctionnement d'un algorithme génétique :

1) Les paramètres du problème sont codés sous forme binaire ;
2) On génère une population aléatoirement. Celle-ci contient un pool génétique qui représente un ensemble de solutions possibles ;
3) On calcule une valeur d'adaptation pour chaque individu. Plus l'individu sera représentatif d'une solution optimale, plus sa valeur sera grande ;
4) On sélectionne les individus devant se reproduire en fonction de leurs parts respectives dans l'adaptation globale ;
5) On croise les génomes des parents ;
6) On fait muter les nouveaux individus ;

7) Si les conditions d'arrêt ne sont pas atteintes, on soumet cette nouvelle population à une nouvelle itération du processus à partir du point 3.

Dans les sections suivantes, on présentera les choix possibles pour chacune des étapes de calcul qui peuvent être optimisées en fonction des besoins posés par le problème :

- la sélection ;
- la reproduction ou le croisement ;
- la mutation.

2.5.3. La sélection

Ce processus consiste à choisir les individus à partir desquels on va créer la génération suivante. En général, on cherchera à régénérer la population en remplaçant une proportion d'individus par leur descendance. Plusieurs approches sont possibles :

Méthode Monte Carlo :

Les parents sont sélectionnés en fonction de leur performance. Si un individu obtient de meilleurs résultats, alors plus grandes sont ses chances d'être sélectionné. On organise une loterie pour laquelle les individus les plus performants obtiennent plus de tickets que les autres. Cette méthode a deux inconvénients :

- si un très bon individu et généré, alors il aura tendance à influencer très rapidement le reste de la population ;
- Si l'ensemble des individus de la population ont atteint un bon niveau de performance alors, celle-ci deviendra relativement homogène et ne pourra plus évoluer.

Méthode basée sur le rang de l'individu dans la population :

Pour éviter de ne sélectionner que les meilleurs individus et d'obtenir trop vite une population très homogène, on peut normaliser les performances en attribuant les tickets de loterie en fonction du classement des individus plutôt que de leur niveau de performance. Dans ce cas, la convergence vers un optimum est plus lente.

Méthode basée sur une transformation de la performance de l'individu :

De même, il est possible d'appliquer une fonction linéaire ou exponentielle [LADD 1999] à la performance de chaque individu. Ceci permet alors de diminuer l'influence des individus les plus forts.

Sélection par tournoi :

On peut organiser des tournois entre des groupes d'individus de manière à n'en sélectionner qu'une certaine proportion. Cette méthode laisse une plus grande chance aux individus un peu moins performants de perdurer dans la nouvelle population.

Méthode élitiste :

Afin de protéger les meilleurs individus déjà détectés, on les transfère directement dans la nouvelle population. Puis, on applique une des autres méthodes de sélection pour créer le reste de la population.

2.5.4. Le croisement

Le croisement est dit aussi hybridation ou crossover [EMECHE 1994]. Il consiste à combiner deux individus (les parents) pour en ressortir deux autres individus (dits enfants). Ceux-ci ne seront pas forcément meilleurs que les parents. Il existe plusieurs variantes de cet opérateur, mais en général il consiste à couper en un ou plusieurs points les chromosomes de deux individus (aux mêmes endroits pour les deux individus) et à échanger les parties situées entre ces points (figure 2.5). Les points peuvent être générés aléatoirement et changés pour chaque individu, ou à chaque cycle. Pour des populations importantes, il semble que le crossover à deux points de rupture soit le plus adapté pour obtenir une convergence [BRIDGES *et al.* 1991].

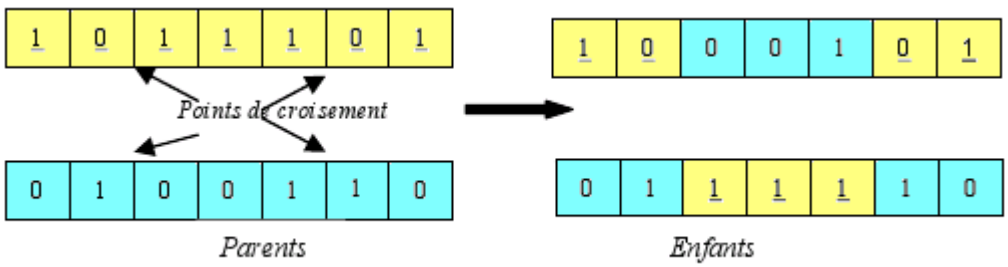


Figure 2.5 : Croisement à deux points de rupture

Dans la variante du croisement uniforme, on utilise un masque, sous la forme d'une chaîne de bits générée aléatoirement et de même longueur que les chromosomes des individus. Les gênes des

individus initiaux sont échangés en fonction de ce masque, si le bit de rang correspondant vaut 1 (figure 2.6). Le croisement uniforme de permutations a les avantages de la simplicité et d'une bonne efficacité [SYSWERDA 1989] surtout dans le cas de populations peu nombreuses [BRIDGES *et al.* 1991].

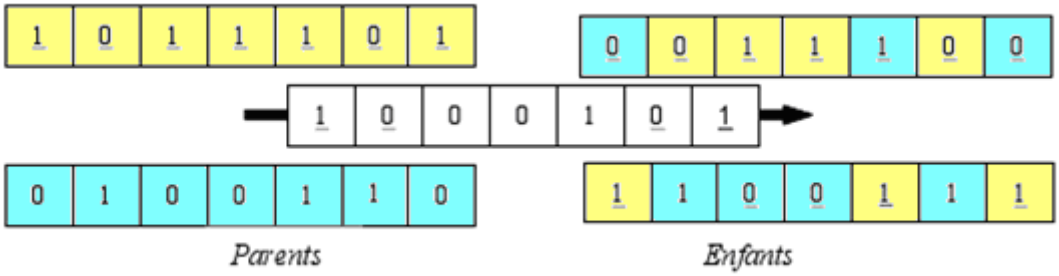


Figure 2.6 : Croisement uniforme de permutations

2.5.5. La mutation

La mutation consiste à modifier ou pas des bits du chromosome de chaque individu de manière à introduire une dose d'incertitude dans la création de la nouvelle génération d'individus. Ceci peut permettre la génération de nouveaux individus suffisamment différents de ceux déjà présents dans la population pour éviter de concentrer tous les individus sur un maximum local de la fonction étudiée (figure 2.7). Un taux de mutation de l'ordre de p=0,1% est généralement utilisé. S'il est plus important, il engendre des changements trop importants dans la population : l'opérateur de mutation devient prédominant par rapport aux autres opérateurs.

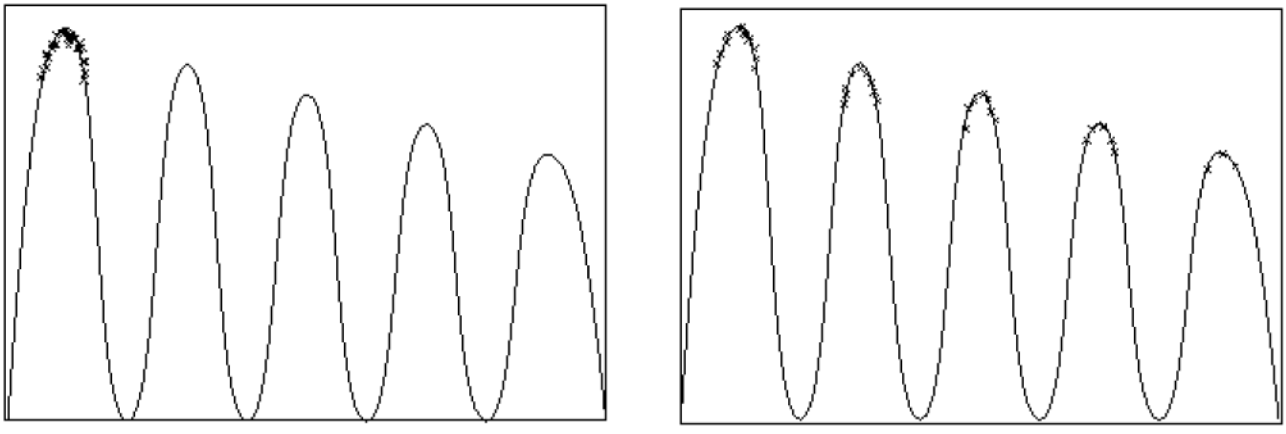


Figure 2.7 : Différence de concentration des individus d'une population sur une fonction

2.6. Conclusion

Ce chapitre a présenté différents supports théoriques nécessaires à la réalisation de ce travail. Dans un premier temps, nous avons rappelé les bases de l'objet de notre étude : la détection de contours puis nous avons exposé les principes de deux outils que nous seront amené à utiliser : la logique floue et l'algorithme génétique.

3. Un nouveau filtre de détection de contours.

Ce chapitre est destiné à donner les bases théoriques du filtre pseudo-statistique. Pour cela, on présentera les techniques sur lesquelles est basé ce filtre puis le fonctionnement du filtre en lui même. Dans une seconde partie, on estimera les gains apportés par un tel filtre en comparant ses résultats aux filtres existants dans des cas simples.

La détection de contours dans une image est un problème complexe à résoudre de manière classique où de multiples difficultés doivent être résolues [SALOTTI *et al.* 1996] :

- la sensibilité au bruit ;
- la détection des contours de faible gradient ;
- la détection des coins incurvés aussi finement que les lignes droites ;
- la détection des contours proches ;
- la préservation de la continuité des contours.

Par contre, dans le domaine médical, on peut noter que le problème des ombres dues aux éclairages des scènes n'existe pas contrairement au domaine de l'analyse de scènes, ou d'images aériennes par exemple.

3.1. Présentation du filtre

Le filtre pseudo-statistique a été proposé par le Professeur Teodorescu sous la forme d'un pseudo algorithme.

Procedure FUZZY_FILTER_1
for i=0 to N
  for j=0 to M
    compute Δ by the CORRECTION procedure
    $S_{i,j} \leftarrow S_{i,j} + \Delta$

END FUZZY_FILTER_1

Procedure CORRECTION
  Compute $\overline{S}_{i,j}^{LW}$
  Compute $\overline{S}_{i,j}^{SW}$
  Compute $d(S_{i,j}, \overline{S}_{i,j}^{LW})$
  Compute $d(S_{i,j}, \overline{S}_{i,j}^{SW})$
  Compute Δ' by the FUZZY_CORRECTION procedure.
END CORRECTION

La détection des points de contours s'effectue en deux étapes principales : Dans un premier temps, on calcule deux gradients moyens entre la luminosité en un point et celle de ses voisins de proximité immédiate ou ceux situés plus loin. Dans un second temps, le filtre flou évalue la qualité de point de contour de chacun des points.

L'architecture de ce filtre est présentée dans la figure 3.1:

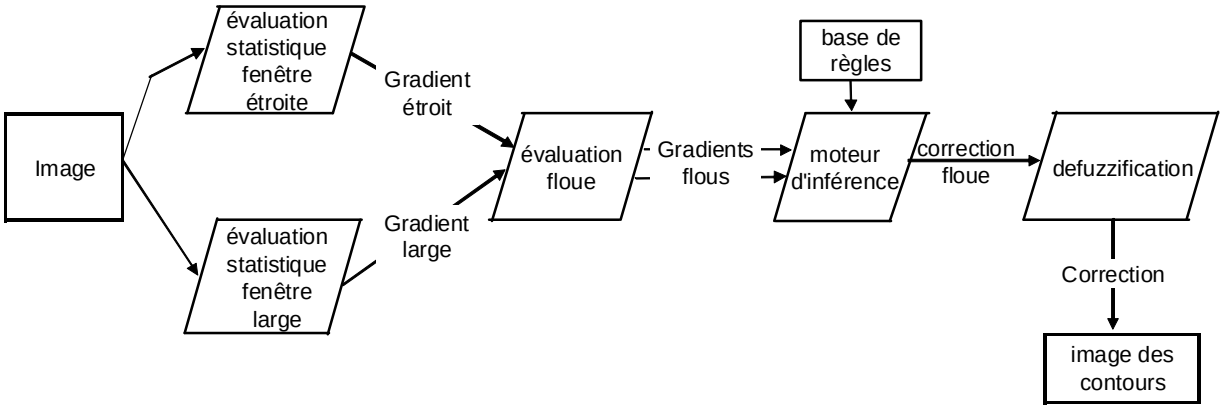


Figure 3.1 : Principe de fonctionnement du filtre de détection floue.

Calcul des gradients

Le calcul des gradients se calcule comme la distance moyenne. Celle-ci se définit comme la valeur absolue de la différence, entre la luminosité de chacun des points et la moyenne des luminosités des points d'une fenêtre, i.e. d'une partie, de l'image.

Ces gradients sont calculés sur deux fenêtres concentriques, dites aussi « fenêtres de convergence ». Si on regarde le dessin ci-dessous, ils seront calculés pour le pixel indiqué par le sigle "@" sur la petite et la grande fenêtre marquées respectivement par des pourtours clairs (c.f. figure 3.2).

Figure 3.2 : Présentation des fenêtres de convergence.

Sur la figure 3.2, on notera que le rayon de la petite fenêtre est égal à 1 et celui de la grande fenêtre est égal à 2. Ces gradients seront évalués pour chacun des points de l'image. Le point noté "A" représente un cas particulier qu'il faut signaler puisqu'il s'agit d'un point situé au bord de l'image. La petite et la grande fenêtre contiennent des points en dehors du cadre de l'image. Dans ce cas, ne seront pris en compte que les points de la fenêtre situés dans le cadre. On ne cherchera pas à faire une extrapolation de la valeur de la luminosité pour les points en dehors du cadre.

Le filtre flou : évaluation des entrées.

Le domaine de définition des gradients est décomposé en trois sous-domaines : "petit", "moyen" ou "large". Les degrés d'appartenance sont mesurés par un filtre composé de fonctions d'évaluation triangulaires. Les seuils de ces fonctions sont notés *a1* à *a7* comme indiqué sur la figure 3.3.

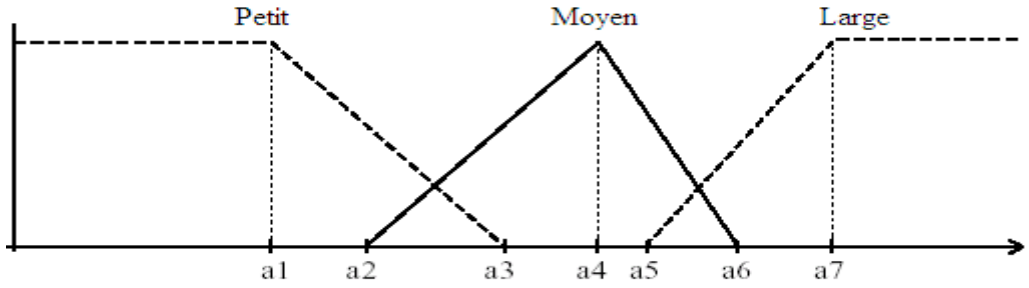


Figure 3.3 : Présentation des domaines d'appartenance.

Le filtre flou : règles et moteur d'inférence.

Pour faire correspondre une valeur floue (dénommée Δ'), à chacune des caractéristiques floues déterminées dans l'étape précédente, on utilise des règles :

$$Si\ d(S_{i,j},\overline{S}_{i,j}^{SW})\ est\ Large,\ et\ d(S_{i,j},\overline{S}_{i,j}^{LW})\ est\ Large,\ alors\ la\ correction\ \Delta'\ est\ TrèsLarge \qquad (Eq.3.1)$$

Ces règles peuvent être présentées sous la forme d'un tableau (c.f. table 3.1).

		$d(S_{i,j},\overline{S}_{i,j}^{LW})$		
	Δ'	S	Z	L
$d(S_{i,j},\overline{S}_{i,j}^{SW})$	S	+L	+S	+Z
	Z	+S	+Z	-S
	L	+Z	-S	-L

Table 3.1 : Règles d'inférence.

Le filtre flou : défuzzyfication.

A l'aide des règles d'inférence, on détermine une valeur numérique pour chacun des couples de distance (sur la grande et petite fenêtre). Ces données sont alors agrégées en se basant sur la méthode proposée par Sugeno [SUGENO 1985]. On peut la traduire par ces deux équations :

$$\lambda_k = f(SWG, LWG) = p_k \cdot SWG + q_k \cdot LWG + r_k \qquad \text{(Eq.3.2)}$$

$$\Delta = \frac{\sum_{i=1}^{k} x_k \cdot \lambda_k}{\sum_{i=1}^{k} \lambda_k} \qquad \text{(Eq.3.3)}$$

On choisit arbitrairement pour simplifier les calculs des valeurs nulles pour les variables p_k et q_k. La variable r_k. est issue de l'étape précédente (Δ').

3.2. Comparaison des filtres à partir de cas de test

Ici, nous voulons à travers des cas concrets comparer les caractéristiques des différents filtres de détection de contours, leurs points forts et leurs faiblesses. Nous observerons en détail la précision avec laquelle sont détectées certaines configurations particulières :

- les contours rectilignes verticaux ;
- les contours rectilignes en diagonale. Le but est de mettre en évidence le caractère non isotropique de l'algorithme de Sobel ;
- les contours aux abords d'un coin ;
- les contours aux abords d'une intersection.

Ces cas sont présentés dans la figure 3.4.

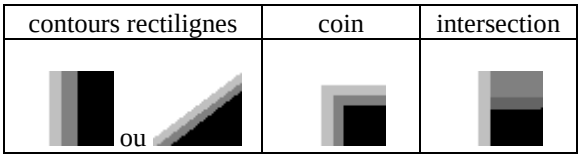


Figure 3.4 : Présentation des contours de test.

Les contours sont répartis sur 3 pixels. Ils sont progressifs et de gradient α. Dans le cas du contour rectiligne, on a donc une relation linéaire entre les niveaux de gris des pixels contigus :

$$I_{i-1,j} = I_{i,j} - \alpha \text{ et } I_{i+1,j} = I_{i,j} + \alpha \qquad \text{(Eq. 3.4)}$$

Nous exprimerons les valeurs calculées pour chacun des algorithmes sous forme d'un multiple du gradient α.

Les algorithmes étudiés sont celui de Laplace, de Canny, les différentes composantes de celui de Sobel et les gradients sur une petite fenêtre (dit « gradient SW ») et sur une grande fenêtre (dit « gradient LW »).

3.2.1. Détection de lignes verticales

Les valeurs numériques sont présentées dans le tableau 3.2.

	$I_{i-2,j}$	$I_{i-1,j}$	$I_{i,j}$	$I_{i+1,j}$	$I_{i+2,j}$
valeur pixel image	$I_{i,j}-\alpha$	$I_{i,j}-\alpha$	$I_{i,j}$	$I_{i,j}+\alpha$	$I_{i,j}+\alpha$
gradient horizontal Sobel	0	$4\cdot\alpha$	$8\cdot\alpha$	$-4\cdot\alpha$	0
gradient vertical Sobel	0	0	0	0	0
Sobel	0	$4\cdot\alpha$	$8\cdot\alpha$	$4\cdot\alpha$	0
Sobel – Canny	0	$4\cdot\alpha$	$8\cdot\alpha$	$4\cdot\alpha$	0
Laplace	0	$3\cdot\alpha$	0	$-3\cdot\alpha$	0
gradient SW	0	$4\cdot\alpha$	$6\cdot\alpha$	$4\cdot\alpha$	0
gradient LW	$8\cdot\alpha$	$18\cdot\alpha$	$20\cdot\alpha$	$18\cdot\alpha$	$8\cdot\alpha$

Table 3.2 : Détection d'une ligne droite (numérique).

La forme graphique ci-dessous donne un aperçu plus visuel de ces résultats (c.f. figure 3.5).

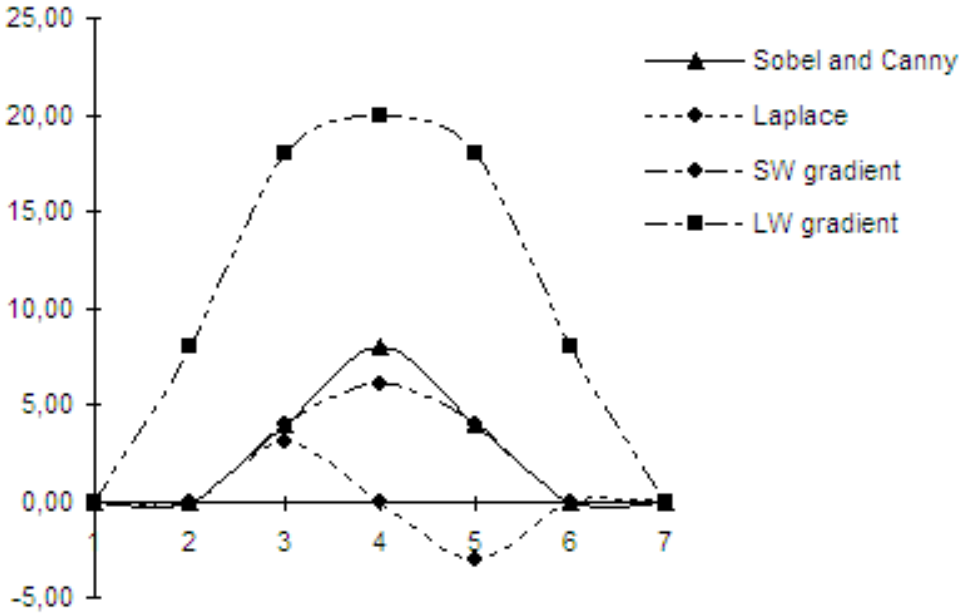


Figure 3.5 : Détection d'une ligne droite (graphique).

3.2.2. Détection de lignes obliques

La ligne en diagonale a une pente de 45° et passe par le point de coordonnées *i* et *j*. Les valeurs numériques sont présentées dans le tableau 3.3.

	$i-2$	$i-1$	i	$i+1$	$i+2$
$j+1$	$I_{i,j}-\alpha$	$I_{i,j}-\alpha$	$I_{i,j}-\alpha$	$I_{i,j}$	$I_{i,j}+\alpha$
j	$I_{i,j}-\alpha$	$I_{i,j}-\alpha$	$I_{i,j}$	$I_{i,j}+\alpha$	$I_{i,j}+\alpha$
$j-1$	$I_{i,j}-\alpha$	$I_{i,j}$	$I_{i,j}+\alpha$	$I_{i,j}+\alpha$	$I_{i,j}+\alpha$
gradient horizontal Sobel	$-\alpha$	$-4\cdot\alpha$	$-6\cdot\alpha$	$-4\cdot\alpha$	$-\alpha$
gradient vertical Sobel	α	$4\cdot\alpha$	$6\cdot\alpha$	$4\cdot\alpha$	α
Sobel	$\sqrt{2}\cdot\alpha$	$4\cdot\sqrt{2}\cdot\alpha$	$6\cdot\sqrt{2}\cdot\alpha$	$4\cdot\sqrt{2}\cdot\alpha$	$\sqrt{2}\cdot\alpha$
Sobel - Canny	$2\cdot\alpha$	$8\cdot\alpha$	$12\cdot\alpha$	$8\cdot\alpha$	$2\cdot\alpha$
Laplace	α	$4\cdot\alpha$	0	$-4\cdot\alpha$	$-\alpha$
gradient SW	$16/9\cdot\alpha$	$16/3\cdot\alpha$	$6\cdot\alpha$	$16/3\cdot\alpha$	$16/9\cdot\alpha$
Gradient LW	$342/25\cdot\alpha \cong 13.7\cdot\alpha$	$96/5\cdot\alpha$	$20\cdot\alpha$	$96/5\cdot\alpha$	$342/25\cdot\alpha \cong 13.7\cdot\alpha$

Table 3.3 : Détection d'une ligne oblique (numérique).

La forme graphique ci-dessous donne un aperçu plus visuel de ces résultats (c.f. figure 3.6).

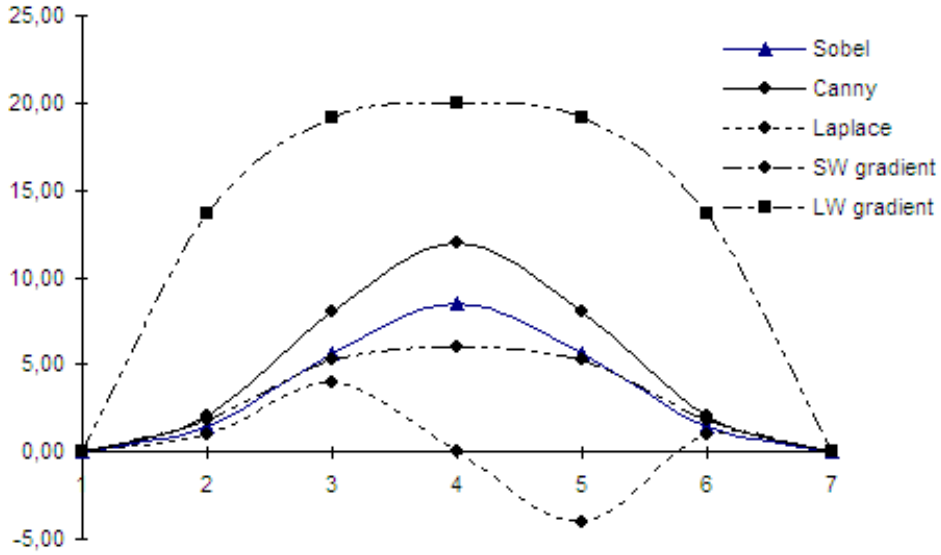


Figure 3.6 : Détection d'une ligne oblique (graphique).

Comme on peut le voir dans les figures 3.5 et 3.6, les courbes du filtre de Laplace sont très différentes des autres. Etant basé sur la dérivée seconde de la "fonction", l'indication du point de

contour est donnée par le croisement avec l'axe des abscisses. Si on compare les courbes basée sur la dérivée du premier ordre, on remarque que le "Gradient Large" est le plus révélateur. Pourquoi? Parce qu'il prend en compte un nombre plus important de valeurs et qu'il n'est pas normalisé.

En fait, la donnée discriminatoire n'est pas la hauteur de la courbe mais combien chacune des méthodes de détection permet de différencier un point de contour des points l'entourant. Autrement dit, quelle est l'importance du rapport entre la valeur entre un point de contour et celle de ses voisins proches. A ce niveau, les algorithmes de Canny et Sobel produisent de bons résultats. Si théoriquement, il est possible de démontrer que l'opérateur de Laplace est meilleur dans la détection des lignes dans toutes les directions, il semble que Canny et Sobel sont aussi tout à fait efficaces dans ce cas là.

Par contre, il semble que les gradients sur les fenêtres étroites et larges ne sont pas aussi déterministes. Ceci explique pourquoi on obtient si difficilement des contours fins. Comme attendu, la détection à l'aide de ce filtre est isotropique. Nous observons cependant que le ratio de détection est supérieur dans le cas de points de contour situés sur des contours obliques par rapport à ceux situés sur des contours droits. Le filtre devrait donc mieux détecter cette catégorie de points, même si la différence par rapport aux autres filtres n’est pas marquante.

3.2.3. Détection en coin

Dans les deux paragraphes précédents, on a présenté les résultats sous forme d'un ratio du gradient réel suivant un axe perpendiculaire au contour. Dans les deux cas suivants, ce n'est pas possible du fait de la multiplicité des axes à prendre en compte lorsqu'on observe un tracé en deux dimensions. La figure 3.7 présente cette information en six tableaux : un pour l'image originale et un par méthode de détection de contours. On utilise les niveaux de gris de la table 3.4 pour donner une interprétation des valeurs des points après détection.

Niveau de gris	arrière plan	point de contour	détection erronée
couleur			

Table 3.4 : Signification des niveaux de gris.

Image

50	50	50	50	50
50	50	50	50	50
50	50	80	80	80
50	50	80	110	110
50	50	80	110	110

Laplace

0	0	0	0	0
0	-1	-2	-3	-3
0	-2	4	1	0
0	-3	1	5	3
0	-3	0	3	0

Sobel

0	0	0	0	0
0	0,35	0,79	1,00	1,00
0	0,79	1,41	1,77	2,00
0	1,00	1,77	1,06	1,00
0	1,00	2,00	1,00	0

Sobel Canny

0	0	0	0	0
0	0,50	1,00	1,00	1,00
0	1,00	2,00	2,00	2,00
0	1,00	2,00	1,50	1,00
0	1,00	2,00	1,00	0

Gradient large

0,08	0,15	0,21	0,27	0,32
0,15	0,34	0,49	0,60	0,67
0,21	0,49	0,67	0,75	0,74
0,27	0,60	0,75	0,72	0,69
0,32	0,67	0,74	0,69	0,46

Gradient étroit

0	0	0	0	0
0	0,20	0,35	0,44	0,44
0	0,35	0,62	0,59	0,67
0	0,44	0,59	0,49	0,44
0	0,44	0,67	0,44	0,00

Figure 3.7 : Détection des contours en coin

Dans cet exemple, tous les opérateurs, sauf le gradient large, situent le contour correctement. Le gradient étroit ne fournit pas un résultat aussi contrasté que les autres. Le gradient Large met en évidence un contour plus large.

3.2.4. Détection d'intersection

Ce cas est plus complexe car au point d'intersection se rencontrent trois zones avec des niveaux de gris différents. En effet, on doit prendre en compte au moins deux gradients : de la première à la seconde, et de la seconde à la troisième zone. Ici, on a choisi d'étudier un cas simple avec un gradient constant (30). Cela nous permettra de présenter les résultats sous forme de multiples de ce gradient.

Le contraste entre les niveaux de gris n'est pas uniforme et nous allons voir que cela peut conduire à des erreurs. Dans ce cas, aucun des filtres ne donne un résultat parfait. Le filtre de Laplace ne détecte pas le point d'intersection. Les résultats du filtre de Sobel ne sont pas aussi contrastés que pour les lignes droites. Les gradients étroits donnent des résultats équivalents contrairement au gradient large qui agrandit la zone prise en compte.

Image

50	80	80	80	80
50	80	80	80	80
50	80	80	80	80
50	80	110	110	110
50	80	110	110	110
50	80	110	110	110

Sobel

0	0,44	0,44	0	0
0	0,44	0,44	0	0
0	0,44	0,57	0,35	0,44
0	0,44	0,79	0,47	0,44
0	0,44	0,89	0,44	0
0	0,44	0,89	0,44	0

Laplace

3	-3	0	0	0
3	-3	0	0	0
3	-2	2	3	3
3	-1	-5	-3	-3
3	0	-3	0	0
3	0	-3	0	0

Sobel Canny

0	0,44	0,44	0	0
0	0,44	0,44	0	0
0	0,44	0,67	0,44	0,44
0	0,44	0,89	0,67	0,44
0	0,44	0,89	0,44	0
0	0,44	0,89	0,44	0

Gradient étroit

0	0,44	0,44	0	0
0	0,44	0,44	0	0
0	0,44	0,52	0,35	0,44
0	0,44	0,59	0,49	0,44
0	0,44	0,67	0,44	0
0	0,44	0,67	0,44	0

Gradient Large

0,32	0,48	0,48	0,32	0
0,32	0,53	0,54	0,37	0,27
0,32	0,58	0,61	0,46	0,44
0,32	0,62	0,67	0,60	0,50
0,32	0,67	0,74	0,69	0,46
0,32	0,72	0,80	0,72	0,32

Figure 3.8 : Détection des contours aux intersections

3.2.5. Conclusion

On note qu'il a été impossible de démontrer que le filtre de Sobel n'est pas isotropique et que sa qualité de détection est dépendante de la direction du contour. Il est vrai que nous n'avons étudié que des cas relativement simples mais qui sont néanmoins représentatifs.

Nous avons montré que le filtre de Laplace est incapable de localiser précisément les contours dans certains cas.

Nous avons vérifié que les informations issues des gradients étroit et large sont pertinentes. Dans la plupart des cas, on ne peut pas montrer théoriquement que le filtre pseudo-statistique est meilleur que le filtre de Laplace ou celui de Sobel. De plus, cette étude a mis en avant la difficulté qu'a ce filtre à tracer des contours fins. Toutefois, les résultats expérimentaux semblent montrer qu'il est possible d'obtenir des contours plus précis. Pour améliorer ces résultats, il serait possible de combiner, par un filtre flou, les informations issues du filtre de Sobel à celles issues des gradients sur les fenêtres. Le rôle du filtre flou serait alors équivalent à celui du module de calcul du filtre de Canny.

4. Evaluation des résultats

Nous avons besoin dévaluer les résultats de la détection des points de contour d'une image dans deux cas : pour comparer les résultats de deux algorithmes différents sur une image ou, dans un processus d'auto apprentissage, comparer automatiquement les résultats de détections effectuées avec des valeurs de paramètres différentes.

L'évaluation peut être effectuée subjectivement ou objectivement. Nous envisageons ces deux solutions puis présentons différentes méthodes objectives et exposons pourquoi elles ne sont pas toutes utilisables pour cette problématique.

4.1. Evaluation subjective ou objective ?

Pour évaluer subjectivement les résultats, on peut demander à un groupe de spécialistes du domaine de classer les images obtenues après la détection des contours. Si elle peut être utilisée pour comparer les résultats d'un petit nombre d'images, cette méthode a des limites lorsque ce nombre augmente et si on doit souvent répéter cette comparaison sur des échantillons différents. En effet, le nombre d'images à étudier dans le cas d'un apprentissage automatique des paramètres est très important (des dizaines de milliers). De plus, l'idée d'automatisation est tout à fait contraire avec l'idée de comparaison subjective et manuelle. Il est donc nécessaire d'utiliser un outil de comparaison, une méthode objective capable de mesurer une distance entre les images obtenues par rapport à une carte idéale des contours. La section suivante présente la démarche utilisée pour établir cette carte modèle.

On utilisera cependant l'évaluation subjective à deux reprises :

- pour finaliser les cartes des contours idéals ;
- pour choisir la meilleure méthode objective, celle mesurant la plus petite erreur possible.

4.2. Le choix du jeu d'images de test

Nous avons choisi cinq images présentant des caractéristiques différentes en termes de luminosité, contraste, complexité du réseau de vaisseaux sanguins, densité et répartition dans l'image. Elles sont présentées au paragraphe 5.1 (page 57).

Ces images n'ont pas des caractéristiques homogènes : certaines ont plus de bruit que les autres. D'autres n'ont pas les mêmes niveaux de contraste. C'est pour cela qu'une sixième image est introduite (c.f. Figure 4.1). Les cinq premières images ont été normalisées : leur niveau de contraste a été homogénéisé, puis elles ont été assemblées les unes à côté des autres.

Figure 4.1 : Image synthétique

4.3. La carte idéale des contours

Nous avons interrogé des experts en imagerie médicale : des ingénieurs biomédicaux. Ils ont dessiné manuellement sur un calque superposé à l'image originale le contour qu'ils jugeaient comme parfait. Les contours obtenus ont été globalement identiques. Cependant, on observe des différences majeures dans l'interprétation de certains détails.

Par exemple, pour la partie située en bas à gauche de l'image, des experts ont proposé des interprétations relativement différentes (voir figure 4.2).

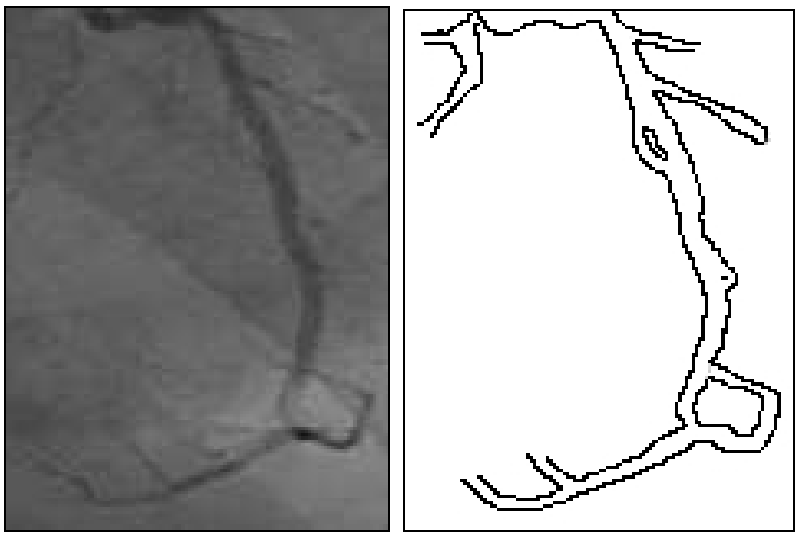


Figure 4.2 : Modèle de carte des contours

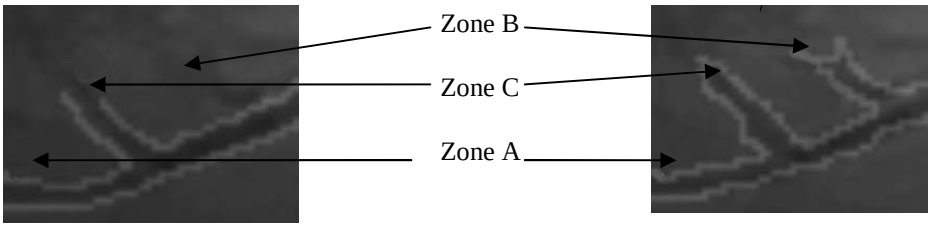


Figure 4.3 : Détail de la carte des contours

- Si globalement le tracé du contour semble identique (voir figure 4.3), à l'échelle des pixels on remarque des différences importantes (zone A).
- Certains détails sont interprétés de manière totalement différente (zones B et C).
- Certaines erreurs sont dues à des erreurs présentes sur l'image originale, des pertes ayant eu lieu lors de la compression de cette image. Dans ces zones, certains spécialistes ont suivi le contour tel qu'il était dessiné, d'autres l'ont interprété pour obtenir le contour du vaisseau tel qu'il est probablement en réalité.

Pour conclure, après un examen de ces résultats, il nous semble qu'il n'existe pas de carte idéale des contours. Celle-ci est extrêmement dépendante de la sensibilité de chacun des experts. Cependant, il est possible d'utiliser leurs dessins pour :

- établir un choix subjectif des contours idéaux et de considérer tout point à une distance proche d'un point de contour comme générant une erreur plus petite ;
- établir une carte moyenne des contours à partir de toutes les cartes dessinées. Cette carte donnant en chaque point une pseudo-probabilité que ce point appartienne à un contour. L'erreur est alors calculée en fonction de cette probabilité.

Selon Salotti [SALOTTI 1996], il est important que cette carte représente les contours qui sont objectivement détectables (de par la réalité de l’image initiale) et non pas l’interprétation que peut en faire un expert : le rôle du détecteur de contours n’est pas d’extrapoler les résultats de la détection.

4.4. Une évaluation basée sur une distance

On note *Nb1* le nombre de points classifiés à tort dans la classe contour et *Nb2* le nombre de points reconnus à tort comme "fond d'image". On note M and N les dimensions de l'image.

La mesure de distance la plus répandue est la mesure Euclidienne. Elle est couramment utilisée pour mesurer le bruit présent sur une image par rapport à un original non bruité. Elle est définie par la relation suivante :

$$d_p^2(I, I^r) = \frac{1}{N \cdot M \cdot P} \cdot \sum_{i=1}^{M} \sum_{j=1}^{N} \left(p_{i,j} - p_{i,j}^r \right)^2 \qquad \text{(Eq. 4.1)}$$

qui est équivalente avec cette relation :

$$d^2 = \frac{1}{M \cdot N} \cdot (nb1 + nb2) \qquad \text{(Eq. 4.2)}$$

Nous voulons utiliser cette distance lors de l'apprentissage automatique. Pour vérifier que cette mesure est appropriée, il faut étudier la fonction entre la distance et les deux paramètres *nb1* et *nb2*. Les valeurs possibles de *nb1* et *nb2* sont :

$$nb1 \in [0, \mathrm{NbEdgP}] \text{ and } nb2 \in [0, \overline{\mathrm{NbEdgP}}]$$

$$\text{et } nb1 + nb2 \in [0,\ M \cdot N] \text{ and } d^2 \in [0,1] \qquad \text{(Eq. 4.3)}$$

La figure 4.4 présente cette fonction.

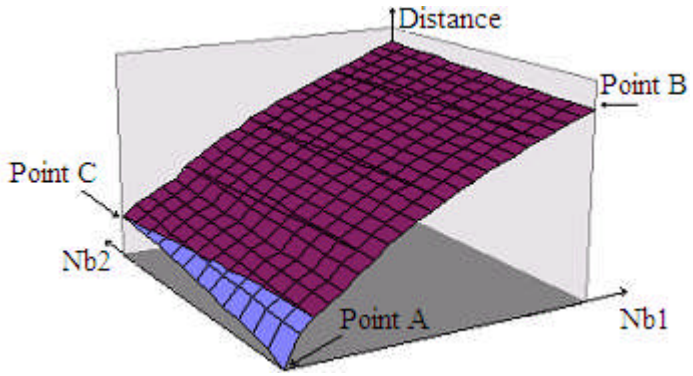


Figure 4.4 : Graphe de la fonction de distance

On distingue 3 points caractéristiques sur ce graphe :

- le point A où Nb1et Nb2 ont une valeur nulle : c'est l'erreur obtenue en analysant une carte des contours parfaite ;
- le point B montre l'erreur quand Nb1 est nul. Tous les points de l'image sont détectés comme étant des points de contour. L'image résultante est blanche ;
- le point C montre l'erreur quand Nb2 est nul. Tous les points de l'image sont détectés comme appartenant au fond de l'image. Aucun point de contour n'est détecté. L'image résultante est noire.

Sur la figure 4.4, la surface sombre représente tous les cas pour lesquels l'erreur calculée sera inférieure à l'erreur au point B et supérieure à l'erreur au point C. Si on veut que l'algorithme d'apprentissage nous donne comme résultat la solution la plus proche du point A, il doit donc pendant sa phase d'investigation, trouver au moins un résultat avec une erreur inférieure à celle du point C pour ne pas trouver que l'image au point C est la meilleure solution. Mais la surface claire est relativement petite.

De plus, en testant l'algorithme de Sobel, on peut voir que les meilleurs résultats obtenus sont au dessus de cette limite. Ce qui veut dire que si notre algorithme n'est pas nettement meilleur que celui de Sobel alors jamais ce calcul de distance ne nous donnera de solution satisfaisante. Cette erreur n'est pas appropriée pour la résolution de notre problème.

4.5. Une proposition d'erreur pondérée

On utilise les mêmes paramètres *Nb1* et *Nb2* que précédemment.

Intuitivement, en regardant la figure 4.4, on peut essayer d'agrandir la taille de la surface claire en changeant la valeur du point C pour qu'elle soit égale à celle du point B. On peut obtenir ceci en pondérant la valeur du paramètre *Nb2* par un facteur *R* supérieur à 1. La définition de la distance est alors la suivante :

$$d_p^2 = \frac{1}{M \cdot N} \cdot (nb1/R + nb2) \qquad \text{(Eq. 4.4)}$$

La variation du paramètre R a une influence majeure sur la variation de la surface claire comme le montre la figure 4.5.

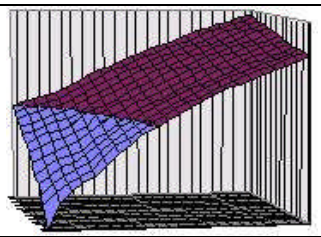
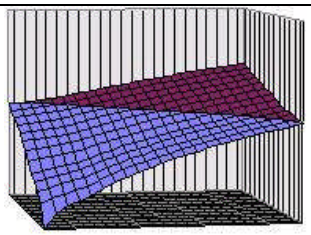
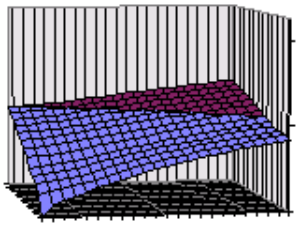
R < $\overline{\text{nbEdgP}}$/nbEdgP (13)	R = $\overline{\text{nbEdgP}}$/nbEdgP (20)	R > $\overline{\text{nbEdgP}}$/nbEdgP (28)
		
La surface claire est si petite que les chances de se retrouver dans le cas de figure de l'erreur euclidienne sont très grandes.	L'erreur aux points B et C sont égales. La taille de la surface claire est importante	Si la taille de la surface claire est importante, elle comprend aussi le point B. Les chances sont grande d'obtenir le point B comme meilleur résultat.

Figure 4.5 : Etude de cas de l'erreur pondérée

Dans un apprentissage, on ne peut pas deviner le nombre de points de contour avant de commencer la détection. Cependant, les résultats de la mesure d'erreur dépendent de la valeur du rapport R qui représente les caractéristiques intrinsèques de l'image. Si on fixe une valeur pour le rapport R, elle sera parfaitement adaptée aux images dont les caractéristiques sont proches de ce rapport. Plus le rapport R choisi est éloigné du rapport réel $\frac{nbPo\,\text{int}\,sCoutour}{nbPo\,\text{int}\,sFond\,\text{Im}\,age}$ plus la mesure d'erreur sur les images sera erronée. Cette erreur n'est donc pas la plus adaptée à notre problématique.

4.6. Une proposition d'erreur statistique

On peut considérer le processus de détection de contours comme un problème de classification. Alors, on peut se baser sur la théorie de la classification et les graphes ROC développé dans [KONISHI *et al.* 2003].

4.6.1. Base théorique

Les graphes ROC (Receiver Operating Characteristics) sont une méthode employée pour évaluer et comparer les résultats de classifications. Cette technique est applicable dans le cas d'une classification d'un ensemble d'éléments entre deux classes non jointes.

La figure 4.6 décrit les notations utilisées.

		classe positive	
		p	n
classe hypothèse	O	Vrais Positifs (*VP)*	Faux Positifs (*FP*)
	N	Faux Négatifs (FN)	Vrais Négatifs (VN)
Total du nombre d'éléments		Po	Ne

Figure 4.6 : Notations basiques applicables à la classification en deux classes

Les taux de cas Faux Positifs et Vrais Positifs se définissent ainsi :

$$fp = \frac{FP}{Ne} \text{ et } vp = \frac{VP}{Po}. \qquad \text{(Eq. 4.5)}$$

Ces graphes peuvent donner une représentation graphique des résultats d'une classification sur un ensemble de jeux de test. Dans notre cas, il permet de représenter la qualité de la détection des contours par un algorithme pour plusieurs images différentes, ou de la même image par plusieurs algorithmes. Chaque point est caractérisé par ses coordonnées :

- En abscisse, la proportion de points du fond de l'image détectés à tort par rapport au nombre de points de fond de l'image soit *vp* ;
- En ordonnée, la proportion de points de contour détectés par rapport au nombre de points de contour soit *fp*.

Un exemple de graphe ROC est donné par la figure 4.7.

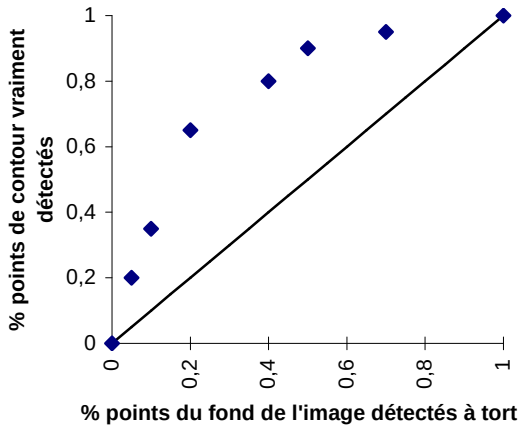


Figure 4.7 : Exemple de courbe d'évaluation ROC

Tout point situé au dessus de ligne $x=y$ est le signe d'un résultat meilleur qu'un choix fait au hasard. La qualité d'une classification n'est pas calculée en faisant la somme ou la moyenne des distances pour tous les éléments du jeu de test. Elle est évaluée en calculant la taille de la surface se trouvant en dessous la courbe. La meilleure valeur est 1.

On peut considérer que tous les points situés sur une ligne parallèle avec la droite $x = y$ ont un niveau d'erreur équivalent. Ainsi, un taux d'erreur pour un échantillon du jeu de test peut être calculé comme l'ordonnée de la projection sur l'axe $x = 0$ et selon la droite $x = y$ du point représentant cet échantillon.

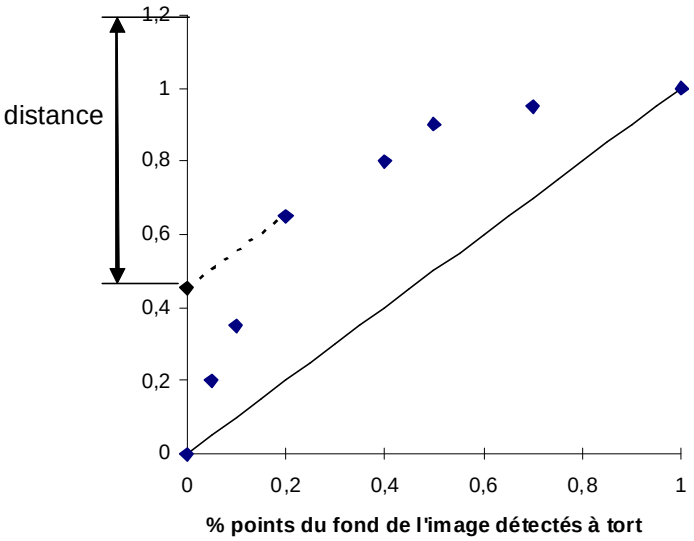


Figure 4.8 Calcul d'une distance à partir de la courbe ROC

Cette distance se calcule donc ainsi :

$$D = 1 - \left(\frac{TP}{Po} - \frac{FP}{Ne} \right) \qquad \text{(Eq. 4.6)}$$

Ces graphes mettent aussi en évidence pourquoi l'utilisation de la distance Euclidienne n'est pas appropriée pour ce type de problème : tous les points situés à la même distance Euclidienne du point parfait (de coordonnées (0,1)) se trouvent sur une courbe ellipsoïdale ayant pour centre ce point.

Note : La courbe n'est pas circulaire puisque ses paramètres caractéristiques sont des multiples de D et $\frac{FP}{Ne}$. Sur la figure 4.9, la courbe est représentée, elle passe par les points A et B qui n'ont pas du tout le même niveau d'erreur dans le sens où elle est définie ci-dessus. On peut donc en conclure que l'usage de la distance Euclidienne ne permet pas le calcul du meilleur résultat.

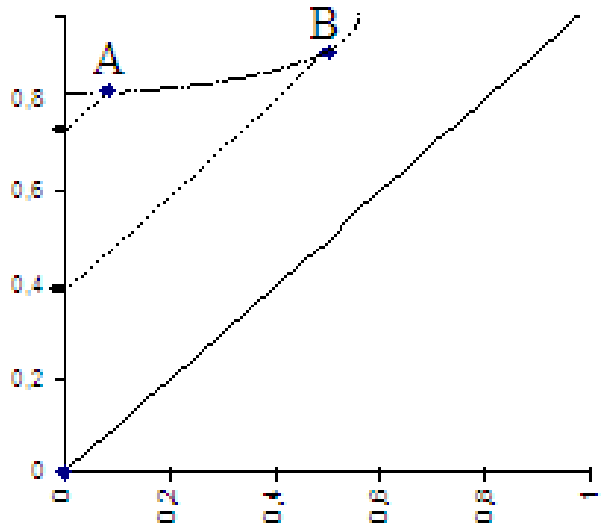


Figure 4.9 : Représentation de la distance Euclidienne dans une courbe ROC

Si le graphe ROC peut donner une appréciation de la qualité des éléments classifiés, il peut aussi être utilisé pour comparer des algorithmes différents. Mais, Holte [HOLTE *et al.* 2004] a mis en évidence qu'une donnée importante manque dans ces résultats : une évaluation de la confiance à

avoir dans les résultats obtenus. L'usage de courbes de coût permettrait d'adresser ce sujet que nous ne traiterons pas.

4.6.2. Application pratique

Idéalement, dans les problèmes traités avec les graphes ROC, le nombre de points de chaque classe est connu. Donc les proportions de points peuvent être exactement évaluées. Dans notre cas, nous ne connaissons pas le nombre de points de contour présents dans une image. Cependant, nous pouvons l'évaluer comme une proportion du nombre de points de l'image. Statistiquement, si on considère les images du jeu de test et les contours dessinés par les spécialistes, le ratio est compris entre 13 et 25. Si on donne à ce ratio une valeur moyenne, quelle sera la conséquence sur l'évaluation des résultats des tests ?

Si on choisit arbitrairement une valeur 19 pour R, alors les coordonnées des points sur le graphe ROC seront affectées puisque, on a :

$$fp = \frac{nb_1}{M \cdot N \cdot (1 - 1/R)} \text{ et } tp = 1 - \frac{R \cdot nb_2}{M \cdot N}. \qquad \text{(Eq. 4.7)}$$

Le choix de cette valeur induit une erreur de plus ou moins deux pour cent sur le taux *fp* et plus ou moins trente pour cent sur le taux *tp*. Cette variation du taux étant équivalente pour tous les tests effectués sur une même image, on peut l'utiliser pour comparer les résultats de différents algorithmes sur une image. On peut noter qu'il est ainsi possible d'obtenir un taux *tp* inférieur à zéro.

Comme précédemment, les paramètres *Nb1* et *Nb2* sont utilisés. Leur définition correspond exactement à celle de *FP* and *FN*.

De l'équation 4.6, on en déduit :

$$D = 1 - \left(\frac{TP}{Po} - \frac{FP}{Ne} \right) = \frac{nb_1}{Ne} - \frac{nb_2}{Po} = \frac{nb_2 \cdot (R-1) + nb_1}{M \cdot N \cdot (1 - 1/R)} \qquad \text{(Eq. 4.8)}$$

soit :

$$Dstat = \left(\frac{nb_1}{(M \cdot N - \frac{M \cdot N}{R})} + \frac{nb_2 \cdot R}{M \cdot N} \right) = \frac{1}{M \cdot N} \cdot \left(\frac{nb_1}{(1 - \frac{1}{R})} + nb_2 \cdot R \right) \qquad \text{(Eq. 4.9)}$$

En utilisant le même ratio *R* pour toutes les images, on ne pourra pas comparer l'erreur calculée sur des images différentes puisque les résultats seront dépendants de la différence de rapport entre ce ratio *R* et le ratio intrinsèque des images, i.e. le vrai rapport entre le nombre de points de contour et le nombre de points de l'image. Si on néglige cet aspect, on peut remarquer que cette mesure est indépendante de la taille de l'image.

Si on veut représenter l'influence de la variation de la valeur du rapport R sur le calcul de l'erreur sous la même forme que dans les paragraphes précédents, on obtient la figure 4.10.

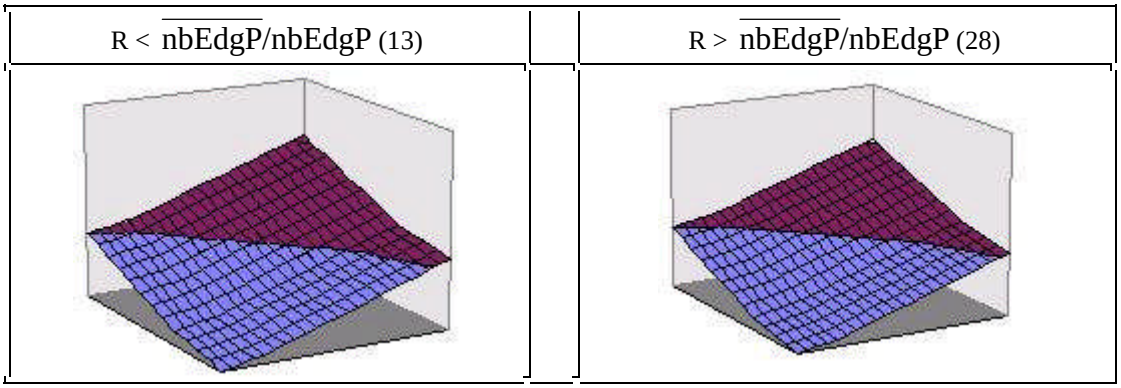


Figure 4.10 : Influence du rapport R sur le calcul de l'erreur

La variation de R n'a que peu d'influence sur la limite de la surface. De plus, les valeurs mesurées sur des détections de contours réalisées avec le filtre de Sobel nous montrent que les résultats se situent à l'intérieur de la surface claire. Tant que R est supérieur à 5, la variation de la taille de la surface est acceptable. Il n'y a pas de limite supérieure mais on remarque que si R tend vers un nombre très grand, alors le premier terme de l'équation 4.9 devient insignifiant par rapport à la valeur du second terme. En d'autres termes, quel que soit le nombre de points considérés comme contour par erreur, il ne sera pas pris en compte dans le calcul de l'erreur.

4.7. Une proposition de mesure de distance floue

On a analysé jusqu'ici des distances déterministes basées sur la comparaison point par point entre une image idéale et une image des contours calculés. Si la valeur en un point est différente dans les deux images alors on considère que la valeur calculée pour ce point est erronée. Cette vision ne prend pas en compte la subjectivité de l'établissement d'une carte des contours idéaux que nous avons déjà mis en évidence. Comme nous en avons déjà introduit l'idée, il peut être intéressant de considérer pour chaque point un calcul de l'erreur proportionnel à la distance qui le sépare d'un point de la même classe sur le modèle.

On définit deux zones autour d'un point :

- Un voisinage très proche où l'erreur est nulle ;
- Un voisinage proche où l'erreur est proportionnelle à la distance du point de même classe le plus proche.

Plus formellement, on peut donner la définition suivante : Pour tout pixel de l'image, on calcule une erreur comme suit. Premièrement, on définit une matrice de voisinage comme représentée sur la figure 4.11.

Figure 4.11 : Distance –fonction de calcul de l'erreur

On construit alors une matrice des contours détectés à tort et on calcule alors l'erreur floue pour chacun de ces points que l'on dénote nb_1^{Fuz} .

On construit une matrice des points de fond de l'image détectés à tort et on calcule alors l'erreur floue pour chacun de ces points que l'on dénote nb_2^{Fuz} .

On a alors les relations suivantes : $nb_1^{Fuz} \in [0, nb1[$ et $nb_2^{Fuz} \in [0, nb2[$ (Eq. 4.10)

Les valeurs E1 et E2 ayant des plages de valeurs comparables à celles de *Nb1* et *Nb2*, on peut donc les utiliser comme variables d'entrées pour la fonction de calcul d'erreur statistique.

4.8. Une autre proposition

Une autre proposition serait d'utiliser la distance d'Hausdorff. Ce concept est basé sur l'idée de mesurer la similarité entre deux objets sur une image. Deux conditions sont liées à sa validité : premièrement, les formes à comparer doivent être limitées. Il doit être possible de les entourer par un disque. Secondement, le contour des formes doit être fermé. Ces deux conditions ne sont pas

remplies dans le cas des images avec des vaisseaux sanguins qui sont ouvertes à au moins une extrémité dans le cas où la détection des contours est parfaite. Nous ne pouvons pas utiliser la distance d'Hausdorff dans ce problème.

4.9. Discussion : qualité de l'évaluation de la distance et apprentissage.

La réalisation des deux objectifs suivants est directement liée à l'objectivité de l'évaluation des résultats de la détection des contours. Premièrement, on veut pouvoir comparer les résultats de filtres différents, et secondement on veut utiliser ce comparateur pour améliorer le réglage des paramètres du filtre pseudo-statistique. Si cette mesure n'est pas assez précise, nous ne verrons pas les petites différences existantes entre des images très proches. S'il avantage certaines erreurs par rapport à d'autres, ou s'il n'est pas objectif par rapport aux caractéristiques intrinsèques de l'image originale alors dans certains cas il pourra évaluer un résultat comme étant le meilleur en contradiction avec l'avis des experts.

Pour cette raison, nous avons mis en concurrence les différentes méthodes de calcul de l'erreur, en particulier la distance floue et l'erreur statistique. Nous avons calculé des cartes des contours de quelques images avec différents filtres. Un classement des résultats a été soumis pour évaluation aux experts. La meilleure carte des contours obtenue avec la mesure de distance floue contenait des tracés extrêmement fins. Par contre, ils étaient souvent tracés comme deux lignes parallèles. C'est ce que Canny nomme une détection double qu'il décrit comme une des difficultés majeures à éviter en détection de contours. Pour cette raison, nous n'utiliserons pas la distance floue pour l'évaluation de nos résultats.

5. Synthèse de la démarche

Cette recherche a été menée durant une année entière et les résultats présentées dans ce document, qu'ils soient théoriques, méthodologiques ou pratiques sont le résultat de ce travail. La présentation qui en est faite n'est pas représentative de l'ordre dans lequel ont été menés les travaux. D'autre part, la réalisation de cette recherche a été enrichie par des échanges au sein du laboratoire comme à l'extérieur. Nous présenterons donc ici dans un ordre plus chronologique les phases qui ont ponctué l'année 2003/2004.

5.1. Choix des images

Cinq images sont choisies pour les tests. Elles présentent des difficultés différentes soit de par leur manque de contraste, de leur exposition (sur-exposition ou sous-exposition) soit à cause d'éléments extérieurs ou bien au bruit naturel. Voici ces images et les contours tels qu'ils ont été définis en suivant la méthode exposée au paragraphe 4.3 (page 44) :

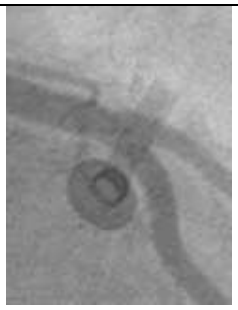
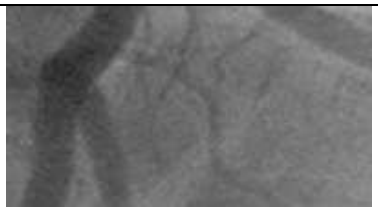
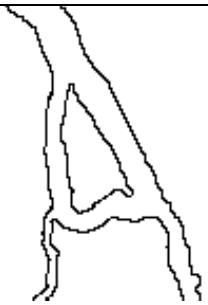
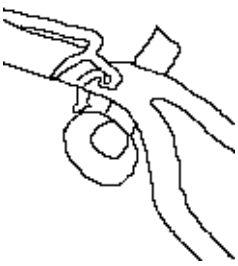
Image 1	Image 2	Image 3	Image 4	Image 5
				
Contour 1	**Contour 2**	**Contour 3**	**Contour 4**	**Contour 5**
				

Table 5.1 : Images de test.

5.2. Développement d'un prototype

Ce prototype a eu pour but de vérifier les principes de fonctionnement du filtre pseudo-statistique. A partir d'un algorithme, j'ai développé une première version qui a permis de visualiser des détections de contours sur des images issues du domaine de l'angiographie médicale. Celles-ci sont réputées comme difficiles parce que peu contrastées et bruitées.

Nous avons choisi arbitrairement les réglages suivants (au moment du développement du prototype, la classe small était définie par un opérateur symétrique) :

- pour la classification :

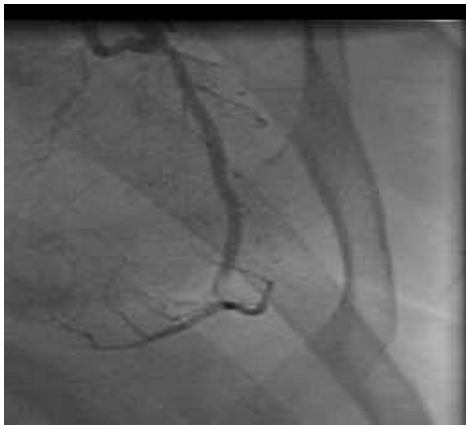
Classe	Seuil bas	Seuil haut	Type fonction
Zéro	1	5	
Small	4	25	
Large	11	20	

- pour le moteur d'inférence :

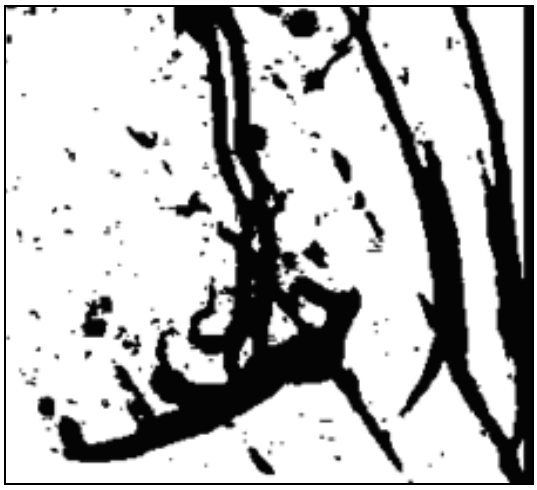
		$d(S_{i,j}, \overline{S}_{i,j}^{LW})$		
		Z	S	L
$d(S_{i,j}, \overline{S}_{i,j}^{SW})$	Z	-L	+L	Z
	S	Z	+L	Z
	L	Z	Z	Z

Table 5.2 : Paramétrage du prototype.

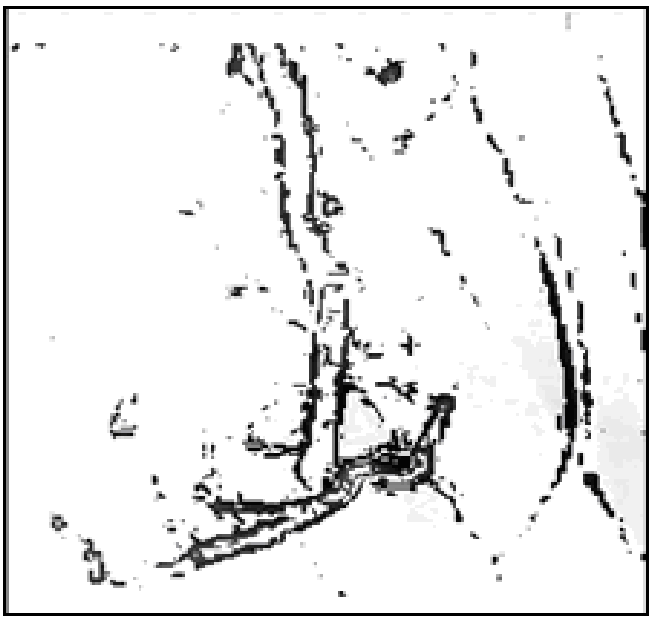
A partir de cette image ci-dessous, on obtient les résultats ci-après.



- Les zones pour lesquelles la petite et la grande fenêtre appartiennent à la classe « Zéro » sont représentatives du fond de l'image.



- Les zones pour lesquelles la grande fenêtre appartient à la classe « Small », et la petite fenêtre appartient à la classe « Small » ou « Zero », sont des zones de contours.



Pour comparaison, avec les autres opérateurs et à partir de la même image, on obtient ces résultats :

- avec le filtre de Sobel :

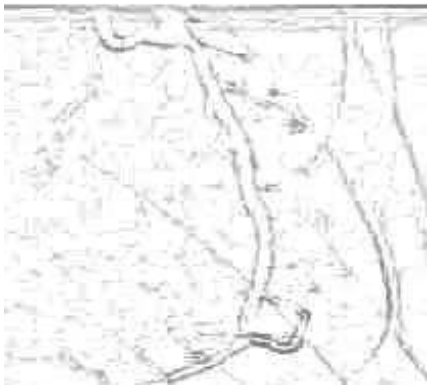
Sans correction	Après une correction du contraste
	

Table 5.3 : Détection par le filtre de Sobel.

- avec le filtre de Laplace :

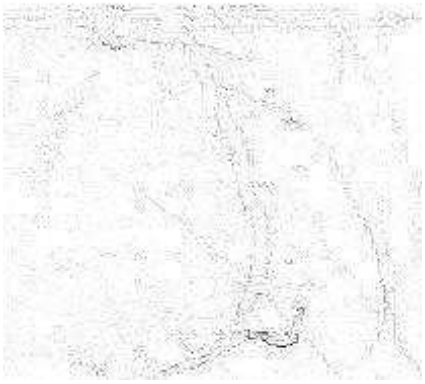
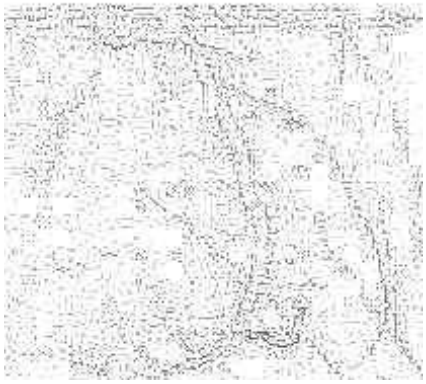
Sans correction	Après une correction du contraste
	

Table 5.4 : Détection par le filtre de Laplace.

Avec notre filtre, nous sommes parvenus à mettre en évidence certains contours mais ils sont gras et imparfaits.

Pour autant qu'on puisse juger de visu ces résultats, il semble que les contours sont nettement plus visibles avec notre filtre qu'avec l'opérateur de Laplace. Par contre, si les contours sont plus nets et plus fins avec l'opérateur de Sobel, nous observons que de nombreux tracés ne représentant pas un contour de vaisseau sont mis en évidence à tort.

5.3. Modélisation

Le développement du filtre pseudo-statistique et la réutilisation nécessaire de ses fonctionnalités dans différents outils nous a incité à modéliser le filtre. Les détails en sont présentés dans le chapitre 6.2.

5.4. Intégration dans GIMP

Pour pouvoir comparer les résultats du filtrage par le filtre pseudo-statistique avec celui d'autres filtres, nous l'avons intégré au sein d'un outil spécialisé dans le traitement des images : GIMP[1]. Cette phase nous a permis de présenter des résultats simplement. D'autre part, nous avons ainsi pu essayer de combiner les résultats de ce filtre avec d'autres outils de pré ou post traitement. C'est ainsi que nous avons eu l'idée d'enrichir le filtre d'une étape de normalisation des résultats. Cette étape répond à plusieurs besoins : les autres filtres de détection de contours indiquent un résultat binaire pour définir l'appartenance de chaque point à un contour : soit un point fait partie d'un contour soit il n'en fait pas partie. D'autre part, la comparaison des résultats avec un modèle nécessite elle aussi un résultat précis.

5.5. Construction du support théorique – présentation aux classes de master 2

Les chapitres 2.1 à 2.3 portent sur la notion de contour et les outils les plus communs utilisés pour la détection de contours. J'ai présenté, en langue Roumaine et Française, le contenu de ces chapitres, que j'ai rédigés, aux étudiants en master 2 dans le cadre du cours de spécialisation en imagerie médicale à l'université technique [TUIASI MASTER].

5.6. Présentation des résultats

5.6.1. Pitesti 04

La faculté de Pitesti a organisé en juillet 2004 un colloque centré sur les applications médicales de recherches en intelligence artificielle et en équipements électroniques. Plusieurs chercheurs du laboratoire de Iasi ont été invités à y présenter leur travaux. Ce fut, pour moi, l'occasion d'exposer les principes sur lesquels est basé le filtre pseudo-statistique. J'ai rédigé, en anglais, l'article qui est présenté en annexe 2 et qui est traduit dans le chapitre 3.

1 - GIMP est un logiciel libre disponible sous les systèmes Linux et Windows. Il autorise l'ajout de greffons, de programmes offrants des fonctionnalités supplémentaires.

5.6.2. ECIT 04

La conférence ECIT s'est déroulée du 21 au 23 Juillet 2004 à Iasi [ECIT 2004]. En tant que membre du comité d'organisation, j'ai participé à la sélection et à la relecture des articles qui y ont été proposés. Si ce temps a été riche d'échange avec les auteurs et les autres membres du laboratoire, il a été couteux par rapport au temps consacré à la recherche. Mais d'un autre point de vue, sans ces rencontres, aurait il été possible de penser aux solutions auxquelles nous sommes parvenus ? À l'occasion d'ECIT, j'ai présenté l'article que j'avais rédigé sur les méthodes de calcul de distance entre les images (voir annexe 3). Cet article est traduit et développé dans le chapitre 4.

5.7. Mesure de la qualité de détection

La mesure de la qualité de la détection s'est faite subjectivement dans un premier temps. Nous avons ensuite soumis les images aux experts afin qu'ils nous dessinent les contours qu'ils y voyaient. Puis nous avons utilisé la distance euclidienne pour mesurer l'erreur.

C'est lors de l'introduction d'outils d'apprentissage systématique que nous avons vu que le meilleur résultat pouvait être une image blanche ou noire. Par la suite, nous avons imaginé que l'erreur statistique pouvait être une bonne solution, ce qui s'est vérifié lors des expérimentations. Plus tard, nous avons mis en évidence que cette solution pouvait être appuyée sur la théorie de la classification.

En parallèle, nous avons enrichi notre expérimentation avec la distance floue. Cette phase de test a duré plusieurs mois durant lesquels nous avons aussi amélioré nos méthodes d'apprentissage automatique. Finalement, nous avons écarté l'utilisation de cette méthode de mesure car les meilleurs résultats mesurés mettaient en évidence des contours doubles (voir figure 5.1).

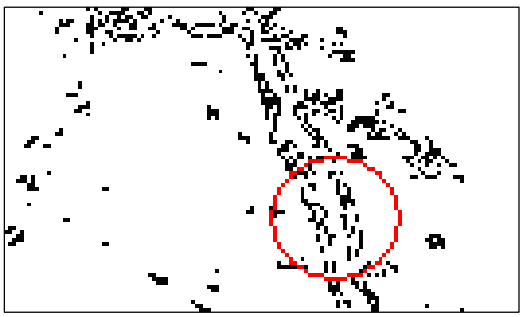


Figure 5.1 : Exemple de double détection de contours.

5.7.1. Exemple de comparaison de la qualité de détection avec les graphes ROC

Les filtres de Sobel ou de Laplace indiquent par un niveau de gris pour chaque point de l'image s'il pourrait faire partie d'un contour. Pour pouvoir comparer la détection des contours entre plusieurs images, cette estimation est normalisée par l'application d'un filtre passe-haut. Le seuil de ce filtre à utiliser pour obtenir la meilleure détection est différent pour chacune des images. En utilisant les graphes ROC [FAWCET 2003], nous pouvons tracer un graphe représentant l'évolution de la qualité de la détection en fonction du niveau du seuil pour chacune des images. La figure 5.2 montre la comparaison du filtrage des cinq images par le filtre de Sobel. En s'appuyant sur les travaux de Holte [HOLTE *et al.* 2002], on peut tracer la courbe noire qui représente la courbe moyenne des cinq images où les valeurs *ft* et *tp* sont calculées, pour chacun des jeux de test selon les formules suivantes :

$$\text{tp}=\sum_i tp_i \Big/ N \qquad \text{(Eq. 5.1)}$$

$$\text{fp}=\sum_i fp_i \Big/ N \qquad \text{(Eq. 5.2)}$$

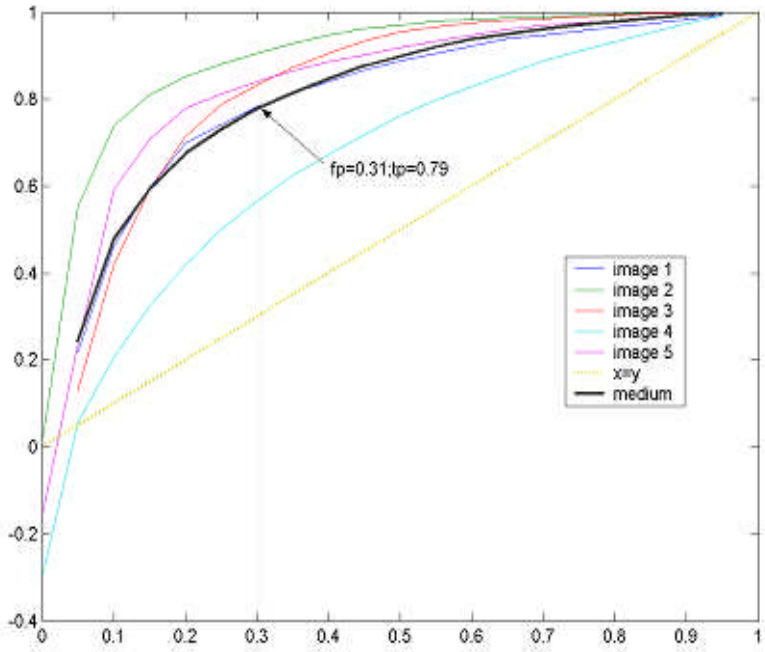


Figure 5.2 : Graphe ROC de la détection des contours sur 5 images (Sobel).

L'erreur minimale est observée au point de coordonnées (0.31, 079) : C'est à cet endroit que la courbe noire est tangente avec la parallèle de la droite $x = y$. L'erreur en ce point est égale avec l'erreur statistique proposée dans le paragraphe 4.6 (page 49).

5.8. Apprentissage automatique

Nous voulons mettre en œuvre un algorithme d'apprentissage des meilleurs paramètres pour le réglage du filtre pseudo-statistique flou. Schématiquement, on peut représenter le cycle d'apprentissage sous forme d'un diagramme (figure 5.3). Les rectangles de couleur représentent des données : les jaunes des données initiales, fournies et les verts, des résultats de l'algorithme d'apprentissage. Les rectangles blancs représentent des traitements.

Nous avons réalisé cet apprentissage en deux étapes :

- une recherche systématique des meilleurs résultats sur un nombre limité de critères ;
- une recherche des meilleurs résultats par algorithme génétique sur l'ensemble des critères possibles.

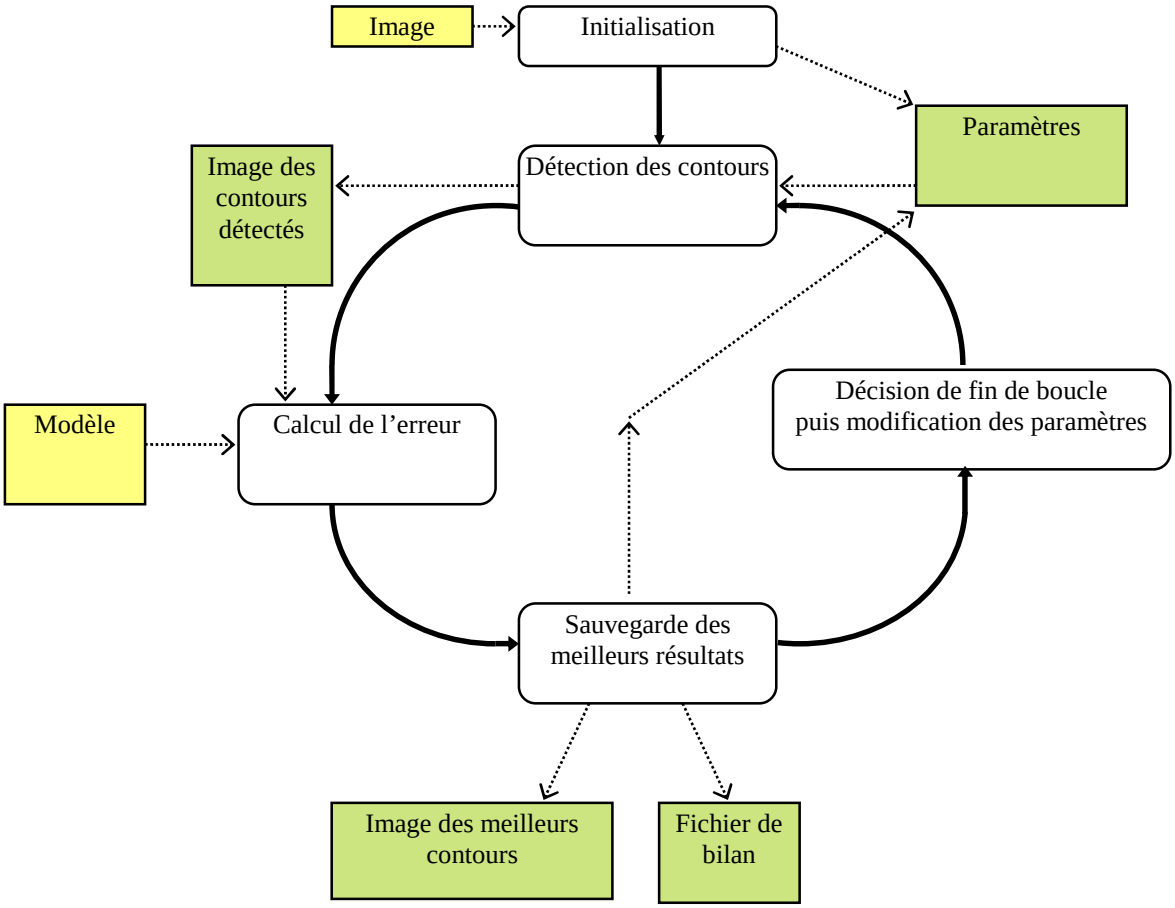


Figure 5.3 : Cycle d'apprentissage

5.8.1. Etude du champ d'apprentissage

Afin de définir quelles seraient les valeurs possibles des fonctions d'appartenance, nous avons regardé précisément quelles sont les valeurs des gradients sur les fenêtres (petite et grande) pour les images de test. Nous pouvons représenter les résultats sous forme d'un histogramme :

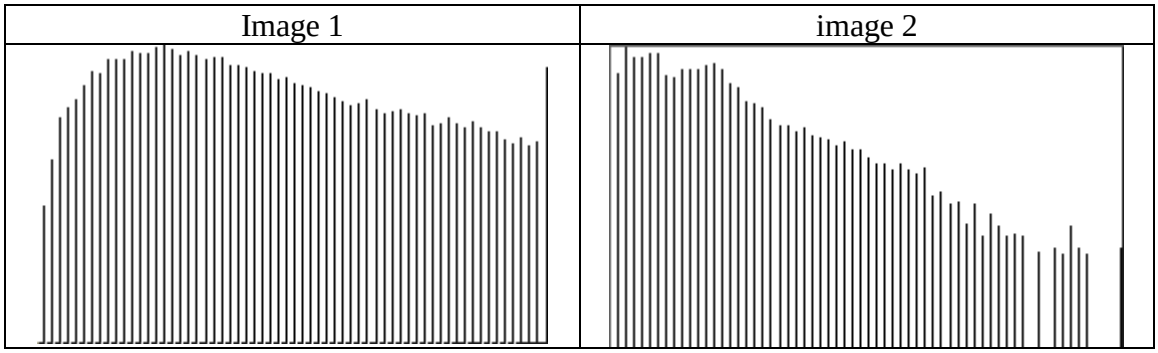


Figure 5.4 : Histogramme des gradients

Les valeurs étant comprises dans l'intervalle $[0,30]$, nous pouvons donc réduire le champ de notre apprentissage aux valeurs de cet intervalle.

Une nouvelle phase de filtrage a été ajoutée suite aux tests faits avec le logiciel GIMP (chapitre 5.3). Nous pouvons étudier trois catégories de filtres à appliquer :

Filtre passe-bas : $$contour_{i,j} = \begin{cases} 1 & si \quad \Delta < LowBandThreshold \\ 0 & sinon \end{cases} \quad \text{(Eq. 5.3)}$$

Filtre passe-haut : $$contour_{i,j} = \begin{cases} 1 & si \quad \Delta > HighBandThreshold \\ 0 & sinon \end{cases} \quad \text{(Eq. 5.4)}$$

Filtre passe-bande : $$contour_{i,j} = \begin{cases} 1 & si \begin{pmatrix} \Delta > LowBandThreshold \quad et \quad \Delta < HighBandThreshold \\ avec \quad LowBandThreshold < HighBandThreshold \end{pmatrix} \\ 0 & sinon \end{cases} \quad \text{(Eq. 5.5)}$$

Le filtre passe-bas met en évidence les valeurs de contours qui sont inférieures à un seuil. A l'inverse, le filtre passe-haut met en évidence les valeurs les plus grandes : ces deux filtres ont des résultats symétriques. Ils sont très identiques et on peut n'en étudier qu'un d'entre eux. Si on choisit arbitrairement le filtre passe-haut, alors les premières étapes du filtre de contours fourniront une valeur numérique qui représentera l'appartenance de chaque point à un contour.

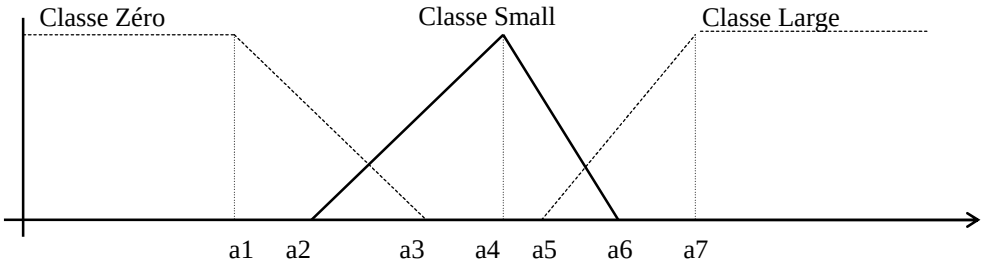
Le filtre passe-bande est configuré avec deux valeurs de seuil : les points dont les valeurs sont situées entre ces deux seuils sont retenus comme contours. Imaginons que les valeurs des

précédentes étapes du filtre sont centrées autour d'une valeur moyenne, quelle qu'elle soit, alors la valeur représentative de l'appartenance à un contour n'a aucune signification : le filtre passe-bande prend un rôle prépondérant sur le filtre flou et nous en perdrons les avantages. Pour cette raison, nous choisissons d'utiliser un filtre passe-haut qui se configure avec un seul paramètre, ce qui a pour avantage de réduire le domaine de recherche.

5.8.2. Apprentissage automatique systématique

Le nombre total de cas à étudier pour l'apprentissage est très élevé. Si nous limitons le nombre de valeurs de seuils à étudier à 30, alors le nombre total de cas s'élève à $30^{7.}$ Ensuite, le moteur de défuzzification comprend 9 règles qui peuvent prendre chacune 6 valeurs soit 9^6 cas. Ce moteur utilise aussi le paramètre r_k qui peut prendre pour chacune des classes « Zero », « Small » et « Large » 255 valeurs soit 255^3 cas. Puis le seuil du filtre passe-haut peut prendre une valeur entre 127 et 255 soit 128 cas. Au total, nous avons donc $30^7*9^6*255^3*127$ soit $2.4*10^{24}$ cas. Si on voulait calculer le résultat dans un intervalle de temps raisonnable, par exemple, inférieur à 24 heures, il faudrait alors calculer $2.8*10^{20}$ cas par seconde, ce qui est nettement supérieur à la puissance de calcul de l'ordinateur à notre disposition. De plus, l'étude d'un cas ne se résume pas à une opération élémentaire mais à un ensemble d'opérations sur une image. Par comparaison, la complexité de la résolution d'un problème de jeu d'échec est de environ 2^{400} [WINSTON 1992].

Nous choisissons de chercher les meilleurs résultats en faisant varier uniquement les valeurs des seuils des fonctions d'appartenance des classes d'entrée. Les autres paramètres seront choisis empiriquement suivant les résultats obtenus durant la phase d'expérimentation. Nous affectons aux seuils systématiquement toutes les valeurs numériques entières inférieures à 30. Le nombre de tests à réaliser étant encore relativement élevé (30^7 soit $2{,}2*10^{10}$), nous limiterons le nombre de tests en imposant certaines règles (c.f. figure 5.5).



a1 < a3, a2 < a4 < a6, a5 < a7, a2 < a3 et a5 < a6.

Figure 5.5 : Définition des règles gérant les fonctions d'appartenance.

Le nombre de tests est alors réduit à $1,5*10^7$. Les valeurs du moteur d'inférence de Sugeno sont alors fixées subjectivement :

		$d(S_{i,j}, \overline{S}_{i,j}^{LW})$		
	Δ '	Z	S	L
$d(S_{i,j}, \overline{S}_{i,j}^{SW})$	Z	-L	+Z	-S
	S	+Z	+S	+L
	L	-S	+L	+L

r_k		
Z	S	L
25	64	140

Table 5.5 : Définition des règles gérant le moteur d'inférence de Sugeno.

Après une première série de tests, nous avons pu définir un intervalle de valeurs plus restreint pour les fonctions d'appartenance ($[0,15]$). Dans le même temps, ceci nous a permis de réduire le pas de recherche de 1 à 0.5 sans augmenter le nombre de cas étudiés. Des tests seront effectués en diminuant le pas, mais ils n'aboutiront pas à un gain significatif sur la qualité de la détection des contours. Les résultats de cette phase de test sont indiqués dans les tables 5.6. Ils mettent en évidence des différences de valeurs de paramètres suivant la méthode utilisée pour calculer l'erreur.

	seuils							distance
image	a1	a2	a3	a4	a5	a6	a7	pondérée
1	0	3.5	4	4	3.5	8	7	0.0187
2	0	6	6.5	6.5	6	7	11	0.0108
3	0	4	4.5	4.5	4	6	6	0.0213
4	0	3.5	4	4	3.5	8	7	0.0308
5	0	5	5.5	5.5	5	6	9	0.0156

	seuils							distance
image	a1	a2	a3	a4	a5	a6	a7	statistique
1	0	4.5	5	5	4.5	9	9	0.1930
2	0	8.5	9	9	9	11	9.5	0.1131
3	0	5	5.5	5.5	6	6.5	6.5	0.2203
4	0	4	4.5	4.5	5	6	7	0.3171
5	0	7	7.5	7.5	9	9.5	9.5	0.1592

Table 5.6 : Résultat de l'apprentissage systématique.

Lors d'un processus d'apprentissage classique pour un problème de classification, il est d'usage de réaliser la phase d'apprentissage sur un échantillon et de mesurer la précision de cet

apprentissage sur un autre échantillon. Dans notre cas, on considère que chacune des images est un échantillon dans lequel chacun des points doit être classifié selon deux catégories. On peut alors faire un apprentissage sur une image et mesurer la qualité de cet apprentissage sur les autres images.

Les résultats de cette phase nous permettent de définir de nouvelles contraintes sur les paramètres du filtre :

- On observe des conjonctions entre certaines paires de valeurs de seuil : (a2, a5) et (a3, a4) sont égales ;
- Le seuil a1 est toujours nul ;
- La valeur maximum est 11.

La restriction du champ de recherche nous a encore permis de réduire le temps de calcul pour investiguer dans deux directions : le test d'autres combinaisons de règles pour le moteur d'inférence et l'utilisation d'autres mesures de la distance entre les images (mesure floue par exemple).

D'autre part, nous avons mis en évidence ceci : lorsque la moyenne des gradients de la petite fenêtre est plus grande (sur une image), alors les seuils des fonctions d'appartenance en entrée du filtre auront des valeurs plus grandes. Nous avons essayé d'établir une relation linéaire entre la moyenne des gradients sur une fenêtre plus large et les valeurs de seuil. Ceci a ajouté deux nouveaux paramètres α et β au filtre :

$$a_{n\,cor} = \alpha \cdot \mathrm{var}_{VLW}(G_{SW}) \cdot a_n + \beta \cdot \overline{G_{SW}} \qquad \text{(Eq. 5.6)}$$

où $\mathrm{var}_{VLW}(G_{SW})$ est la moyenne des gradients sur les petites fenêtres dans une très grande fenêtre.

5.8.3. Apprentissage automatique par algorithme génétique

Dans notre cas, on cherchera à optimiser le filtre pseudo-flou dont les résultats attendus sont clairement spécifiés en calculant automatiquement des paramètres optimums qui donneront un résultat correct, un résultat dont l'erreur mesurable est inférieure à un seuil acceptable. En effet, même si nous connaissons tous les détails de l'implémentation du filtre, il est très difficile d'envisager un algorithme d'optimisation de ce filtre qui calculerait les valeurs optimum des paramètres d'entrée.

On utilise donc l'algorithme génétique pour une période d'apprentissage durant laquelle il s'adaptera au problème. La solution identifiée comme acceptable est retenue.

Dans le chapitre 5.8.2, nous avons montré que le nombre de cas était égal à $2.4*10^{24}$ cas, soit 2^{84}. Nous pourrions donc coder notre problème avec un chromosome de 84 bits. Or dans la phase précédente, nous avions réduit l'espace de recherche en limitant la précision de la définition des seuils des fonctions d'appartenance en entrée du filtre (limitation aux valeurs inférieures à 30 définies avec un pas de 0.5). En utilisant un algorithme génétique, étendre l'espace de recherche n'a pas un cout aussi rédhibitoire que lors d'une recherche systématique. Nous choisissons donc une longueur de chromosome de 382 bits. Nous l'utiliserons en codant les valeurs des onze variables sur 32 bits et les règles du moteur de défuzzification sur 3 bits.

Les choix suivants sont faits concernant les stratégies de sélection, croisement, mutation et d'insertion.

- La sélection, qui permet de choisir quels sont les parents qui seront utilisés pour créer la future génération, est basée sur la comparaison de la qualité des parents par rapport à la moyenne de la population. C'est une méthode élitiste.

- Le croisement, qui simule les phénomènes naturels de reproduction, permet de combiner les chromosomes de deux individus. Nous choisirons un croisement en deux points.

- La politique de mutation est la suivante : plus les parents sont semblables, alors plus les chances qu'une ou plusieurs mutations se produisent seront importantes. Cette politique dite « anti-inceste » permet d'éviter une homogénéisation de la population. Avec un taux de mutation de l'ordre de 0.03, elle est reconnue comme efficace [MICHALEWICZ 1992].

- L'insertion : Un algorithme génétique classique peut rencontrer certaines difficultés lorsque la population se concentre sur un minimum local. Il est possible de l'éviter en créant des groupes au sein de la population. Nous choisissons donc une taille de population fixe constituée de 500 groupes de 500 individus. Nous insérons dans chaque nouvelle génération de population les nouveaux éléments si ceux-ci sont meilleurs qu'un élément tiré au hasard et qu'un individu identique n'est pas déjà présent dans le groupe. Les meilleurs individus sont insérés dans les autres groupes si aucun meilleur individu n'y est déjà présent.

Lors de chaque recherche, nous avons obtenu des résultats meilleurs que ceux détectés par le filtre de Sobel. Par contre, le temps de recherche a été relativement prohibitif : de quelques jours à quelques semaines sur un ordinateur équipé d'un pentium à 2ghz.

L'approche génétique montre que tous les paramètres n'ont pas la même influence sur la qualité des résultats :

- La valeur du filtre passe-haut ne varie pas beaucoup durant une phase d'apprentissage ;
- Les valeurs choisies pour les clasess « Small » et « Zero » du moteur de défuzzification sont aussi extrêmement stables durant cette phase ;
- Les valeurs des paramètres α et β sont très volatiles. Des petits changements sur leurs valeurs sont souvent combinés à des changements importants sur les valeurs des seuils des classes en entrée du filtre et sur les règles du moteur d'inférence ;
- La plupart des règles du moteur d'inférence peuvent varier jusqu'aux derniers cycles de la phase d'apprentissage. Une seule est vraiment stable : lorsque le gradient appartient à la classe « Zero » sur les fenêtres petites et larges, alors le coefficient de correction appliqué lors de la défuzzification est « largement négatif ». En fait, aucune variation de gradient n'est observée à ces points ; ils correspondent à des zones de fond de l'image sans contour. Pour les autres règles, il est difficile de distinguer une stabilité dans les résultats.

Concernant les valeurs des seuils de fonctions d'appartenance en entrée du filtre, il n'est pas possible de déterminer des valeurs types même si, statistiquement, on observe des tendances qui sont présentées dans la table 5.7.

Seuil	a1	a2	a3	a4	a5	a6	a7
Valeur	3.48	7.14	5.59	9.32	15.07	11.47	15.43
Ecart statistique	0.13	0.16	0.21	0.43	0.26	0.3	0.34

	r^k					Seuil du
Seuil	Zero	Small	Large	α	β	filtre passe-haut
Valeur	3.93	21.09	86.3	6.33	0.83	62.02
Ecart statistique	0.11	0.82	2.36	0.05	0.14	1.69

Table 5.7 : Résultats statistiques de l'apprentissage génétique.

Les règles énoncées à la suite de l'apprentissage systématique ne sont pas vérifiées :

- Plus aucune corrélation entre les valeurs des seuils des classes d'appartenance en entrée du filtre n'est observable ;
- La valeur a1 n'est pas égale à 0 mais à 3.48 ;
- La valeur maximum (pour a6 ou a7) n'est pas de 11 mais de 15.

Cette approche nous apporte de bons résultats, et pourtant nous ne pouvons garantir que nous avons obtenu le meilleur qui puisse exister. Durant ce processus d'apprentissage, nous avons remarqué une cinquantaine de solutions que nous avons jugées comme très bonnes puisqu'elles permettent de mesurer une erreur jusqu'à 25% meilleure que celle obtenue par le filtre de Sobel. Ces résultats sont présentés dans un tableau en annexe 4. Ils sont relativement disparates et nous n'avons pas réussi à trouver une corrélation fine entre tous les paramètres en utilisant une méthode graphique (figure 5.6).

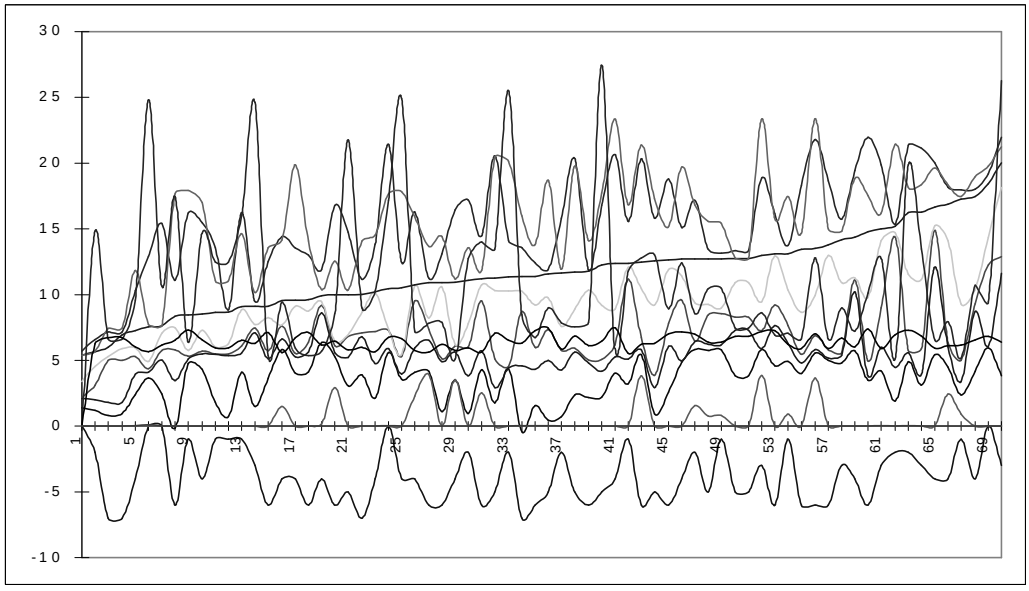


Figure 5.6 : Vision graphique des valeurs des paramètres de réglage.

Nous pouvons présenter un tableau des meilleures erreurs mesurées et quelques exemples de détection de contours dans la figure 5.7.

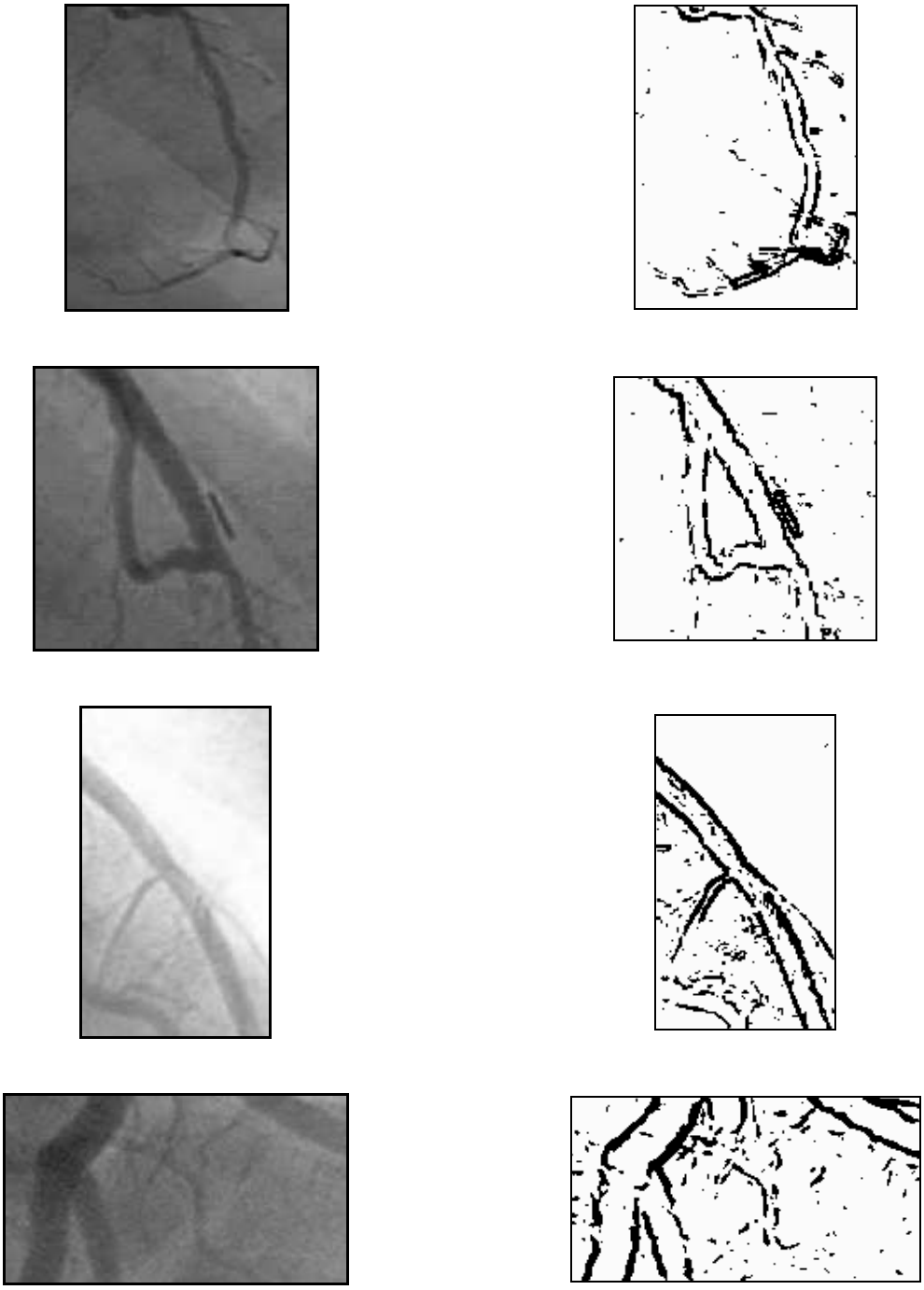


Figure 5.7 : Images issues de l'apprentissage génétique.

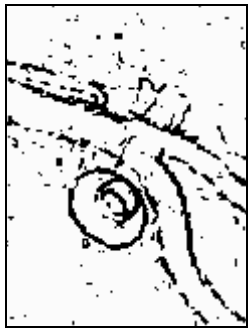


Figure 5.7 (suite) : Images issues de l'apprentissage génétique.

5.9. Présentation des résultats aux journées académiques de Iasi.

Ces journées d'échange ont eu lieu en Septembre 2004. Elles ont été l'occasion de présenter une partie des résultats de la performance du filtre pseudo-statistique dans la détection des contours. Elles n'ont pas l'objet d'une publication.

5.10. Etude de la résistance des filtres au bruit.

Nous avons introduit un bruit gaussien sur l'image 1 (voir table 5.1). Ensuite, nous avons évalué la résistance au bruit des algorithmes pseudo-statistique et de Sobel. La comparaison des résultats est visible dans la figure 5.8.

On constate que la performance du filtre pseudo-statistique est peu affectée par une augmentation du bruit. Par contre au delà d'un certain seuil, la dégradation des performances est beaucoup plus rapide pour le filtre pseudo-statistique.

On pourrait étudier la variation de ce seuil suivant les qualités des images testées.

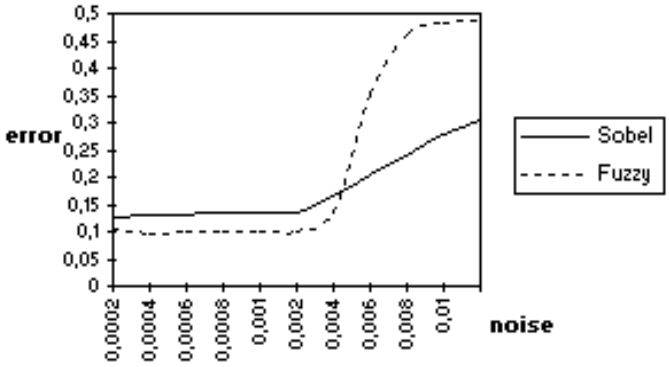


Figure 5.8 : Résistance au bruit.

5.11. Conclusion

Ce chapitre a exposé l'essentiel des travaux réalisés depuis le choix des images de test jusqu'à la réalisation du logiciel permettant l'optimisation du paramétrage du filtre par algorithme génétique, en passant par la présentation d'articles portant sur les résultats obtenus ou de cours sur les principes du traitement des images. Nous y avons montré que le filtre pseudo-statistique, avec quelques améliorations par rapport au principe de base, peut apporter un gain de 25% au filtre de Sobel sans réglage préalable.

6. Réalisation

Ce chapitre présente les différentes phases d'implémentation du filtre et la démarche que nous avons suivie pour aboutir aux résultats que nous présentons. Dans une première phase, la réalisation d'un prototype a permis la vérification de la validité du concept. Dans un second temps, nous avons développé des outils pour améliorer les résultats obtenus et les comparer avec ceux obtenus par des filtres équivalents. La première phase a été réalisée rapidement en modifiant un outil existant, sans analyse poussée de la problématique (c.f. section 6.1). Par contre, au début de la seconde phase, même si nous ne connaissions pas encore la liste précise des outils que nous aurions à développer, nous avions conscience que une réutilisation du code serait probablement nécessaire entre ces outils. Nous avons réalisé une analyse du modèle objet (c.f. section 6.2). Ces objets ont alors été utilisés dans plusieurs outils (c.f. section 6.3). Nous avons apporté régulièrement des modifications au modèle pour nous adapter aux nouveaux besoins du projet. Etant à la fois client et réalisateur du logiciel, nous avons pu utiliser une démarche en spirale. Le cycle <Objectifs, Expériences, Abstraction, Application> a été répété à plusieurs reprises pour obtenir les outils que nous présentons ici.

6.1. Phase 1 : le prototype

6.1.1. Objectif

L'objectif du prototype est de démontrer la validité du principe du filtre à partir d'un développement léger. Nous utilisons donc comme base un programme en C sous DOS qui avait été écrit dans le laboratoire. Nous avons implémenté les trois classes sur des fenêtres de taille variable et calculé la modification à apporter aux différents points de l'image initiale suivant leurs degrés d'appartenance aux classes. Nous utilisons des images de taille fixe extraites d'un film représentant une angiographie des vaisseaux cardiaques. Nous avons généré plusieurs types de résultats :

- des tableaux contenant les valeurs numériques des moyennes et des distances pour chacun des points. Ils nous ont permis de déterminer des valeurs statistiquement intéressantes pour le réglage des comparateurs fuzzy ;
- des images résultantes représentant la répartition des points de l'image initiale dans les différentes classes ;
- une image représentant l'image initiale modifiée par l'algorithme implémenté ;
- une image représentant les modifications apportées à l'image initiale.

6.1.2. Analyse des outils

Le développement en C sous DOS présente plusieurs points faibles :

- Les images sont disponibles sous différents formats qu'il faut pré-traiter afin de les transformer dans le format accepté par ce logiciel existant. Il n'est pas possible de les interpréter directement, les possibilités offertes par les librairies existantes en C sous Dos étant limitées. Le logiciel travaille donc à partir d'un tableau de 256 sur 240 points représentant chacun le niveau de gris codé sur 8 bits. La méthode actuelle pour convertir les images dans ce format consiste à les enregistrer sous un format BMP où les niveaux de gris sont codés sur 8 bits, puis à supprimer l'en-tête du fichier obtenu pour n'en garder que le tableau représentant l'image ;
- Le programme en C a une approche fonctionnelle. La gestion des filtres, des images, et des opérateurs flous selon une orientation « objet » permet de bénéficier des propriétés d'abstraction, d'héritage. Le choix est donc fait d'une réécriture en C++ qui va permettre une meilleure adaptabilité aux contraintes de flexibilité : paramétrage des seuils des comparateurs fuzzy par exemple. Cette phase passe par une modélisation des objets utilisés dans notre problématique. Elle est décrite dans la section suivante.

6.2. Phase 2 : La modélisation

Les programmes sont basés sur trois familles d'objets :

- le filtre ("modifier") dont la fonction principale est de fournir une correction à la valeur de niveau de gris d'un pixel pour le mettre en évidence si il fait partie du contour ;
- les opérateurs flous ("fuzzy_operator") sont les filtres implémentant les fonctions d'entrée du filtre flou ;
- les images ou ("ImageTeodor") contiennent toutes les informations relatives aux images.

6.2.1. La famille "modifier"

Elle représente n'importe quel filtre. Son interface propose les méthodes suivantes :

void initialise () Cette méthode permet la configuration initiale de l'objet.
float analyse (Image*, Image*) Cette méthode permet de donner une correction sur la valeur d'un point en fonction des données dans deux fenêtres.
start_debug () Cette méthode permet d'actionner un mécanisme de trace plus complet lors de l'analyse suivante.

Ce filtre est dérivé sous la forme de filtres flous pseudo-statistiques. Ces filtres représentent différents outils pour traiter l'image. Les filtres "fuzzyModifierPseudoStat" implémentent les méthodes supplémentaires suivantes :

void initialiseRule (dist *SmallDistance*, dist *LargeDistance*, fuzDelta *Delta*) Cette méthode permet d'initialiser le tableau des règles de défuzzication.
void initialiseDeffuzying (fuzDelta *fuzzyDistance*, float *crispDistance*) Cette méthode permet d'initialiser la table de défuzzyfication.
void setOperatorClassZero (fuzzy_operator *) void setOperatorClassSmall (fuzzy_operator *) void setOperatorClassLarge (fuzzy_operator *) Ces méthodes permettent d'initialiser les opérateurs qui serviront à évaluer les degrés d'appartenance aux classes zero, small et large.
float analyse (float *smallDst*, float *largeDst*) Cette méthode calcule la correction à apporter en fonction des valeurs de gradient sur la petite et la grande fenêtre.
void initStat () Cette méthode initialise le tableau des statistiques.
void printDump () Cette méthode affiche à l'écran les paramètres actuels du filtre.
void printDump (FILE *) Cette méthode affiche dans un fichier les paramètres actuels du filtre.

Les filtres " fuzzyModifierPseudoStat" sont dérivés en plusieurs classes; chacune d'entre elles propose une initialisation par défaut différente et calcule la correction à apporter différemment. Le filtre suivant a été le plus utilisé : fuzzyModifierPseudoStatMedia. Il implémente une défuzzication basée sur les algorithmes de Sugeno (avec $p = q = 0$).

6.2.2. Les opérateurs flous

Ces opérateurs sont utilisés comme filtres d'entrée du filtre fuzzy : associés aux classes Small, Zero et Large, ils implémentent les différentes sortes de fonctions d'appartenance.

Leur interface propose les méthodes suivantes :

void initialise (float *seuilBas*, float *seuilHaut*) Cette méthode permet de configurer les valeurs des seuils haut et bas de la fonction d'appartenance.
float getMinThreshold () Cette méthode retourne la valeur du seuil bas de la fonction d'appartenance.
float getMaxThreshold () Cette méthode retourne la valeur du seuil haut de la fonction d'appartenance.
float compute (float) Cette méthode renvoie le degré d'appartenance de la valeur à la fonction d'appartenance de ce filtre

Quatre classes différentes d'opérateurs flous existent. Ils représentent quatre fonctions différentes comme indiqué dans la table 6.1.

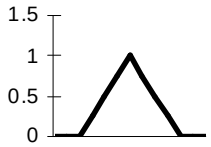
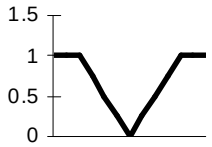
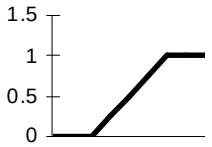
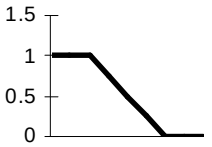
opérateur	MinMaxMin	MaxMinMax	MinMax	MaxMin
Fonction				

Table 6.1 : Définition des classes d'opérateurs flous.

Deux fonctions supplémentaires sont offertes par l'interface des opérateurs MaxMinMax et MinMaxMin :

void initialise (float *seuilBas*, float *seuilMed*, float *seuilHaut*) Cette méthode permet d'initialiser une valeur pour le seuil intermédiaire de la fonction d'appartenance qui n'est donc pas forcément symétrique.
float getMediumThreshold () Cette méthode renvoie la valeur du seuil intermédiaire de la fonction d'appartenance.

6.2.3. La famille "ImageTeodor"

Les objets "ImageTeodor" gèrent les images. La structure contenant les informations relatives à chaque point de l'image est un tableau de valeurs. On utilise ces objets pour gérer les

images à analyser, les modèles de contours, les petites et les grandes fenêtres. Un exemple d'utilisation de ces classes est montré en Annexe 1.1.

On peut classifier les méthodes en trois catégories : les fonctions d'initialisation, les fonctions d'accès à une information de l'image et celles de traitement de l'image.

Fonctions d'initialisation et de gestion des images :

Initialise(int *lHeight*,int *lWidth*) Cette méthode permet de créer une image.
void ImageTeodor::initialiseImage(int *bright*) Cette méthode donne la valeur *bright* à tous les points d'une image.
void ImageTeodor::initialiseFromBMP(const char * *imageName*) Cette méthode initialise une image à partir d'un fichier sous le format BMP.
void ImageTeodor::saveToBMP(const char * *fileName*, const char * *modeleName*) Cette méthode sauvegarde une image sous le format BMP. L'en-tête de l'image est généré à partir du contenu du fichier *modeleName*.
void dump() Cette méthode permet d'écrire dans un fichier texte les valeurs de tous les pixels d'une image.

Fonctions d'accès à une information de l'image :

int getPixel(int *i*,int *j*) Cette méthode retourne la valeur du pixel de coordonnées (i, j).
setPixel(int *lX*, int *lY*, int *bright*) Cette méthode permet de modifier la valeur du pixel de coordonnées (lX,lY).
int getHeight() Cette méthode retourne la hauteur de l'image
int getWidth() Cette méthode retourne la largeur de l'image
ImageTeodor * getScare (int *i*, int *j*, int *radius*) Cette méthode retourne une image contenant l'image de centre (i, j) et de rayon *radius*
float Average() Cette méthode retourne la moyenne des valeurs de luminosité de tous les points de l'image

float distance() Cette méthode retourne la distance moyenne pour tous les points de l'image
float distance (float avg) Cette méthode retourne la distance moyenne par rapport à la valeur *avg* pour tous les points de l'image

Fonctions de traitement de l'image :

void normalise () après cette normalisation, tous les points ayant une valeur de niveau de gris supérieure à 127 seront blancs, les autres noirs.
void normalise (int *lowStep*, int *highStep*) transforme une image en niveaux de gris en une image noir et blanc : après la normalisation, tous les points ayant une valeur de niveau de gris comprise entre lowStep et highStep seront blancs, les autres noirs.
void normaliseLow (int *lowStep*) après cette normalisation, tous les points ayant une valeur de niveau de gris supérieure à *lowStep* seront blancs, les autres noirs.
void equalise (int* *miValue*, int* *maValue*) Les valeurs de niveaux de gris initiales sont comprises dans l'intervalle $[miValue, maValue]$. Ce filtre applique une translation linéaire des valeurs de niveaux de gris initiales vers l'intervalle $[0, 255]$.
void evaluateDistance(ImageTeodor* *Modele*, float * *distRelativ*, float * *distNormal*, float * *distTeodor*, float * *distCorrected*, int **class1*, int **class2*, int **nbModelPoint*) renvoie une évaluation de la distance entre l'image courante et le Modèle selon plusieurs méthodes de calcul différentes. Les variables class1 et class2 contiennent le nombre de points détectés à tort et *nbModelPoint* le nombre de points de l'image qui a servi de modèle.

6.3. Développement des outils d'analyse fine

Au-delà de cette modélisation, nous avons pu intégrer ces objets dans une librairie et les utiliser au sein de plusieurs outils :

- un outil d'apprentissage des paramètres basé sur un test systématique de toutes les valeurs possibles d'un nombre restreint de paramètres ;
- Un outil d'apprentissage basé sur une recherche à base d'algorithme génétique ;
- Un plugin intégré au logiciel GIMP ;

- Un programme de mesure des erreurs entre une image modèle et un résultat de détection de contours.

6.3.1. Apprentissage systématique

Utilisation

Notre algorithme d'apprentissage nous fournit plusieurs données en sortie :

- un tableau mesurant le meilleur résultat pour chacun des opérateurs de comparaison (c.f. figure 6.1 : Sauvegarde des résultats lors de la recherche systématique.);
- l'image correspondant au meilleur résultat.

Les résultats sont agrégés dans des tableaux (Open Office ou Excel) qui permettent une analyse a posteriori.

relative error :0, 5, 6, 6, 6, 7, 7 =>9636.105.

absolu error :0, 5, 6, 6, 6, 7, 7 =>0.148.

Corrected error :0, 4, 5, 5, 5, 6, 6 =>0.226.

Teodor error :0, 5, 6, 6, 6, 7, 7 =>778.001.

Figure 6.1 : Sauvegarde des résultats lors de la recherche systématique.

6.3.2. Apprentissage par algorithme génétique

Objectif

Ce programme permet de rechercher un paramétrage du filtre en basant cette recherche sur un algorithme génétique.

Utilisation

Ce programme est écrit en C++ et s'exécute sous Dos. Il utilise la libraire libGA dont la version 0.21 du code source est disponible gratuitement sur le site http://sourceforge.net/projects/cpplibga/files/.

La commande en ligne de ce programme est succincte : AG.exe. Il n'existe pas d'options.

Le répertoire "C:/test/images/result/" doit exister préalablement. Les cinq images à analyser doivent se situer dans le répertoire " C:/test/images" et s'appeler "base1.bmp" à "base5.bmp". Les modèles de contours doivent aussi se situer dans le répertoire " C:/test/images" et s'appeler "

modele1.bmp" à " modele5.bmp". Toutes les images traitées doivent être dans un format BMP non compressé et elles doivent être encodées en niveau de gris.

Durant l'exécution, le programme affiche régulièrement quelques informations relatives à la recherche en cours (figure 6.2, ligne 14) :

- Nombre de cycles effectués ;
- Nombre de sous-ensembles de solutions déjà évalués ;
- Nombre de solutions évaluées par seconde ;
- Meilleur résultat connu.

Il affiche aussi les paramètres de toutes les solutions trouvées pour lesquelles la distance mesurée est inférieure à un seuil paramétrable. La figure 6.2 présente un exemple de sortie d'écran.

```
1   zero : 0.0372        0.2348      small : 0.2175      0.9065      0.9277      large: 0.9150      0.9208
2   rule table:
3   SW \ LW      Zero   Small  Large                high <=> low
4   Zero          5      6      2          0<=>0       0<=>0          0<=>0
5   Small         0      3      2          0<=>0    2117<=>0       3626<=>0
6   Large         1      5      4          0<=>0     198<=>0        214<=>91936
7   deffuzzying table:
8   Zero         pos: -3      neg:  3
9   Small        pos: -4      neg:  4
10  Large        pos:-46      neg: 46
11  filter low-high: 156 256      min/max :256, 81255.
12  factor grade   : 3.983          avg: 2.402            offset: -6.592.
13  erreur: 0.418269
14  ****  **** **** **** **** **** ****
15  step: 30 solutions: 5389 SpS: 4.26326 F: -0.418269
```

Figure 6.2 : Affichage des résultats lors de la recherche par algorithme génétique.

Sur la première ligne, on trouve les paramètres de seuils des fonctions d'entrées du filtre flou. Les lignes 4 à 6 décrivent les réglages du moteur d'inférence sur la partie gauche du tableau. La partie droite contient des statistiques sur la répartition du nombre de points appartenant à chaque combinaison de classes avec un degré d'appartenance de 100 % pour chacune des classes. Les lignes 8 à 10 contiennent les valeurs utilisées pour la traduction de la correction exprimée de manière floue dans une valeur numérique. La ligne 11 présente l'échelle des valeurs calculées après cette correction (à droite) et le réglage des seuils du filtrage final sur la partie gauche. La ligne 12 affiche les trois paramètres de correction. Sur la ligne 13 est affichée la distance évaluée. La ligne 15 indique des paramètres de performance : le nombre de cycles d'apprentissage effectué, le nombre de solutions dans le cycle, le nombre de solutions par seconde et la dernière erreur mesurée. Ce message est rafraîchi à intervalle régulier.

Trois possibilités d'interaction existent :

- l'appui de la touche 's' permet de sauvegarder la prochaine solution qui sera évaluée par l'algorithme. On trouve alors dans le répertoire "C:/test/images/result/" des fichiers contenant les contours détectés pour chacune des images ;
- l'appui de la touche 'v' permet de visualiser la prochaine solution qui sera évaluée par l'algorithme ;
- l'appui de la touche 'd' permet une visualisation très détaillée de l'analyse qui est faite en un point de coordonnées *(12,12)* lors de la prochaine évaluation d'une solution par l'algorithme.

L'arrêt du programme se fait par l'utilisation de la touche "CTR+C".

6.3.3. GIMP

Objectif

Ce programme permet d'appliquer le filtre sur une image au sein du logiciel GIMP d'une manière similaire aux filtres de Sobel ou de Laplace. En 2010, la question se poserait de l'utilisation du logciel imageJ à la place de GIMP. Celui-ci propose une solution équivalente implémentée sous java [IMAGEJ] pour laquelle les filtres de Deriche et de contours actifs sont disponibles [BOUDIER].

Utilisation

Le principe est relativement simple puisque le filtre pseudo-statistique apparait dans le même menu que les filtres de Sobel ou de Laplace. Le processus d'utilisation est donc le suivant :

- Ouvrir une image avec le logiciel GIMP ;
- Sélectionner une zone de l'image ;
- Appliquer le filtre en cliquant avec le bouton droit puis en choisissant "Filtres/Detection de bord/PseudoStat" comme sur la figure 6.3.

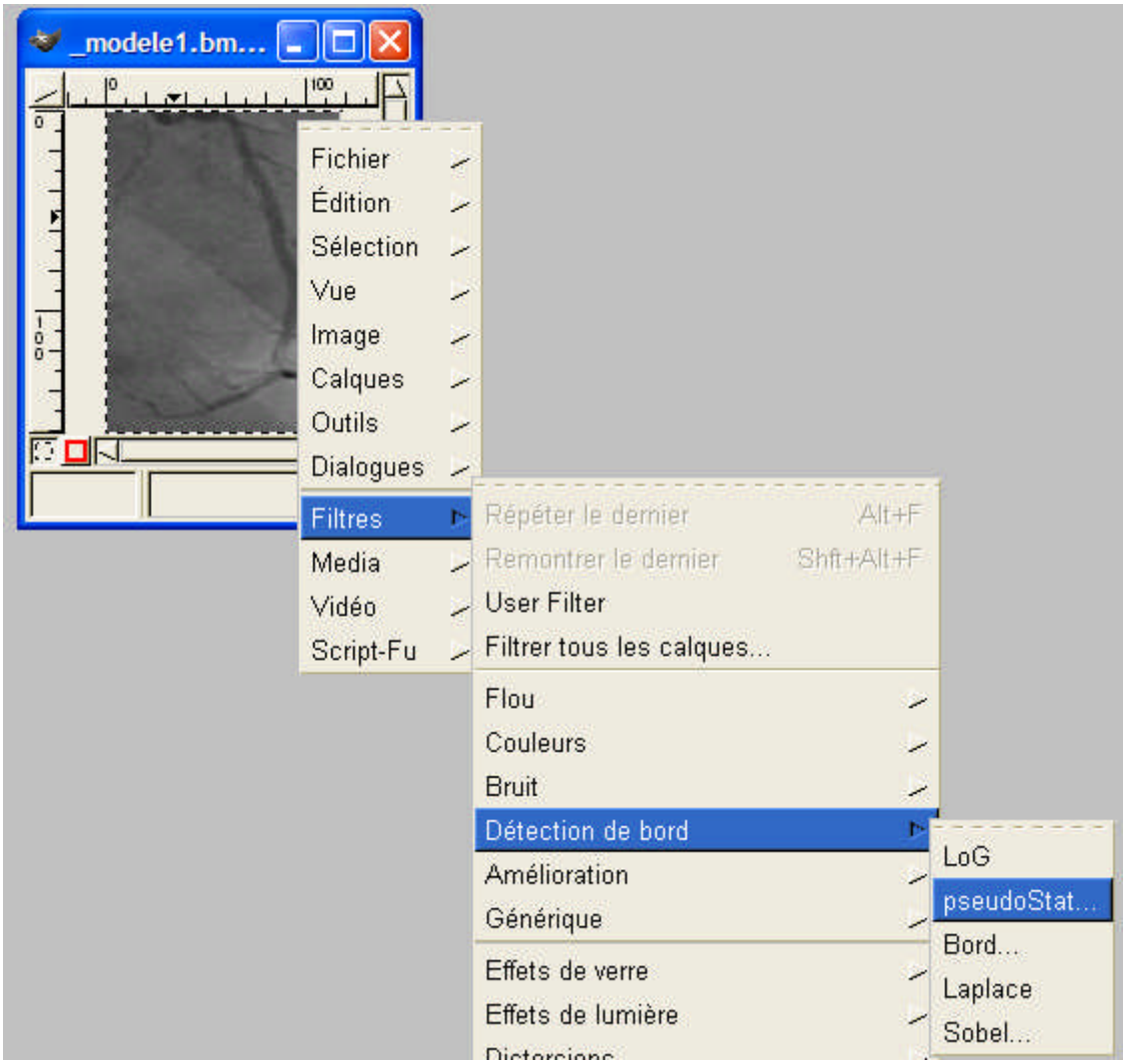


Figure 6.3 : Application du filtre à une image ou une zone sélectionnée de l'image.

Le résultat s'affiche alors dans la fenêtre contenant l'image.

Compilation

Elle s'effectue sous Visual C++ 6.0. La version 1.2.5 de Gimp a été utilisée en 2004 avec la librairie gtk en version 1.3.0. (En 2010, le logiciel Gimp est maintenant en version 2.6.7 et le greffon compilé pour la version 1.3.0 n'est pas compatible avec cette nouvelle version).

Le développement du code a été relativement simple : d'une part, le code source de Gimp étant disponible, il a été possible de se baser sur les implémentations des autres greffons pour construire celui du filtre pseudo-statistique. D'autre part, les qualités de réutilisabilité du code de la librairie ont été mises à profit.

Configuration

Pour que GIMP prenne en compte le greffon, il faut ajouter le répertoire où il est compilé dans la liste. Pour ceci, ouvrir la fenêtre "Preferences" dans le menu "Fichiers", puis choisir l'option "Greffon" et ajouter le nom du répertoire a la liste déjà présente (voir figure 6.4).

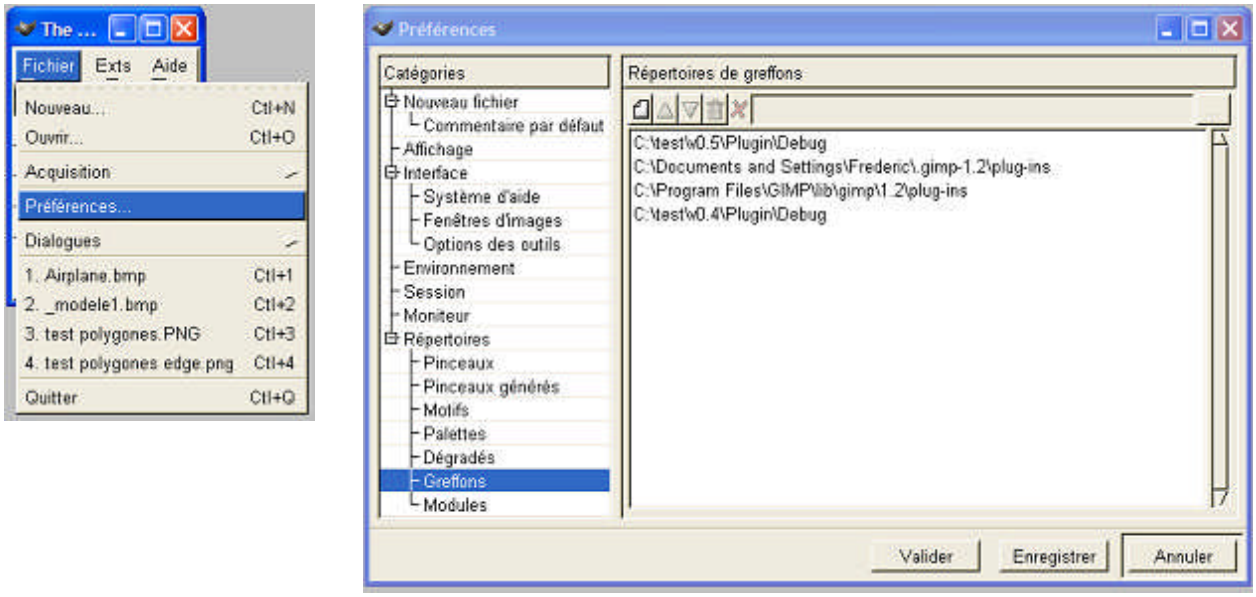


Figure 6.4 : Configuration de l'application GIMP.

Amélioration à envisager

Le greffon demande à être compilé à nouveau après chaque changement de configuration. Il serait souhaitable qu'il soit initialisé grâce à un fichier de paramétrage.

6.3.4. Mesure des erreurs

Objectif

Ce programme donne une mesure de la distance entre un contour détecté et un contour idéal selon deux méthodes de calcul :

- Distance euclidienne ;
- Distance euclidienne pondérée.

Il donne aussi trois autres informations :

- nombre de points de l'image ;
- nombre de points de contours non détectés (Classe 2) ;
- nombre de points de contours détectés à tort (Classe 1).

Utilisation

Ce programme est écrit en C et s'exécute sous Dos. La commande en ligne de ce programme est succincte : *" difeucli.exe fichier1 fichier2 "* où :

- *fichier1* est le nom de l'image à comparer avec le contour de référence ;
- *fichier2* est le nom du fichier du contour de référence.

Les valeurs par défaut sont : C:\Image.bmp pour le *fichier1* et C:\Modèle.bmp pour le *fichier2*.

Les images traitées doivent être dans un format BMP non compressé et elles doivent être encodées en niveau de gris.

Un fichier c:/result.txt est créé. Il contient les résultats de l'analyse ligne par ligne de l'image : numéro de ligne; erreur calculée sur la ligne; erreur cumulative;

Les images peuvent être modifiées et une nouvelle évaluation de l'erreur faite sans relancer le programme en appuyant sur la touche "espace". Un appui sur une autre touche arrête le programme.

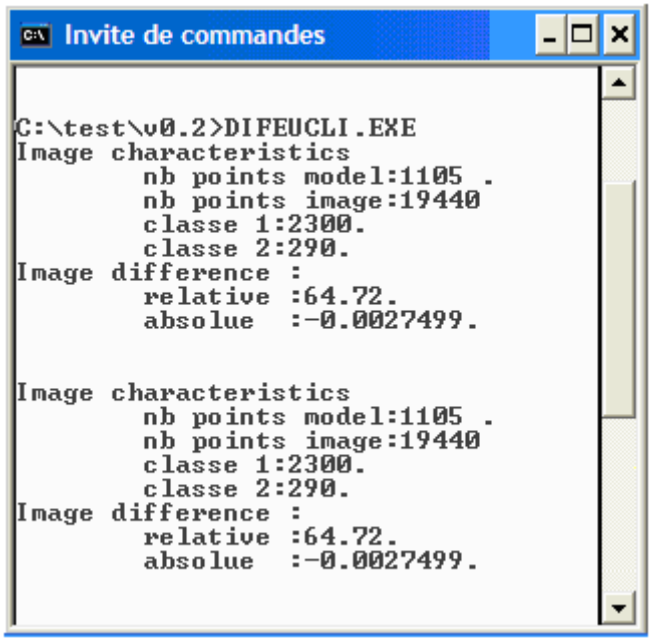


Figure 6.5 : Exemple de sortie d'écran.

Compilation

Ce programme se compile en utilisant le langage Borland C++, version3.1.

7. Conclusion

7.1. Bilan scientifique

Le but de cette recherche était d'étudier la possibilité d'utiliser une architecture basée sur la logique floue pour filtrer les contours présents sur une image.

Pour cela nous avons recherché des méthodes de quantification de la validité des résultats pour les comparer entre eux ou avec ceux obtenus par d'autres algorithmes plus classiques. Nous avons démontré que la distance Euclidienne n'est pas adaptée dans le cadre de ce problème au contraire de l'erreur statistique. Cette dernière a, de plus, l'avantage d'être indépendante des caractéristiques intrinsèques de l'image si les points de contours représentent moins de 20 % de points de l'image.

Cela nous a aussi permis d'améliorer les performances du filtre en effectuant des réglages fins sur l'ensemble des paramètres qui le caractérise pour qu'il puisse être utilisé sans réglage préalable. Finalement, nous avons pu mesurer une qualité de détection environ 30% supérieure à celle du filtre de Sobel, considéré, à l'époque, comme une référence dans le domaine.

7.2. Travaux futurs

Lors de l'étude de la résistance au bruit de l'algorithme pseudo-statistique, on a observé l'existence d'un seuil à partir duquel la qualité de détection se dégrade. On pourrait prolonger l'étude en cherchant une corrélation entre la variation de ce seuil et les qualités des images testées.

Le filtre a été utilisé pour détecter les contours. Il serait aussi envisageable d'utiliser le même algorithme avec d'autres réglages pour effectuer des modifications différentes sur les images comme le lissage, le débruitage ou tout autre opération réalisée par une convolution uniforme sur tous les points de l'image.

De même, il serait intéressant d'étendre son champ d'application à des images en couleur en l'appliquant en parallèle sur les trois canaux de couleur. Les images disponibles sur le site [MARATHON] pourraient être utilisées dans ce but et aussi pour comparer les résultats d'une détection de contours sur un jeu d'images plus usuelles que le domaine de l'angiographie médicale.

7.3. Applications envisageables

Cette méthode de filtrage peut être appliquée partout où les filtres de Sobel ou de Canny sont utilisés. Les applications dans le domaine de la vision artificielle sont très nombreuses. Certaines applications de segmentation d'image utilisent le résultat d'un tel filtrage pour alimenter un filtre à contour actif qui vient affiner et compléter le résultat de l'évaluation. On constate aussi qu'en vision artificielle, aujourd'hui, la reconnaissance des contours n'est plus une étape indispensable du processus d'interprétation des images. Par exemple, dans le domaine de l'angiographie médicale, les champs de Markov sont utilisés pour segmenter les images et reconstruire une image en deux dimensions pour détecter les sténoses (rétrécissement des artères coronaires) [CHAKCHOUK *et al.* 2009].

7.4. Bilan personnel

Ce stage a été très riche de par l'intérêt du sujet et des rencontres que j'ai faites durant cette année passée en Roumanie. Travailler dans un laboratoire de recherche est passionnant surtout lorsque, guidé dans la recherche, on aboutit à des résultats probants. Il est aussi intéressant de partager ses connaissances avec les collègues, les étudiants et d'obtenir en retour des interrogations, des rapprochements avec des méthodes utilisées par ailleurs qui permettent des avancées.

Enfin, ce stage m'a aussi permis de vivre un an dans ce pays aux portes de l'Europe. Je confirme que, malgré la distance, le peuple roumain est très proche de la France de par la culture et la mentalité.

Références bibliographiques

[BELLET 1998] Bellet F., Une approche incrémentale a base de processus coopératifs et adaptatifs pour la segmentation des images en niveaux de gris, rapport de thèse / Juin 1998, INPG Grenoble

[BEZDEK at al. 1994] J.C. Bezdek and D. Kerr, Training Edge Detecting Neural networks with Model-Based Examples, Proc 3rd International Conference on Fuzzy Systems, FUZZ-IEEE'94, Orlando, Florida, USA. pages 894-901, June 26 - 29, 1994

[BOW 2002] Sing-Tze Bow, Pattern Recognition and Image Pre-processing, second Edition, 698pp., ISBN 0-8247-0659-5, Marcel Decker Inc., New York, 2002

[BRIDGES *et al.* 1991 C.L Bridges and D.E Goldberg. An analysis of multipoint crossover. In Proceedings of the Foundation Of Genetic Algorithms. FOGA, 1991.

[CANNY 1986 J. F. Canny. A computational approach to edge detection. IEEE Transactions on pattern analysis and machine intelligence, pages 679-698, 1986

[CHAKCHOUK *et al.* 2009] M. CHAKCHOUK, S. SEVESTRE-GHALILA, C. GRAFFIGNE, M. JAÏDANE, H. RAJHI, R. HAMZA Reconstruction 3D de sténoses par modélisation markovienne à partir de plus de deux images d'angiographie rotative par rayons X, thèse soutenue le 20 Février 2009 à l'université Paris Descartes

[CHOW *et al.*1972] C.K. Chow T. K.aneko Boundary detection of radiographic images by a threshold method : Frontiers of pattern recognition by Satosi Watanabe, 1972

[DERICHE 1990] R. Deriche, Fast algorithms for low level vision, IEEE transactions on pattern analysis and machine intelligence, Vol. 12, No. I. January 1990, pages 78-87.

[EFFORD 2000] N. Efford, Digital Image Processing, Addison-Wesley, Reading, MA, 2000, pages 164–173

[EMMECHE 1994] Emmeche C., Garden in the Machine. The Emerging Science of Artificial Life, Princeton University Press, 1994, pages 114 et suivantes.

[FAWCET 2003] Fawcet T., ROC Graphs : Notes and Practical Considerations for Researchers, document http://www.hpl.hp.com/personal/Tom_Fawcett/papers consulté sur internet en Septembre 2004.

[FOGEL *et al.* 1966] L. Fogel, A. Owens, M. Walsh, Artificial Intelligence Through Simulated Evolution, John Wiley & Sons, Inc, New York, 1966.

[GACOGNE 1997] Louis Gacôgne, Eléments de logique floue, Hermès, 1997.

[HARVEY *et al.* 1997] Harvey A. Cohen Craig McKinnon and J. You, Neural-Fuzzy Feature Detectors, DICTA-97, Auckland, N.Z., Dec 10-12, pages 479-484.

[HOLLAND 1975] Holland J.H. Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, 1975.

[HOLTE *et al.* 2002] Drummond C. and R. C. Holte, 'Explicitly representing expected cost : An alternative to ROC representation', in Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000, pages 198–207.

[HOLTE *et al.* 2004] Drummond C. and R. C. Holte, What ROC Curves Can't Do (and Cost Curves Can), ECAI 2004 Workshop on ROC Analysis in Artificial Intelligence.

[KASS *et al.* 1987] Kass M., Witkin A. and Terzopoulos D. Snakes : Active contour models, International Journal of Computer Vision, Volume 1, Number 4 / janvier 1988, pages 321-331

[KONISHI *et al.* 2003] S. Konishi A. L. Yuille, J. M. Coughlan, and S. C. Zhu, Statistical Edge Detection : Learning and Evaluating Edge Cues, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 25, no 1, Jan. 2003, pages 57-74

[LADD 1999] S.R. Ladd, Genetic Algorithm in C++, 1999-2000

[LIANG *et al.* 2003] Lily R. Liang, Carl G. Looney. Competitive Fuzzy Edge Detector, International Journal of Applied Soft Computing. 3(2) 2003, pages 123-137

[MARTIN *et al.* 2000] Alberto Martin and Sabri Tosunoglu, Image Processing Techniques for Machine Vision, FIU, Florida Conference on the Recent Advances in Robotics - 2000

[MICHALEWICZ 1992] Zbigniew Michalewicz, Genetic Algorithms + Data. Structures = Evolution Programs., Springer-Verlag, 1992 pages 235-236

[SALOTTI *et al.* 1996] Salotti, M., Bellet, F., Garbay, C., Evaluation of Edge Detectors : Critics and Proposal. Workshop on Performance Characteristics of Vision Algorithms, Cambridge – 1996

[SUGENO 1985] M. Sugeno, Industrial applications of fuzzy control, Elsevier Science Pub. Co., 1985

[SYSWERDA 1989] Syswerda, G., Uniform crossover in genetic algorithms. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2-8. Morgan Kauffman.

[WINSTON 1992] Winston P., Artificial Intelligence, Third Edition, Addison-Weysley, 1992

[ZADEH *et al.* 1987] L. Zadeh and R.Yager, Fuzzy Sets and Applications : Selected Papers by L.A. Zadeh, John Wiley, 1987

[ZADEH 1965] L. Zadeh,, Fuzzy sets, information and controls 8, 1965, pages 338-353

Les publications liées à ce travail

[MAYER *et al.* 04a] F. Mayer, Horia N. Teodorescu, Pseudo Statistic Edge Detector using Neuro Fuzzy Computation: Error Distance Evaluation. Proc. ECIT - Iasi (2004)

[MAYER *et al.* 04b] F. Mayer, Horia N. Teodorescu, On a Pseudo Statistic Edge Detector using Neuro Fuzzy Computation

[MAYER *et al.* 04c] F. Mayer, Horia N. Teodorescu, Neuro-Fuzzy Edge Detector: Experimental Results. Proc. MIC - Roma (2004) - never published

Sites Internet

[ACAD] http://www.acad.ro/ consulté sur internet en Janvier 2010 (en partie en langue française).

[BOUDIER] http://www.snv.jussieu.fr/~wboudier/softs.html consulté sur internet en Avril 2010.

[CERFS] http://www.etc.tuiasi.ro/sibm/old/Cercetare/CERFS.htm consulté sur internet en Janvier 2010 (en langue roumaine).

[ECIT 2004] http://www.etc.tuiasi.ro/sibm/ECIT%272004/index_en.htm consulté sur internet en Janvier 2010.

[GIMP] http://www.gimp.org/ consulté sur internet depuis Janvier 2004.

[IASI] http://www.bucovine.com/fr/pages/villes/iasi.shtml consulté sur internet en Janvier 2010.

[ICS] http://www.iit.tuiasi.ro/institute_tp.php consulté sur internet en Janvier 2010 (en langue roumaine).

[IMAGEJ] http://rsb.info.nih.gov/ij consulté sur internet en Avril 2010.

[MARATHON] http://marathon.csee.usf.edu/edge/edge_detection.html consulté le 10 Janvier 2010.

[TUIASI MASTER] http://www.etc.tuiasi.ro/documents/planuri_invatamant/master/Plan_SEIII.pdf consulté sur internet en Janvier 2010 (en langue roumaine).

Annexes

Annexe 1: Exemple d'utilisation des classes d'objets ImageTeodor.

L'algorithme ci-dessous montre un cas d'utilisation des familles d'objets (c.f. section 6.2) à travers la réalisation d'un filtre pseudo-statistique.

```
// on instancie le filtre
        ourModifier = new fuzzyModiferPseudoStatMedia();

//on instancie et initialise les opérateurs, les seuils de défuzzification
        operatorZero  = new fuzzy_operator_MaxMin();
        operatorSmall = new fuzzy_operator_MinMaxMin();
        operatorLarge = new fuzzy_operator_MinMax();
        operatorZero->initialize(0,5);
        operatorSmall->initialize(2,8,12);
        operatorLarge->initialize(7,15);
        DeffuzzingStep1 = 6;
        DeffuzzingStep2 = 26;
        DeffuzzingStep3 = 56;

// on initialise le filtre
        ourModifier->setOperatorClassZero( operatorZero);
        ourModifier->setOperatorClassSmal( operatorSmall);
        ourModifier->setOperatorClassLarge(operatorLarge);

        ourModifier->initialiseRule (ZERO, ZERO, 5);
        ourModifier->initialiseRule (ZERO, SMALL, 5);
        ourModifier->initialiseRule (ZERO, LARGE, 5);
        ourModifier->initialiseRule (SMALL, ZERO, 5);
        ourModifier->initialiseRule (SMALL, SMALL, 5);
        ourModifier->initialiseRule (SMALL, LARGE, 5);
        ourModifier->initialiseRule (LARGE, ZERO, 5);
        ourModifier->initialiseRule (LARGE, SMALL, 5);
        ourModifier->initialiseRule (LARGE, LARGE, 5);

// valeurs de fuzDelta {ZEROneg, ZEROpos, ZEROnull, SMALLneg, SMALLpos, LARGEneg, LARGEpos};
        ourModifier->initialiseDeffuzying((fuzzyModiferPseudoStat::fuzDelta) 0, (float) (DeffuzzingStep1*(-1)));
        ourModifier->initialiseDeffuzying((fuzzyModiferPseudoStat::fuzDelta) 1, (float) DeffuzzingStep1);
        ourModifier->initialiseDeffuzying((fuzzyModiferPseudoStat::fuzDelta) 2, (float) 0);
        ourModifier->initialiseDeffuzying((fuzzyModiferPseudoStat::fuzDelta) 3, (float) DeffuzzingStep2*(-1));
        ourModifier->initialiseDeffuzying((fuzzyModiferPseudoStat::fuzDelta) 4, (float) DeffuzzingStep2);
        ourModifier->initialiseDeffuzying((fuzzyModiferPseudoStat::fuzDelta) 5, (float) DeffuzzingStep3*(-1));
        ourModifier->initialiseDeffuzying((fuzzyModiferPseudoStat::fuzDelta) 6, (float) DeffuzzingStep3);


// On créé l'image originale, et le modèle ainsi que une copie pour y stocker les contours détectés.
// L'initialisation à partir e l'image originale permet de créer une image de mêmes dimensions
        ourImage0 = new ImageTeodor ();
        ourImage0->initialiseFromBMP( imageName);
        ourModele0 = new ImageTeodor ();
        ourModele0->initialiseFromBMP( modeleName);
        ourImageModified0 = new ImageTeodor ();
        ourImageModified0->initialiseFromBMP( imageName);
```

Annexes

```
// On créé et on initialise les trois fenêtres servant au calcul de la correction
        theSmallWindow = new ImageTeodor();
        theLargeWindow = new ImageTeodor();
        theSmallWindow->Initialise(3,3); // fenêtre de 3 pixels de côté
        theLargeWindow->Initialise(5,5);

        for ( int row = 0; row < ourImage0->getHeight(); row++){
                for ( int j = 0; j < ourImage0->getWidth() ; j++ ){
                        theSmallWindow->initialiseFromImageScare( ourImage0, j, row);
                        theLargeWindow->initialiseFromImageScare( ourImage0, j, row);

                        int newBrightness = ourModifier->analyse(theSmallWindow->distance(),
                                                                theLargeWindow ->distance() );
                        ourImageModified0->setpixel( i, j, newBrightness);
                }
        }
        ourImageModified0->equalise(45, 157);
        ourImageModified0->evaluateDistance(ourModele, &distRelativ, &distNormal,  &distTeodor, &distCorrected,
&class1, &class2, &nbModelPoint);
```

L'algorithme ci-dessous, montre un processus d'initialisation du filtre dans le cas particulier d'une recherche cyclique (algorithme génétique ou systématique) où des objets de type image sont utilisés comme tableaux de stockage pour mémoriser le résultat des calculs qui ne varient pas d'un cycle à l'autre.

```
        ourMediaSmallW0 = new ImageTeodor ();
        ourMediaSmallW0->initialiseFromBMP( imageName);
        ourMediaLargeW0 = new ImageTeodor ();
        ourMediaLargeW0->initialiseFromBMP( imageName);
        ourMediaVeryLargeW0 = new ImageTeodor ();
        ourMediaVeryLargeW0->initialiseFromBMP( imageName);
        for ( int row = 0; row < ourImage0->getHeight(); row++){
                for ( int j = 0; j < ourImage0->getWidth() ; j++ ){
                        theSmallWindow->initialiseFromImageScare( ourImage0, j, row);
                        theLargeWindow->initialiseFromImageScare( ourImage0, j, row);
                        theVeryLargeWindow->initialiseFromImageScare( ourImage0, j, row);
    //  ImPrecision est un facteur permettant de conserver un minimum deprecision malgré le stockage sous forme de
nombres entiers.
                        ourMediaSmallW0->setPixel(j, row, ImPrecision*theSmallWindow->distance());
                        ourMediaLargeW0->setPixel(j, row, ImPrecision*theLargeWindow->distance());
                }
        }
```

Annexes

Annexe 2: Article Pitesti 2004

On a Pseudo Statistic Edge Detector using Neuro Fuzzy Computation

Frédéric Mayer[1], Horia-Nicolai Teodorescu[2],

1-Conservatoire National des Arts et Métiers - Paris and TIMC - SIC team - Grenoble France and Institute for Theoretical Informatics of the Romanian Academy - fmayer@etc.tuiasi.ro
2-Technical University of Iasi and Institute for Theoretical Informatics of the Romanian Academy - hteodor@etc.tuiasi.ro

Abstract: This paper presents a new edge detector algorithm using a neuro-fuzzy architecture. This study also includes a comparison of the best known algorithms in edge detection. The analysis of edginess can be seen as a classification problem. Historically, the automation of this process was first supervised. Later, the detectors have tried to detect by themselves the edges, as the human eye sees them. As the resulting edge maps are usually used during upper level analysis leading to image understanding, their quality is critical. Then, new ideas to improve the results of this task are still welcome.

Keywords: edge detection, neural fuzzy detector

1. Introduction

The aim of this paper is to present an enhancement in the domain of computer vision [1]. The purpose of computer vision is to program a computer to "understand" a scene or features in an image. Typical goals of computer vision include [2]:

- The detection, segmentation, location, and recognition of certain objects in images (e.g., human faces)
- The evaluation of results (e.g., segmentation, registration)
- Registration of different views of the same scene or object
- Tracking an object through an image sequence
- Mapping a scene to a three-dimensional model of the scene; such a model might be used by a robot to navigate the imaged scene
- Estimation of the three-dimensional poses of humans and their limbs.

These goals are achieved by means of pattern recognition, statistical learning, projective geometry, image processing, graph theory and other fields.

Edge detection is a research field within computer vision and feature extraction. The goal of edge detection is to mark the points in an image at which the intensity changes sharply. Sharp changes in image properties usually reflect important events and changes in world properties. Then an *edge enhancement* can increase the contrast between the edges and the background in such a way that edges become more visible. *Edge tracing* is the process of following the edges, usually collecting the edge pixels into a list. A related field to edge detection is segmentation. Segmentation is the division of a given digital image into contiguous regions. In current computer vision algorithms, the similarity of image parts is usually defined in terms of color and texture. The goal to automatically segment images into semantically meaningful parts is very difficult to achieve. Segmentation and edge detection are complementary as one detects characteristic areas in the image and the other one detects the boundaries between these regions.

For most of the non-supervised systems, the best performances are obtained by the treatment of the variation of gradients. Following P[r] Teodorescu proposal, we chose to explore a new idea classifying the gradient information with a three sets fuzzy system, computing the memberships values by a rule engine and using the Sugeno defuzzification

Annexes

method.

In the second section, we give some basic definitions about the image information, what an edge is and state some denotations. Next, we present several algorithms as Sobel, Laplace, Canny, and CFED. And then, in the fourth section, we present the basics of our new neuro-fuzzy algorithm. In the discussion, we will compare the algorithms.

2. Definitions and basics

We denote:

- *I* an image.
- *i* and *j* the coordinates of a pixel over the *x* and *y* axes.
- *M* and *N* the dimensions of the image over the *x* and *y* axes.
- *f(i,j)*, the function which returns the luminosity or the intensity at the pixel with coordinates *i* and *j*.

We assume that all the algorithms are representing white edges over a black background.

In an image, color or luminosity information is coded for each pixel, that is each point. The luminosity can be evaluated from the color information depending on how it is coded. Every coding system (RGB, CYG...) has established some rules permitting this computation.

The gradient at one pixel is the difference of luminosity between this pixel and an adjacent one or over a range of pixels.

The eye sensibility over the entire range of possible values is not equally shared. So, the visibility of the gradients is linked to the amplitude of the difference of data and their location over the range. The data representing in Figure 1 is more understandable than the printing a picture because small numeric variations are better appreciated in this way [3]. However, representing large images in a paper document with such a method is unrealistic. Therefore, we can use this representation only for details. Another way to represent the variation of luminosity is to draw a graph. But this can only be easily done in a single direction as in Figure 2.

Pixel coordinates	1	2	3	4	5	6	7
1	5	7	6	4	153	148	149
2	3	6	9	10	145	150	136
3	5	7	30	60	90	120	150

Figure 1. Image luminosity representation

A model of an edge can be ideally represented by the *Step Edge* [8]. It is simply a change in gray level occurring at one location. For instance, if we consider the image in Figure 1, a step edge can clearly be detected between the 4th and 5th pixels on the first two ranks. The step edge is an ideal model because in a real image rarely a change in a gray level occurs due to noise and illumination disturbances. Due to digitization and compression loses, it is unlike that the image will be sampled in such a way that all of the edges happen to correspond exactly with a pixel boundary. The actual position of the edge is considered to be in the center of the ramp connecting the low gray level to the high gray level. When the variation of luminosity is progressive, this is a *Ramp Edge*. Figure 2 shows these two cases:

step Edge *ramp edge*

Figure 2. Edge configuration

Taking an edge to be a change in intensity taking place over a number of pixels, edge detection algorithms generally calculate a derivative of this intensity function.

Edges may be viewpoint dependent, i.e. edges may change as the viewpoint changes, and typically reflect the geometry of the scene, objects occluding one another and so on, or may be viewpoint independent - these generally reflect properties of the viewed objects such as markings and surface shape. In two dimensions and higher, the concept of a projection has to be considered.

For instance, a typical edge might be the border between two blocks of different colors; in contrast a line can be a small number of pixels of a different colors on an otherwise unchanging background. There will be one edge on each

Annexes

side of the line.

The convolution is the mathematical operation used to calculate the new value in a point depending on the neighbors values. If we take a 3x3 window, we apply a 3x3 mask and the resultant value is:

$$g(i,j)=\sum_{b=1}^{3}\sum_{a=1}^{3} a_{a,b}\cdot f(i+a-1,\ j+b-1) \text{ using the mask } \begin{vmatrix} a_{1,1}, a_{2,1}, a_{3,1} \\ a_{1,2}, a_{2,2}, a_{3,2} \\ a_{1,3}, a_{2,3}, a_{3,3} \end{vmatrix} \quad (1)$$

3. The existing algorithms

This section describes and compares four approaches. Both of them compute the gradient information. The complexity of these edge detector is variable: the Canny or CFED algorithms do not only detect the edge points but apply some more steps to upgrade the result.

2.1. Sobel

The Sobel operator is a simple edge detection algorithm using the 1st derivative of the intensity information [1][4]. It searches for peaks in the intensity gradient. The operator uses two 3x3 kernels convolved with the original image to produce a map of intensity gradient. The areas of highest gradient are where the intensity of the image changes rapidly over a few pixels, and are thus likely to represent edges.

On a fixed 3x3 neighborhood centered on any pixel position, the gradient in the y axis is :

$$\partial f(x,y)/\partial y=\left[f(i,j+1)-f(i,i)+f(i,i)-f(i,j-1)\right]/2=\left[f(i,j+1)-f(i,j-1)\right]/2 \quad (2)$$

If we ignore the ½ factor, the equivalent mask to be used under convolution is: $\begin{vmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & +1 & 0 \end{vmatrix}$

Some similar masks can be calculated along all the other axis: *x*, and the lines *x=y* and *x=-y*:

$$\begin{vmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{vmatrix}, \begin{vmatrix} -1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & +1 \end{vmatrix}, \begin{vmatrix} 0 & 0 & +1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{vmatrix}$$

If these masks are computed together, we obtain the Prewitt masks:

$$\begin{vmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{vmatrix}, \begin{vmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{vmatrix}$$

To improve the results, Sobel's idea is to give more importance to the current *x* and *y* axis, giving them a bigger weight. The two convolution kernels are used to establish two maps of gradients over horizontal and vertical axis.

$$Sob_{horizon}=\begin{vmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{vmatrix}\cdot I \text{ and } Sob_{vertical}=\begin{vmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{vmatrix}\cdot I \quad (3)$$

Then the edginess of a pixel can be computed as:

$$Edge(i,j)=\sqrt{Sob_{horizon}+Sob_{vertical}} \quad (4)$$

Using this information, being careful of keeping the signs, we can also calculate the direction of the tangent of the curb in this point: the gradient's direction is:

$$\theta=\arctan(Sob_{verticel}/Sob_{horizon}). \quad (5)$$

However, an error is generated whenever $Sob_{horizon}$ is equal to zero. So, there is a restriction set whenever this takes place. If the gradient in the *x*-direction ($Sob_{horizon}$) is equal to zero, then the edge direction has to be equal to 90 degrees or 0 degrees, depending on what the value of the gradient in the *y*-direction is equal to. If $Sob_{vertical}$ has a null value, the edge direction will equal 0 degrees.

2.2. Laplace:

These edge-detection operators are based upon the 2nd derivative of the intensity [4][1]. This is essentially the rate

Annexes

of change in intensity gradient and is best at detecting lines. As a line is a double edge, hence we will see an intensity gradient on one side of the line, followed immediately by the opposite gradient on the opposite site. Therefore, we can expect to see a very high change in intensity gradient where a line is present in the image. To find lines, we can search the results for zero-crossings of the change in gradient.

$$\partial^2 f(x,y)=\partial^2 f(x,y)/\partial x^2+\partial^2 f(x,y)/\partial y^2 \quad (6)$$

Recalling that the second derivative can be approximated by the difference of two derivatives at two adjacent points, the second derivatives on the y-axis become:

$$\partial^2 f(x,y)/\partial y^2=\left[(f(i,j+1)-f(i,j))/\alpha-(f(i,j)-f(i,j-1))/\alpha\right]/\alpha=\left[f(i,j+1)+2\cdot f(i,j)-f(i,j-1)\right]/\alpha^2 \quad (7)$$

Therefore, ignoring the α^2 member, the 3x3 Laplacian mask is the sum of the two second difference masks, which is:

$$\begin{vmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{vmatrix} + \begin{vmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{vmatrix} = \begin{vmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{vmatrix}$$

If one also consider the axis $x=y$ and $x=-y$, one can obtain the Laplacian mask: $\begin{vmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{vmatrix}$

The use of Laplacian mask can be extend to another use as enhance an image accentuating smooth edges without more calculation: Adding an identity mask, then the result image will contain the edge map added to the original image.

2.3. Mathematical computation: Canny

John Canny's team defined what an optimal edge detector needs to satisfy [5][8]:

- The edge detector should respond only to edges, and should find all of them; no edges should be missed.
- The distance between the edge pixels as found by the edge detector and the actual edge should be as small as possible.
- The edge detector should not identify multiple edge pixels where only a single edge exits.

The Canny edge detector separates the edge detection process in six major steps:

- The first step is to filter out any noise in the original image before trying to locate and detect any edges. Once a suitable mask has been calculated, the Gaussian smoothing can be performed using standard convolution methods. The larger the width of the Gaussian mask, the lower is the detector's sensitivity to noise. The localization error in the detected edges also increases slightly as the Gaussian width is increased.
- After smoothing the image and eliminating the noise, the next step is to find the edge strength by taking the gradient of the image. The Sobel operator performs a 2-D spatial gradient measurement on an image. Then, the approximate absolute gradient magnitude (edge strength) at each point can be found. The magnitude, or edge strength, of the gradient is then approximated using the formula:
- $|G|=|\mathrm{Sob}_{\mathrm{horizon}}|+|\mathrm{Sob}_{\mathrm{vertical}}|$ (8)
- As, we have seen in equation 5, it is also possible to compute the edge direction.
- Once the edge direction is known, the next step is to relate the edge direction to a direction that can be traced in an image. So if the pixels of a 5x5 image are aligned as follows:

x x x x x
x x x x x
x x a x x
x x x x x
x x x x x

Then, it can be seen by looking at pixel "a", there are only four possible directions when describing the surrounding pixels - 0 degrees (in the horizontal direction), 45 degrees (along the positive diagonal), 90 degrees (in the vertical direction), or 135 degrees (along the negative diagonal). So now the edge orientation has to be resolved into one of these four directions depending on which direction it is closest to (e.g. if the orientation angle is found to be 3 degrees, make it zero degrees). Think of this as taking a semicircle and dividing it into 5 regions.

Annexes

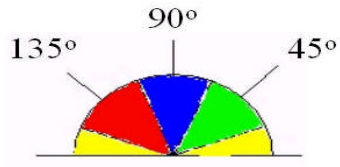


Figure 3. Canny edge analysis

Therefore, any edge direction falling within the yellow range (0 to 22.5 & 157.5 to 180 degrees) is set to 0 degrees. Any edge direction falling in the green range (22.5 to 67.5 degrees) is set to 45 degrees. Any edge direction falling in the blue range (67.5 to 112.5 degrees) is set to 90 degrees. And finally, any edge direction falling within the red range (112.5 to 157.5 degrees) is set to 135 degrees.

- After the edge directions are known, non-maximum suppression now has to be applied. Non-maximum suppression is used to trace along the edge in the edge direction and suppress any pixel value (sets it equal to 0) that is not considered to be an edge. This will give a thin line in the output image.
- Finally, hysteresis is used as a means of eliminating streaking. Streaking is the breaking up of an edge contour caused by the operator output fluctuating above and below the threshold. If a single threshold, T1 is applied to an image, and an edge has an average strength equal to T1, then due to noise, there will be instances where the edge dips below the threshold. Equally it will also extend above the threshold making an edge look like a dashed line. To avoid this, hysteresis uses 2 thresholds, a high and a low. Any pixel in the image that has a value greater than T1 is presumed to be an edge pixel, and is marked as such. Then, any pixels that are connected to this edge pixel and that have a value greater than T2 are also selected as edge pixels. Following an edge, a gradient of T2 is needed to start, and the stop occurs when a gradient below T1 is reached.

2.4. Neuronal Computation: Bezdek

The key feature of the Bezdek approach is the use of a training set based on a square window in a binary image [6]. Considering a 3x3 window, Bezdek used the Sobel operator to "score" all possible binary pixel populations. For a 3x3 window, there are 2^9 different possible window sets. These binary windows were used by Bezdek in training a neural net. The edginess, E, is considered as a fuzzy measured of membership of the set of edge points, and the neural net, with one or two hidden layers, is trained to give the appropriate value of E for each window.

What is striking is that Bezdek's approach does in fact succeed in producing an excellent edge detector, that agrees with the Sobel on binary images, but has better behavior on low contrast gray-scale images.

However Bezdek's approach is limited in a very basic way. It cannot be extended to larger window sizes. The argument is simple: For a 3x3 there are only some 512 = 2^9 prototypes, and training is readily achieved in a matter of minutes. For a 4x4 NN operator, there are 2^{16} binary prototypes, and for 5x5 =2^25. Assuming training times are linear in the number of inputs we compute as follows: Training times for the 3x3 neural network based operators take of the order of minutes, whereas for 4x4 the corresponding time would be of order of 2^7 = 128 minutes. But for a 5x5 operator training times would take of the order of 2^16 = 64K minutes = 45 days. In fact, the linearity is not reasonable, and combinatorial explosion would be far worse.

2.5. Fuzzy computation: CFED

Fuzzy logic can also be applied on image edge detection. This fuzzy classifier first classifies image pixels as different kinds of edge pixels or non-edge pixels [7]. It then applies competitive rules on them according to their classification so that not all the edge pixels are output as edges, which thins the edges. This fuzzy classifier is called the *competitive fuzzy edge detector*.

The competitive fuzzy classifier is made up of the fuzzy classifier part and the rule-based competition part. In the fuzzy classifier, there are six classes: background, vertical edge, horizontal edge, two kinds of diagonal edge classes and a speckle edge classes. Either a Gaussian or an extended Epanechnikov is used for each class as its fuzzy set membership function. The fuzzy truth for a pixel to be a member of a class is given by the evaluation of its function on the vector of features for that pixel.

Each pixel determines a vector of differences (d(1), d(2), d(3), d(4)) in the four directions across that pixel in its 3x3 window. The vertical, horizontal, and two diagonal differences are

$$d(1) = |f(a_{1,2}) - f(a_{2,2})| + |f(a_{3,2}) - f(a_{2,2})|, \ldots, d(4) = |f(a_{1,1}) - f(a_{2,2})| + |f(a_{3,3}) - f(a_{2,2})| \quad (9)$$

Annexes

A feature vector is evaluated by putting it through each fuzzy set membership function that is centered on a prototypical difference vector for that class. Thus each pixel has a fuzzy truth value for each class and the maximum fuzzy truth determines the class of that pixel.

In the competition, neighboring edge pixels in the same direction compete with each other in directional edge strength. To be a black edge pixel in the output image, a classified directional edge pixel must have larger edge strength on its edge direction in comparison to its neighbors on that direction. If an edge pixel does not win the competition, then it will be output as white (background). The result is a black line drawing on a white background.

Speckle edge pixels are mapped to black directly without competition. This may introduce isolated single/double-pixel speckles. A despeckler operates on the image after all other processes have been completed so as to remove these speckles.

Fuzzy Classification
Step1: set parameters for the fuzzy set membership functions; open the image file;
Step2: for each pixel in the image compute gray level change magnitudes in the 4 different directions and construct the pixel feature vector from those magnitudes
for each class compute the fuzzy truth of the feature vector
determine maximum fuzzy truth and record pixel class for that pixel

Edge Strength Competition
Step1: for each pixel in the image
if (edge class) then apply competitive rule and record pixel value
if (background class) then write black pixel
if (speckle edge class) then write white pixel

Despeckling
Step1: for each pixel in the image
if (pixel is isolated single/double speckle) then change to black.

2.6. Description

Several neuro fuzzy systems have been described in the literature and are applied to different fields. This edge detector is based on the analysis of the gradient for every point of the image on a small and a larger window. The basic idea is to classify the pixels according to these two data. For instance, if the gradient is small on both windows, then the pixel belongs to the background but if the gradient on the small window is high and the gradient on the large window is small, then some noise could have spoilt the information for this pixel.

So, each pixel gradient data are evaluated by a fuzzy classifier built with a set of three classes, a inference engine and a SUGENO defuzzified system. The edge detector provides a correction on the original pixel value. This value is then normalized and a filter is applied to give a final binary information about the edge property value. The overall system scheme is sketched in Figure 4:

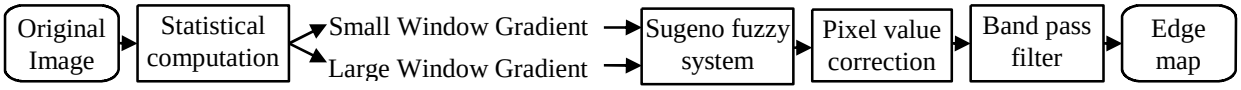


Figure 4. Neuro Fuzzy Edge detector scheme

We characterize the image properties through the properties of two windows: a Small Window (denoted SW) and a Large Window (LW). We compare *f(i,j)* with the statistic properties of these windows, that is the average luminosity over these windows.

$$\bar{S}_{i,j}^{SW} = \frac{1}{N_{SW}} \sum_{(i,j)\in SW} f(i,j) \text{ et } \bar{S}_{i,j}^{LW} = \frac{1}{N_{LW}} \sum_{(i,j)\in LW} f(i,j) \quad (10)$$

Then, we compute the average gradient, that is the difference between the local luminosity of each point of the window and the average as computed in eq. 10.

Annexes

$$d(S_{i,j},\overline{S}_{i,j}^{SW}),\ d(S_{i,j},\overline{S}_{i,j}^{LW}) \qquad (11)$$

We translate these distances in a membership to a fuzzy rule. We define three fuzzy classes Zero, Small and Large using triangular functions as input membership functions.

The procedure produces a correction to the luminosity of the image points, according to the values of the above distances, i. e. according to the proximity to the points of their nearby and distant environment. The correction is computed as in equation (12):

$$S_{i,j}^{l} \leftarrow S_{i,j}+\Delta \qquad (12)$$

An inference engine first evaluates this correction. It is based on rules as:

If $d(S_{i,j},\overline{S}_{i,j}^{SW})$ *is Large, Et* $d(S_{i,j},\overline{S}_{i,j}^{LW})$ *is Large, Then correction* Δ*' is TrèsLarge* (13)

This inference engine can be described in the Table 2. The values are only for exemplification purposes in this table:

		$d(S_{i,j},\overline{S}_{i,j}^{LW})$		
Δ'		S	Z	L
$d(S_{i,j},\overline{S}_{i,j}^{SW})$	S	+L	+S	+Z
	Z	+S	+Z	-S
	L	+Z	-S	-L

Table 1. Fuzzy values of luminosity correction

A SUGENO system is used to Deffuzzify this value. For each combination of the nine entries of the inference engine table, we have a rule$_{k:}$

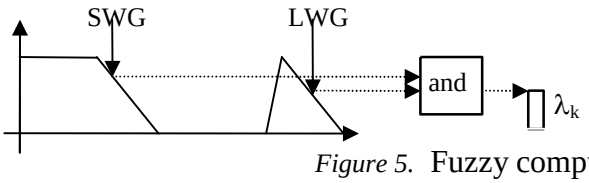


Figure 5. Fuzzy computation scheme

with $x_k=F(SWG,LWG)=p_k\cdot SWG+q_k\cdot LWG+n_k$ and $\Delta=\frac{\sum_{i=1}^{k} x_k\cdot\lambda_k}{\sum_{i=1}^{k}\lambda_k}$ (14)

The filter is set with seven thresholds for the input membership functions; the parameters for each rule: *p, q, r*; and the thresholds of the output filter. To simplify the computation, we arbitrary set each p_k and q_k with the null value. The r_k value is issue from the conversion of the fuzzy value from table 2 to a crisp value.

Algorithm (in pseudo code)

Procedure FUZZY_FILTER_1
1.for i=0 to N
2. for j=0 to M
3. apply procedure CORRECTION
4. $S_{i,j} \leftarrow S_{i,j}+\Delta$
END FUZZY_FILTER_1

Procedure CORRECTION
1.Compute $\overline{S}_{i,j}^{LW}$

Annexes

2.Compute $\overline{S}_{i,j}^{SW}$

3.Compute $d(S_{i,j},\overline{S}_{i,j}^{LW})$

4.Compute $d(S_{i,j},\overline{S}_{i,j}^{SW})$

5.Compute Δ' by the FUZZY_CORRECTION procedure.

END CORRECTION

5. Discussion and Conclusions

The Sobel and Laplace operators give generally good results [8][9] even if they miss precision and continuity in the layout of edges. The Sobel operator is not isotropic, i.e. its sensitivity depends on edge orientation. This property is used to produce an information about the edge direction. It is very effective to detect the vertical and horizontal edges but can miss some corners, round or diagonal edges. It is also sensitive to noise.

The disadvantages of Laplacian operators are that they give no direction for the edge and they amplify noise polluting the image with some speckle points. Derivatives and differences are sensitive to noise and second differences derivative is even more sensitive: small errors in the input data can cause large errors in the output data. Noise should be reduced before edge detection is performed. Indeed, as Laplacian operator is based on zero crossing, it produces a one-pixel boundary and situates the edge in the place of the larger luminosity variation. In case the original edge gradient variation is not uniform, the Laplacian operator could produce double edges. Using a Gaussian filter on the image both reduces the sensibility to noise and to changing edge gradient variation.

When a Gaussian filter is applied on the image before detecting edge, it reduces the sensibility to the noise, but some information is also lost: the fine details can become non-significant and the location of the edges can be misinterpreted. The quality of edge detection is reduced if using such a filter on a clear image with sharp edges.

Usually, another step is added after the edge detection: Once the derivative has been calculated, one is used to thresholding the data to determine where the results suggest an edge is present. The lower the threshold is, the more lines will be detected, and the results will become increasingly suspected of noise and will also pick out irrelevant features from the image. Conversely, a high threshold may miss subtle lines or sections of lines.

The comparison between the algorithms should be made regarding to the class of complexity of these algorithms. One can determine two classes of algorithms: the simplest one to which Laplace and Sobel belong, and the more complex one to which CFED and Canny belong. The Laplace and Sobel are respectively simple or double convolutions. The neuronal approach cannot be compared in term of performances because it works differently: the learning phase is long [10], but the computation phase is very fast. The results have to be objectively measured.

The canny operator is efficient in detecting most of the edges but it can miss to connect some Y-junctions. It produces a fine edge and is not very sensitive to noise. According to [7], the CFED method is much quicker than the Canny method. The canny filter, as the Sobel and Laplace operator when they are used with a post-treatment threshold, must be set by an operator to adapt the result of the edge detection to the quality of the source image and to the needs of the application.

Until now, a first implementation permitted to check the viability of the principle of this new neuro fuzzy edge detector. To go further, we compare objectively the edge detection performed by several algorithms. So, we measure quantitatively the quality of the obtained results. More explanations on the measurement method will be presented on ECIT 2004 [11]. The evaluation of the capabilities of this new algorithm will be presented on MIC 2004 [12].

References

[1]. Sing-Tze Bow, Pattern Recognition and Image Pre-processing, second Edition, 698pp., ISBN 0-8247-0659-5, Marcel Decker Inc., New York, 2002

[2]. http://encyclopedia.thefreedictionary.com/Computer%20vision distributed under GNU Free Documentation License

[3]. Fabrice Belet, Une approche incrémentale à base de processus coopératifs et adaptatifs pour la segmentation des images en niveaux de gris, rapport de thèse / Juin 1998, INPG Grenoble

[4]. N. Efford, Digital Image Processing, Addison-Wesley, Reading, MA, 2000, pp.164–173.

[5]. J. F. Canny.: A computational approach to edge detection. IEEE Trans. Pattern Analysis and Machine Intelligence, pages 679-698, 1986

[6]. J.C. Bezdek and D. Kerr, Training Edge Detecting Neural networks with Model-Based Examples, Proc 3rd international Conference on Fuzzy Systems, FUZZ-IEEE'94, Orlando, Florida, USA. pp 894-901, June 26 - 29, 1994

Annexes

[7]. Lily R. Liang, Carl G. Looney. International Journal of Applied Soft Computing. 3(2): 123-137 (2003)
[8]. Alberto Martin and Sabri Tosunoglu, Image Processing Techniques for Machine Vision, FIU, Florida Conference on the Recent Advances in Robotics - 2000
[9]. Salotti, M., Bellet, F., Garbay, C.,: Evaluation of Edge Detectors: Critics and Proposal. Workshop on Performance Characteristics of Vision Algorithms, Cambridge - 1996
[10]. Harvey A. Cohen Craig McKinnon , and J. You, Neural-Fuzzy Feature Detectors, DICTA-97, Auckland, N.Z., Dec 10-12, pp 479-484.
[11]. Frédéric Mayer, Horia N. Teodorescu, Pseudo Statistic Edge Detector using Neuro Fuzzy Computation: Error Distance Evaluation. Proc. ECIT - Iasi (2004) - to be published
[12]. Frédéric Mayer, Horia N. Teodorescu, Neuro-Fuzzy Edge Detector: Experimental Results. Proc. MIC - Roma (2004) - to be published

Annexes

Annexes

Annexe 3: Article ECIT 2004

Pseudo Statistic Edge Detector using Neuro Fuzzy Computation: Error Distance Evaluation

Frédéric Mayer[1], Horia-Nicolai Teodorescu[2]

1-Conservatoire National des arts et métiers - Paris and TIMC - SIC team - Grenoble France and Institute for Theoretical Informatics of the Romanian Academy - fmayer@etc.tuiasi.ro
2-Technical University of Iasi and Institute for Theoretical Informatics of the Romanian Academy - hteodor@etc.tuiasi.ro

Abstract: During our investigations to improve a new type of edge detector based on fuzzy logic, we had to confront with several questions: how to objectively estimate the correctness of edge detection? Which distance should be used to measure it? Does it exist a true reference? Several algorithms are investigated here to measure the distance between images, starting from the simplest one - the Euclidean distance. We finally defined a statistical measure used in combination with a fuzzy approximation of the distance between an image and a model.

Keywords: edge detection, low level processing, image processing, fuzzy logic

1. Introduction

The image pre-processing is a wide area of research with multiple applications. Edge detection is a step of the process that permits a computer to understand the image content. Moreover, edge detection evidences elements that help the humans to make decisions. In computer vision applications, the image pre-processing stage aims to emphasize elements in

Annexes

the picture that are essential in specific fields of applications. The stages of this information process are numerous and one of them is classification [1]. The latest trend to solve this task combines two technical areas: edge detection and segmentation [2]. The edge detection treatment is usually shared in two phases: a low level and a high level. In a classical approach, with gray level images, the low level edge detection is based on a statistical computation of the luminosity, extracting gradients and leveling the higher variations. The well-known Laplace and Sobel methods are both based on this idea [3]. Another known method, the Canny edge detector tries to counter a major weaknesses of Sobel and Laplace detector, namely their noise sensibility [4]. The Canny detector is more sophisticated and acts in several steps: it filters noise before processing, then post-processes the result to thin the drawing of the edge map. The fuzzy pseudo statistical edge detector is also based on such a scheme and its principle will be presented in section 2.

In this paper, we present the methodology we followed to evaluate this new detector versus the other ones and, also, as this detector has many possible parameters conjunctions, we had to learn the best ones, comparing the outputs to an ideal result. In the third section, we present how we determined what is the best edge map for a given image. In the fourth one, we justify why the Euclidean distance measure cannot be used in this case. In the next sections, several comparison tools are evaluated and we study the conditions in which they can be applied to our problem. We will present some results based on an angiographical image.

2. System Description

This system is described in more details in [5]. This edge detector to provide some statistics properties computes every point of the image. These ones are evaluated by a fuzzy system: based on a set of three rules and a fuzzy inference engine type SUGENO, the fuzzy system provides a correction on the original pixel value. This value is then normalized and a filter is applied to give a final binary information about the edge property value of every pixel. This system scheme is sketched in figure 1:

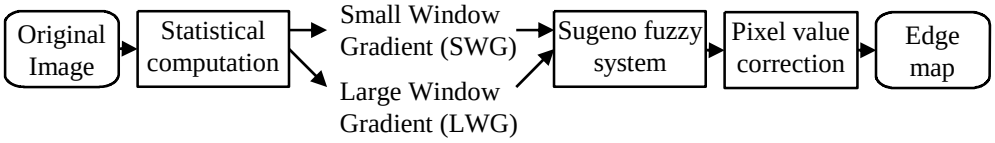
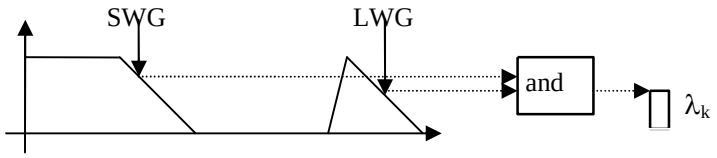


Fig. 1. Fuzzy Pseudo Statistical Edge detector scheme

The SUGENO fuzzy system is based on *k* inference rules. For each rule, we have:
$rule_{k:}$



and $\Delta = \frac{\sum_{i=1}^{k} x_k \cdot \lambda_k}{\sum_{i=1}^{k} \lambda_k}$

Annexes

Fig. 2. Fuzzy Computation scheme

where $x_k = F(SWG, LWG) = p_k \cdot SWG + q_k \cdot LWG + n_k$

The edge detector depends on many settings: input membership functions, *p, q, r* parameters for each rule, and the thresholds of the output filter. To simplify the computation, we arbitrary set each p_k and q_k with the null value.

To find the best settings, a system has been implemented that computes the different possibilities and compares them. In a first step, the possibilities are calculated in two different ways: the system tries successively some sets of arbitrary chosen parameter conjunctions, or it looks for the best choice using a genetic algorithm. On the second step, the system compares the results confronting an edge map to a reference one and makes an evaluation of the differences using a comparison tool. To do so, we need the reference edge map and the evaluation tool that are presented in the next sections.

3. Reference Edge Map

We worked with several medical images extracted from a angiographical examination. We asked some bio-medical engineers to draw reference edge maps. They did it manually on a layer over the original image (see figure 3). Even if they are not experts, we have considered they have enough knowledge to make a good interpretation from a radiographic image.

The obtained maps show results that look close one to each other, but comparing them in the details, we noticed some important differences.

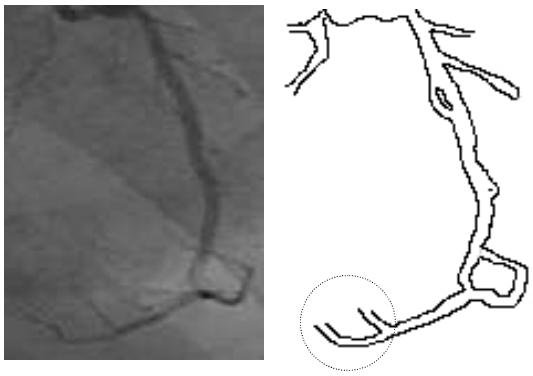


Fig. 3. Edge map example

Annexes

For instance, if we focus on the circled zone, we get different versions, as in figure 4:

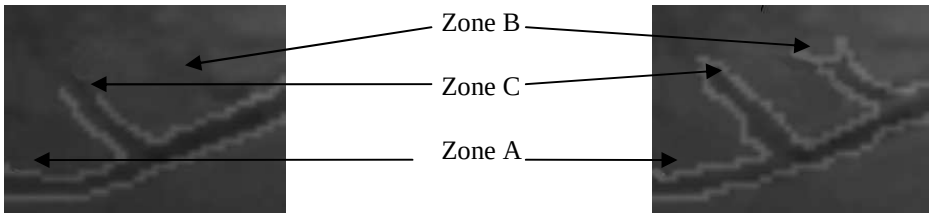


Fig. 4. Edge map details

* If the edge seems to be the same, at the pixel scale, there is a important difference (zone A).

* Some details are clearly understood in different ways (zones B and C).

Concluding, after evaluation of these different results, it seems that it doesn't exist an ideal reference edge map: the "true" edge map depends very much on the experts' interpretation. Nevertheless, it is possible to use the manual evaluations in two different ways:

* Establish a subjective choice on a map that defines the edges. Then, we consider the area in the neighborhood of an edge line with a smaller error.

* Establish an average edge map from all the drawn maps. This edge map will then give a pseudo probability for a pixel to be on an edge. Then, this probability can be used to compute an error.

4. The comparison tool

The comparison tool will be used to evaluate the best settings of the edge detector parameters. We can see this process from different perspectives: Firstly, as a simple measure of the distance between two images with the Euclidean distance. It is frequently used to measure the noise between an image and a reference. Secondly, if the edge detection is seen as a classification process then we can try to count the number or the proportion of points that are wrongly classified. We will subsequently determine, which is the best choice for our application and the rational for optimization.

Notations

- The number of edge points of the image will be defined as $nbEdgP$.
- The number of points that are not part of the edge will be noted as $\overline{nbEdgP}$.
- We consider that an image I has dimensions M and N.
- Then: $\overline{nbEdgP}=M\cdot N-nbEdgP$.

Euclidean distance definition

If I and I^r are two gray level images and $p_{i,j}$ is the gray level at the point (i,j), then:

$$d^2(I,I^r)=\sum_{i=1}^{M}\sum_{j=1}^{N}\left(p_{i,j}-p_{i,j}^r\right)^2$$

Then, the average distance between the points of the two images is defined as:

$$d^2(I,I^r)=\frac{1}{N\cdot M}\cdot\sum_{i=1}^{M}\sum_{j=1}^{N}\left(p_{i,j}-p_{i,j}^r\right)^2$$

If the image is coded on P gray levels, then we define the averaged Euclidean distance:

Annexes

$$d_p^2(I,I_r)=\frac{1}{N\cdot M\cdot P}\cdot\sum_{i=1}^{M}\sum_{j=1}^{N}\left(p_{i,j}-p_{i,j}^r\right)^2$$

In our case, the blood vessel edges are expressed with a binary information: arbitrarily, we chose that the white points show the edges. So, we have *P=1* and then:

$$d_p^2(I,I^r)=d^2(I,I^r)$$

Comparing the two images, we determine:

* the points wrongly marked as edge: *nb1*;
* the points marked as background when they are edge: *nb2*.

Then, we obtain the relation:

$$d^2=\frac{1}{M\cdot N}\cdot(nb1+nb2)$$

If we compare the reference edge image with a black one, then $nb1$ is null because the black image does not contain any point detected as an edge. Then, the computed distance is directly linked to the number of edge points in the edge reference map:

$$d^2=\frac{1}{M\cdot N}\cdot nb2$$

The comparison tool should is used to learn the best parameters of the edge detector. We investigated theoretically on the possible values to check if this distance would be able to provide improved results. We have:

$$nb1\in[0,\mathrm{NbEdgP}] \text{ and } nb2\in[0,\overline{\mathrm{NbEdgP}}]$$
$$\text{so } nb1+nb2\in[0,M\cdot N] \text{ and } d^2\in[0,1]$$

The figure 5 represents the plot of the error values depending on the $nb1$ and $nb2$ parameters:

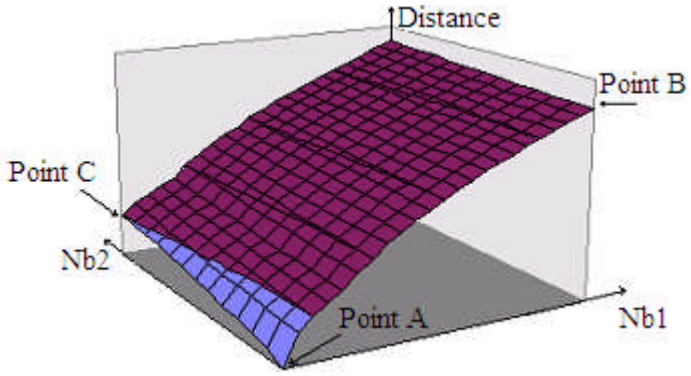


Fig. 5. Error function graph

There are three outstanding points in this graph:

- The point A shows the error when $nb1$ and $nb2$ are null; this is the perfect edge detection.
- The point B shows the error when $nb1$ is null; this is a white image where all points are detected as edges.
- The point C shows the error when $nb2$ is null; this is a black image where no edge points are detected.

The darkest surface represents all the error cases for which the error is smaller than the error in point B and bigger than the error in point C (the black image). We want to implement an algorithm to learn the nearest solution to that in point A. Fix this, the algorithm has to find at least one solution with an error smaller than the error in point C not to reach the point C as best solution. But the light surface is rather small and an experiment on Sobel's filter showed that its best results are above this limit. Even with a good filter,

Annexes

chances are nearly null to reach such a solution. Therefore, the use of this measure is inadequate in this problem.

Weighted error proposal

The parameters are the same as above. Intuitively, looking to the graphics, we can try to "lift" the point C value to be equal to the point B value to increase the light surface. We can obtain this if we weight the $nb1$ parameter with an integer R greater than one. Then relationship between the parameters becomes:

$$d_w^2 = \frac{1}{M \cdot N} \cdot (nb1/R + nb2)$$

The R parameter has an major influence one the lightly-colored surface variation, as showed in figure 6:

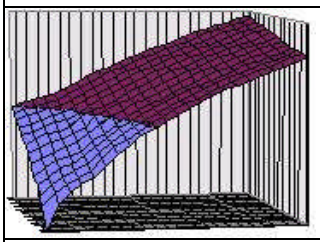
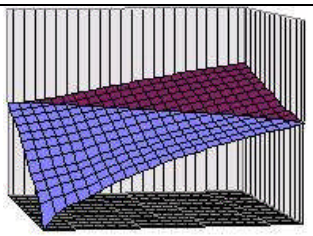
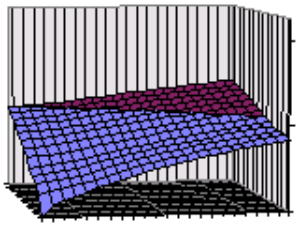
$R < \overline{nbEdgP}/nbEdgP$ (13)	$R = \overline{nbEdgP}/nbEdgP$ (20)	$R > \overline{nbEdgP}/nbEdgP$ (28)
		
The lightly-colored surface is rather small. The chances to encounter the already known situation are important.	The error on points B and C are equal. The lightly-colored surface is important.	The lightly-colored surface is bigger but it includes the point B. The chances are great to find the point B as the best point during the learning.

Fig. 6. Weighted error graph

In a learning process, we can not guess the number of edge points at the beginning of the process. We can only make estimations. But the results will depends on this intrinsic characteristics of the image. Therefore, this weighted error might not be the best answer to our problem.

Statistical error proposal

The same parameters can be used in a more statistical view. The parameters $nb1$ and $nb2$ define two classes of errors. We can consider the distance as a global error for the image: this error is computed as the average of the error in each class:

$$Dstat = \frac{1}{2} \cdot \left(\frac{nb1}{\overline{NbEdgP}} + \frac{nb2}{NbEdgP} \right)$$

If we consider $R = \frac{M \cdot N}{NbEdgP}$

then

$$Dstat = \frac{1}{2} \cdot \left(\frac{nb_1}{(M \cdot N - \frac{M \cdot N}{R})} + \frac{nb_2 \cdot R}{M \cdot N} \right) = \frac{1}{2} \cdot \frac{1}{M \cdot N} \cdot \left(\frac{nb_1}{(1 - \frac{1}{R})} + nb_2 \cdot R \right)$$

The figure 7 presents the influence of the variation of the parameter R on the error computation.

Annexes

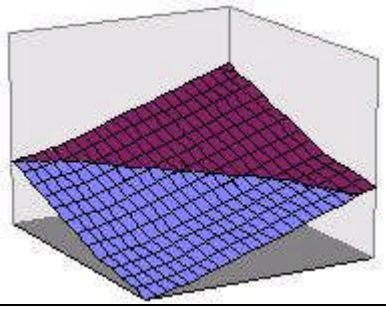
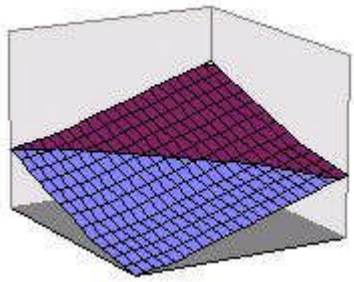
R < $\overline{\text{nbEdgP}}$/nbEdgP (13)		R > $\overline{\text{nbEdgP}}$/nbEdgP (28)
		

Fig. 7. Statistical error graph

As the variation of *R* has nearly no influence on the limit of the surface, this error can be considered as a good indicator for our experimentation. As long as $R>5$, the variation of the surface can be considered as acceptable. There is no upper limit, but we remark that if R tends to a big number then the first term of the equation becomes insignificant in front of the second term. Perhaps, R should be set at a higher value in the case of very high number of edge points in the images.

We can also note that this distance measure is dependent neither on the image size, nor on the edge point count number. Therefore, we also use it to compare the quality of the edge detection between two different images.-

Fuzzy distance proposal

We analyzed two distances both deterministic, and both based on pixel comparison: the value for each pixel on the computed edge map was compared with the value for the pixel with the same co-ordinates on the reference map. If the detected edge doesn't match exactly the reference edge map then the detection is considered as erroneous. As we proposed in the edge map definition section, it can be profitable to consider an area around the ideal edge where the error is less weighted than at a larger distance.

We define two areas around a pixel point:

* A nearby space where the error is considered as null.
* A small distance area where the error is proportional to the distance to the closest edge pixel.

The distance function graph is showed in figure 8:

Fig. 8. Distance – error function

This calculation is to be used for the edge points but also for the non-edges points. Then, we can obtain nb_1^{Fuz} and nb_2^{Fuz} where $nb_1^{Fuz} \in [0, nb1[$ and $nb_2^{Fuz} \in [0, nb2[$

As nb_1^{Fuz} and nb_2^{Fuz} have some comparable values to nb_1 and nb_2, we can use them to compute the statistical error with comparable results.

5. Applications

The distance evaluation is to be used in:

Annexes

- the learning process of the edge filter parameters as a feedback function;
- the comparison between different algorithms results.

The Figure 9 presents edge maps images obtained with the filter from the image in figure 1. The lower row contains the distance evaluation with the comparison tool.

Edge maps			
Distance evaluation	0.077654	0.028419	0.02113

Fig. 9. Distance evaluation examples

The statistical error is able to provide a quite reliable estimation. However, we notice that the edge map measured as the best one is faulty: some edges appear as a double line. If we consider our detector as the low level of a more complex architecture, then this error should be managed at a higher level.

6. Perspectives

The corrected distance takes in account the lack of precision of the estimation of the reference edge map. But this lack of precision is equally evaluated on all the surface of the image. It could be possible to use the specialists' edge drawings to establish a likely edge map. The probability information contained in this map could be used in place of the described fuzzy distance function. We would then consider the different points of view of the specialists and the lack of imprecision of the eye perception about on edge at one point.

7. Conclusions

By this paper we proposed and analyzed a tool to evaluate the distance between two edge maps. We demonstrated why the Euclidean error is not appropriate in this problem. The statistical error associated to the fuzzy distance provides a realistic measure to compare two computed edges and sort the best one. This statistical error has the advantage to be independent of the intrinsic characteristics of the image and the edge map, at least if the proportion of edge points compared to non-edge points is greater than five.

With this distance measure, we were able to learn the best parameters of our fuzzy edge detector and secondly to compare the results obtained with other edge detection algorithms.

References

[1]. Sing-Tze Bow, Pattern Recognition and Image Pre-processing, second Edition, 698pp., ISBN 0-8247-0659-5, Marcel Decker Inc., New York, 2002

[2]. Fabrice Belet, Une approche incrémentale à base de processus coopératifs et adaptatifs pour la segmentation des images en niveaux de gris, rapport de thèse / Juin 1998, INPG Grenoble

[3]. N. Efford, Digital Image Processing, Addison-Wesley, Reading, MA, 2000, pp.164–173.

[4]. Alberto Martin and Sabri Tosunoglu, Image Processing Techniques for Machine Vision, Proc. of Florida Conference on the Recent Advances in Robotics, from http://www.eng.fau.edu/conf/fcrar2000/ , FIU 2000

[5]. Horia N. Teodorescu, Frédéric Mayer, On a Pseudo Statistic Edge Detector using Neuro Fuzzy Computation. Proc. EEIA symp. - Pitesti (2004)

Annexes

Annexe 4: Tableau des résultats de la recherche par algorithme génétique

Low Zero	high Zero	low Small	medium small	high small	low large	high large	defuz zero	defuz small	defuz large	filter low	grade / Beta	avg / alpha	error
0,93	3,86	2,93	9,21	15,8	12,58	17,3	-5	-10	-86	77	13,05	6,29	0,01321
3,83	8,29	7,28	10,93	13,34	12,74	12,78	-5	-33	-52	59	7,44	6,83	0,01385
3,84	12,86	11,6	18,13	21,96	20,04	21,23	-3	-13	-108	71	26,27	6,38	0,01396
3,83	8,29	7,28	10,93	13,34	12,74	12,78	-5	-33	-52	59	7,44	6,83	0,01399
3,56	5,28	3,98	5,45	12,44	10,5	17,82	-4	-34	-44	90	24,92	6,67	0,01408
0,97	6,01	3,84	7,79	17,19	11,07	13,57	-2	-18	-60	87	12,74	5,95	0,01416
4,87	9,6	4,95	11,42	15,12	12,71	19,69	-4	-32	-106	62	12,42	7,17	0,01447
5,72	11,11	10,21	11,27	19,41	14,36	18,83	-4	-6	-101	87	7,21	6,02	0,01474
3,54	6,1	5,64	6,21	16,42	10,95	11,18	-4	-29	-123	122	5,11	5,77	0,01481
-0,16	5,8	3,47	7,49	11,09	8,37	17,65	-6	-9	-44	69	17,45	6,51	0,01488
4,61	9,23	7,66	12,93	16,12	13,1	15,76	-6	-12	-89	127	6,92	7,26	0,01504
1,89	14,36	4,47	14,71	15,35	15,21	21,42	-2	-14	-49	74	5,12	6,97	0,0152
2,36	5,93	5,63	8,89	20,36	11,71	19,76	-5	-8	-94	51	7,57	6,85	0,01526
2,17	5,04	4,81	10,36	11,88	11,78	14,11	-6	-37	-89	94	7,88	6,11	0,01533
4,09	9,52	5,95	10,67	16,29	10,74	15,94	-4	-32	-77	63	7,24	5,75	0,01548
4,32	5,7	5,49	7,28	15,47	8,51	17	-4	-22	-114	62	14,68	6,56	0,01555
1,52	7,44	7,09	7,73	9,47	9,12	10,15	-3	-29	-62	78	24,6	6,28	0,01558
4,31	9,53	5,74	10,76	14,47	11,2	11,83	-6	-36	-112	62	14,02	5,63	0,01561
5,83	7,61	6,61	7,69	14,39	9,54	14,15	-4	-32	-123	47	9,43	5,61	0,0157
3,12	5,88	3,95	11,15	21,13	16,28	18,37	-3	-32	-95	65	13,46	6,87	0,0159
5,43	14,88	12,09	15,23	20,01	16,68	19,63	-4	-33	-112	83	6,56	5,92	0,01591
5,82	6,45	6,42	9,38	17,18	12,71	16,84	-2	-4	-58	51	8,55	6,98	0,01593
2,59	7,82	6,05	11,9	18,8	12,64	15,16	-6	-35	-69	67	8,93	7,03	0,01601
4,78	5,59	5,41	10,9	15,72	14,16	14,82	-3	-33	-96	50	8,98	6,69	0,01617
4,02	6,2	5,26	8,87	20,66	12,37	23,36	-4	-7	-69	51	5,94	7,48	0,0162
6,36	9,18	8,63	9,45	11,9	9,92	10,39	-4	-25	-93	42	5,69	6,15	0,01639
5,28	6,14	5,59	6,47	16,78	9,97	12,53	-6	-10	-88	57	8,57	6,45	0,01645
4,03	5,52	5,27	9,13	13,55	9,58	19,85	-4	-13	-33	59	5,93	6,71	0,01649
5,8	8,57	6,19	8,97	13,15	12,72	15,41	-1	-25	-117	49	10,47	6,39	0,01677

Annexes

Annexes

Low Zero	high Zero	low Small	medium small	high small	low large	high large	defuz zero	defuz small	defuz large	filter low	grade / Beta	avg / alpha	error
5,83	7,27	5,89	9,44	18,85	12,97	23,36	-3	-24	-102	49	8,62	7,2	0,01704
3,07	6,81	5,22	6,95	14,74	9,97	10,34	-5	-10	-100	55	21,74	5,8	0,01723
4,12	6,39	5,64	8,75	13,02	9,62	14,06	-6	-31	-90	40	5,37	7,14	0,01728
5,55	6,89	5,84	9,94	21,78	13,51	23,39	-6	-36	-116	55	12,78	7,04	0,0175
4,92	7,64	6,3	10,34	13,75	13,12	17,47	-1	-31	-48	57	7,07	6,22	0,01751
5,76	8,46	6,22	9,26	13,44	12,71	15,56	-5	-13	-105	53	10,45	6,4	0,01758
3,23	11,17	5,11	12,07	15,58	12,38	16,82	-1	-31	-126	54	11,79	5,53	0,0176
5	5,8	5,15	12,99	18,65	13,79	15,01	-6	-17	-112	39	6,63	5,89	0,01777
5,42	6,71	5,78	10,44	20,32	12,48	21,38	-6	-36	-116	55	12,77	6,23	0,01789
3,62	5,72	5,14	8,25	12,52	9,12	13,56	-6	-31	-90	40	5,37	7,14	0,01813
5,95	12,23	6,05	14,33	19,18	18,42	19,76	0	-30	-114	53	9,44	6,84	0,01817
4,16	9,32	8,73	10,37	18,02	17,46	18,95	-4	-29	-114	52	10,7	6,55	0,01834
2,37	5	3,41	9,27	17,97	17,28	17,46	-1	-20	-106	123	5,15	6,17	0,01857
4,11	6,08	5,57	8,85	16,23	9,1	14,65	-1	-22	-73	54	16,04	6,51	0,01888
4,17	7,35	6,54	8,22	11,26	10,91	13,69	-6	-37	-113	62	7,82	5,63	0,01906
2,42	5,75	5,04	7,05	15,41	7,63	7,67	0	-1	-55	54	10,65	6,17	0,01907
0,72	5,77	4,26	7,58	15,79	11,65	11,93	-2	-25	-94	45	7,84	5,86	0,01974
3,99	5,46	4,8	8,64	18,3	13,45	14,67	-6	-17	-96	39	6,63	5,89	0,01978

MEMOIRE D'INGENIEUR C.N.A.M. en INFORMATIQUE

Etude d’un détecteur de contours basé sur la logique floue

Frédéric Mayer **Grenoble, le 1er Juillet 2010**

Résumé

Cette étude évalue la possibilité d'utiliser une architecture basée sur la logique floue pour filtrer les contours présents sur une image en niveau de gris. Nous avons implémenté un prototype pour vérifier que cette méthode peut donner des résultats intéressants. Nous étudions une méthode de quantification de la validité de ces résultats pour pouvoir les comparer avec ceux obtenus par d'autres algorithmes plus classiques et, d'autre part, améliorer les résultats du filtre en effectuant des réglages fins sur l’ensemble des paramètres qui le caractérisent. Dans ce but, une méthode basée sur l’algorithmie génétique est utilisée. Les résultats ont été présentés lors de plusieurs congrès en Roumanie.

Mots-clés : détection de contours, filtre de Sobel, filtre de Canny, filtre de Laplace, logique floue, algorithme génétique

---

Abstract

This aim of this study is to evaluate a new grey scaled images edge detector based on fuzzy logic architecture. We first implemented a prototype to check if it could be an interesting way of filtering edges. We studied several methods to evaluate the results and compare them with standard filter performance (Sobel, Laplace, Canny). We improved the performance of the pseudo-statistical filter by using a genetic algorithm. Several articles on this subject were presented in Romania.

Keywords : edge detection, Sobel filter, Cannyfilter , Laplace filter, fuzzy logic, genetic algorithm.