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

Ecole Nationale Polytechnique
Département d'Electronique

Mémoire de fin d'études
En vue de l'obtention du diplôme d'ingénieur d'état en Electronique

Thème

Etude et implémentation de détecteurs de contours

Proposé et Dirigé par :

Dr L. HAMAMI

Réalisé par:

Melle CHAOUCHI Chahinez

Promotion : Septembre 2004

ملخص :

يهدف هذا العمل الى دراسة وزرع كواشف الهوامش (أو الحدود). نتعرض في أول دراستنا الى بعض أجهزة المعالجة الأولية، هاته الخطوة تعتبر الموالية لعملية كشف الحدود، ثم نتطرق الى هذه الأخيرة.

العملية المستهدفة في هذا العمل هي العملية الاشتقاقية، لهذا سوف نهتم بأجهزة الاشتقاقات من الدرجة الأولى والثانية كذلك الكاشفات المثالية، كاشف كاني (Canny) وكاشف دريش (Deriche) .

الكلمات المفتاحية :

المعالجة الأولية– التجزئة – كاشف الحدود.

Résumé

Ce travail consiste en l'étude et l'implémentation de détecteurs de contours. Nous étudierons donc au préalable quelques opérateurs de prétraitement, étape qui précède la détection de contours dans une image, puis nous aborderons cette dernière. L'approche envisagée est l'approche dérivative. Nous nous intéresserons donc aux opérateurs dérivateurs du premier et du second ordre ainsi qu'à deux détecteurs optimaux : le détecteur de Canny et celui de Deriche.

Mots clés : prétraitement, segmentation, détecteurs de contours.

Abstract

This work consists in studying and implementing methods to detect edges in the image. First, we will study some operators of preprocessing, step which precede the detection of edges in the image, then, we will be interested in the detectors. The approaches considered are the derivative approaches of first and second orders. We will finish by studying two optimal detectors : the detector of Canny and the detector of Deriche.

Key words: preprocessing, segmentation, detectors of edges.

Table des matières

Introduction générale

De nos jours, l'image s'intègre naturellement dans notre environnement quotidien, une image est porteuse d'un message. Le traitement d'images qui consiste à extraire de l'information dans ces dernières trouve alors bon nombre d'applications dans le champ des activités humaines.

Une image contient une grande quantité d'information, il faut donc réduire son champ d'analyse .Nous pouvons partitionner l'image, en analyse/traitement d'images cette opération s'appelle la *segmentation*. Les contours constituent des indices pertinents dans une image, leur détection est l'une des approches possibles de la segmentation, et les opérateurs qui permettent cette détection sont les *détecteurs de contours*.

Ces détecteurs sont très nombreux, et le choix de l'un d'entre eux dépend de l'image étudiée et des résultats que l'on veut obtenir.

Nous nous proposons de présenter différents détecteurs de contours, il est à noter cependant, que toute expérimentation nécessite la production et la manipulation ultérieure d'images numériques.

Le sujet s'organise autour de la présentation de différents détecteurs de contours.

Pour ce, il nous a d'abord semblé nécessaire de situer, d'une manière générale, les détecteurs de contours dans une chaîne de traitement d'images .l'étape de prétraitement, qui précède la détection de contours est abordée d'une manière plus détaillée. (Chapitre I).

La détection de contours est l'une des approches possibles de la segmentation, cette notion fait donc l'objet du chapitre II.

Le chapitre III sera consacré aux détecteurs de contours proprement dits. Nous présenterons les détecteurs basés sur l'approche dérivée première, ceux basés sur l'approche dérivée seconde, ainsi que deux détecteurs optimaux (Canny et Deriche).

Nous présenterons également la transformation de Hough qui permet la détection des droites dans une image.

Les détecteurs cités précédemment feront l'objet de fonctions sous Matlab (Chapitre IV), ces fonctions seront testées sur différentes images.Les tests et les résultats sont présentés dans le dernier chapitre (Chapitre V).

Introduction

Le développement considérable du traitement numérique, son accessibilité, nous permet de créer, traiter, transmettre des images. Le traitement d'images est un ensemble d'opérations permettant d'améliorer la qualité d'une image.

La segmentation est l'un des maillons de cette chaîne, la détection de contours peut être l'une des méthodes utilisées.

Dans ce chapitre nous présenterons d'abord des généralités sur le traitement d'images avec une reconstitution de la chaîne de traitement, puis nous aborderons d'une manière plus détaillée le prétraitement, étape indispensable avant la segmentation.

A. Généralités- Reconstitution d'une chaîne de traitement

1. Traitement d'images [1] [3] [4]

1.1. Définition

Le traitement d'images est né de l'idée et de la nécessité de remplacer l'observateur humain par la machine. L'image ou les signaux provenant des capteurs ont alors été numérisés pour pouvoir être traités par l'ordinateur. Dans un deuxième temps, l'image a été codée puis mémorisée sur différents supports. La vision intervenant dans un grand nombre d'activités humaines, le champ des applications est très vaste.

Le traitement d'images peut se définir comme l'ensemble des opérations appliquées à une image afin d'en améliorer la qualité et d'en faciliter l'interprétation.

La qualité étant une notion subjective elle est soumise à la réalisation d'un objectif. (La qualité d'une image pour un ordinateur et un opérateur humain n'est pas forcément la même). Ceci explique la diversité et la richesse des méthodes utilisées, leur choix sera fonction de l'application donnée. L'analyse d'images consiste à extraire, traiter, interpréter les informations contenues dans l'image. On peut alors distinguer les traitements bas niveau (données de nature numérique) et les traitements de haut niveau (données de nature symbolique).La segmentation (donc *détection de contours*) est un *traitement de bas niveau*.

1.2 Vue générale : le domaine et ses applications

1. Un domaine pluridisciplinaire

Le traitement d'images fait appel à :

- Optique
- Traitement de signal
- Analyse numérique
- Informatique
- Théorie de l'information
- Etude Statistique
- Electronique
- ....

2. Des applications très variées

- Compression d'images
- Amélioration d'images en vue d'un meilleur rendu visuel (télévision)
- Imagerie médicale (compter des cellules dans une image)
- Authentification (billets de banque)
- Aide à la conduite
- Imagerie satellitaire
- Reconnaissance de l'écriture
- Débruitage d'images
- Cryptage d'images
- .....
-

2. Chaîne de traitement

Une chaîne de traitement numérique peut être modélisée comme suit :

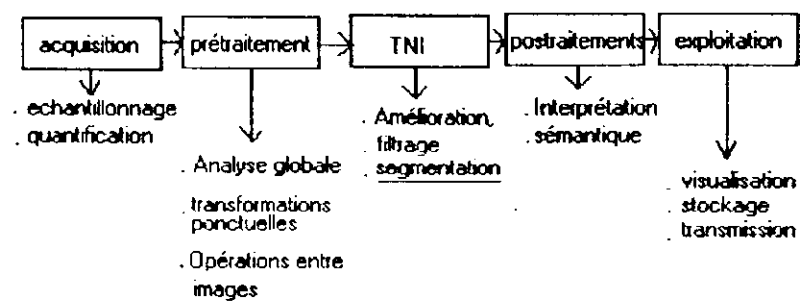


Figure I.1 : Chaîne de traitement numérique.

L'acquisition consiste en :

- la mesure du rayonnement lumineux
- l'échantillonnage et la quantification

Les prétraitements concernent les images *acquises* (correction gamma, filtrage..)

Le **Traitement** numérique d'images (TNI) consiste en :

- l'application d'algorithmes
- l'*extraction* d'informations

Les post traitements concernent les images *traitées* (reconnaissance de formes, de caractères..)

L'exploitation englobe la visualisation, la transmission, l'impression, la compression ...

Au cœur de la chaîne de traitement, on trouve l'image.

3. Image [6] [8]

3.1 Définitions

Au sens large, l'image est la projection d'une scène sur un plan.

- Image numérique : L'image peut être représentée par une matrice de points élémentaires ou pixels (image bitmap).Chaque point est porteur d'un attribut couleur qui lui est spécifique.

Image = A[i,j] où $1 \leq i \leq N$, $1 \leq j \leq M$ ×

- Dimensions de l'image : hauteur et largeur de l'image pouvant être exprimées en pixels (taille de la matrice NxM).
- Pixel (picture element): composant carré de l'image. (unité de base, pas de discrétisation). Sa valeur représente le rayonnement quantifié sur cette surface.
- Couleur : intensité de lumière associée à chaque valeur d'un pixel (il existe différents modes de représentation RVB, TSL, CMJN..)
- **Images en niveaux de gris :** l'image est représentée par une matrice. Chaque pixel représente l'intensité lumineuse comprise entre 0 et 255 (pour un codage sur 8 bits) soit en tout 256 niveaux de gris (le niveau 0 correspondant au noir et le niveau 255 au blanc).

3.2. Acquisition numérique

C'est l'opération qui consiste à numériser une image qui sera enregistrée sous forme de fichier graphique.

3.2.1. Outils d'acquisition

On peut citer

- Le scanner
- L'appareil photographique

3.2.2. Formats de l'image

Parmi les formats les plus utilisés, on a :

- BMP (Microsoft Windows Bitmap): bitmap : une matrice de bits, codé en couleur jusqu'à 24 bits par pixel, lu uniquement sur des logiciel Windows.
- JPEG (Joint Photographic Experts Group): compression d'images photographiques
- compression DCT
- PCX (Paintbrush)
- PNG (Portable Network Graphics)
- TIFF (Tagged Image File Format).....

3.2.3. Profondeur de codage, nombre de couleurs et poids de l'image

Le *nombre de couleurs* est donné par le nombre de bits alloués pour le coder ou *profondeur de codage*.

$$\text{Nombre de couleurs} = 2^N$$

N : nombre de bits alloués

La taille de l'image est la place qu'elle occupe en mémoire vive lors de l'affichage.

Le *poids* de l'image est l'encombrement résultant de son stockage.

Taille et poids de l'image ne sont pas égaux.

3.2.4. Restitution d'une image

Tout système de traitement d'image est doté d'un dispositif de visualisation qui permet par exemple d'observer les images obtenues après différentes étapes d'un traitement. Afin de visualiser l'image numérisée « brute » ; divers types de restituteurs sont utilisés, dont le rôle est l'inverse de celui des numériseurs c'est-à-

dire qui transforme le tableau de nombres qu'est l'image numérique en une image analogique visible par l'oeil de l'observateur. Pour cela divers types de supports peuvent être employés : écran cathodique, cliché photographique, impression sur papier, etc... Dans tous les cas pour chaque échantillon de l'image numérique on recrée un nouvel élément d'image, c'est-à-dire un nouveau pixel dont on choisit la forme de façon à reconstituer une image qui soit la plus proche possible de l'image avant numérisation compte tenu des erreurs introduites par les diverses opérations (préfiltrage, échantillonnage, quantification,...).

Après cette brève présentation de l'image, et de son acquisition nous allons passer aux prétraitements qui peuvent être appliqués à cette dernière.

4. Prétraitements [4] [5]

Avant de traiter une image il est nécessaire d'en définir la qualité. Un examen visuel permet de nous orienter sur la question. Cependant, pour une analyse objective de la qualité, il existe des outils statistiques (l'histogramme par exemple), qui permettent une analyse globale de l'image à partir de laquelle nous pouvons effectuer des modifications sur celle ci.

L'étape de prétraitement permet d'améliorer la qualité d'une image. C'est une étape importante dans une chaîne de traitement d'images.

Pour la segmentation, elle permet

- d'éliminer une partie du bruit et,
- de mettre en valeur les contours.

Plusieurs méthodes de prétraitement (modification d'histogramme, réduction de bruit par filtrage, rehaussement de contraste) seront présentées dans la deuxième partie de ce chapitre.

5. Traitement numérique d'images

Une fois acquise, prétraitée, l'image peut encore subir d'autres transformations en vue d'extraire de l'information. Cela consiste à appliquer des algorithmes permettant la reconnaissance et la localisation d'éléments primitifs (régions, *contours*, segments de droite...) qui représentent des informations importantes pour une interprétation ultérieure.

C'est dans cette étape que s'inscrit la *segmentation*, il peut s'agir alors :

- D'extraire des *contours*
- Ou d'extraire des régions d'une image.

6. Postraitements [4] [9]

Après extraction des primitives, ces dernières peuvent être interprétées. Ces traitements sont dits de haut niveau car ils opèrent sur des données de nature symbolique associées à la réalité de l'image et sont relatifs à l'interprétation et la compréhension de l'image.

L'interprétation automatique d'images est utilisée de manière limitée à cause des problèmes de fiabilité qu'elle pose, (images aériennes, microscope électronique..). De manière encore expérimentale, elle intervient dans la commande de robots qui contrôlent leurs mouvements par une caméra de prise de vue; par contre le contrôle optique d'une production industrielle est monnaie courante.

Les difficultés sont de nature et d'importance diverses, elles dépendent évidemment du problème posé par l'application.

En pratique il s'agira, par exemple, de reconnaître des chromosomes d'une certaine forme dans une image fournie par le microscope, ou encore de déterminer la présence d'un objet préalablement décrit dans un champ délimité.

B. Les prétraitements

L'étape de prétraitement regroupe toutes les techniques visant à améliorer la qualité d'une image, c'est-à-dire son interprétation. La qualité d'une image n'est pas forcément la même pour un ordinateur et un opérateur humain, il peut s'agir alors de produire une image la plus proche de la réalité physique, ou de satisfaire l'œil de l'observateur humain.

La phase de prétraitement, située en amont de la segmentation, a pour but de faciliter cette dernière en renforçant la ressemblance entre pixels d'une même région, ou en accentuant leur dissemblance lorsqu'ils appartiennent à des régions différentes.

Dans ce chapitre nous présenterons les techniques de prétraitements les plus courantes :

- La modification d'histogramme
- La réduction du bruit par filtrage
- Le rehaussement de contraste

1. La modification d'histogramme [1] [3] [4]

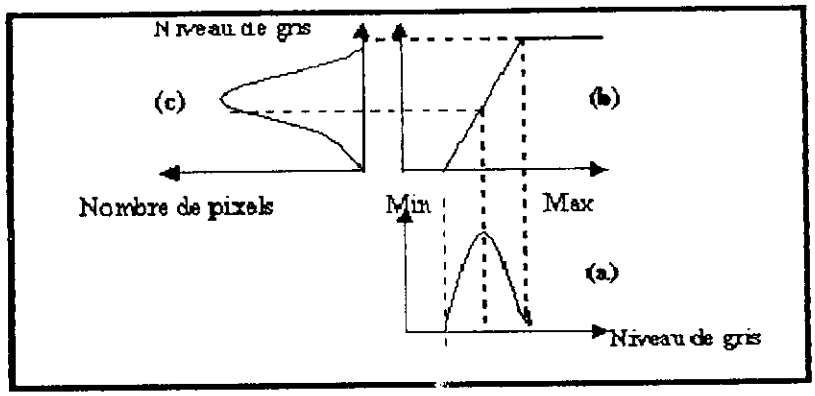
Cela consiste à améliorer l'image en lui appliquant une transformation ponctuelle d'intensité.

A tout pixel on associe une intensité. La transformation T est choisie croissante de façon à préserver les contours relatifs entre régions. Du fait de leur caractère ponctuel, ces méthodes *n'affectent pas la forme des régions. Elles modifient uniquement l'apparence visuelle.*

1.1. Notions d'histogramme, d'histogramme cumulé de Look Up Table (LUT)

1.1.1 Histogramme

On peut pour une image en niveaux de gris déterminer pour chaque niveau le nombre de pixels représentatifs : c'est le tracé d'*histogramme* .On peut donc assimiler l'histogramme à la densité de probabilité des intensités lumineuses (à un facteur de normalisation près).



a)- Histogramme de l'image originale

b)- Fonction de transformation

c)- Histogramme recadré

Figure I.2 : Principe de la modification d'histogramme

L'histogramme est un outil privilégié en analyse d'images

1.1.2 Histogramme cumulé

Pour chaque niveau de gris, la valeur de l'histogramme cumulé est égale au nombre de pixels dont la valeur est inférieure ou égale au niveau de gris dans l'image.

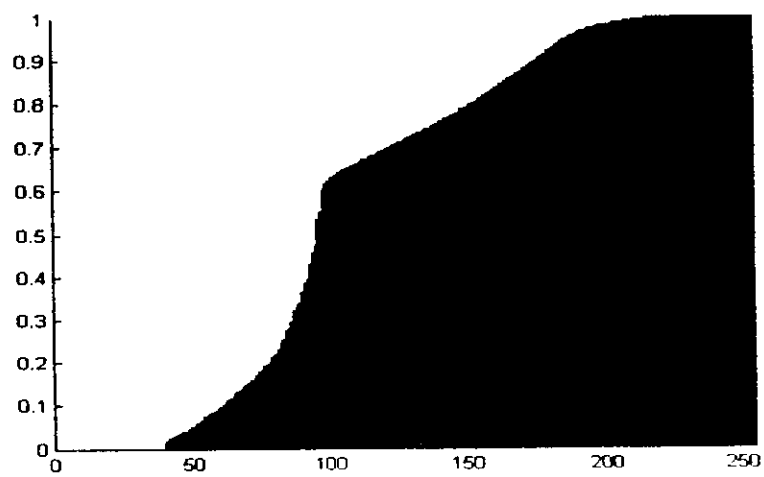


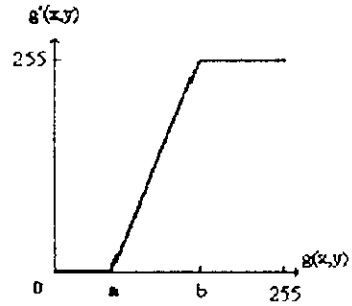
Figure I.3: Histogramme cumulé normalisé.

1.1.3 Look Up Table (LUT)

Une LUT est une bijection (fonction, notée T par la suite) qui transforme un niveau de gris I en un niveau de gris J. Cette transformation est ponctuelle et

s'applique donc à chaque pixel. Il n'y a pas de modification spatiale de l'image (déformation par exemple)

Exemples de quelques transformations (LUT) :



Recadrage: permet d'obtenir une image de dynamique maximale

Figure I.4 : Recadrage

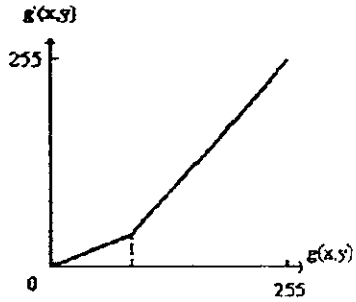
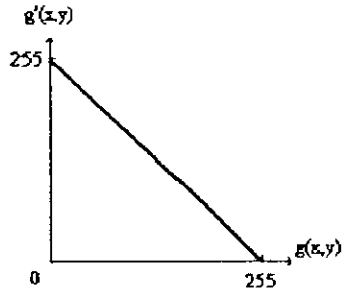
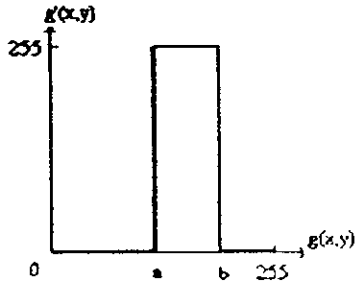


Figure I.5 : Dilatation des zones claires



Inversion de dynamique : on inverse les extrêmes noir et blanc, car pour l'œil humain certains détails se distinguent mieux en blanc sur fond noir qu'en noir sur fond blanc (ou inversement)

Figure I.6 : Inversion de dynamique



Extraction d'une fenêtre d'intensité : inverse du recadrage de dynamique. Permet une mise en évidence rapide de certains détails si l'on possède une connaissance *a priori* de leur distribution de niveaux de gris.

Figure I.7 : Extraction d'une fenêtre d'intensité

1.2. Extension de dynamique (recadrage)

Le but de cette transformation est d'utiliser au mieux l'échelle de niveaux de gris. Soit A[i,j] l'image de départ et A'[i,j] l'image après transformation. Soient $[a_0, a_1]$ l'intervalle des intensités présentes dans l'image et $[a_{min}, a_{max}]$ l'intervalle disponible. L'extension de dynamique correspond à la transformation linéaire T suivante :

$a_s' = \alpha + \beta . a_s$ telle que : $\forall$ a $\in [a_0, a_1]$ a $\xrightarrow{T}$ a' $\in [a_{min}, a_{max}]$

avec :

$$\alpha = \frac{a_{min}.a_1 - a_{max}.a_0}{a_1 - a_0}, \quad \beta = \frac{a_{max} - a_{min}}{a_1 - a_0}$$

- Une extension de dynamique de l'histogramme permet l'amélioration de l'aspect visuel l'atténuation des erreurs d'arrondi et la possibilité d'éviter le recours à l'utilisation des nombres en virgule flottante.

1.3 Egalisation d'histogramme

Il s'agit de rendre l'histogramme le plus plat possible.

Pour cela on considère que l'image est un ensemble de réalisations d'une variable aléatoire A de densité de probabilité f(a) (l'histogramme normalisé de l'image sur l'intervalle de $]a_{min} ; a_{max}[$) et de fonction de répartition F(a) (l'histogramme cumulé de l'image).

La transformation T (choisie continue, dérivable au moins par morceaux et strictement croissante) telle que la variable aléatoire B= T(A) soit uniformément répartie sur $[b_{min} ; b_{max}]$.

Avec g(b) la densité de probabilité de B et T' la dérivée de T on a les relations :

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

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

ceci équivaut à : $T'(a) = (b_{max} - b_{min}) f(a)$

la transformation T est alors définie par :

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

Elle permet un renforcement du contraste sur des détails de l'image, une amplification des fluctuations dans les zones où elles sont faibles. Le sens et la position des transitions sont conservés

En général, il n'est pas possible d'établir une égalisation parfaite.

2. Réduction de bruit par filtrage (lissage) [1] [3] [4]

2.1 Les sources de bruit

1. bruit lié au contexte de l'acquisition : lié aux évènements inattendus (bougé, modification ponctuelle des conditions d'éclairage..)
2. bruit lié au capteur : (mal utilisé ou de mauvaise qualité)
3. bruit lié à l'échantillonnage et reflète essentiellement des problèmes de quantification (CCD) :
   - précision d'environ 1/512
   - problèmes dans le cas d'applications de grande précision
4. bruit lié à la nature de la scène : l'environnement où se situe l'acquisition n'est pas toujours maîtrisable (couverture nuageuse en télédétection, poussière dans un atelier..)

2.2 Types de bruit

Le bruit peut être considéré comme

1. additif : I=I'+B

(dans la plupart des cas)

2. multiplicatif :I=I'.B

(cas des speckle dans les images radar ou grain sur les films radiographiques)

3. convolutif : I=I'*B

(Utile pour les dégradations dues à un effet de bougé ou mauvaise mise au point)

Où **I** représente l'image à traiter, **I'** l'information utile et **B** un champ aléatoire.

2.3 Filtrage

On peut considérer une image comme étant constituée de plusieurs zones homogènes représentant les objets. Dans la réalité, des fluctuations des niveaux de

gris sont présentes à cause du bruit. On cherchera donc à diminuer l'amplitude de ces fluctuations sans toucher aux transitions.

Contrairement aux manipulations d'histogrammes, qui sont des opérations ponctuelles sur tous les pixels de l'image, les filtrages sont des produits de convolution qui mettent en jeu l'environnement (voisinage) de chaque pixel.

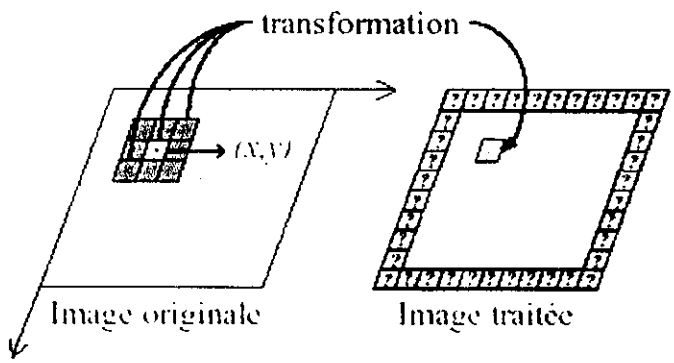
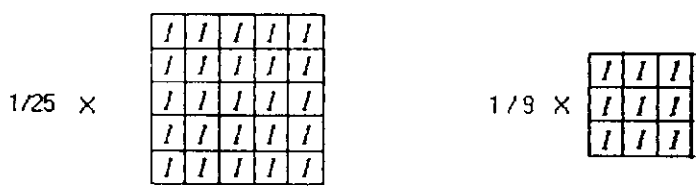


Figure I.8 : Application d'un masque à une image.

2.3.1 Filtre moyenneur

Il s'agit d'effectuer une moyenne des niveaux de gris autour du pixel central
On peut utiliser des masques du type :



La taille du masque est un paramètre variable, plus la taille du masque est grande, plus le filtrage est important. De manière générale, on cherche à avoir un filtrage isotropique, pour cela le voisinage considéré devra avoir une forme circulaire, mais il faudra tenir compte du problème de la définition d'un cercle sur un maillage carré.
Exemple :

1/29 ×

0	0	0	1	0	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
1	1	1	1	1	1	1
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	0	1	0	0	0

L'inconvénient du filtre moyenneur est l'introduction de flou dans l'image.

2.3.2 Filtre Gaussien

L'expression de la gaussienne en deux dimensions est donnée par :

$$h(x,y) = \frac{1}{2.\pi.\sigma^2}.\exp\left(-\frac{(x^2+y^2)}{2\sigma^2}\right)$$

L'avantage d'un tel filtre est que l'on règle très facilement le degré de filtrage à travers le paramètre σ.

Par rapport au filtre moyenneur, il accorde plus d'importance aux pixels proches du pixel central, mais dégrade lui aussi les contours.

2.3.3 Filtre médian

Ce filtre est *non linéaire*. Il consiste à affecter au pixel central le niveau de gris séparant la population en deux effectifs égaux.

Exemple :

30	10	20
10	250	25
20	25	30

→ 10, 10, 20, 20, 25, 25, 30, 30, 250

médiane = 25

L'avantage de ce filtre est qu'il donne de très bons résultats sur le bruit impulsionnel (poivre et sel).

Ses inconvénients sont le fait, qu'il supprime les détails fins et qu'il conduit à des temps de calcul élevés.

2.3.4 Filtre de Nagao

Ce filtre est également appelé filtre à sélection de voisinage. Chaque fenêtre 5x5 est divisée en 9 domaines.

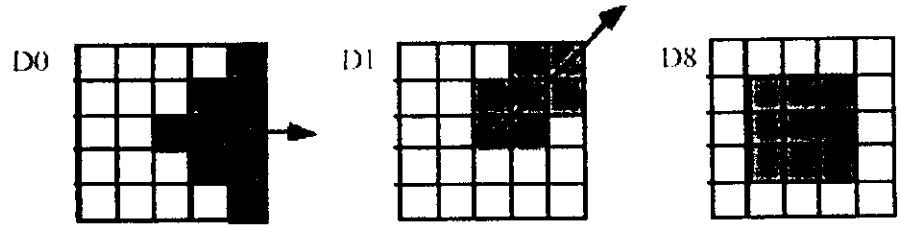


Figure I.9 : Filtres directionnels de Nagao D0, D1 et D8.

Les filtres D2, D4 et D6 sont déduits de D0 par rotation de 90° ; D3, D5 et D7 sont déduits de la même manière à partir de D1.

Pour chaque domaine Di on calcule la moyenne et la variance : moy (i), var (i). On cherche ensuite le domaine où la variance est la plus petite et on affecte la moyenne des niveaux de gris de ce domaine au pixel central.

Les avantages de ce filtre sont:l'élimination du bruit impulsionnel, la préservation et l'amélioration du contraste. De plus, les zones avant et après la transition sont plus homogènes

Parmi les améliorations possibles de ce filtre on peut citer la régularisation de la forme géométrique des domaines et le remplacement de la variance par l'étendue 'Max (gris)-Min (gris)'.

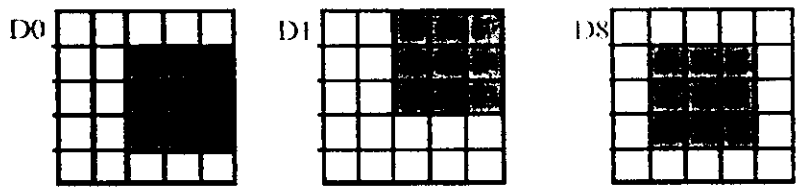


Figure I.10 : Régularisation de la forme géométrique.

2.3.5 Filtrage morphologique

2.3.5.1 Image binaire

Une image binaire est une image codée sur 2 niveaux de gris. Cette opération permet de faire apparaître certains détails ou de détacher les objets ou certaines primitives du fond.

Tous les pixels dont le niveau de gris est inférieur à un seuil sont mis à 0, les autres seront mis à 1.

Algorithme :

Si I(i,j)< seuil **alors** I(i,j)=0 **Sinon** I(i,j)=255

Le problème réside dans le choix du seuil.

2.3.5.2 Opérateurs morphologiques

1. Dilatation

Elle consiste à dilater l'image. De ce fait, les points noirs isolés au milieu de parties blanches sont éliminés. Le processus est réalisé en balayant sur l'image une fenêtre carrée de taille (N+1)*(N+1) et en effectuant pour chaque pixel de l'image le **OU** logique des $(N+1)^2$-1 voisins.

La dilatation peut être interprétée comme une contraction de 1 pixel, elle élimine les trous.

2. Erosion

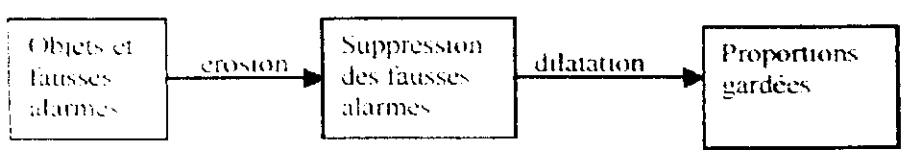
Le procédé est dual à la dilatation. Il s'agit d'une croissance de 1 pixel.

On effectue cette fois le **ET** logique des $(N+1)^2$-1 voisins. De ce fait, les points blancs isolés au milieu de parties noires sont éliminés mais des pixels noirs sont ajoutés au contour des objets présents dans l'image. L'érosion supprime les « fausses alarmes ».

L'érosion et la dilatation peuvent être programmées en utilisant des masques de convolution. Le type de masque dépend de la connexité avec laquelle on travaille.

3. Cascade d'opérations

- **Ouverture :** elle est constituée par une opération d'érosion suivie d'une dilatation. Elle permet de retrouver la taille normale des objets de l'image. L'ouverture servira à supprimer les fausses alarmes :



- **Fermeture :** c'est l'opération inverse de l'ouverture, qui consiste à faire subir à l'image une dilatation suivie d'une érosion. Elle permet aussi de retrouver la taille normale des objets de l'image.

La fermeture supprimera les trous :

Remarque sur le filtrage

Il existe une dualité filtrage/raideur de la transition : un faible lissage préserve les transitions mais élimine mal le bruit, un fort lissage élimine mieux le bruit mais les transitions sont plus étalées, il faudra donc faire un compromis.

2.3.5.3 Contours d'une image binaire

- Contours intérieurs : L'érosion supprime un pixel aux bords de l'objet. En faisant (Image – Erodé) on obtient les contours intérieurs.

- Contours extérieurs : on peut obtenir les contours extérieurs de l'objet en faisant (Dilaté - Image).

3. Rehaussement de contraste [1] [4] [7]

3.1. Définition du contraste

C'est l'opposition marquée entre deux régions d'une image, plus précisément entre les régions sombres et les régions claires de cette image. Le contraste est défini en fonction des luminances de deux zones d'images.

Si L_1 et L_2 sont les degrés de luminosité respectivement de deux zones voisines A_1 et A_2 d'une image, le contraste C est défini par le rapport :

$$C = \frac{L_1 - L_2}{L_1 + L_2}$$

C'est une mesure relative des différences dans l'image

3.2. Principe

L'intérêt des opérateurs précédents est de réduire les fluctuations d'intensité dans une même région en essayant de préserver les transitions.

Cependant, si la transition est floue, on peut faire un 'rehaussement de contraste', ce qui revient à diminuer l'étendue de la transition sans affecter l'intensité moyenne des régions situées de part et d'autre de cette transition.

L'opérateur de rehaussement devra, si possible, réduire le bruit dans les zones stationnaires et éviter les phénomènes de dépassement.

Il s'agit de réaliser l'opération :

Signal – K * dérivée seconde (où K est un paramètre à régler)

Soit pour une image

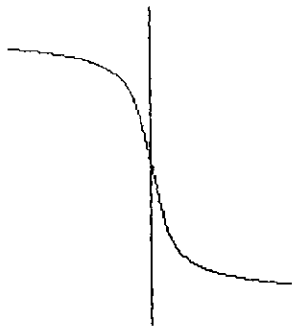


Figure I.11 : Signal original

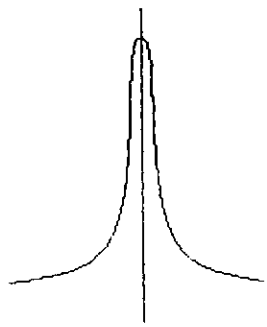


Figure I.12 : Dérivée première

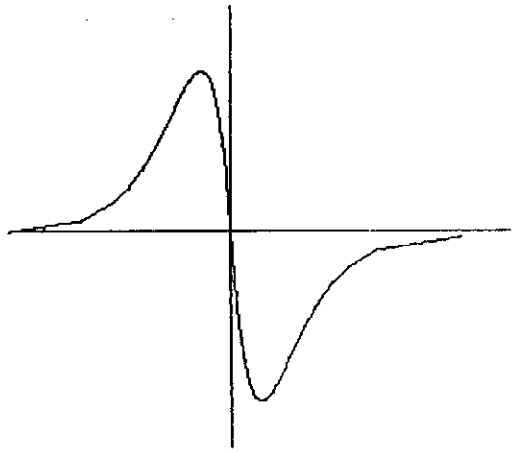


Figure I.13 : Dérivée seconde

Figure I.14 : Signal – K* dérivée seconde

Commentaire : Le signal obtenu possède une transition plus abrupte. K est le paramètre permettant de régler la raideur de la transition

3.3. Utilisation du Laplacien

Le Laplacien, qui constitue une bonne approximation de la dérivée seconde, est souvent utilisé en amélioration d'images pour rehausser le contraste :

$$I'(x.y) = I(x.y) - c\nabla^2 I(x.y)$$

Pour calculer le Laplacien d'une image, on peut utiliser des masques du type

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Le principal inconvénient est la sensibilité au bruit.

Conclusion

Dans ce chapitre nous avons d'abord présenté une vue d'ensemble du traitement d'images. L'image une fois acquise subit une succession de transformations qui peut aboutir à une éventuelle interprétation de cette dernière. Le volume des informations à traiter et des paramètres intervenant tout au long du processus peut être très important, réduire le champ d'analyse par l'extraction de primitives semble alors intéressant, mais l'image doit d'abord être prétraitée.

Par la suite, plusieurs méthodes de prétraitement ont été abordées, il s'agit de transformations ponctuelles ou locales permettant d'améliorer la qualité de l'image acquise. Le but est de réduire le bruit tout en préservant les transitions entre les différentes régions afin de préparer l'image à l'étape suivante : *la segmentation.*

Introduction

La phase de prétraitement a consisté à restaurer et améliorer l'image, soit à la préparer à l'analyse. Il faut maintenant extraire les aspects intéressants de l'image (les primitives).

Pour ce faire, l'image peut d'abord être divisée en différentes zones homogènes .La segmentation est une opération qui permet de diviser l'image en plusieurs régions délimitées par des contours.

Deux approches de la segmentation sont alors possibles :

- Approche frontière (détection de contours)
- Approche région

Dans ce chapitre nous allons donc présenter ces deux approches.

1. Généralités [2][4]

La segmentation est un traitement bas niveau qui consiste à créer une partition de l'image en sous ensembles appelés régions.

Une région est, un ensemble connexe de points ayant des propriétés communes (intensité, texture...) qui les différencient des régions voisines.

Le choix d'une technique de segmentation est lié à :

- La nature de l'image
  - Eclairage non homogène, reflets,
  - présence de bruit, de zones texturées,
  - Contours flous, en partie occultés
- Aux opérations situées en aval de la segmentation
  - Localisation, mesure, calcul 3D,
  - Reconnaissance des formes, interprétation,
  - Diagnostic, contrôle qualité,
- Aux primitives à extraire
  - Contours, segments de droite, angles...
  - Régions, formes,
  - Textures,
- Aux contraintes d'exploitation
  - Complexité algorithmique, fonctionnement en temps réel,
  - Taille de la mémoire disponible en machine.

La segmentation fait référence aux notions de différence et de similarité comme les perçoit le système visuel humain et ceci donne naissance aux deux approches « frontière » et « région ».

1.1. Définitions

- Contour : un contour est une transition marquée entre deux régions ayant chacune une luminosité distincte.
- Texture : la texture peut être définie comme la répétition spatiale d'un même motif dans plusieurs directions. Il s'agit d'une notion utilisée pour traduire un aspect homogène de la surface ou pour la décrire qualitativement à l'aide des adjectifs : fine, lisse, granuleuse.....

Intérêt de l'étude de la texture: beaucoup d'images (en particulier les images naturelles) possèdent des zones texturées (non homogènes au sens des niveaux de gris). Pour segmenter une image en régions, en général on ne se limite pas aux niveaux de gris. Malheureusement il n'existe pas de définition universelle de la texture à l'heure actuelle...

1.2. Partitions d'une image [4]

1.2.1. Partitions élémentaires

1.2.1.1. Pavage, maillage d'une image

- Un *pavage* est une partition du plan, utilisant toujours les mêmes formes élémentaires, les tesselles.

Figure II.1 : Pavages utilisés en imagerie

Remarque : Le pavage rectangulaire (carré) est le plus utilisé.

- Un *maillage* est un graphe non orienté dont les nœuds sont les sites. Tout site est associé à une tesselle du pavage. (dualité pavage-maillage)

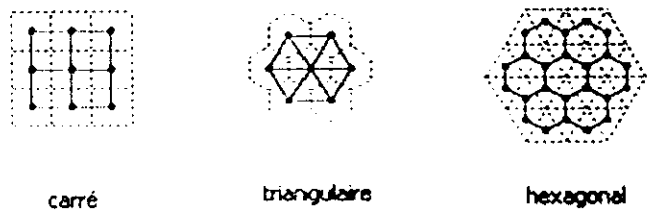


Figure II.2 : Maillages associés

Le maillage est donc une connexion des points centres des pavés par des lignes. Il faut que les pavés se touchent par un coté. Les centres des pavés sont des points discrets correspondant aux pixels

Le pavage carré :

- Le pavage carré donne un maillage carré
- Pour le pavage carré les points discrets coïncident avec les points à coordonnées entières selon deux axes :

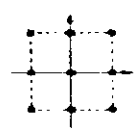


Figure II.3 : Axes associés

L'avantage du pavage carré est que chaque pavé a 8 pavés voisins (4 pavés le touchant par un côté, 4 le touchant par un sommet)

Chaque point a 8 voisins, 4 selon les axes et 4 selon les diagonales.

1.2.2. Adjacence dans le cas du maillage carré

Tout pixel d'une image se code comme un couple (i, j) d'entiers. Dans un maillage carré, tout pixel a 2 types de voisins, à savoir ses 4 voisins selon les axes, et ses 4 voisins selon les diagonales. Deux pixels p et q sont dits :

1. 4-adjacents s'ils sont voisins suivant un axe (*voisinage en 4-connexité*)
2. 8-adjacents s'ils sont voisins suivant un axe ou une diagonale (*voisinage en 8-connexité*)

Figure II.4 : Voisinage d'un pixel

Etant donné un pixel (i,j), en maillage carré les pixels adjacents à (i,j) sont :

- (i+1,j), (i-1,j), (i,j+1), et (i,j-1) pour la 4-adjacence
- les mêmes, plus (i+1,j+1), (i+1,j-1), (i-1,j+1),
- et (i-1,j-1) pour la 8-adjacence

On vérifie alors les relations suivantes :

- (i,j) est 4-adjacent à (i',j') si et seulement si $|i - i'| + |j - j'| = 1$
- (i,j) est 8-adjacent à (i',j') si et seulement si $\max(|i - i'|,|j - j'|) = 1$

2. Approche frontière

2.1. Définition

L'approche frontière regroupe les techniques de détection de contours. Ces méthodes *ne conduisent pas directement* à une segmentation de l'image car les contours obtenus sont rarement connexes, il faut donc procéder à une fermeture des contours si l'on souhaite une partition complète de l'image. En effet, après fermeture de contours, la dualité contour-région apparaît nettement.

D'une manière générale, la segmentation par approche frontière consistera en :

- une détection des contours,
- la fermeture des contours,
- et le codage des contours.

2.2. Détection de contours

Les contours peuvent être définis comme des discontinuités de la fonction d'intensité dans les images. Le principe de la détection de contours repose donc sur l'étude de la variation de la fonction d'intensité dans l'image. Plusieurs approches peuvent être envisagées (approche gradient, Laplacien....). Le problème réside dans la présence de bruit dans l'image. Il faudra donc déterminer des critères qui nous permettront cette détection, et qui définiront la 'nature' des contours à extraire.

2.2.1. Opérateurs de détection

On peut distinguer

2.2.1.1. Opérateurs dérivatifs du premier ordre (approche gradient)

- Les opérateurs de Prewitt et de Sobel,

- Les opérateurs directionnels de Roberts.

2.2.1.2. Opérateurs dérivatifs du deuxième ordre (approche Laplacien)

- Laplacien sur voisinage réduit,
- Marr Hildreth.

2.2.1.3. Filtrage optimal

- opérateur de Canny,
- opérateur de Deriche.

2.2.1.4. Localisation des contours et seuillage

L'application des opérateurs de détection de contours (c'est à dire des discontinuités de la fonction d'intensité dans les images) peut conduire à des contours épais, des contours non détectés, ou encore du bruit. Il s'agit alors d'extraire un maximum de contours puis de trier ce qui a été détecté.

On peut envisager :

- L'extraction des maximums locaux de la norme du gradient,
- Le seuillage par hystérésis,
- Ou le suivi d'une ligne de crête.

Remarques :

Cette étape permet de détecter les contours mais ne nous donne pas toujours les points contours (contours d'une 'épaisseur' d'un pixel).

L'étude détaillée des détecteurs de contours se fera dans le chapitre suivant.

2.3. Fermeture des contours

La détection de contours n'aboutit pas toujours à des contours fermés (si le seuil d'extraction est trop élevé par exemple),or par définition la segmentation se définit comme une partition de l'image, nous devons aboutir à des régions définies comme l'intérieur de lignes fermées ,d'où l'intérêt de cette étape.

2.3.1. Recherche du meilleur chemin entre 2 extrémités

L'image numérique peut être associée à un maillage où chaque sommet est un site ou pixel. Le maillage est un graphe, la fermeture des contours peut s'apparenter à la

recherche d'un chemin entre les deux sommets du graphe associés aux deux sites So et Sr qui sont les extrémités de la lacune à combler.

Il existe plusieurs algorithmes (non spécifiés ici) pour réaliser cette opération. En général il s'agit de minimiser une fonction d'évaluation qui nous donne le chemin solution de coût minimum.

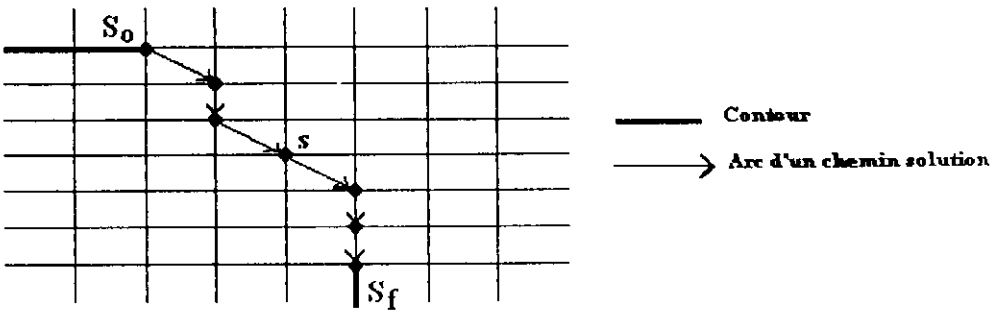


Figure II.5 : Recherche de chemin sur un maillage.

2.3.2. Recherche du meilleur chemin à partir d'une extrémité

Il s'agit d'extraire les contours « surs » en fixant un seuil suffisamment élevé pour ne pas détecter le bruit. Cette méthode suppose une extraction au préalable des contours par un opérateur donnant des contours d'une « largeur » d'un pixel et fournissant la norme du gradient.

2.3.1.1. Identification des extrémités

Les contours étant fins, il est possible d'identifier une extrémité par l'étude du voisinage 3x3 de chaque point contour dont les huit voisins sont désignés comme l'indique la figure II.6. Si le pixel voisin est un point de contour, x_i vaut 1, sinon il vaut 0. Ainsi, chaque voisinage peut être codé sur un octet.

$$V = \sum_{i=1}^{7} x_i \, 2^i$$

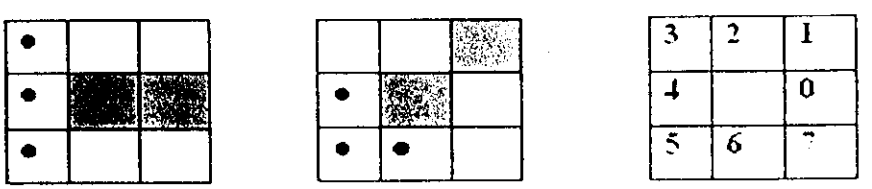


Figure II.6 : Exemple de configuration extrémités : Pixels hachurés représentent les contours, les points noirs représentent les candidats à la fermeture et le carré de droite représente la numérotation des voisins pour le calcul du code V.

On construit alors une table T de 256 éléments (LUT) dont la fonction d'adressage est le code V du voisinage. Si le pixel examiné est une extrémité, l'élément T[V] adressé par le code V du voisinage contiendra un 1 et les coordonnées relativement à s des 3 pixels candidats à la fermeture.

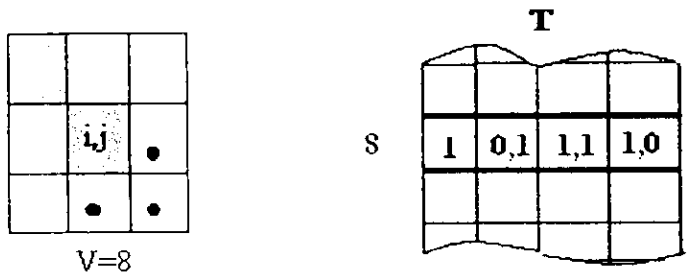


Figure II.7 : Codage d'une extrémité et table d'examen

2.3.2.2. Algorithme de fermeture

L'image doit être balayée ligne par ligne du haut vers le bas. Lorsqu'une extrémité est rencontrée, la procédure de recherche du meilleur candidat à la fermeture est activée. Quand la condition d'arrêt est atteinte, le balayage de l'image reprend là où il s'est arrêté.

2.4. Codage des contours

2.4.1. Code de Freeman

Il s'agit du codage local le plus simple pour stocker une ligne fermée ou non : il caractérise le passage d'un pixel à son successeur.

Le code de Freeman associe à chaque déplacement élémentaire (vers un des voisins définis par la 8-connexité) un code entier dans l'intervalle [0,7] suivant la convention donnée par la figure II.8. On parle de direction de Freeman.

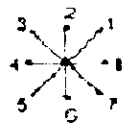


Figure II.8 : Directions de Freeman

Toute information contenue dans une image et qui peut être parcourue de manière linéaire se code par la succession des directions de Freeman le long de ce parcours.

Exemple :

Figure II.9 : Exemple de codage de contour

Le code de Freeman est utilisé pour le suivi de contour dans une image binaire ou d'une fermeture de contour après extraction de points de frontière dans une image en niveaux de gris.

Il est aussi utilisé pour le codage d'un arc, d'une courbe.... (entités linéaires en général). Il permet un codage exact et une compression de données

2.4.2. Codage de la frontière d'un objet en 8-connexité [4]

Le suivi de contour permet de générer le code de Freeman de la frontière d'un objet. L'objet est une composante 8-connexe. On appelle point frontière, les points de l'objet 4-connectés au fond. Lorsque l'on est sur un point de contour P_i, on recherche le point de contour suivant en parcourant dans un sens donné (trigonométrique) les 8 voisins de Pi à partir d'un point dont on est certain qu'il appartienne au fond. Il s'agit du point de fond qui précédait Pi dans l'exploration du voisinage de P_{i-1}. On désigne dir_{ini} par la direction qui permet de passer du point P_i à ce point de fond pour commencer la localisation de P_{i+1}. Au départ on part du point de fond situé à gauche de P_0 pour trouver P_1.

Le principe de l'extraction de la chaîne de Freeman correspondant à la frontière d'un objet est présenté par l'algorithme suivant :

Parcours de l'image du haut vers le bas et de la gauche vers la droite pour trouver le premier point du parcours ;
Soit P_0 le premier point, dans la direction dir_{init} =4 on trouve un point de fond la chaîne des directions est stockée dans CH/
SI tous les voisins de P_0 sont dans le fond ALORS P_0 est un point isolé ;
SINON

```
        /* soient :
s le site courant et t le site suivant
dir la direction désignant le point frontière suivant à partir de s
dir_init la direction à partir de laquelle on est sur de trouver un point de fond à partir de
s*/
DEBUT
Fini := FAUX ; t := [0, 0] ; dir :=4 ; k :=1 ;
TANT QUE NON fini FAIRE
DEBUT
TANT QUE t ∉ objet FAIRE
DEBUT
dir :=(dir+1) mod 8 ;
calculer le nouveau site t à partir de s et dir ;
dir_init := (dir+5) mod 8 ;
FIN
SI t est le deuxième point
ALORS mémoriser t dans P1 ;
SINON SI (s=P0) ET (t=P1) ALORS fini :=VRAI
s :=t ;
t := point déduit de t suivant dir_init;
CH[k] := dir ; k := k+1 ;
Dir := dir_init ;
FIN
FIN
```

3. Approche région [1] [4]

3.1. Définition

La notion de *région* fait référence à des groupements de points ayant des propriétés communes. Les méthodes de l'approche région *aboutissent directement* à une partition de l'image, chaque pixel étant affecté à une région unique.

Nous présentons ici le principe de quelques méthodes.

- par agrégation de pixels,
- par division,

- par division-fusion,
- par quadtree.

3.2. Segmentation par agrégation de pixels

Au départ, il s'agit de considérer chaque pixel comme une région. Ensuite on essaie de les regrouper à partir d'un double critère :

- critère de similarité des niveaux de gris : exemple : variance des niveaux de gris dans une région R
- critère d'adjacence

Principe : on choisit un germe et on fait croître ce germe tant que les pixels de son voisinage vérifient les tests d'homogénéité. Lorsqu'il n'y a plus de candidat on choisit un autre germe.

3.3. Segmentation par division

Cette méthode, contrairement à la précédente suppose au départ que tous les pixels appartiennent à la même région. Si les pixels de cette région ne vérifient pas le critère d'homogénéité, la région est divisée. Le processus est réitéré sur chaque région et s'arrête lorsque le critère d'homogénéité est vérifié (obtention d'une région homogène). La variance des niveaux de gris (inférieure à un certain seuil), peut être utilisée comme critère d'homogénéité. On peut également rajouter un critère sur la taille des régions.

Il existe plusieurs techniques de division

Exemple :

L`une des techniques utilisées consiste à diviser les différents modes de l'histogramme et à affecter à chacun d'entre eux une ou plusieurs régions.

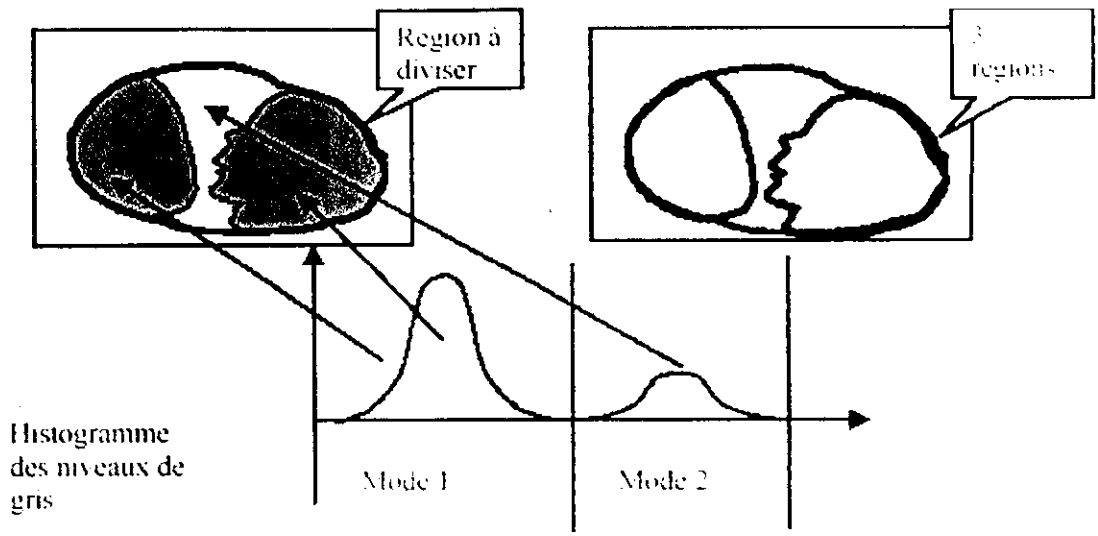


Figure II.10 : Principe de la segmentation par division

3.4. Segmentation par division-fusion

Cette méthode consiste à reprendre les résultats de la segmentation par division puis à fusionner les régions malencontreusement séparées. Pour ce, on utilise le *graphe d'adjacence.*

Le graphe d'adjacence est un graphe non orienté où chaque nœud représente une région et chaque branche représente l'adjacence entre les régions.

Ainsi, on peut remonter aux régions sans passer par les pixels.

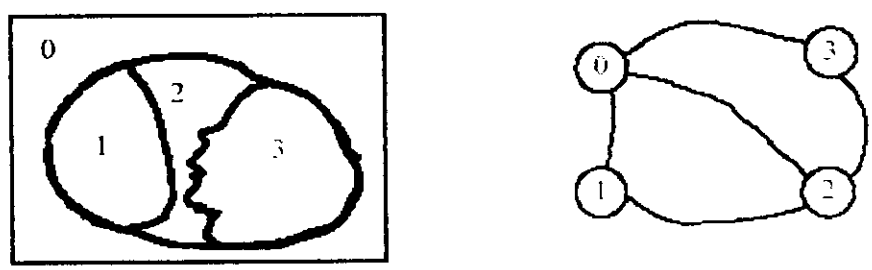


Figure II.11 : Principe de la division-fusion

3.5. Quadtree

Il s'agit d'une méthode de découpage récursif de l'image. Au départ l'image est constituée d'un seul rectangle découpé en 4 quarts s'il n'est pas jugé homogène.

Les quatre nouveaux rectangles sont ses fils dans un arbre (QUADTREE).chacun des fils est à nouveau découpé s'il n'est pas jugé homogène. Cette méthode peut être utilisée en segmentation ou en compression d'image.

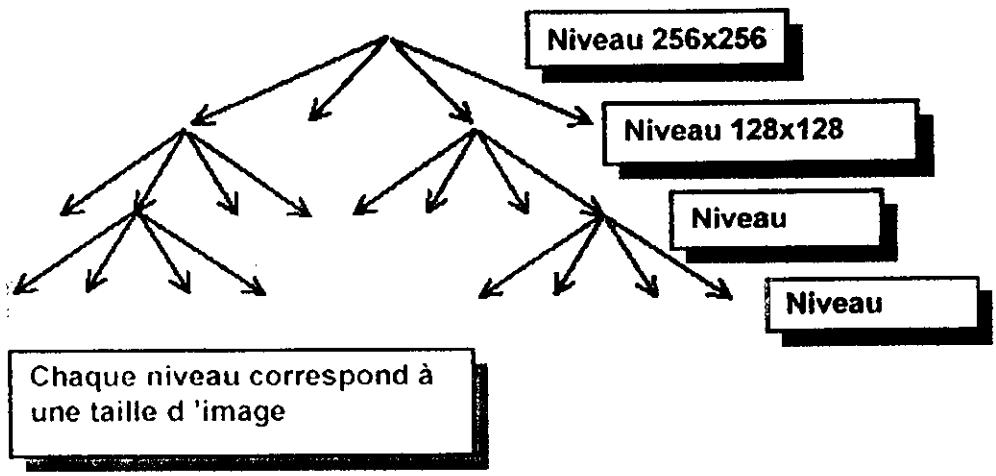
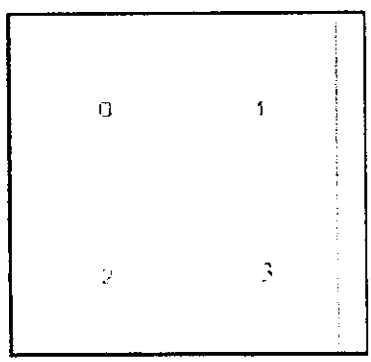


Figure II.12 : Segmentation de l'image en Quadtree

Dans le quadtree linéaire, chaque feuille du quadtree est codée de manière à pouvoir retrouver sa position. Le principe de codage retenu est le suivant :



00	01	10	11
02	03	12	13
20	21	30	31
22	23	32	33

000	001	010	011	100	101	110	111
002	003	012	013	102	103	112	113
020	021	030	031	120	121	130	131
022	023	032	033	122	123	132	133
200	201	210	211	300	301	310	311
202	203	212	213	302	303	312	313
220	221	230	231	320	321	330	331
222	223	232	233	322	323	332	333

Figure II.13 : Codage des différents niveaux dans la segmentation en Quadtree

3.6. Etiquetage en composantes connexes

Une image d'étiquettes est une image où les pixels d'une même région ont tous la même valeur appelée étiquette de région. La plupart des méthodes citées nécessitent un algorithme d'étiquetage en composantes connexes : tous les pixels connexes ayant le même attribut doivent être affectés à une même région.

Il existe plusieurs algorithmes d'étiquetage. La méthode la plus classique repose sur un balayage séquentiel de l'image. On considère pour chaque point P ses voisins déjà traités : deux points pour la 4-connexité et quatre points pour la 8-connexité. Ces points sont les prédécesseurs de P lors du balayage ligne par ligne de l'image.

prédécesseurs en 4-connexité prédécesseurs en 8-connexité

Figure II.14 : Masque L en 4 et 8-connexité

Pour réduire à deux le nombre de balayage de l'image, on construit une table d'équivalence T qui permet de gérer les équivalences d'étiquettes qui apparaissent lors du parcours séquentiel.

Exemple d'algorithme :

- On suppose qu'à chaque pixel est affecté un attribut,
- On veut regrouper tous les pixels ayant le même attribut ;
- On fait appel à une table de correspondance, pour réduire le nombre de balayage à 2. Il s'agit d'un vecteur où à chaque étiquette, on affecte une autre étiquette qui lui correspond.

Initialisation : correspondance (i)=(i) pour tout (i)

Premier balayage :

On déplace le masque L (Figure II.14 ,4-connexité, C=P)

1- Si att (C)=att(A) et att (C)# att (B)
Alors étiquette (C)= étiquette (A),

2- Si att (C)=att(B) et att (C)# att (A)
Alors étiquette (C)= correspondance(étiquette (B)),

3- Si att (C)#att(B) et att (C)# att (A)
Alors étiquette (C)=nouvelle étiquette ,

4- Si att (C)=att(B) et att (C)= att (A) et étiquette (A) = étiquette(B)
Alors étiquette (C)= étiquette (A),

5- Si att (C)=att(B) et att (C)= att (A) et étiquette (A) # étiquette (B)
Alors étiquette (C)=min(correspondance((étiquette (B)), étiquette (A)),

Correspondance (étiquette (C)) = étiquette (C)

Correspondance (étiquette (A)) = étiquette (C)

Correspondance (Max (correspondance (étiquette (B), étiquette (A)))= étiq (C)

Mise à jour des correspondances :

- toutes les étiquettes telles que correspondance (i)=i, correspondent à de vraies régions. On leur affecte alors un numéro de région (numérotation dans l'ordre croissant).
- Pour toutes les étiquettes ne validant pas l'hypothèse précédente, on fait correspondance (i)= correspondance (correspondance (i)).

Deuxième balayage.

A chaque pixel i on affecte l'étiquette correspondance (i).

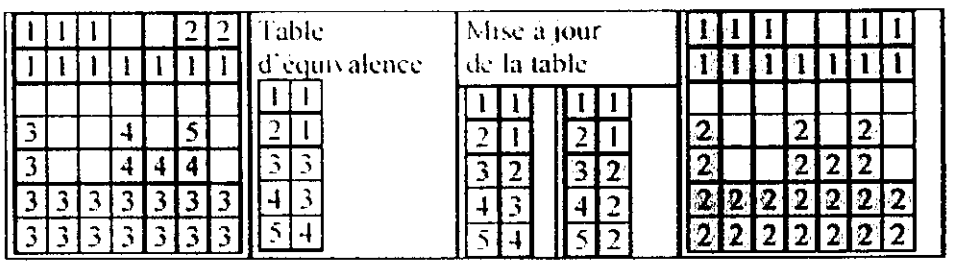


Figure II.15 : Etiquetage en composantes connexes

Cette méthode ne nécessite que deux balayages.

Conclusion

La segmentation est une phase clé dans l'extraction de primitives pour la description des aspects intéressants de l'image.

La segmentation obtenue par la détection de contours aboutit à une partition de l'image en deux classes : les points contours et les autres. La partition en régions s'obtient à partir de la fermeture des contours. Il existe donc une dualité des deux approches. Dans ce chapitre nous avons présenté sommairement quelques méthodes pour les deux approches, dans les chapitres qui vont suivre nous allons nous intéresser à la détection de contours, objet de ce travail.

Introduction

Les contours sont considérés comme des variations locales d'intensité de l'image. Les méthodes présentées permettent la détection de ces variations d'intensité.

Dans ce chapitre, nous commencerons par définir les contours et ce qu'ils caractérisent dans la scène observée, puis nous présenterons différentes approches possibles permettant l'extraction de contours :

- l'approche dérivative du premier ordre,
- l'approche dérivative de second ordre,
- l'approche par filtrage optimal.

Nous terminerons le chapitre avec la transformation de Hough, transformation globale de l'image qui opère sur les contours en eux-mêmes et qui permet d'aboutir à des primitives plus adaptées à une analyse structurelle du contenu de l'image que ne l'étaient les contours.

1. Notions Générales [10]

1.1. Les contours

Les contours sont définis comme les variations brusques d'intensité dans l'image et correspondent à des caractéristiques physiques de la scène tridimensionnelle observée.

Ces caractéristiques qui apparaissent sur les images d'éclairement d'entrée sont engendrées par des phénomènes physiques qui se produisent dans la scène observée .Les caractéristiques physiques de la scène qui sont traduites par des contours sur les images d'éclairement peuvent être:

1. **Le contour d'un objet** qui masque partiellement un autre objet de la scène observée. Ce contour d'occultation empêche de voir les caractéristiques de l'objet masqué qui est derrière.
2. **Les arêtes 3D**, qui correspondent à une *discontinuité du vecteur normal à la surface externe de l'objet.*
3. **Les différentes marques** qui apparaissent sur des surfaces visibles de la scène. Ces marques peuvent provenir de la **texture** de la surface ou d'un **changement d'albédo**.
4. **Le contour de l'ombre** projetée sur des objets de la scène et provoquée par une **discontinuité de l'éclairage incident.**

Les contours de l'image sont donc associés à des **caractéristiques, physiques ou géométriques, identifiables de la scène** tridimensionnelle sous observation.
Ces variations d'intensité représentent alors, dans un grand nombre de cas des informations importantes pour les opérations situées en aval de la segmentation.

Les contours d'une image correspondent donc aux lieux géométriques où le signal présente une forte discontinuité. Pour un signal à une dimension, on a :

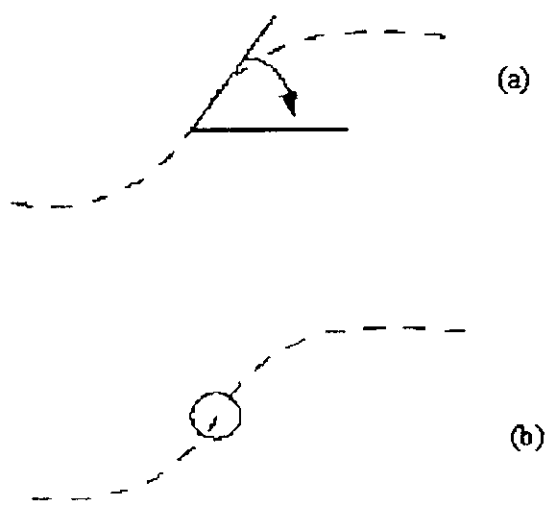


Figure III.1 : Définition des contours

(a) les contours correspondent à une pente élevée de la transition.(recherche des maxima locaux du gradient)
(b) les contours correspondent à un changement de concavité. (passage par zéros de la dérivée seconde, le laplacien pour les images).

1.2 Différents types de contours

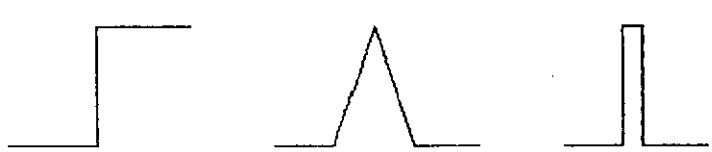


Figure III.2 : Différents types de contours : marche, toit et pointe

1. contours de type « saut d'amplitude » : les contours se situent entre les pixels appartenant à des régions ayant des intensités moyennes différentes.
2. contours de type toit : les contours correspondent à une variation locale d'intensité présentant un maximum ou un minimum.

1.3. Filtrage linéaire d'une image :

Le filtrage linéaire d'une image consiste à convoluer sa fonction d'intensité $I(x, y)$ avec une fonction $h(x, y)$ appelée réponse impulsionnelle du filtre.

$$I'(x,y) = h(x,y) * I(x,y),$$

$$I'(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} h(u,v).I(x-u,y-v)dudv,$$

$$I'(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} h(x-u,x-v).I(u,v)dudv,$$

Dans le cas discret on a :

$$I'(x,y) = \sum_{u=-H/2}^{+H/2}\sum_{v=-H/2}^{+H/2} h(u,v)I(x-u,y-v)$$

Où H correspond à la dimension du masque de filtrage.

1.4 Filtre séparable

Un filtre à réponse impulsionnelle h (x, y) séparable selon x et y est un filtre pour lequel :

$$h(x,y) = h_x(x).h_y(y)$$

Ce qui se traduit pour l'image filtrée par :

$$I'(x,y) = h(x,y) * I(x,y),$$
$$I'(x,y) = h_y(y) * (h_x(x) * I(x,y))$$

et pour les dérivées par :

$$\frac{\partial I'(x,y)}{\partial x} = I(x,y) * (\frac{\partial h_x(x)}{\partial x} h_y(y)),$$

$$\frac{\partial I'(x,y)}{\partial y} = I(x,y) * (h_x(y)\frac{\partial h_y(x)}{\partial x}),$$

$$\Delta I'(x,y) = I(x,y) * (\Delta h_x(x)h_y(y) + h_x(x)\Delta h_y(y))$$

Les principaux intérêts des filtres séparables sont :

- Ramener le problème du filtrage d'un signal bidimensionnel à celui du filtrage d'un signal monodimensionnel.

- Réduire le temps de calcul. Pour une convolution par un masque de filtrage de dimension , la complexité est de 2H au lieu de H*H
- .Possibilité d'implanter récursivement le filtre.

1.5. Conventions

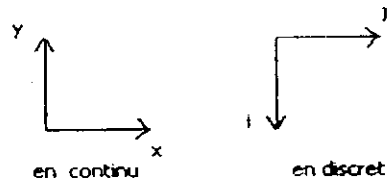


Figure III.3 : Conventions

2. Approche dérivée première [1] [5] [10] [11]

2.1 Opérateur gradient

Cet opérateur permet de caractériser et de repérer les zones de variation de niveaux de gris. La détection de contours revient à déterminer *les extréma locaux dans la direction du gradient.*

L'avantage est que cette méthode est relativement peu bruitée (il est reconnu que l'opération de dérivation est bruitée parce qu'elle amplifie les variations brusques)

2.2 Gradient d'une image

Le gradient d'une image est le vecteur $\nabla I(x,y)$ par :

$$\nabla I(x,y) = \left(\frac{\partial I(x,y)}{\partial x}, \frac{\partial I(x,y)}{\partial y} \right)^t$$

Il est donc caractérisé par un module m, et une direction ϕ dans l'image

$$m = \sqrt{\left(\frac{\partial I(x,y)^2}{\partial x} + \frac{\partial I(x,y)^2}{\partial y}\right)}$$

$$\phi = \arctan\left(\frac{\partial I(x,y)}{\partial y} / \frac{\partial I(x,y)}{\partial x}\right)$$

La direction du gradient maximise la dérivée directionnelle.

2.3 Gradient d'une image filtrée

Le gradient d'une image filtrée

$$\nabla I'(x,y) = \nabla(I(x,y) * h(x,y)) = \nabla I(x,y) * h(x,y) = I(x,y) * \nabla h(x,y)$$

2.4 Approximation par différences finies

En développant au premier ordre, avec une approximation, la dérivée en un point f'(x) peut s'exprimer par :

$$f(x) = \frac{1}{2}[f(x+1) - f(x-1)]$$

Les dérivees peuvent être alors calculées par convolution de l'image avec un masque de différences

Pour une image numérique, on peut définir

- une dérivée partielle df/dx suivant les colonnes par le filtre de matrice suivante

$$\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$ dérivée horizontale

- une dérivée partielle df/dy suivant les lignes de l'image définie par la matrice

$$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$ dérivée verticale

2.5 Opérateurs

2.5.1 Opérateurs de Roberts (1962)

$$h1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad h2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Cet opérateur présente une forte sensibilité au bruit en raison de la taille des masques (nécessité de faire un préfiltrage).

2.5.2 Masques de Prewitt

$$h1 = 1/3\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad h2 = 1/3\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

La convolution de l'image par les masques ci-dessus correspond au calcul des dérivées de l'image préfiltrée par le filtre séparable:

- h1 : dérivation horizontale
- h2 : dérivation verticale

2.5.3 Masques de Sobel (1972)

$$h1 = 1/4\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad h2 = 1/4\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

La dérivation accentue le bruit de l'image, c'est-à-dire les pixels de valeur parasite et de répartition aléatoire. Les opérateurs de Sobel, qui effectuent une moyenne locale sur trois pixels en largeur, sont moins sensibles au bruit :

Cet opérateur est très populaire. Il correspond à la convolution de l'image par : $[1 \quad 2 \quad 1] * [-1 \quad 0 \quad -1]$

Ce sont des masques directionnels également (Nord Est)

$$\begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$$ dérivation oblique

2.5.4. Masques dérivée de Gaussienne

Le filtrage se fait à l'aide d'un masque Gaussien G.

La dérivée en x s'écrit :

$$I_x = \frac{\partial (I * G)}{\partial x} = I * \frac{\partial G}{\partial x}$$

On raisonne de la même manière pour le calcul de la dérivée en y.

$$G(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \text{ soit } \frac{\partial G(x,y)}{\partial x} = -\frac{x}{2\pi\sigma^4}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

Le paramètre permet de régler le degré de lissage.

2.6. Calcul de la norme du gradient

Une fois les dérivées en x et en y calculées, il faut calculer la norme du gradient.

De manière générale, son expression est :

$$\left\|\overrightarrow{grad}\right\| = \left(|I_x|^N + |I_y|^N\right)^{1/N}$$

N=2, il s'agit de la distance euclidienne, mais le temps de calcul est long.

On peut avoir recours aux approximations :

N=1, somme des valeurs absolues, ou N=infini .Ces deux approximations sont utilisables pour réduire le temps de calcul mais amènent à des approximations importantes.

2.7 Détection de contours

Les opérateurs présentés précédemment permettent de calculer la norme du gradient. Pour détecter les contours il faut réaliser un traitement ultérieur.

Ce traitement consiste en un seuillage simple ou par hystérésis. On peut également faire un suivi de contour.

2.7.1. Seuillage simple

Après avoir calculé la norme du gradient il faut la seuiller pour décider si un pixel fait partie ou non d'un contour.

Principe :

On fixe un seuil S, si la norme du pixel est supérieure à S alors il s'agit d'un contour.

Cependant un seuil trop bas aboutit à des sur détections, et un seuil trop élevé implique des contours non fermés.

2.7.2. Seuillage par hystérésis

L'utilisation d'un seuil unique peut mener à de fausses alarmes ou des contours non détectés.

Le seuillage par hystérésis permet d'y remédier.

Principe :

- On introduit 2 seuils (seuil bas et seuil haut) :
- **Si** norme < Seuil bas, **alors** il ne s'agit pas d'un contour
- **Si** norme > Seuil haut, **alors** il s'agit d'un contour (contour « sur »)
- **Si** Seuil bas<norme < Seuil haut, **alors** il s'agit d'un contour hypothétique.

Les contours hypothétiques sont gardés s'ils sont voisins à des contours « surs ».Il s'agit d'une recherche d'adjacence à partir des points extrémités des contours surs. Cette recherche peut se faire soit par suivi soit par étiquetage des voisins connexes.

2.7.3. Suivi de contour, lignes de crêtes

Le seuillage par hystérésis n'élimine pas le problème des contours épais lorsqu'un fort lissage a été effectué. On peut y remédier par la recherche de lignes de crêtes ; un contour est gardé s'il est supérieur à un seuil et maximum local. On aboutit alors à des contours plus fins.

2.7.4. Extraction des maxima locaux

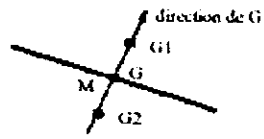


Figure III.4 : Extraction des maxima locaux

Le principe est de comparer le point G contour en un point M avec les gradients G1 et G2 pris dans la direction du gradient. Si G>G1 et G>G2 alors M est un maximum local.

2.8 Approche dérivative précédée d'un filtrage non linéaire

Le filtre de Nagao se prête très bien à une détection de contours : il supprime le bruit sans toucher à la raideur des transitions. Pour détecter les contours il suffit de faire un filtrage de Nagao suivi d'un calcul de dérivée très simple :

$$\begin{cases} G_x = \begin{bmatrix} -1 & 1 \end{bmatrix} \\ G_y = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \end{cases}$$

3. Approche dérivative du second ordre [1][5] [10] [11]

3.1. Laplacien d'une image

Dans ce cas la détection de contours revient à déterminer les *passages par zéro* du laplacien.

Le laplacien d'une image d'intensité $I(x,y)$ est donné par :

$$\nabla^2 I(x,y) = \frac{\partial^2 I(x,y)}{\partial x^2} + \frac{\partial^2 I(x,y)}{\partial y^2}$$

L'opérateur laplacien donne une approximation directe de la somme des dérivées secondes

Il est invariant aux rotations de l'image.

Cependant, la sensibilité au bruit est accrue par rapport au gradient.

3.2. Laplacien d'une image filtrée

Le laplacien d'une image filtrée

$$\Delta I'(x,y) = \Delta I(x,y) * h(x,y) = I(x,y) * \Delta h(x,y)$$

3.3. Différences finies

L'estimation du laplacien d'une image se fait de la même manière par convolution de l'image avec un masque. Le laplacien est approximé par différences finies :

$$\frac{\partial f^2(x,y)}{\partial x} = f'(x,y) = f'(x+1,y) - f'(x,y) = [f(x+1,y) - f(x,y)] - [f(x,y) - f(x-1,y)]$$

$$= f(x+1,y) - 2f(x,y) + f(x-1,y)$$

$$\nabla^2 = [1 \; -2 \; 1]$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{ou} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$ masques d'approximation du laplacien

Le laplacien ne nécessite qu'une convolution, le gradient en nécessite deux.

3.4. Opérateurs de Marr Hildreth

On filtre d'abord l'image *A(x,y)* avec un filtre gaussien *g(x,y)*, puis on calcule la dérivée seconde dans la direction **n** du gradient, et on cherche les passages par zéro dans l'image obtenue *B(x,y)*.

$$B(x,y) = \frac{\partial^2 (A * g(x,y))}{\partial n^2} \quad \ldots\ldots\ldots (1)$$

Pour simplifier on suppose que les passages par zéro de la dérivée seconde correspondent à ceux du laplacien, l'expression (1) devient :

$$B(x,y) = \Delta(A * g(x,y))$$

Le but est de réduire le bruit associé aux filtres différentiels et de faciliter la détection du passage par zéro (contour).

Cette démarche est équivalente à appliquer directement sur l'image l'opérateur laplacien d'une gaussienne $\Delta g(x,y)$ qui peut être approché par la différence de deux gaussienne (opérateur DOG).

En monodimensionnel l'opérateur DOG s'écrit :

$$DOG\ (x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} - \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma_i^2}}$$

En posant $\sigma_i = \sigma + \delta\sigma$, il vient :

$$\sqrt{2\pi}\, DOG\ (x) = \frac{1}{\sigma} e^{-\frac{x^2}{2\sigma^2}} - \frac{1}{\sigma + \delta\sigma} e^{-\frac{x^2}{2(\sigma+\delta\sigma)^2}}$$

Il apparaît que :

$$\sqrt{2\pi}\, DOG\ (x) = \delta\sigma \frac{\partial}{\partial\sigma}\left(\frac{1}{\sigma} e^{-\frac{x^2}{2\sigma^2}}\right) = -\left(\frac{1}{\sigma^2} - \frac{x^2}{\sigma^4}\right) e^{-\frac{x^2}{2\sigma^2}}$$

Or :

$$\left(\frac{1}{\sigma^2} - \frac{x^2}{\sigma^4}\right) e^{-\frac{x^2}{2\sigma^2}} = \frac{\partial g^2(x)}{\partial x^2}$$

DOG(x) approche donc la dérivée seconde de g(x) si σ et σ_i sont voisins.

3.5. Détection des points contours

Les points de contour sont caractérisés par des passages par zéro du laplacien. La détection de ces points s'effectue en deux étapes :

1. Détection des passages par zéros. Les pixels pour lesquels le laplacien change de signe sont sélectionnés.
2. Seuillage des passages par zéros de fortes amplitudes (par hystérésis par exemple).

4. Dérivation par filtrage optimal [10] [11]

Les approches précédentes sont dépendantes de la taille des objets traités, elles sont aussi très sensibles au bruit car la convolution de l'image se fait par des masques de petites dimensions

La dérivation par filtrage optimal repose sur la définition de critères d'optimalité de la détection de contours ; ces critères débouchant sur des filtres de lissage *optimaux*.

4.1. Approche de Canny

On se place dans le cas monodimensionnel.

Les critères d'optimalité portent sur :

1. la détection : le contour doit être détecté, il faut minimiser les fausses réponses,
2. la localisation : le contour doit être localisé avec précision, il s'agit de minimiser la distance entre les points détectés et le vrai contour,
3. la réponse unique : il s'agit de minimiser le nombre de réponse pour un seul contour.

4.1.1. Modèle de contours-détection

On suppose que la détection est effectuée en convoluant le signal par un filtre de réponse impulsionnel, les contours étant caractérisés par les extrema de la sortie du filtre.

Les contours envisagés ici sont des contours de types marche et le bruit est supposé blanc (de moyenne nulle).

$f(x) = A.u_{-1}(x) + \eta(x)$

u_{-1} : Fonction de Heaviside

$$u_{-1}(x) = \begin{cases} 0 & \text{Pour x dans } [-\infty,0] \\ 1 & \text{Pour x dans }]0,+\infty] \end{cases}$$

η (x) : bruit gaussien

$$\eta_0^2 = E[\eta^2(x)]$$

La détection se fait par convolution de

$$\Theta(x_0) = \int_{-\infty}^{+\infty} I(x).f(x_0 - x).dx$$

4.1.2. Détection

Une bonne détection peut être caractérisée par une faible probabilité de ne pas détecter un vrai point de contour et une faible probabilité de marquer de faux points de contours, soit un faible rapport signal sur bruit au point de discontinuité (RSB).

$$\Sigma = \frac{A \int_{-\infty}^{0} f(x).dx}{\eta_0 \sqrt{\int_{-\infty}^{+\infty} f^2(x).dx}}$$

4.1.3. Localisation

Une bonne localisation peut être caractérisée par des contours détectés qui doivent être proches du vrai contour. Ceci revient à maximiser l'écart type de la position des passages par zéro (inverse de l'espérance de la distance (point de contour vrai – détecté).

$$\Lambda = \frac{A|f'(0)|}{\eta_0 \sqrt{\int_{-\infty}^{+\infty} f^2(x).dx}}$$

4.1.4. Réponse unique

Nous ne devons pas aboutir à des réponses multiples.

On utilise alors la distance moyenne entre les passages par zéro de la réponse au bruit gaussien :

$$x_{\max} = 2\pi \left(\frac{\int_{-\infty}^{+\infty} f'^2(x).dx}{\int_{-\infty}^{+\infty} f''^2(x).dx} \right)$$

Ce critère utilise l'hypothèse du bruit gaussien. Il est différent de la détection car il prend en compte le voisinage.

4.1.5. Critère d'optimalité

Il s'agit de maximiser le produit sous la contrainte du troisième critère.

Ceci nous conduit à l'équation différentielle suivante

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

Une solution approchée de cette équation est le filtre gaussien :

$$g(x) = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2.\sigma^2}}$$

4.2. Filtres de Deriche

Deriche a proposé un filtre de lissage dont la dérivée est la solution exacte de l'équation de Canny étendue aux filtres à supports infinis.

$$g(x) = S.x.e^{-\alpha.x}$$

L'implantation récursive

$$y_1(n) = x(n-1) + b_1 y_1(n-1) + b_2 y_2(n-2)$$

$$y_2(n) = x(n+1) + b_1 y_2(n+1) + b_2 y_2(n+2)$$

$$y(n) = c[y_1(n) + y_2(n)]$$

Avec $b_1 = -2.e^{-\alpha}$, $b_2 = e^{-2\alpha}$ et $c = (1-e^{-\alpha})^2$

4.2.1. Filtres dérivés de Dériche

4.2.2.1 Filtre de lissage

$$l(n) = k_0(\alpha|n| + 1)e^{-\alpha|n|}$$

4.2.2.2. Dérivée seconde

$$d2(n) = k_1(1 - \alpha|n|)e^{-\alpha|n|}$$

4.2.3. Passage en deux dimensions

Le gradient en x correspond au lissage sur les colonnes, les dérivées premières sur lignes (séparable).

L'orientation locale d'un contour s'obtient par:

$$I_0(i,j) = arctg\left(\frac{I_y(i,j)}{I_x(i,j)}\right)$$

Pour la norme du gradient on utilise :

$$I_g(i,j) = \sqrt{I_x(i,j)^2 + I_y(i,j)^2}$$

5. Transformation de Hough [1] [9]

Cette méthode a pour but de détecter les droites lorsque l'on a les points de contour. Les segments de droite sont des primitives simples et économiques, dont l'interprétation ultérieure est souvent naturelle. Dans une image, il n'est cependant pas toujours facile de déterminer les pixels qui appartiennent en fait au même segment de droite. La transformée de Hough permet, par une méthode statistique, de délimiter les points approximativement colinéaires.

Principe :

Une droite dans le plan (x, y) a pour équation :

$y = ax + b$ ou $x \sin \phi + y \cos \phi = \rho$ ..... (2)

Si $\Phi \in [0, \Pi]$, les paramètre de la droite (Φ, ρ) sont uniques. (Équation (2)).

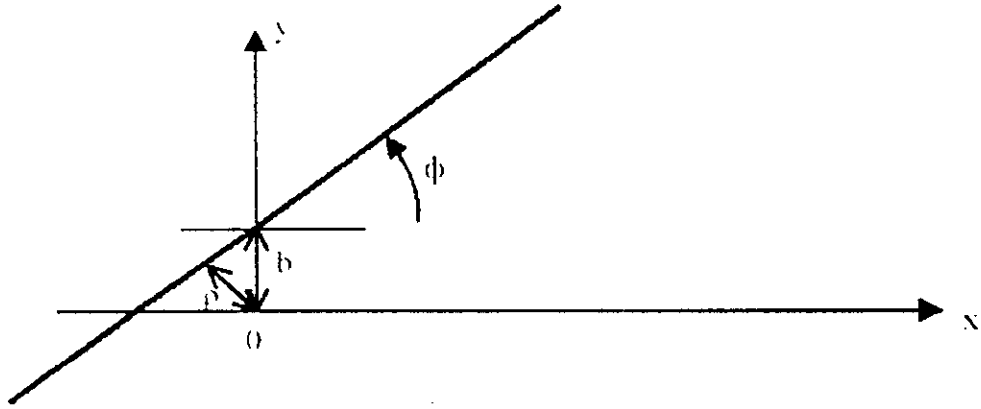


Figure III.5 : représentation d'une image dans le plan (x, y) et correspondance des paramètres

La transformation de Hough est une transformation qui permet de passer du plan image au plan des paramètres. Le plan image est parfaitement défini, le plan des paramètres devra être discrétisé en N*M cellules.

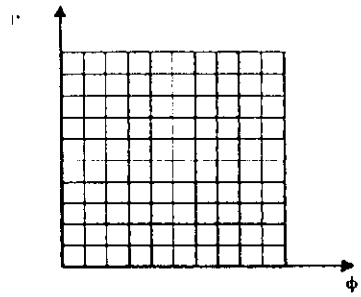


Figure III.6 : discrétisation du plan des paramètres

Pour chaque pixel représentant un point dans l'image discrète on reporte, dans le plan dual la droite correspondante, également discrétisée: si une cellule (Φ, ρ) est touchée plusieurs fois on compte le nombre de droites auxquelles elle appartient. Les fréquences ainsi cumulées représentent un histogramme bidimensionnel, dont on extraira les modes (maxima locaux). A chaque mode dans le plan dual discrétisé correspond une droite, contenant une concentration de points, dans l'image.

L'avantage de la méthode de Hough est de prendre en considération l'image globale. Ses inconvénients sont un manque de précision, qui en limite l'emploi à une première approximation, ainsi qu'un coût important en calcul.

Pour chaque point de contour on peut faire passer en théorie une infinité de droites. Dans ce cas comme les orientations sont discrétisées on ne peut faire passer que M droites.

Transformée de Hough généralisée.

Une amélioration peut être apportée si l'on connaît en plus, l'orientation des points de contour (image binaire). Pour cela, il suffit d'utiliser l'orientation du gradient : pour chaque point de contour, on calcule ρ uniquement pour l'orientation perpendiculaire à la direction du gradient. L'accumulation dans l'espace des paramètres est simplifiée.

Pour coder les contours on peut utiliser le codage de Freeman.

On peut terminer sur un critère de qualité, caractérisant la propriété commune aux points du segment avec une tolérance (par exemple colinéarité, valeur de gradient -ou du pixel proche d'un maximum, etc.).

Le résultat escompté est une liste d'éléments géométriques primitifs segments de droite et, plus généralement, arcs de courbe ou régions uniformes constituant une image. Chaque élément primitif sera caractérisé par les coordonnées de ses extrémités ainsi qu'une série d'attributs intrinsèques (couleur moyenne, pente, taille) et éventuellement de relations avec d'autres primitives (voisins directs, angles, distances). Une image ainsi segmentée est plus adaptée à une analyse structurelle de son contenu que ne l'était un tableau de pixels.

Conclusion

Il existe différentes possibilités d'extraction des contours. Dans ce chapitre nous avons présenté certaines d'entre elles. Les approches dérivatives sont les plus immédiates pour détecter et localiser les variations du signal. Il peut s'agir d'approches dérivatives du premier ou du second ordre (approximation par le laplacien).Les détecteurs optimaux reposent sur la définition d'un modèle de contours et de critères d'optimalité (type marche dans notre cas). La transformation de Hough permet d'aboutir à une représentation de l'image plus facile à interpréter.

Tous les opérateurs cités vont être implémentés, ceci fera l'objet du prochain chapitre.

Introduction

Le but de ce travail est de présenter différentes méthodes permettant d'extraire les contours d'une image. Ces différentes méthodes seront ensuite testées sur plusieurs images. Cependant, comme cela a été spécifié précédemment, nous ne pouvons pas réellement isoler cette étape dans la chaîne de traitement d'images.

Nous avons donc inclus dans ce travail un ensemble d'outils généraux et de fonctions permettant le prétraitement (étape qui précède la détection de contours dans l'image), avant d'aborder les détecteurs de contours proprement dits.

Une autre difficulté à noter , est le fait que nous ne disposons pas d'images 'précises' (provenant de la même source par exemple), ou d'une application donnée.

L'image est abordée ici comme une matrice dont les coefficients sont les valeurs des pixels en niveaux de gris (codés sur 8 bits).

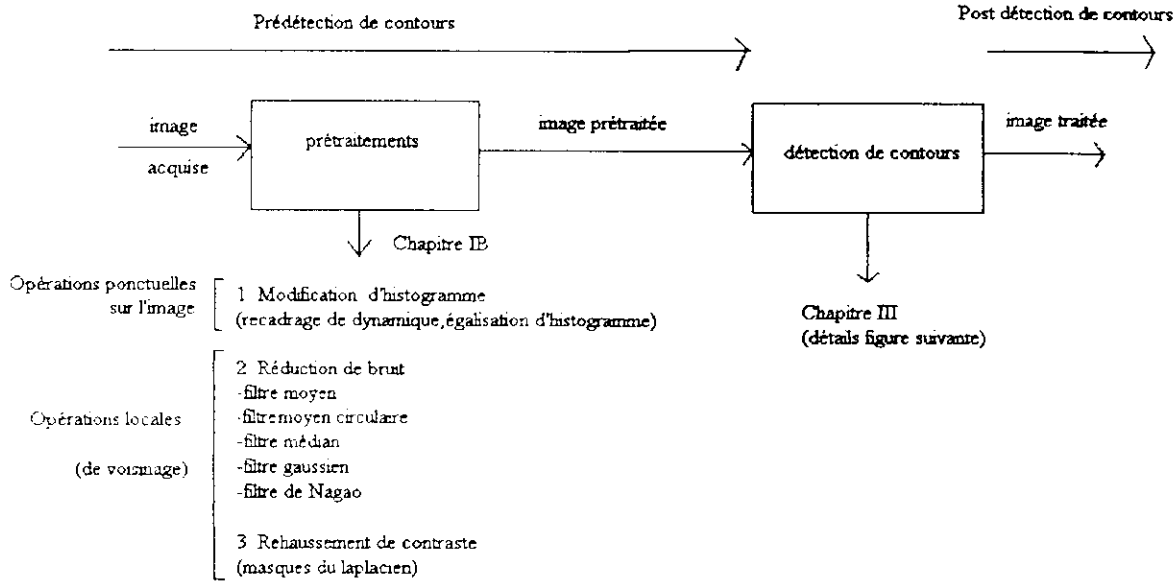


Figure VI.1 : Synoptique du travail effectué

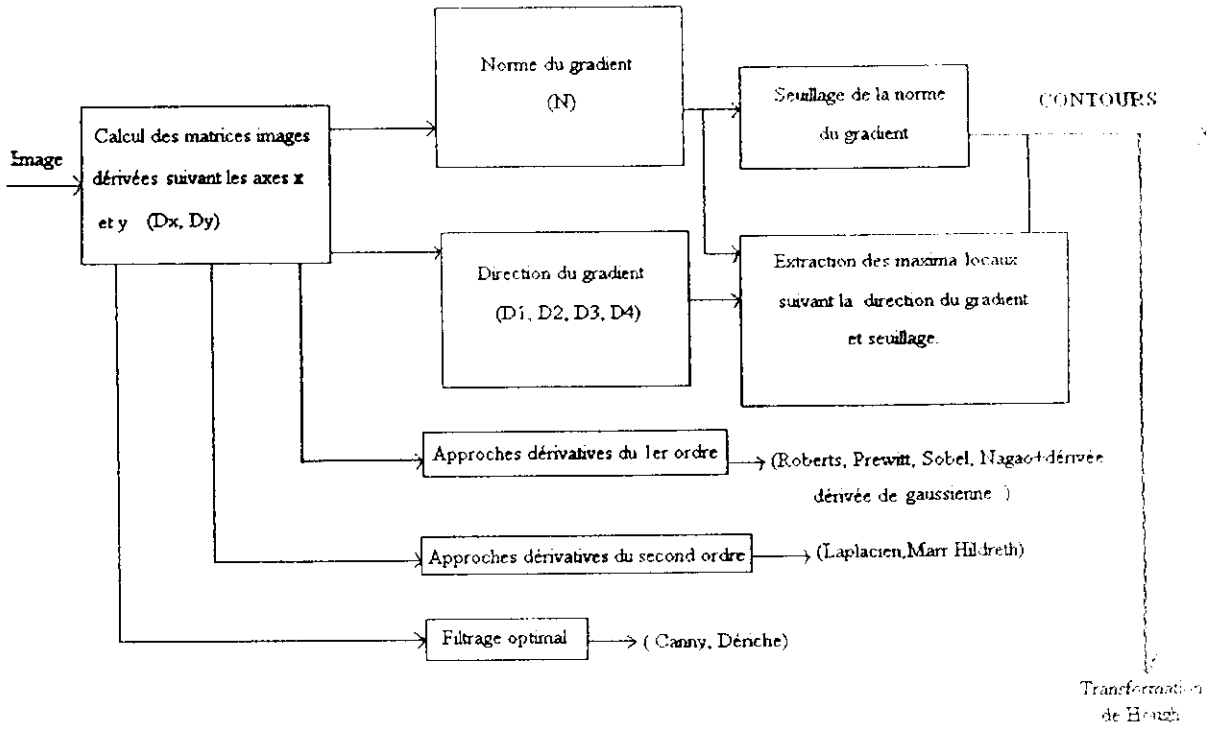


Figure VI.2 : Détection de contours

Nous avons donc créé nos propres fonctions (précisées dans les tableaux), certaines font appel à des fonctions matlab que nous avons spécifié (FMA : Fonctions Matlab Auxiliaires). L'utilisation de ces fonctions ne doit pas apporter un aspect réducteur, il s'agit de simplifier certains aspects du traitement vu les conditions citées précédemment.

Version utilisée : Matlab 5.3.

Remarque : la description des fonctions matlab (en gras dans le texte) de la toolbox image utilisées est jointe en annexe

1. Lecture du fichier graphique

1.1. Informations générales

La commande **imfinfo** de **matlab** nous fournit tous les renseignements dont nous pouvons avoir besoin pour traiter l'image (format de fichier, taille, profondeur de codage de l'image...)

1.2. Lecture de l'image

Nous travaillerons sur des images de type matriciel. Pour lire l'image on fait appel à la fonction 'lecture_image (u)'. Cette fonction fait directement appel à la fonction de la toolbox **imread (voir annexe pour les propriétés de cette fonction)**

Description

[I]=lecture_image (u) :

1. u désigne le nom du fichier et son extension.

2. I est la matrice image sur 256 niveaux de gris

2. Affichage de l'image

2.1. Fonctions matlab utilisées

Pour afficher l'image sur 256 niveaux de gris nous utilisons la fonction **Imagesc** (cf. annexe 3 fonctions matlab) ainsi que les palettes créées (fonction 'palette_nvg256', fonction 'palette_nvg256inv' voir ci après).

2.2. Palette des couleurs

Pour afficher nos résultats sur 256 niveaux de gris nous créons une palette de 256 niveaux de gris (fonction 'palette_nvg256').

Nous créons également une palette de 256 niveaux de gris inversés (fonction' palette_nvg256inv'), car les résultats sur certaines images traitées ont une meilleure apparence visuelle avec cet affichage.

3. Traitement

Nous considérons que les traitements se font sur des *images en niveaux de gris* (maximum 256). Les méthodes exposées peuvent cependant être étendues à d'autres types d'images).

4. Outils généraux

Dans cette partie nous présentons des fonctions d'utilité générale, pouvant être utilisées aussi bien au niveau des prétraitements que pour la détection de contours.
Il s'agit des fonctions permettant :

- L'affichage de l'histogramme normalisé de l'image,
- L'affichage de l'histogramme cumulé normalisé de l'image,

Ces fonctions constituent de bons indicateurs pour le choix des transformations à opérer sur l'image.

- Palette 256 niveaux de gris,
- Palette 256 niveaux de gris inversés.

Fonctions	Entrée(s)	Sortie(s)	Description
1. hist1 (I)	I : (matrice image sur 256 nvg)	X	Affiche l'histogramme *normalisé* de I
2. hist_cum1 (I)	I : (matrice image sur 256 nvg)	X	Affiche l'histogramme cumulé *normalisé* de I
3. [map]=palette_nvg256	X	Map : palette	Palette 256 niveaux de gris
4. [map]=palette_nvg256inv	X	Map : palette inversée	Palette 256 niveaux de gris inversés

Tableau IV.1 : Outils généraux

5. Prétraitement

Dans cette partie nous présentons les fonctions permettant la réalisation des opérations de prétraitement décrites dans la partie B du chapitre I.
Les prétraitements agissent sur l'image acquise, et ont pour but d'améliorer la qualité de l'image. Le choix de l'opérateur est soumis à l'image et aux moyens dont

nous disposons ainsi que du but à atteindre. Les effets de chacun d'entre eux ont été expliqués chapitre I.B.

5.1. Modification d'histogramme

5.1.1. Recadrage de dynamique

Fonction 'rec_dyn1 (I)', cette fonction affiche l'histogramme recadré, l'image initiale et l'image recadrée

5.1.2. Égalisation d'histogramme

Fonction 'egal_hist (I)', cette fonction affiche l'histogramme égalisé et l'image égalisée.

5.2. Opérateurs de pré traitement

5.2.1. Réduction de bruit par filtrage

5.2.1.1 Rajout de bruit à une image

Pour rajouter du bruit à l'image on fait appel directement à la fonction **imnoise** de matlab (cf. annexe 3 fonctions matlab).

On peut rajouter du bruit des types :

1. Gaussien
2. Poivre et sel
3. Speckle

5.2.1.2. Gestion des bords

L'application de filtres sur l'image, nous amène au problème de la gestion des bords, c'est-à-dire des pixels du bord de l'image : doivent ils être traités ou ignorés ?

Sous matlab cette gestion peut se faire de différentes manières :

-soit les pixels de bords sont ignorés, ce qui nous conduit à une image traitée de taille inférieure à celle de l'image initiale.

-soit ces pixels sont pris en compte, ceci est réalisé par le 'zero padding', il s'agit de rajouter des zéros sur le pourtour de l'image (cf. annexe 3 fonctions matlab) de telle sorte que l'on puisse traiter les pixels du bord de l'image avec le masque.

Cette opération peut aboutir à deux résultats en fonction des traitements recherchés :

-Option 'same' : les pixels du bord sont calculés avec le 'zero padding', l'image de sortie a les mêmes dimensions que l'image d'entrée.

-Option 'full'

L'utilisation de la fonction **filter2** (cf. annexe 3 fonctions matlab) dans nos fonctions se fait avec l'option 'same'.

Cette fonctionnalité peut trouver son intérêt lorsque la taille du masque augmente ou dans des applications particulières telles que la restauration.

Les fonctions dont nous disposons sont :

- **Filtre moyen** : la fonction 'filtre_moy1' permet le filtrage de l'image I par un filtre moyen de taille n*n, le nombre n doit être impair de préférence
- **Filtre moyen circulaire :** la fonction 'filtre_moycir1' permet d'opérer un filtrage moyen circulaire sur l'image (taille du filtre 7*7).
- **Filtre médian :**

La fonction 'filtre_med1 (I, n)' permet d'opérer un filtrage médian sur l'image I la taille du masque est de n (n=3 ,5 ou 7).

- **Filtre gaussien**

Pour filtrer l'image à l'aide d'un filtre gaussien, on utilise la fonction 'filtre_gauss (I, t, sig)'.

Paramètres d'entrée :

- I : matrice image
- t : t*t est la taille du masque, t doit être impair
- sig : sigma résolution du filtre.

- **Filtre de Nagao**

Pour filtrer l'image à l'aide des filtres de Nagao on fait appel à la fonction 'filtre_nagao (I)' : cette fonction renvoie l'image filtrée ainsi qu'une matrice index donnant le masque de Nagao correspondant pour chaque pixel (valeurs de 1 à 9 correspondant aux filtres D_0 à D_8 respectivement tels que définis chapitre IB)

5.3. Rehaussement de contraste

Pour rehausser le contraste de l'image, on peut utiliser la fonction 'reh_contrast'

Cette fonction est basée sur la formule Is=I-K*lap (I) où :

- Is désigne la matrice image de sortie,

- I la matrice image d'entrée,
- lap : le laplacien de l'image I et
- k le paramètre permettant de régler la raideur de la transition

Cette fonction renvoie 2 matrices de sortie

- Is1 correspond à un rehaussement de contraste opéré à l'aide du masque lap1

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

- Is2 correspond à un rehaussement de contraste opéré à l'aide du masque lap2

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

5.4. Opérateurs morphologiques

5.4.1. Dilatation

On fait appel à la fonction 'dilat_ibin (I_bin)', l'image d'entrée doit être binaire. Cette fonction donne en sortie 2 images dilatées :

- Is_dil1 : image dilatée en utilisant la 4-connexité
- Is_dil2: image dilatée en utilisant la 8-connexité

I_bin doit contenir des 1 et des 0

5.4.2. Erosion

On fait appel à la fonction 'eros_ibin (I_bin)', l'image d'entrée doit être binaire. Cette fonction donne en sortie 2 images érodées :

- Is_eros1 : image érodée en utilisant la 4-connexité
- Is_eros2: image érodée en utilisant la 8-connexité

I_bin doit contenir des 1 et des 0

Ces deux fonctions interviennent pour la détermination des contours d'une image binaire

Fonctions	Entrée(s)	Sortie(s)	FMA	Description
1. rec_dyn1 (I)	I : (matrice image sur 256 nvg)	X	X	Recadrage de dynamique
2. egal_his1 (I)	I : (matrice image sur 256 nvg)	X	X	Egalisation d'histogramme
3.[Is]=filtre_moy1(I,n).	I : (matrice image) n : taille du filtre	Is : (matrice image filtrée)	Filter2	Filtrage Moyenneur d l'image
4. [Is]=filtre_moycir1 (I).	I : matrice image entrée	Is : matrice image filtrée	Filter2	Filtrage Moyenneur circulaire de l'image
5.[Is]=filtre_med1(I,n)	I : matrice image entrée n : taille du filtre	Is : matrice image filtrée	ordfilt	Filtre médian
6.[Is1,index]=filtre_nagao(I)	I	Is1 index	Filter2 Ordfilt	Filtres de Nagao*
7 [Is, h_gauss]=filtre_gauss (I, t, sig)	I, t, sig	Is, h_gauss	Filter2	Filtre gaussien*
8. [Is1, Is2]=reh_contrast (I, k)	I : matrice image entrée K : facteur de rehaussement	Is1 Is2*	Filter2	Rehaussement de contraste
9.[J]=bin_im1(I,s)	I : matrice image entrée S=seuil	J : matrice image binarisée	X	Binarisation manuelle d'une image e son affichage
10.[Is_dil1,Is_dil2]=dilat_ibin(I_bin)*	I_bin : matrice	Is_dil1	Conv2	Dilatation

	image binarisée (0 et 1)	Is_dil2		d'une image binaire
11.[Is_eros1,Is_eros2]=eros_ibin(I_bin)	I_bin(0 et 1)	Is_eros1, Is_eros2	Conv2	Erosion d'une image binaire
12.[Is_cont1,Is_cont2]=cont_dil_ibin(I_bin)	I_bin(0 et 1)	Is_cont1 Is_cont2	Conv2	Contours d'une image binaire à partir de l'image dilatée*
13. .[Is_cont1,Is_cont2]=cont_eros_ibin(I_bin)	I_bin(0 et 1)	Is_cont1 Is_cont2	Conv2	Contours d'une image binaire à partir de l'image érodée*

Tableau IV.2 : Prétraitement

6. Détection de contours

6.1 Approche dérivée première

Les masques de Roberts, Prewitt, Sobel sont définis par les matrices correspondantes puis implémentés à l'aide de la fonction filter2.

Sorties pour chaque filtre

Is1 : matrice image filtre par h1

Is2 : matrice image filtre par h2

N : norme euclidienne du gradient

6.1.1. Roberts

h1= [1 0; 0 -1];

h2= [0 1;-1 0];

6.1.2. Prewitt

h1= (1/3)*[-1 0 1;-1 0 1;-1 0 1];

h2= (1/3)*[-1 -1 -1; 0 0 0; 1 1 1];

6.1.3. Sobel

h1 = (1/4)*[-1 0 1;-2 0 2;-1 0 1];

h2 = (1/4)*[-1 -2 -1; 0 0 0; 1 2 1];

6.1.4. Dérivée de Gaussien

Fonctions	Entrée(s)	Sortie(s)	FMA	Description
1.[Is1,Is2,N]=det_rob(I)*	I	Is1 Is2 N	Filter2	Détecteurs de Roberts
2.[Is1,Is2,N]=det_prew(I)*	I	Is1 Is2 N	Filter2	Détecteurs de Prewitt
3.[Is1,Is2,N]=det_sob(I)*	I	Is1 Is2 N	Filter2	Détecteurs de Sobel
4. [Is]=deriv_gauss (I, t, sig)	I, t, sig	Is	Filter2	Filtre dérivée de gauss
5.[Is]=filtre_nagao_deriv(I)	I	Is	Filter2	Filtrage de Nagao suivi d'une dérivation

Tableau IV.3 : Approche dérivée première

6.2. Approche dérivée seconde

6.2.1. Laplacien

La fonction 'lap (I)' renvoie deux matrices de l'image d'entrée I :

Is1 calculée à l'aide du masque lap1

Is2 calculée à l'aide du masque lap2

6.2.2. Détecteur de Marr Hildreth

La fonction 'filtre_marrhildreth (I,t,sig)' filtre l'image à l'aide du filtre de Marr Hildreth.
- I : matrice image d'entrée,
- t : détermine la taille du filtre ((2*t+1)* (2*t+1)), t doit être impair,
-sig : résolution du filtre.

6.2.3. Laplacien de gaussienne

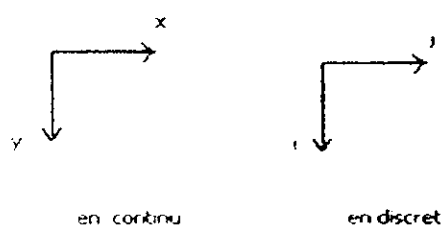
La fonction 'lap_de_gaussienne (I, t, sig)' ' filtre l'image à l'aide du laplacien d'une gaussienne
- I : matrice image d'entrée,
- t : détermine la taille du filtre ((2*t+1)* (2*t+1)), t doit être impair,
-sig : résolution du filtre.

Fonctions	Entrée(s)	Sortie(s)	FMA	Description
1. [Is1, Is2]=lap (I)*	I	Is1 Is2	Filter2	Laplacien
2[Is]=filtre_marrhildreth(I,t,sig).	I, t, sig,	Is	Filter2	Filtre de Marr Hildreth
3.[Is]=lap_de_gaussienne (I, t, sig)	I, t, sig,	Is	Filter2	Filtre laplacien de gaussienne

Tableau IV.4 : Approche dérivée seconde

6.3. Détecteurs optimaux

6.3.1. Conventions sous matlab



6.3.2. Réalisation du détecteur de Canny

Canny a montré que l'opérateur optimal pour détecter un contour bruité est la dérivée première directionnelle d'une fonction gaussienne de résolutionσ. Le contour est défini comme le maximum local de l'opérateur G_n convolué avec l'image I.

$$G_n = \frac{\partial}{\partial n} G_\sigma$$

Où n est la direction du gradient. L'image est donc convoluée avec l'opérateur G_n ou de façon équivalente puisque les opérations de dérivation et de convolution sont linéaires :

$$I * G_n = I * \frac{\partial}{\partial n} G_\sigma = \frac{\partial}{\partial n} (I * G_\sigma)$$

6.3.2.1. Etapes de la réalisation d'un détecteur de Canny

L'image peut d'abord être filtrée par le filtre passe-bas gaussien, puis la dérivée directionnelle est calculée en chaque pixel de l'image.

Les étapes pour la réalisation d'un détecteur de Canny sont alors :

1. Convolution de l'image avec un filtre passe bas gaussien $I * G_\sigma$
2. Calcul du gradient selon l'axe horizontal et vertical $\nabla (I * G_\sigma) = D_x, D_y$
3. Calcul de l'amplitude du gradient $|\nabla (I * G_\sigma)| = \sqrt{D_x^2 + D_y^2}$
4. Un pixel est activé (indique la présence d'un contour) si son amplitude de gradient est maximum dans la direction du gradient. La valeur de l'amplitude du pixel doit donc être la plus grande par rapport aux deux valeurs d'amplitude (obtenues par interpolation linéaire) de part et d'autre du pixel dans la direction du gradient. Cette opération de sélection a pour effet d'amincir considérablement la largeur détectée des contours.
5. Sélection des contours significatifs. Une opération de seuillage avec hystérésis est appliquée sur l'image de contour obtenue à l'étape précédente.

Les pixels de l'image de contour qui ont été activés à l'étape précédente :

- sont conservés si leur valeur d'amplitude de gradient est supérieure à la valeur supérieure de seuil
- ne sont pas conservés (mis à zéro) si l'amplitude est inférieure à la valeur inférieure de seuil,
- sont conservés si la valeur d'amplitude est comprise entre les deux valeurs de seuil et qu'un des pixels de contours voisins a été conservé parce que sa valeur d'amplitude dépassait la valeur supérieure du seuil. C'est un processus itératif qui

évolue de telle sorte que les pixels dont l'amplitude de gradient est située entre les deux valeurs de seuil sont convertis en pixels de contour s'il existe un parcours pouvant les relier à un pixel de contour dont l'amplitude est supérieure au seuil supérieur.

Lors de notre implémentation, nous n'avons pas réalisé le suivi de contour, le seuillage se limite à un seuillage simple, mais l'index (b) des pixels susceptibles de subir un suivi de contour est disponible dans le programme (valeurs 1 de la matrice b dans le fichier 'detect_canny1.m')

6.3.2.2. Implémentation sous matlab

Etapes 1,2

Ces deux étapes sont réalisées en même temps.

Les deux opérations :

- Filtrage à l'aide d'un filtre passe bas (gaussien),
- Dérivation (filtre dérivée de gaussienne),

sont combinées en une seule opération (les 2 opérations sont réalisées par un filtre convolution des deux précédents, h= gauss*dgauss).

L'opération s'effectue sur un signal monodimensionnel, on effectue donc le traitement suivant les lignes, puis les colonnes de l'image, pour revenir à un signal bidimensionnel.

Etape 3

Le traitement précédent nous permet d'obtenir les matrices images dérivées suivant les directions x et y. A partir de ces matrices il nous est possible de calculer la norme du gradient (ainsi que sa direction).

$$Grad = \sqrt{D_x^2 + D_y^2}$$

Notre fonction nous permet d'afficher les deux résultats suivants :

- Les contours de l'image sans extraction des maxima dans la direction du gradient (seuillage de la norme du gradient calculée)
- Les contours de l'image après extraction des maxima locaux et seuillage (étapes 4 et 5)

Etape 4 : extraction des maxima locaux dans la direction du gradient

Cette étape nécessite d'abord la définition des directions adoptées, 4 directions sont choisies.

La direction du gradient à partir des images dérivées est obtenue en calculant la tangente en chaque pixel.

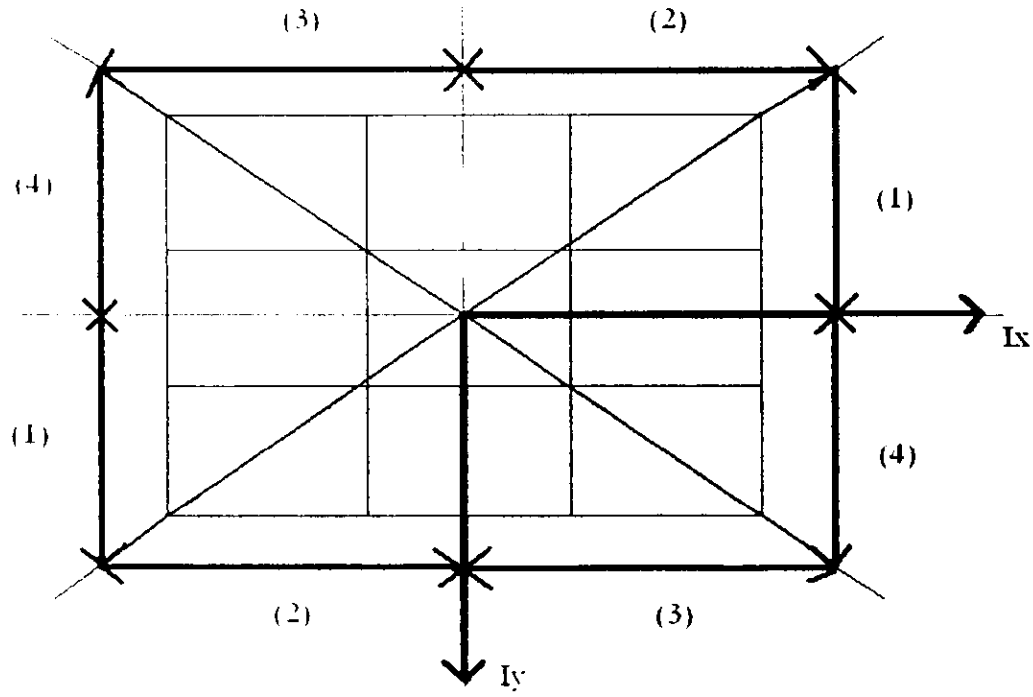


Figure VI.3 : Définition des directions du gradient

Les directions sont définies par:

Direction 1

Iy<=0 et Ix>0 et |Iy| <|Ix| ou Iy>=0 et Ix<0 et |Iy|<= |Ix|

Direction 2

Iy<0 et Ix>=0 et |Iy| >=|Ix| ou Iy>0 et Ix<=0 et |Iy| >=|Ix|

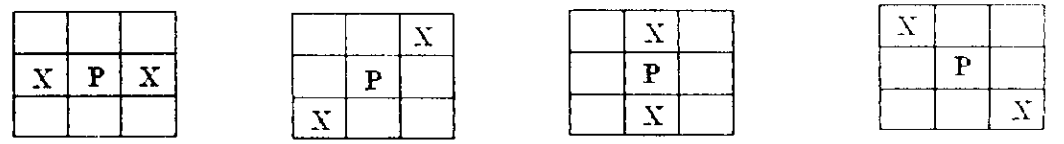
Direction 3

Iy<=0 et Ix<0 et |Iy|> |Ix| ou Iy>0 et Ix<=0 et |Iy|>= |Ix|

Direction 4

Iy<=0 et Ix<0 et |Iy| <|Ix| ou Iy>=0 et Ix>0 et |Iy|< |Ix|

Pour déterminer la direction du gradient 4 configurations sont possibles.



Configuration 1 **Configuration 2** **Configuration 3** **Configuration 4**

Figure VI.4 : Configuration possibles de la direction du gradient

Ces configurations sont obtenues par interpolation linéaire

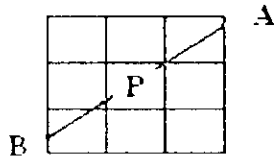


Figure VI.5 : Illustration configuration2

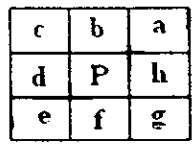


Figure VI.6 : Définition du voisinage du pixel central

Direction 1

$Grad_{A(h,a)}$= CL (grad(h), grad(a))

$Grad_{B(d,e)}$= CL (grad(d), grad(e))

Direction 2

$Grad_{A(a,b)}$= CL (grad(a), grad(b))

$Grad_{B(e,f)}$= CL (grad(e), grad(f))

Direction 3

$Grad_{A(b,c)}$= CL (grad(b), grad(c))

$Grad_{B(f,g)}$= CL (grad(f), grad(g))

Direction 4

$Grad_{A(c,d)}$= CL (grad(c), grad(d))

$Grad_{B(g,h)}$= CL (grad(g), grad(h))

CL: Combinaison Linéaire

La valeur de Grad (P) est comparée aux valeurs de $Grad_A$ et $Grad_B$

Il s'agit d'un maximum local si Grad (P) >= GradA et Grad (P) > $Grad_B$

Etape 5 seuillage.

Cette étape correspond à un seuillage simple des maxima locaux extraits (valeurs > sh introduit par l'utilisateur). Notre fonction renvoie également une matrice index contenant les pixels candidats à un suivi de contours ultérieur.

6.3.2.3. Fonction

detect_canny1(I,sig,sb,sh)

1. Entrées:

I : matrice image d'entrée

Sig : résolution des filtres de gauss et dérivée de gauss

sb, sh : seuil bas et seuil haut pour l'opération de seuillage par hystérésis

2. sorties :

Is: matrice image de sortie avec extraction des maxima locaux

N : norme euclidienne du gradient

ix : matrice gradient suivant x

iy : matrice gradient suivant y

direction : calcul des directions du gradient suivant les conventions adoptées.

6.3.3. Dériche

6.3.3.1. Présentation

Dériche a également proposé un détecteur optimal. Ce filtre est de type RII, implémenté de manière récursive. Le filtre est également séparable.

Il s'agit d'effectuer d'abord un lissage suivant les lignes (respectivement les colonnes) puis une dérivation suivant les colonnes (respectivement les lignes)..On obtient alors la matrice dérivée suivant y (respectivement x) pour la convention matlab.

L'étape précédente nous permet d'obtenir les matrices dérivées on peut alors calculer le gradient de l'image.

La démarche à suivre en ensuite est la même que celle adoptée pour le détecteur de Canny :

- Calcul de la norme du gradient
- Affichage de l'image suivant deux raisonnements possibles :

1. Sans extraction des maxima locaux de la norme du gradient (après seuillage simple de la norme du gradient).
2. Avec extraction des maxima locaux dans la direction du gradient. La définition des directions et les calculs par interpolation sont les mêmes que ceux effectués pour le détecteur de Canny.

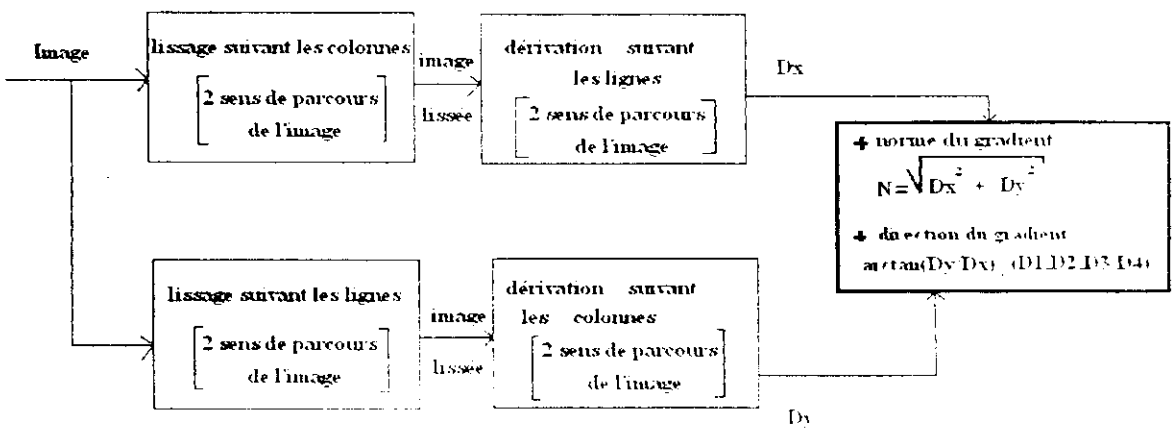


Figure VI.7 : Détecteur de Canny

6.3.3.2. Algorithmes

D'une manière générale (en considérant le signal monodimensionnel), on a :

1. Lisseur de Deriche

$B_1[i]=b^* A[i] +b^*\exp(-\alpha) *(\alpha -1)^*A[i-1]+2^* \exp(-\alpha)^* B_1[i-1]-; \exp(-2\alpha)^* B_1[i-2];$

pour i=1,….,M

$B_1[i]=b^*\exp(-\alpha) *(\alpha +1)^*A[i+1]-b^* \exp(-2\alpha)^*A[i+2]+2^* \exp(-\alpha)^* B_2[i+1]- \exp(-2\alpha)^* B_2[i+2];$

pour i=M,…..,1

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

$b=-(1-\exp(-\alpha))^2/ (1+2^* \alpha *\exp(-\alpha)- \exp(-2\alpha))$

2. Dérivateur de Deriche

$B_1[i]=c^*\exp(-\alpha) A[i-1] +2^*\exp(-\alpha) B_1[i-1]- \exp(-2\alpha)^* B_1[i-2]$; pour i=1,….,M

$B_2[i]=-c^*\exp(-\alpha) A[i+1] +2^*\exp(-\alpha) B_2[i+1]- \exp(-2\alpha)^* B_2[i+2]$; pour i=M,…..,1

$c=-(1-\exp(-\alpha))^2/ \exp(-\alpha)$

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

3. Opérateur bidimensionnel

$H_x(x, y)=-b^* \alpha^2*x*\exp. (-\alpha|x|)*b*(\alpha|y| +1)^* \exp. (-\alpha|y|)$

$H_y(x, y)=-b^* \alpha^2*y*\exp. (-\alpha|y|)*b*(\alpha|x| +1)^* \exp. (-\alpha|x|)$

Dérivée directionnelle suivant la direction des x

$B_i[i,j]=[A^*f[j]]^*h[i]$

Dérivée directionnelle suivant la direction des y

$B_j[i,j]=[A^*f[i]]^*h[j]$

Pour l'image, il s'agira d'effectuer l'incrémentation/décrémentation suivant les indices i, puis j et inversement

6.3.3.3. Fonction 'detect_deriche (I, alpha, sb, sh)'

1. Entrées

I: : matrice image d'entrée

Alpha : permet de régler la résolution du filtre

sb, sh: valeurs respectives des seuils bas et haut pour le seuillage par hystérésis.

2. Sorties

Is : matrice image de sortie avec extraction des maxima locaux

N : norme euclidienne du gradient

Ix : lissage suivant y, dérivée suivant x

Iy : lissage suivant x, dérivée suivant y

Direction (1, 2, 3, 4): calcul des directions du gradient suivant les conventions adoptées. (cf. Figure IV.4)

Fonctions	Entrée(s)	Sortie(s)	FMA	Description
1.[Is,N,ix,iy,direction]=detect_canny1(I,sig,sb,sh)	I, sig, sb, sh	Is, N, ix, iy, direction	Conv Conv2	Detecteur optimal de Canny**
2.[Is,N,ix,iy,direction]=detect_deriche(I,alpha,sb,sh)	I alpha, sb, sh	Is N, ix, iy, direction	X	Detecteur optimal de Deriche**

Tableau IV.5 : Détecteurs optimaux

7. Extraction des contours

Seuillage simple

Fonction 'seuil_simp (I, s)', où I est l'image d'entrée et s le seuil (les valeurs supérieures ou égales à s sont mises à 0 les autres à 1).

Seuillage par hystérésis

Fonction 'seuil_hys (I,sb,sh)' , ou I est l'image d'entrée et sb le seuil bas, sh est le seuil haut.

Fonctions	Entrée(s)	Sortie(s)	FMA	Description
[Is]=seuil_simp(I,s)	I:matrice image S: seuil manuel	Is: matrice image seuillée	X	Seuillage simple
[Is]=seuil_hys(I,sb,sh)	I: matrice image Sb: seuil bas Sh : seuil haut	Is : matrice image seuillée	X	Seuillage par hystérésis

Tableau IV.6 : Extraction de contours

8. Calcul des normes

I1 et I2 sont des matrices contenant les normes des projections du gradient (directions perpendiculaires).

Fonctions	Entrée(s)	Sortie(s)	FMA	Description
1.[N_euc]=Dist_euc(I1,I2)	I1 I2	N_euc	X	Distance euclidienne
2. [nor_somva]=norm_somva(I1,I2)	I1 I2	nor_somva	X	Somme des valeurs absolues
3.[nor_max]=norm_max(I1,I2)	I1 I2	Nor_max	X	Maximum

Tableau IV.7 : Calcul des normes du gradient

9. Affichage des contours

Pour avoir une meilleure appréciation visuelle, nous afficherons les contours de l'image traitée en couleur, sur l'image initiale : fonction 'affichage_des_contours (I, N, s)' où :

I est l'image originale,

N : matrice norme gradient

S : seuil (seuillage de la norme du gradient)

10. Critères de qualité et mesure de distorsions dans une image : Calcul d'erreur

En plus de l'analyse visuelle nous rajoutons deux fonctions permettant une analyse quantitative .Il s'agit du MSE et du PSNR définis comme suit :

Soient une image originale f de taille NxM et f' l'image traitée.

- La **métrique de distorsion d'erreur quadratique moyenne** (Mean Square Error ou **MSE**) est :

$$MSE = \frac{1}{M * N} \cdot \sum_{j=0}^{N-1} \sum_{k=0}^{M-1} (f(j,k) - f'(j,k))^2$$

- Si l'on considère une image de dynamique [0,255] on définit le **PSNR** (Peak Signal to Noise Ratio).

$$PSNR = \frac{M * N * 255}{\sum_{j=0}^{N-1} \sum_{k=0}^{M-1} (f(j,k) - f'(j,k))^2}$$

Fonctions	Entrées	Sorties	Description
[err_mse]=calcul_MSE(I1,I2)	I1:image référence I2:image comparée	Err : erreur	Calcul du MSE
[err_psnr]=calcul_PSNR(I1,I2)	I1:image référence I2:image comparée	Err : erreur	Calcul du PSNR

Tableau IV.8 : Calcul d'erreur

11. Transformation de Hough

Il s'agit en quelque sorte d'une 'application' des contours puisque cette transformation opère sur les contours de l'image.

Pour opérer cette transformation on fait appel à la fonction 'transf_hough (I)', où I est l'image contours.

Conclusion

Dans ce chapitre nous avons présenté les différentes fonctions créées afin de pouvoir opérer la détection de contours.
Il s'agit d'un ensemble d'outils qui nous permettent d'opérer des traitements sur l'image. D'une part nous avons des fonctions nous permettant d'améliorer la qualité de l'image étudiée afin de la préparer à la détection de contours. D'autre part, nous disposons des détecteurs de contours et des fonctions qui leur sont associées afin d'effectuer leur extraction. Toutes ces fonctions ne sont pas indépendantes les unes des autres, pour pouvoir utiliser ces outils toutes les fonctions créées doivent etre disponibles dans l'espace de travail.
Ces fonctions ont été testées sur différentes images, les résultats sont présentés dans le chapitre qui va suivre.

Introduction

Dans ce chapitre nous exposons différents traitements que nous appliquons à plusieurs images. Ces traitements sont effectués à partir des fonctions présentées dans le chapitre précédent.

Le but de notre travail est la détection de contours. Nous allons donc effectuer cette détection sur les images dont nous disposons à l'aide des outils créés.

Pour chaque image nous créons un programme (faisant appel aux fonctions créées) .tous les programmes sont joints en annexe (Annexe 4).

Ce chapitre s'organise de la manière suivante :

- Présentation des images traitées,
- Traitements réalisés,

	Prétraitement	Images/fichiers
Réduction de bruit (poivre et sel)	- Filtre moyen (5*5) - Filtre médian (5*5) - Filtre Nagao : 1 application 2 applications	Image 'test' / pretraitem_nagao.m
Modification d'histogramme	- Egalisation d'histogramme - Recadrage de dynamique	Image 'os.bmp'/ egal_os
Rehaussement de contraste	X	Image route1.bmp'/ reh_route1.m

Tableau V.1 : Prétraitements et programmes associés

Image non prétraitée	Détecteurs de contours						
	Roberts	Prewitt	Sobel	Canny (SEMLG)	Canny (AEMLG)	Deriche (SEMLG)	Deriche (AEMLG)
Muscle.bmp	muscle1.m			muscle21.m		muscle22.m	
Os.bmp	os1.m			os21.m		os22.m	
Route1.bmp	route1.m			route21.m		route22.m	
Chromo.bmp	Binarisation de l'image, extraction de contours dilatation (4-connexité)						

Tableau V.2 : Détection de contour sur images non prétraitées et programmes associés

Image prétraitée	Détecteurs de contours				
	Roberts	Prewitt	Sobel	Canny (SEMLG)	Canny (AEMLG)
Os.bmp (égalisation d'histogramme)	os1_egal (0.1)			os21_egal.m	
Chromo.bmp	Dilatation érosion (4-connexité), extraction de contours dilatation (4-connexité)				

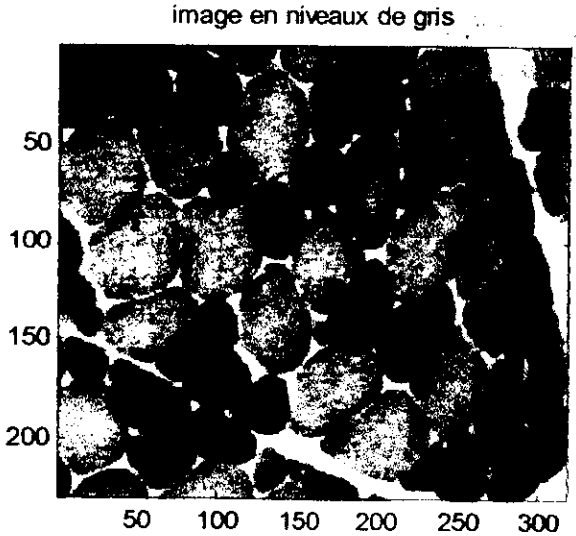
Tableau V.3 : Détection de contour sur images prétraitées et programmes associés

- interprétation des résultats obtenus.

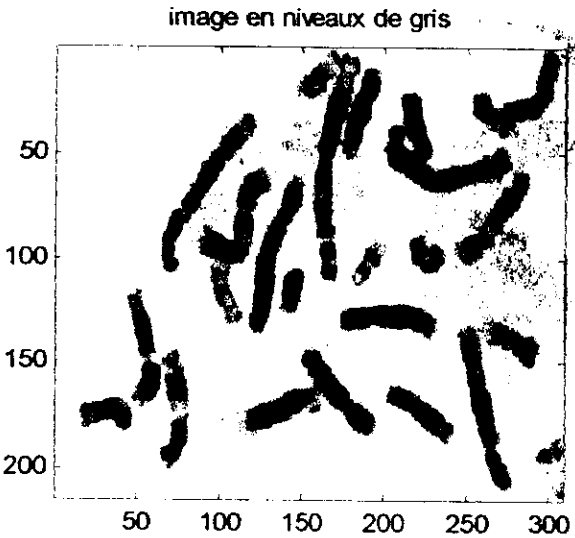
1. Images étudiées sur 256 niveaux de gris

Nous disposons d'images bitmap, 'vraies couleurs ('true colors').Des modifications, nécessaires pour passer de l'image RGB à une image en niveaux de gris, ont été réalisées. En effet, tous les traitements qui seront effectués sur l'image ne se basent pas sur la couleur, on met donc l'image en niveau de gris, les traitements n'en seront que plus rapides.

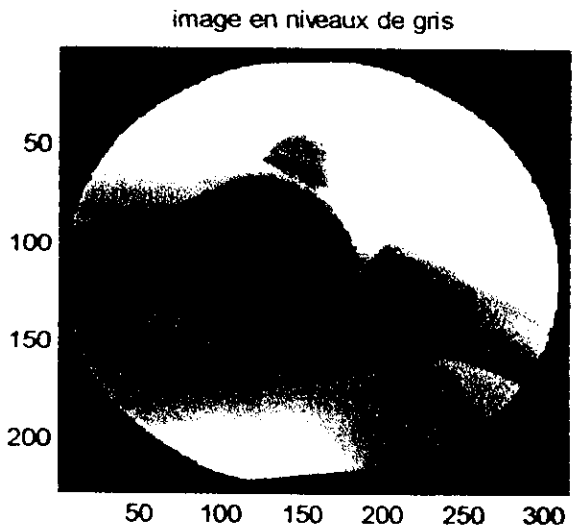
Des informations concernant le fichier, obtenues par la fonction **'imfinfo'** ont été rajoutées (encadrés).



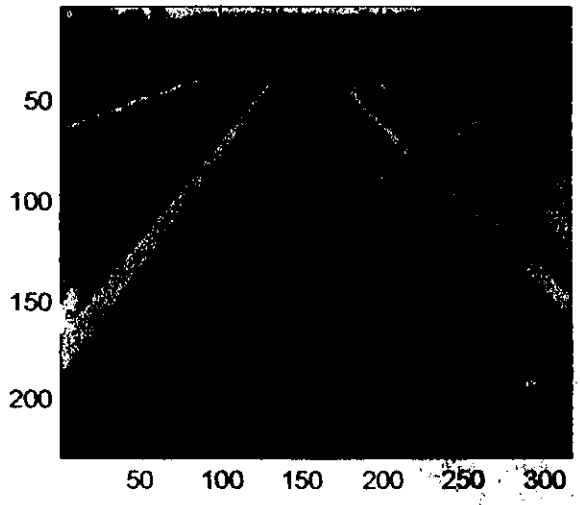
Fichier original : 'muscle.bmp'
Hauteur: 318
Largeur : 229
Profondeur de codage: 24 bits
Type d'image : 'truecolor'



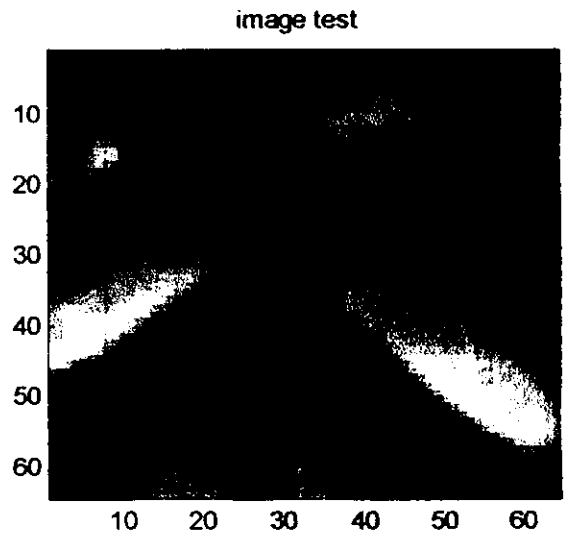
Fichier original: 'chromo.bmp'
Hauteur: 309
Largeur : 214
Profondeur de codage: 24 bits
Type d'image: 'truecolor'



Fichier original:'os.bmp'
Hauteur: 315
Largeur : 226
Profondeur de codage: 24 bits
Type d'image : 'truecolor'

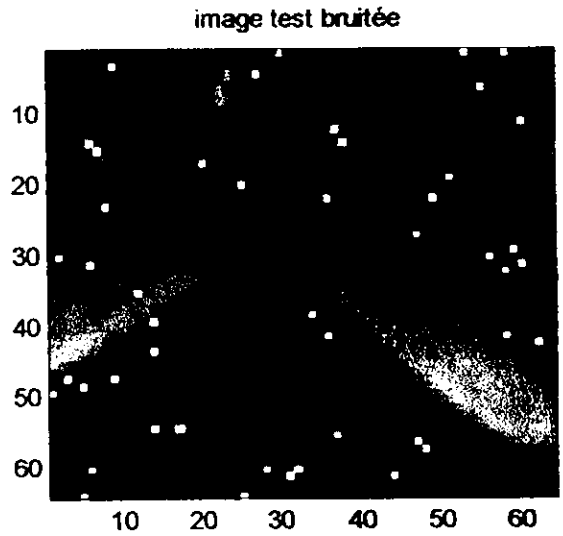


Fichier : 'route1.bmp'
Hauteur: 317
Largeur : 228
Profondeur de codage: 24 bits
Type d'image : 'truecolor'



2. Prétraitements

2.1. Réduction de bruit (Rajout de bruit 'poivre et sel' : d=0.02, d=0.05)



2.1.1. Bruit poivre & sel d=0.02

(fichier pretraitem_nagao.m (pretraitem_nagao.(d) : d, densité du bruit))

- **pretraitem_nagao.(0.02)**

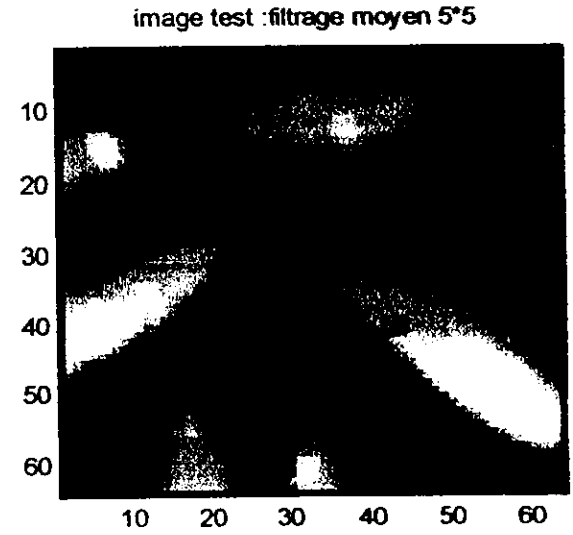


image test :filtrage median 5*5

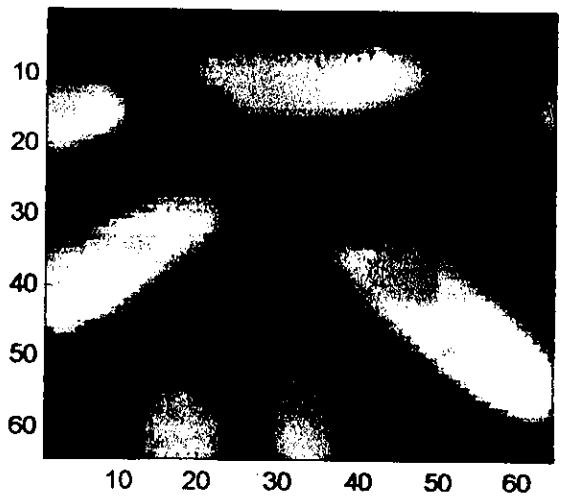


image test :filtrage nagao *1

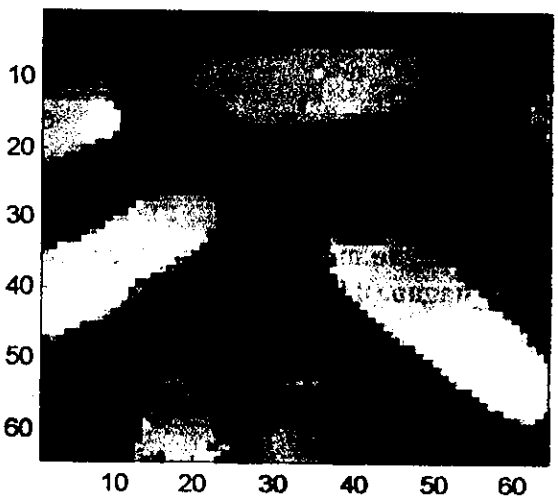
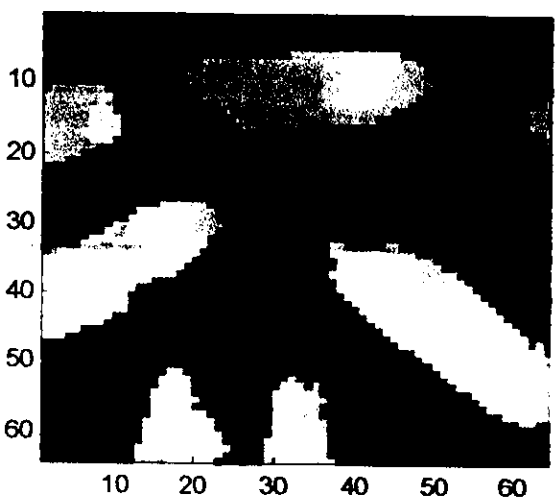
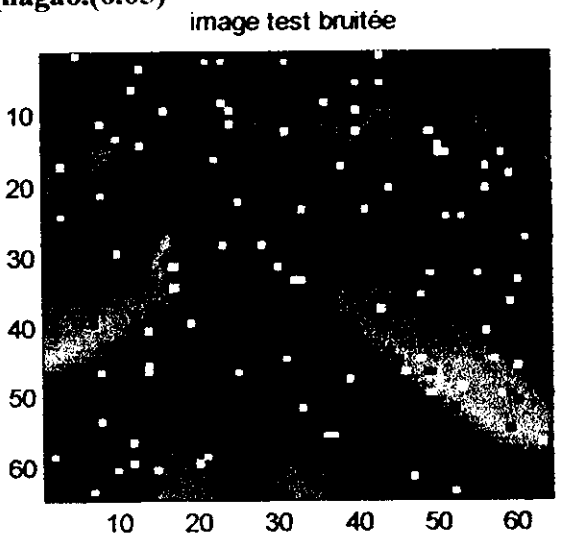


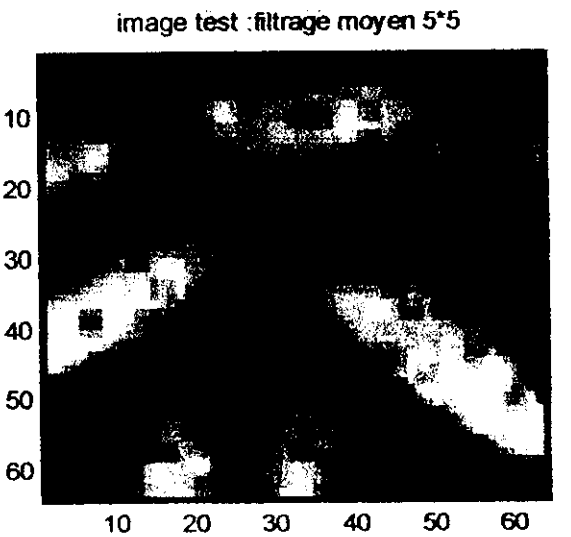
image test :filtrage nagao *2

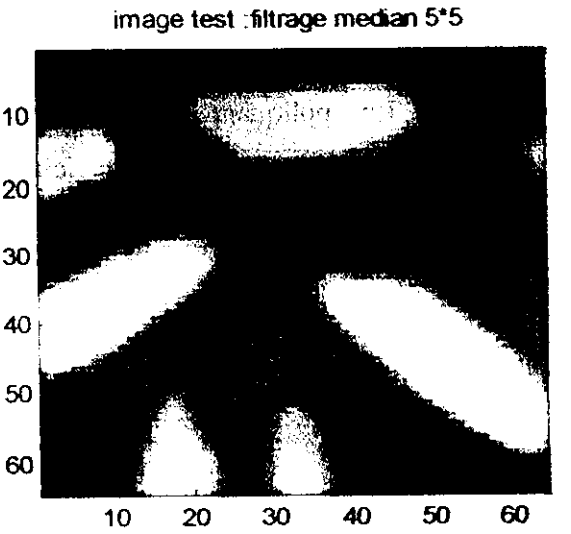


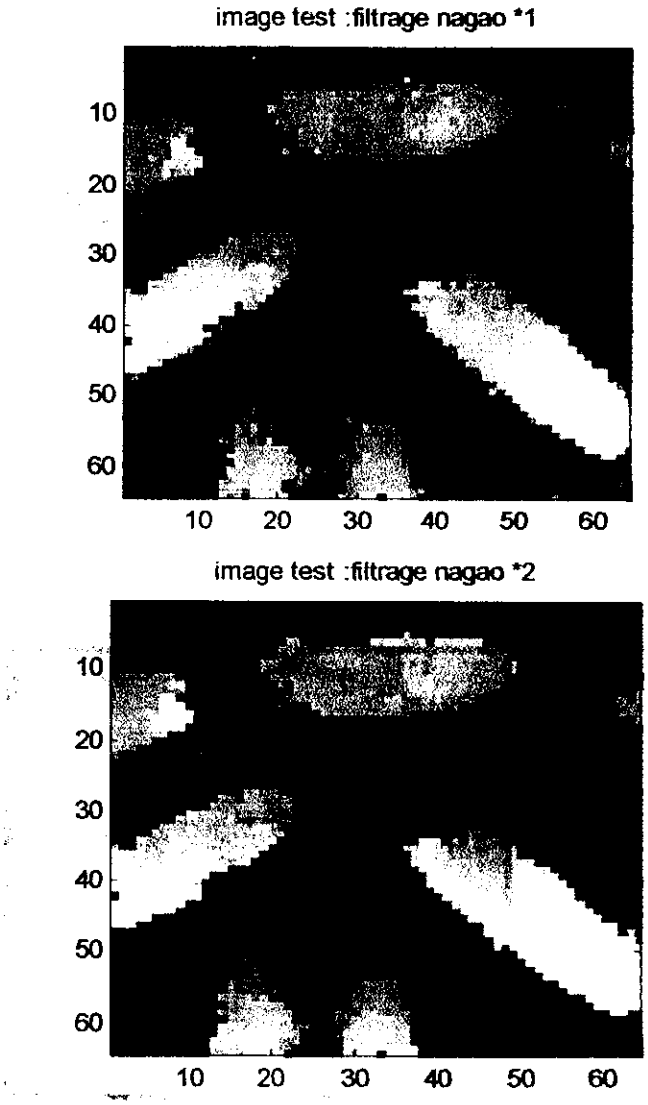
2.1.2. Bruit poivre et sel d=0.05

- **pretraitem_nagao.(0.05)**









2.1.3. Interprétation des résultats (visuelle)

Filtrage moyen (5*5) : En appliquant ce filtrage à l'image bruitée (d=0.02), on remarque que celui-ci est inefficace, même si la valeur des niveaux de gris a changé, la grosseur du grain a augmenté. On remarque également l'apparition de flou dans l'image.

Les résultats sont d'autant plus médiocres lorsqu'on augmente la densité du bruit (d=0.05).

Filtrage médian (5*5) : Ce filtre permet l'élimination totale du bruit dans les deux cas. Par rapport à l'image originale, la valeur des niveaux de gris de l'image a changé (on peut le remarquer à travers le fond).

Filtrage de Nagao :

Dans le cas d=0.02, en appliquant une fois le filtre à l'image, on obtient une réduction du bruit, mais reste insuffisante. On remarque par contre, une amélioration du contraste, les zones de transitions sont plus franches (par rapport à l'image originale).

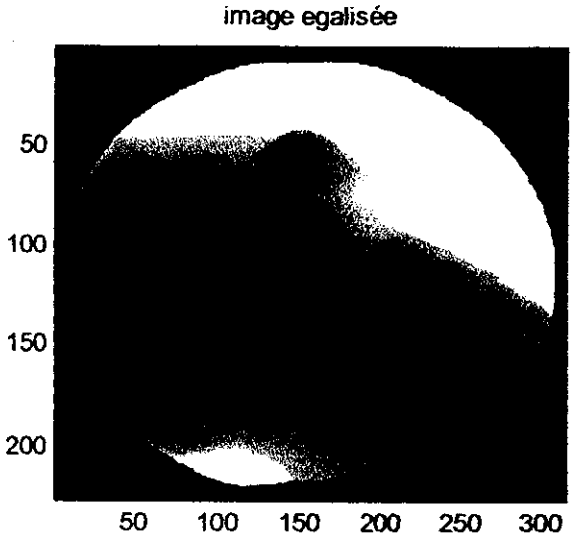
Si l'on applique une deuxième fois ce filtre, les résultats au niveau de l'élimination du bruit sont améliorés.

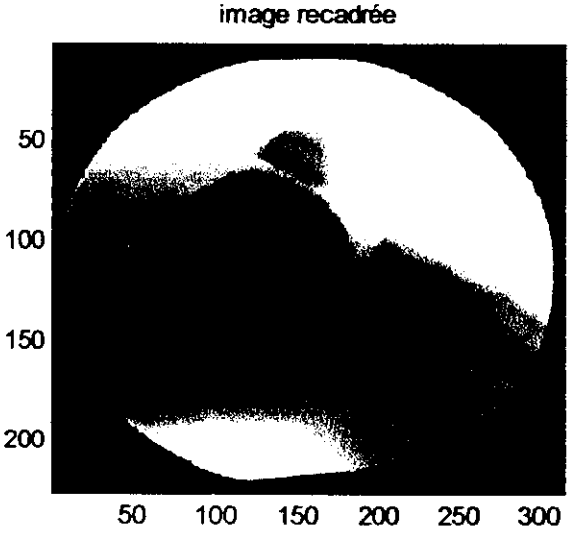
Dans le cas d=0.05, les résultats la densité de bruit présent (grains) est plus importante si l'on applique une fois le filtre à l'image. On peut améliorer les résultats si l'on effectue une deuxième fois le filtrage. Comme dans le cas précédent, on remarque que le contraste a été amélioré.

Général : Dans les deux cas (d=0.02,d=0.01), on remarque que le filtre médian est le plus efficace concernant l'élimination du bruit poivre et sel. Le filtre de Nagao donne également de bons résultats concernant l'élimination du bruit, mais présente surtout l'avantage d'améliorer le contraste, ce qui est intéressant pour une éventuelle détection de contours.

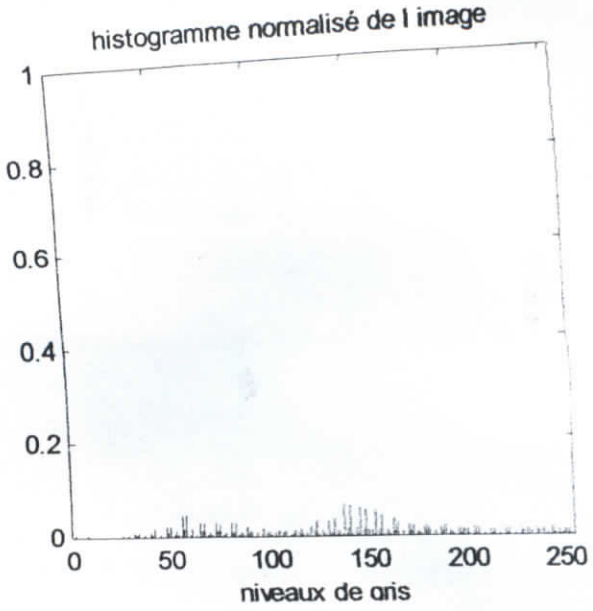
2.2. Modification d'histogramme

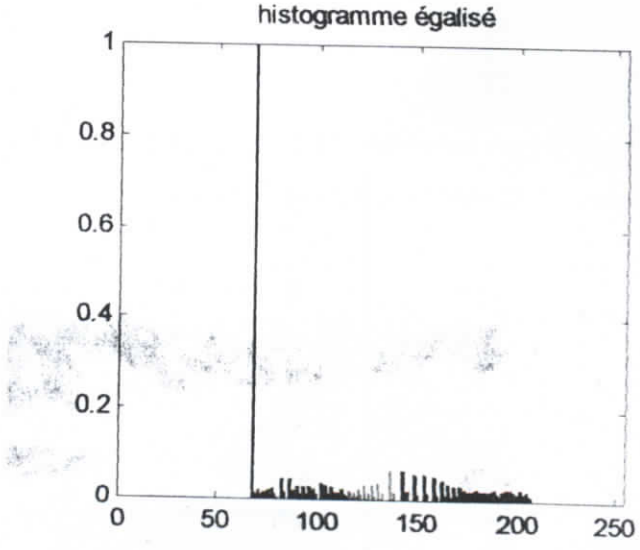
2.2.1. Egalisation d'histogramme. (Image 'os.bmp')

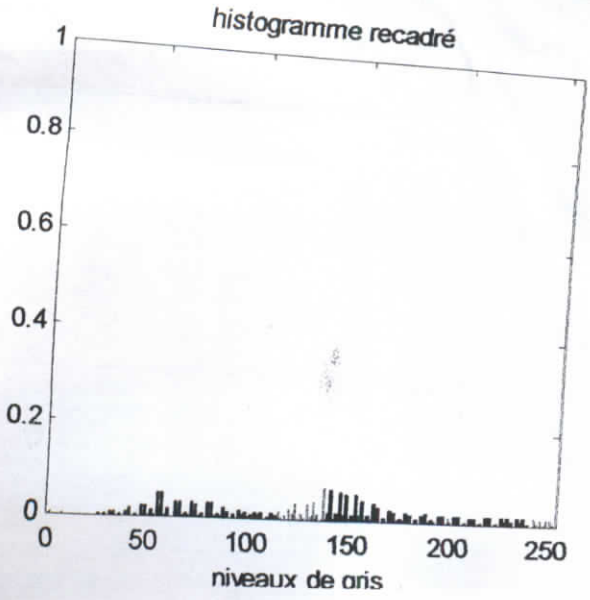




Histogrammes correspondants







Interprétation des résultats (visuelle)

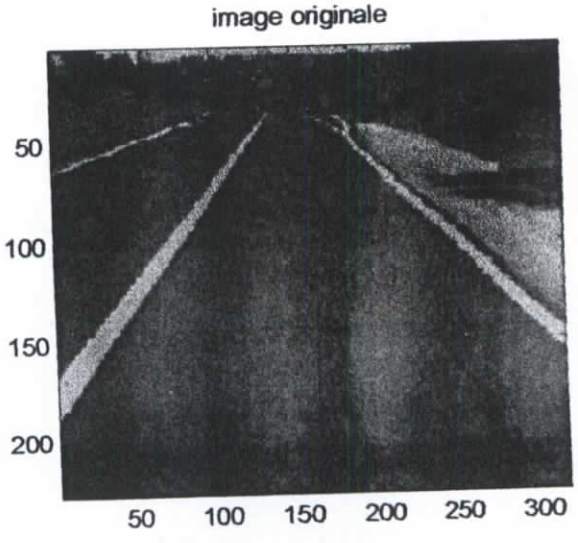
Egalisation d'histogramme : on remarque que le contraste est renforcé. Certaines zones de l'image (chair) ont été mises en valeur.

Recadrage de dynamique : pas de changement par rapport à l'image initiale, ceci s'explique par le fait que l'image originale a déjà une dynamique maximale.

2.3. Rehaussement de contraste (image'route1.bmp')

fichier reh_route1.m (route(k), k paramètre réglant la raideur de la transition)

route1 (-1)



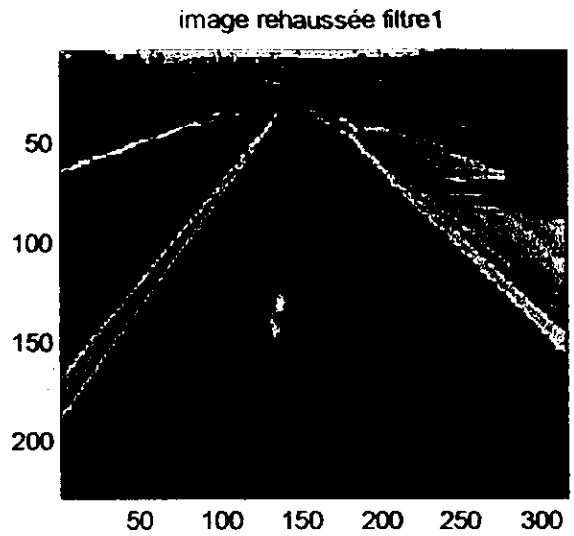


image rehaussée filtre1
50
100
150
200
50
100
150
200
250
300

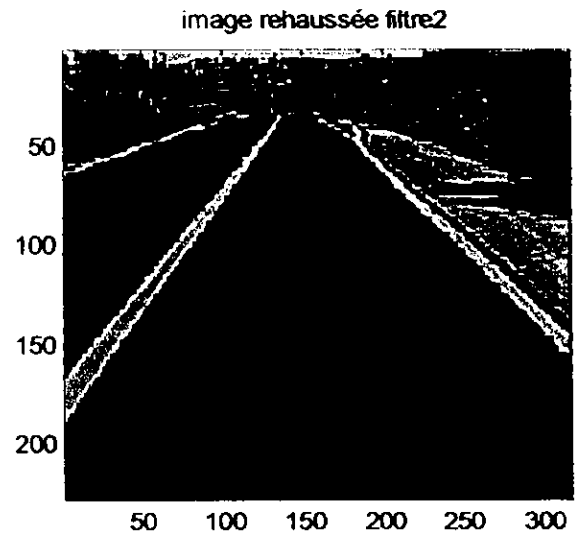


image rehaussée filtre2
50
100
150
200
50
100
150
200
250
300

route1 (-2)

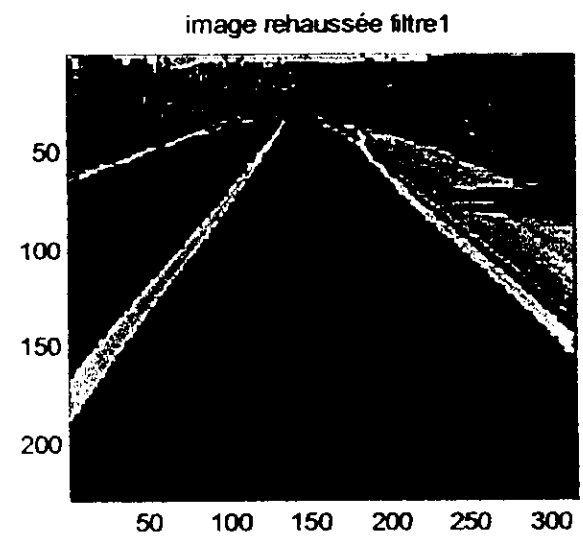
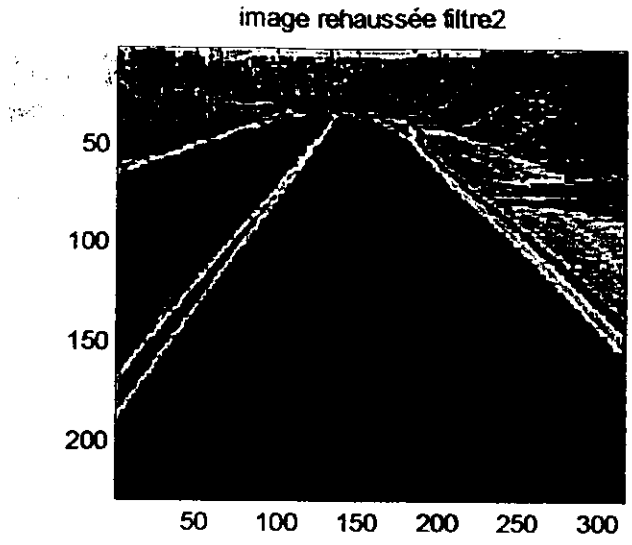


image rehaussée filtre1
50
100
150
200
50
100
150
200
250
300



Interprétation des résultats (visuelle)

Dans les deux cas (k=-2, k=-1), le contraste est amélioré avec les deux filtres, avec un rehaussement plus net à l'aide filtre 2. Cependant, on remarque surtout une grande sensibilité au bruit.

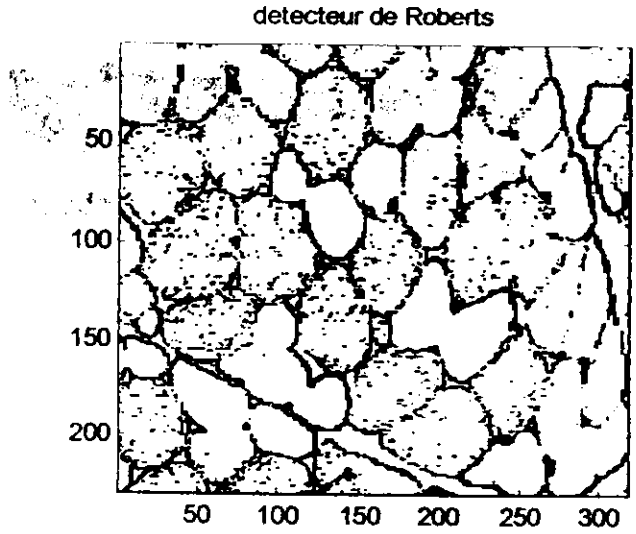
3. Détecteurs de contours

3.1. Images non prétraitées

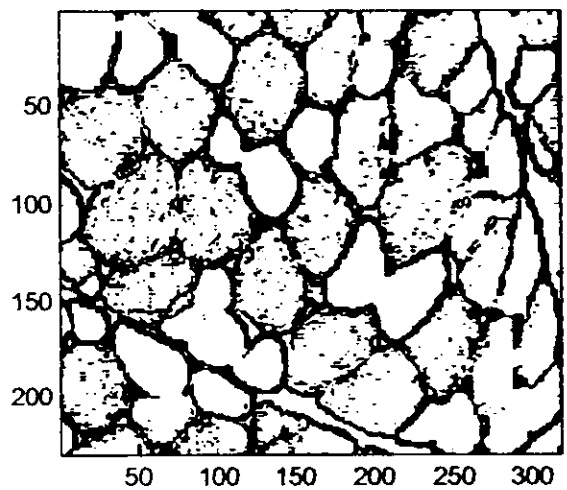
3.1.1. Image 'muscle.bmp'

Fichier muscle1.m (muscle1(s), s seuil d'extraction)

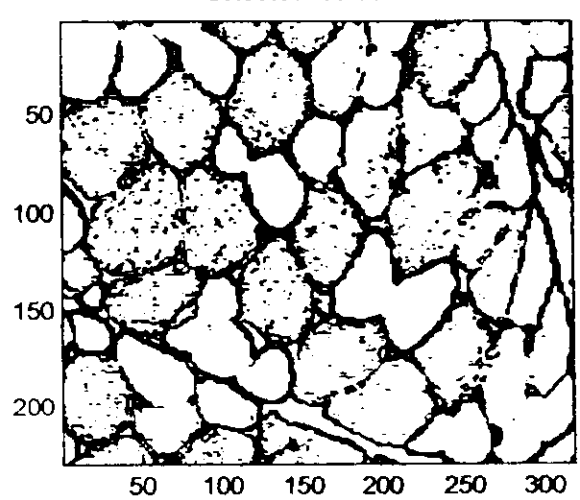
muscle1 (0.1)



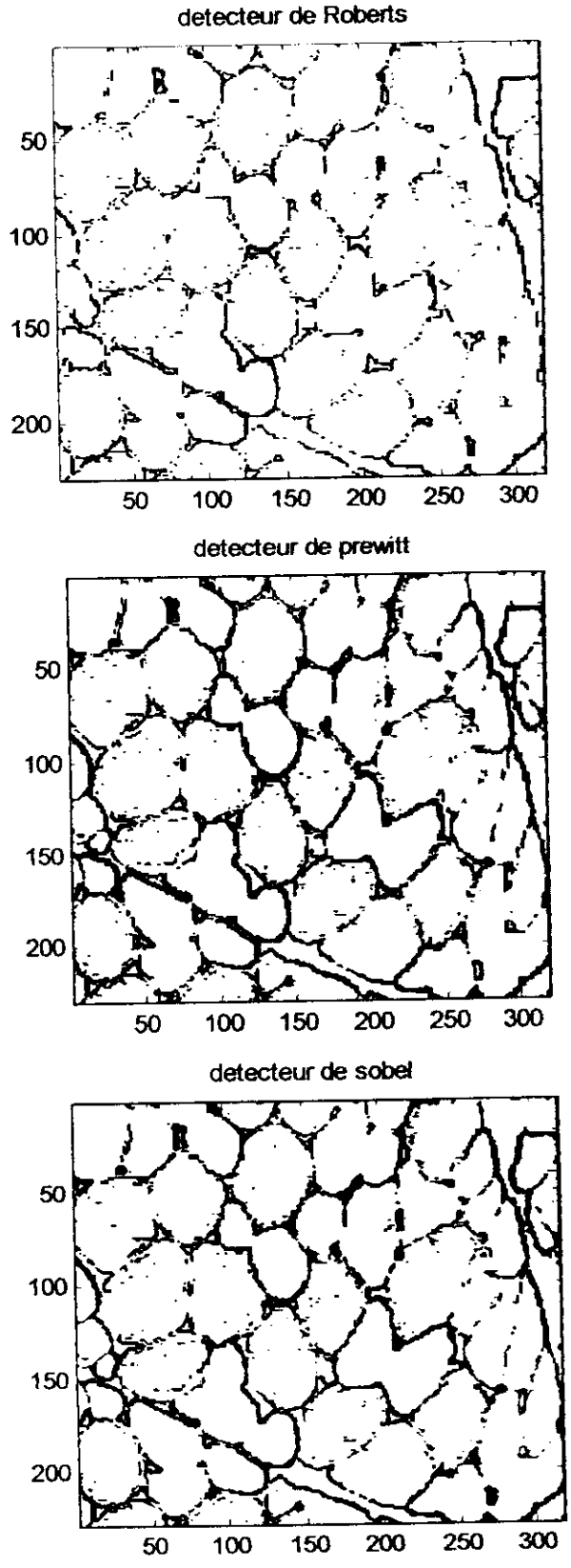
detecteur de prewitt



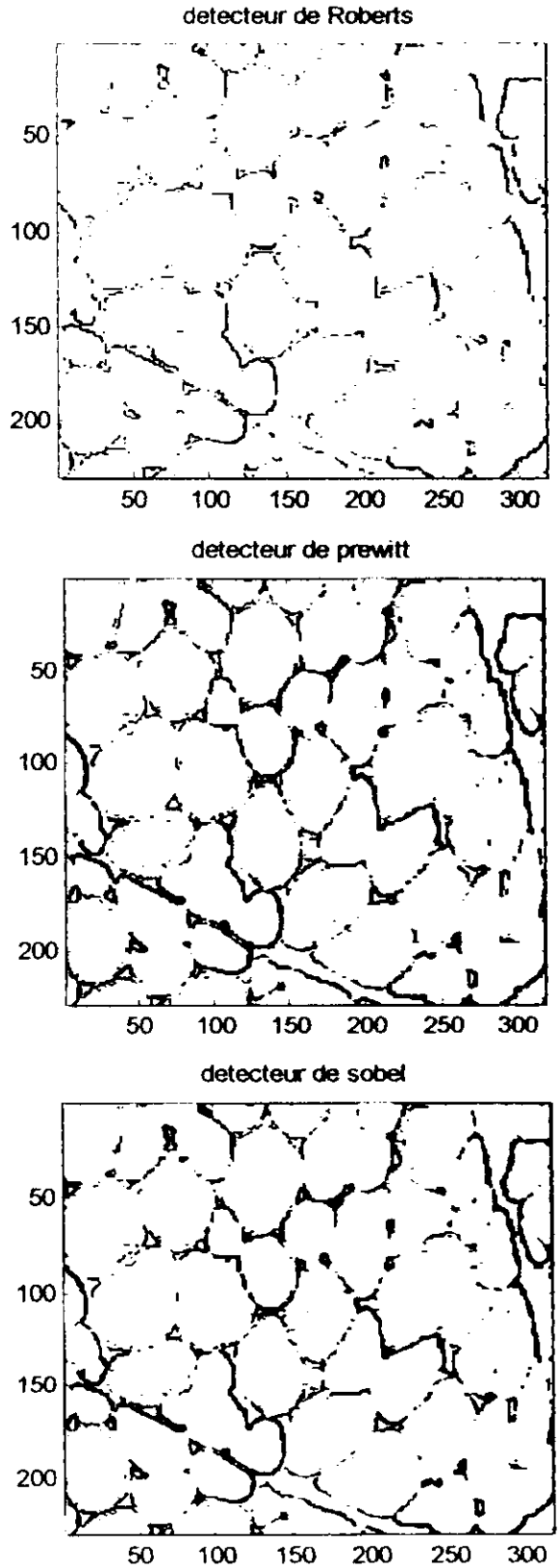
detecteur de sobel



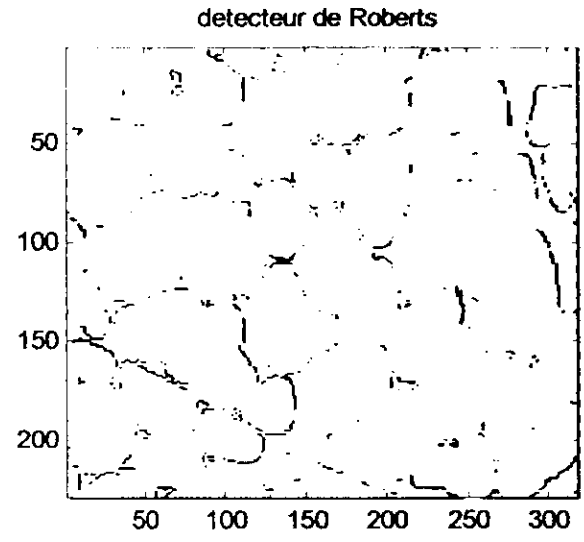
muscle1 (0.15)

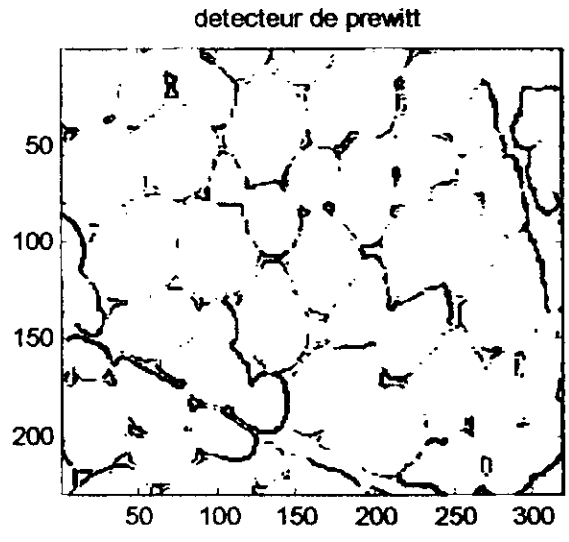

detecteur de Roberts
50
100
150
200
50 100 150 200 250 300
detecteur de prewitt
50
100
150
200
50 100 150 200 250 300
detecteur de sobel
50
100
150
200
50 100 150 200 250 300

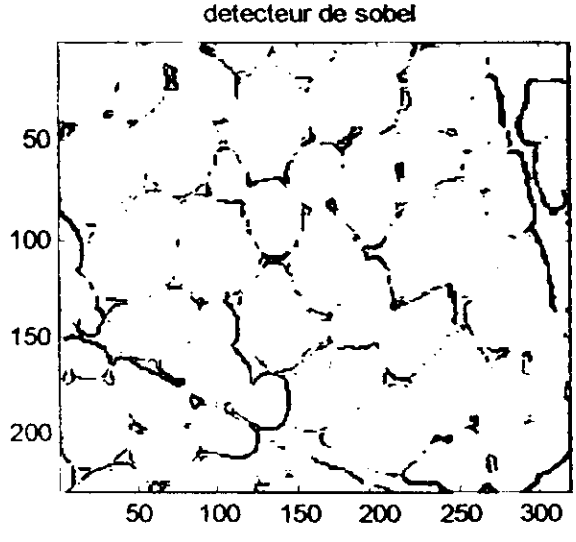
muscle1 (0.2)


detecteur de Roberts
50
100
150
200
50 100 150 200 250 300
detecteur de prewitt
50
100
150
200
50 100 150 200 250 300
detecteur de sobel
50
100
150
200
50 100 150 200 250 300

muscle1 (0.25)

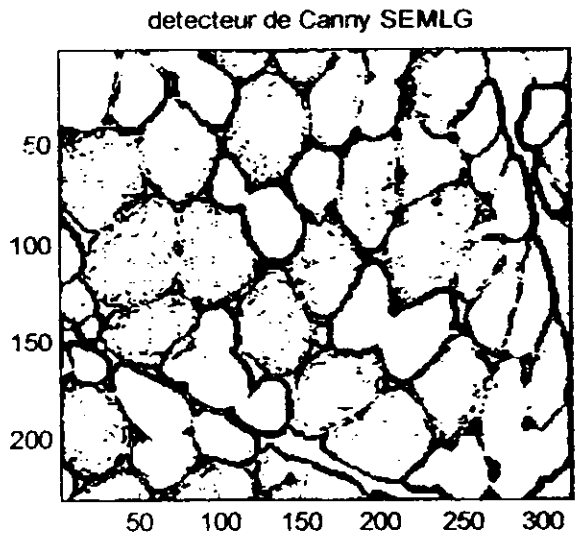

detecteur de Roberts
50
100
150
200
50
100
150
200
250
300

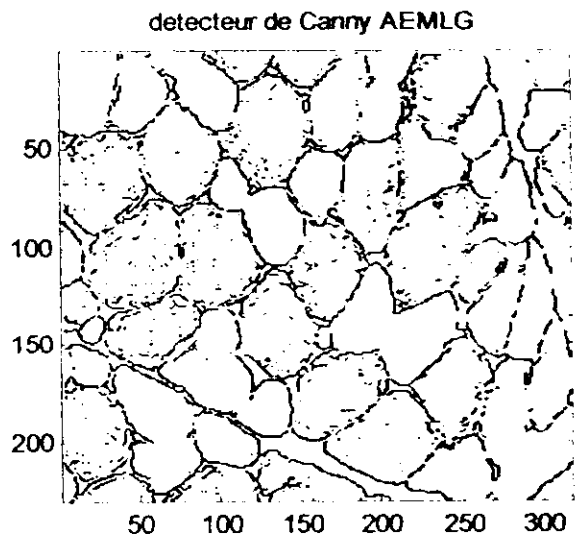

detecteur de prewitt
50
100
150
200
50
100
150
200
250
300


detecteur de sobel
50
100
150
200
50
100
150
200
250
300

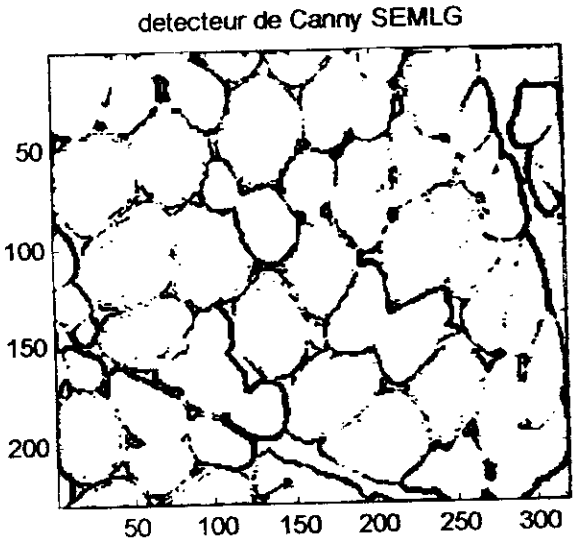
fichier muscle21;m (muscle21(s,sig) , s, seuil ;sig résolution du filtre)

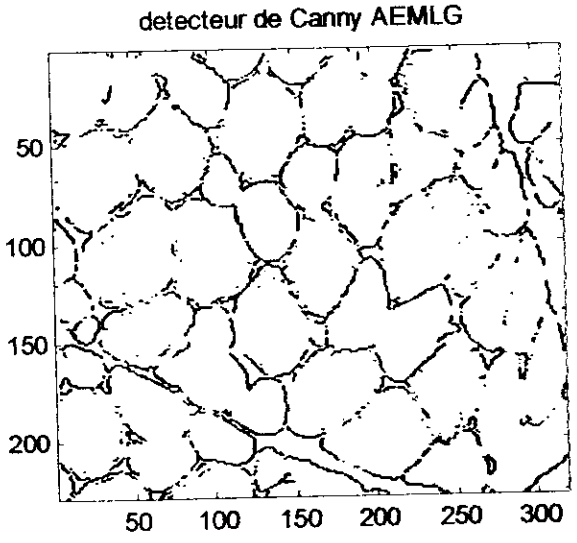
muscle21 (0.1, 0.5), et_fil =2



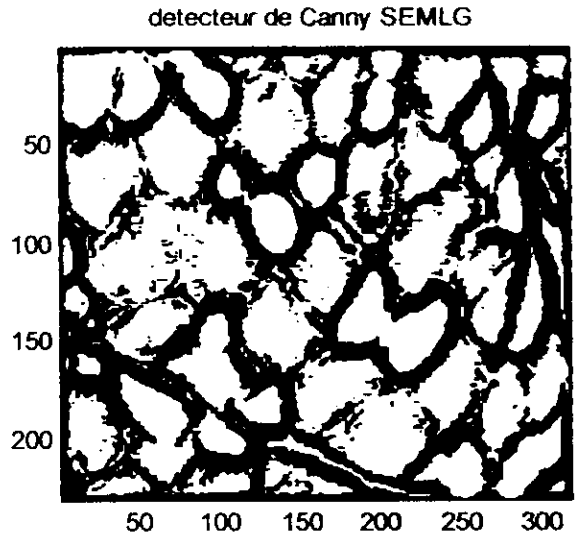


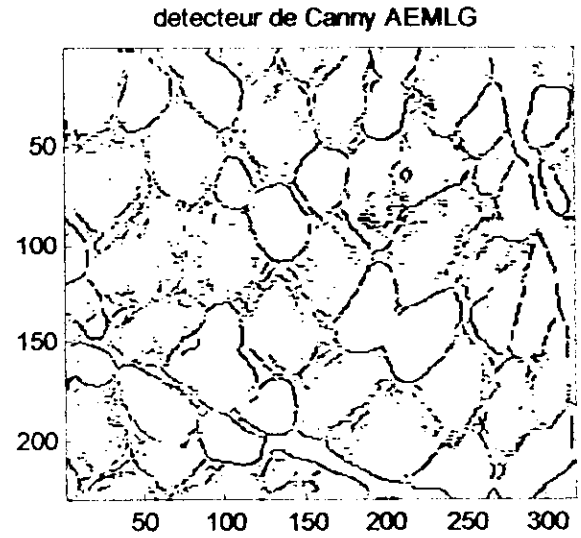
muscle21 (0.15, 0.5), et_fil =2



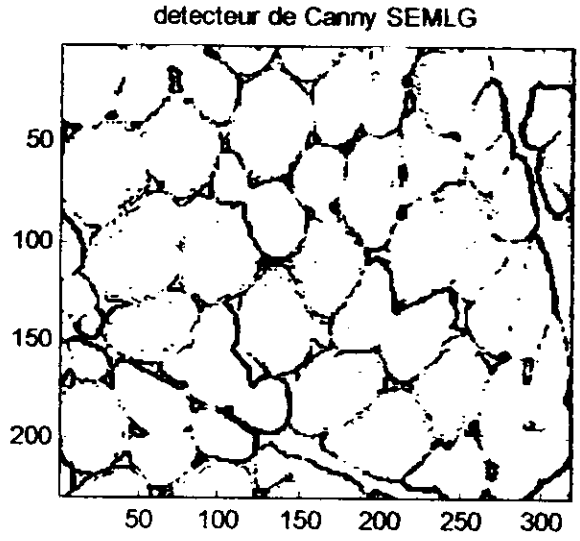


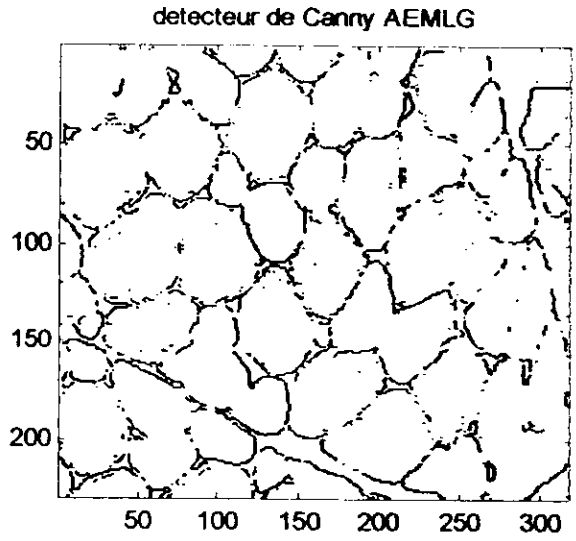
muscle21 (0.1, 2), et_fil =8





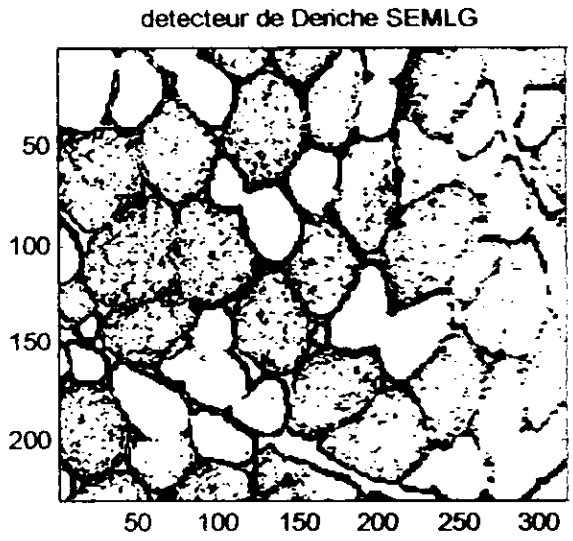
muscle21 (0.15, 0.25), et_fil =1

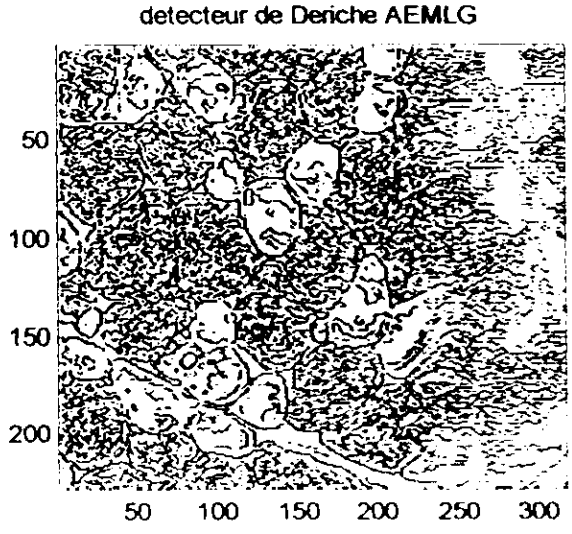




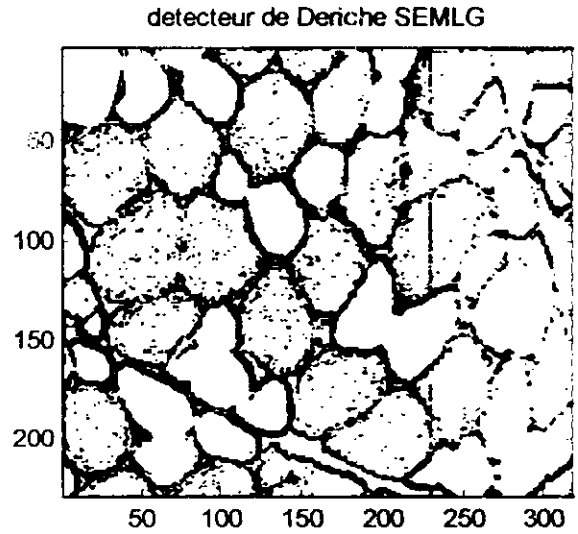
Fichier muscle22.m (muscle22(s, alpha), s : seuil, alpha paramètre détecteur Deriche)

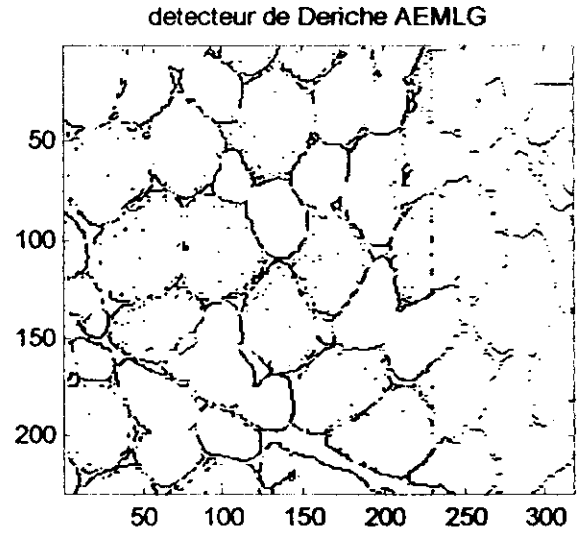
muscle22 (0.1, 4)





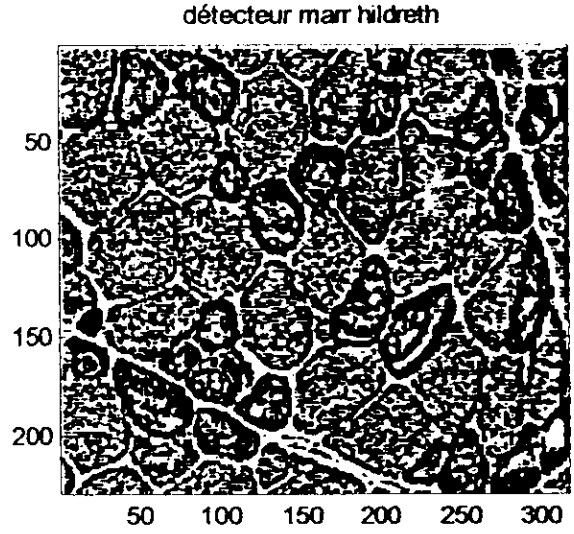
muscle22 (0.1, 2)

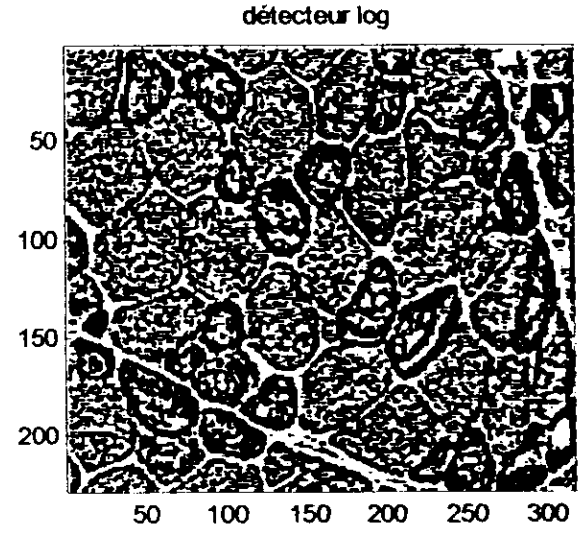




Fichier muscle3.m (muscle3 (t, sig), t réglage taille, sig résolution du filtre)

muscle3 (1,0.5)





Interprétation des résultats (visuelle)

1.Opérateurs de Roberts, Prewitt et Sobel

Général : lorsqu'on augmente la valeur du seuil, les contours sont plus fins mais certains d'entre eux disparaissent. On remarque également que la sensibilité au bruit diminue.

Pour une même valeur du seuil, les contours détectés par les opérateurs de Prewitt et Sobel sont plus épais, ils détectent plus de bruit également.

Ces résultats sont dus à la taille des opérateurs et à la nature texturée de l'image d'origine.

Les images traitées par les opérateurs de Prewitt et Sobel ont un aspect visuel proche.

2.Détecteur de Canny

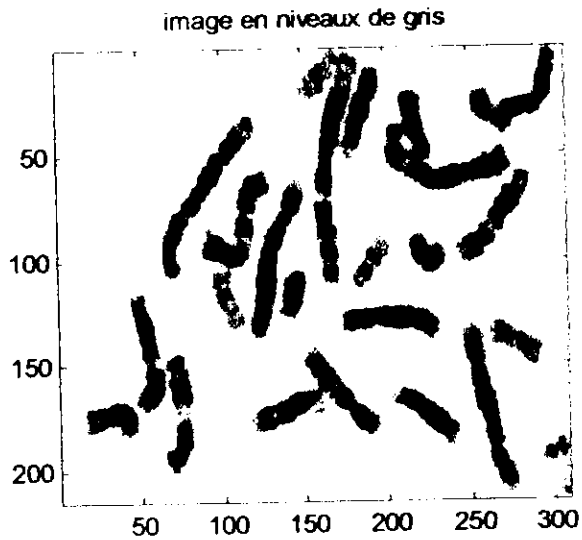
Pour un seuil fixe, et un 'sig' variable : en diminuant la valeur de sigma, on augmente l'étalement du filtre, les contours détectés sont plus épais (cas SEMLG).Lorsqu'on applique l'extraction du maximum local du gradient (ELMG) on obtient des contours plus minces, mais ils sont plus fermés lorsque 'sig' diminue.

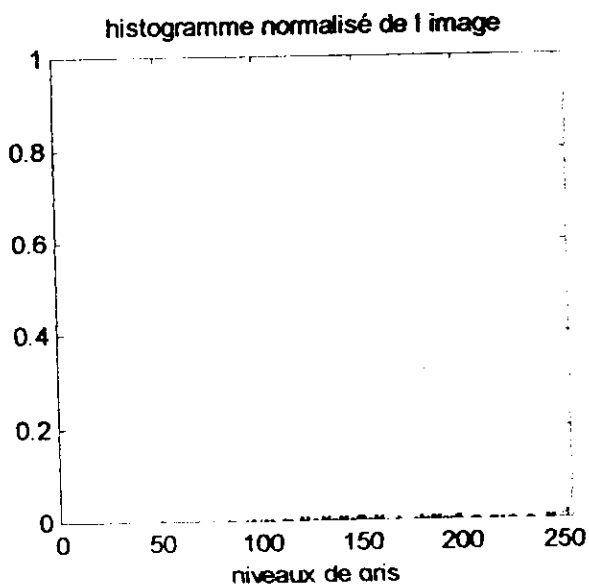
Pour un sigma fixe et seuil variable : quand on augmente la valeur du seuil, le bruit diminue (cas SEMLG et AEMLG).Les contours (cas SEMLG) sont plus fins mais disparaissent pour certaines parties de l'image.

Pour une même valeur du seuil on remarque que la détection de Canny pour les valeurs de 'sig' choisies est moins sensible au bruit comparé aux détecteurs précédents.

3.**Détecteur de Deriche :** le seuil est fixe (seuil =0.1), quand alpha diminue, le bruit diminue. Quand on applique l'extraction du maximum local du gradient avec alpha=4, l'opérateur est très sensible au bruit. Par contre le choix alpha=2 donne de meilleurs résultats.

3.1.2 Image 'chromo.bmp'





Fichier chromo1_bin.m (chromo1_bin (s) ,s seuil de binarisation)

chromo1_bin (210)

image binarisée

image contours par dilatation 4-connexité

chromo1_bin (230)

image binarisée

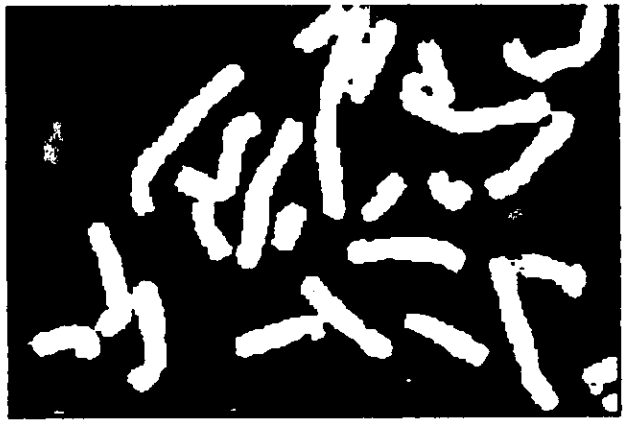


image contours par dilatation 4-connexité

chromo1_bin (240)

image binarisée

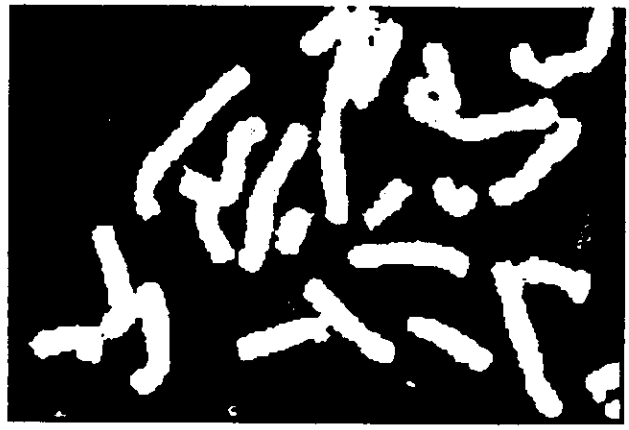
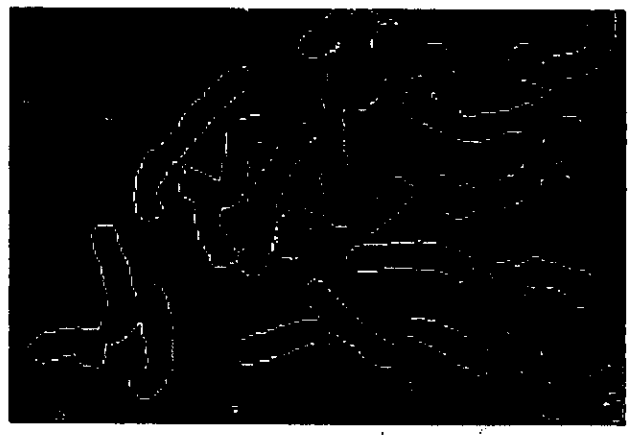


image contours par dilatation 4-connexité



Interprétation des résultats

Pour cette image on effectue une binarisation de celle-ci puis on effectue une détection de contours à l'aide de la dilatation.

Les résultats obtenus dépendent du seuil de binarisation : quand on augmente le seuil des taches apparaissent dans l'image, ceci s'explique par leur présence dans l'image originale, le choix du seuil détermine si elles seront détectées ou pas.

Pour cette image ces taches sont considérées comme du bruit. D'autre part le seuil détache ou non les chromosomes les uns des autres.

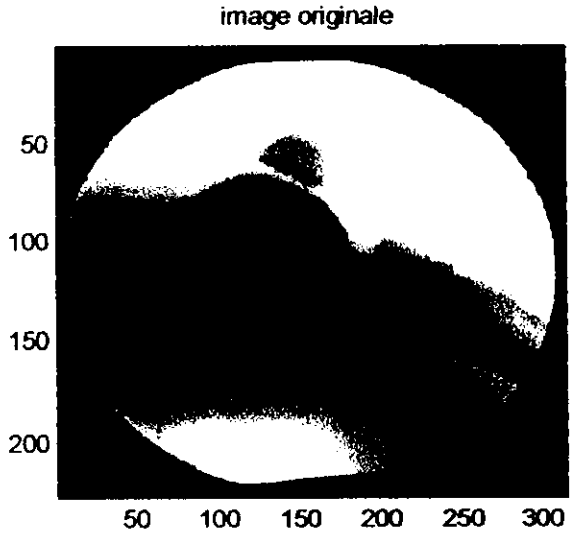
Cette approche de la détection est simple à réaliser et nous permet d'aboutir directement à des contours d'une épaisseur d'un pixel et fermés.

err_mse (230,240) =0.0274

err_mse (210,240)=0.0650

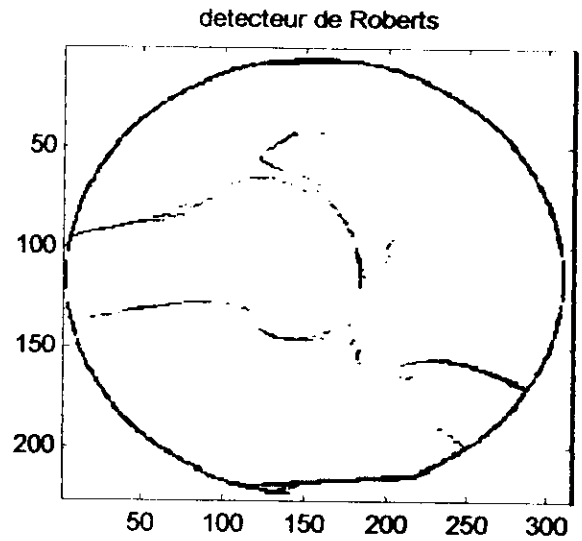
err_mse (210,230)=0.0464

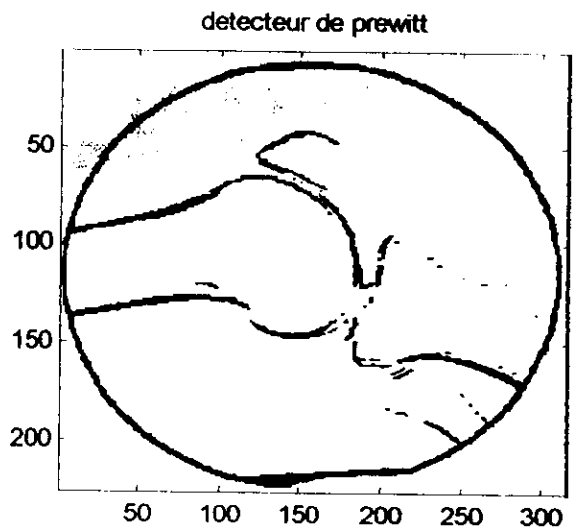
3.1.3. Image 'os.bmp'

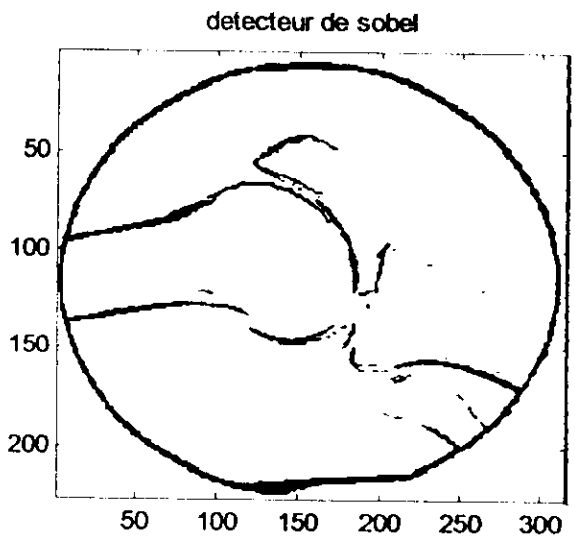


Fichier os1.m (os1 (s), s : seuil)

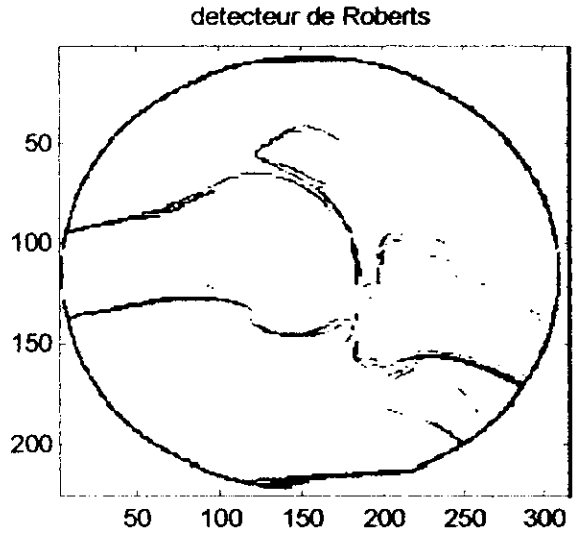
os1 (0.1)

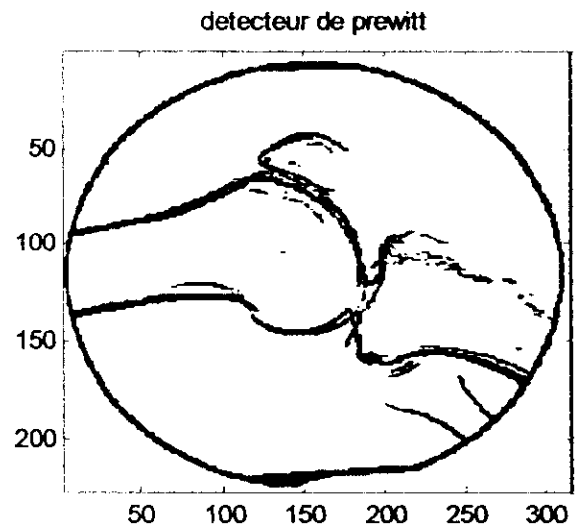


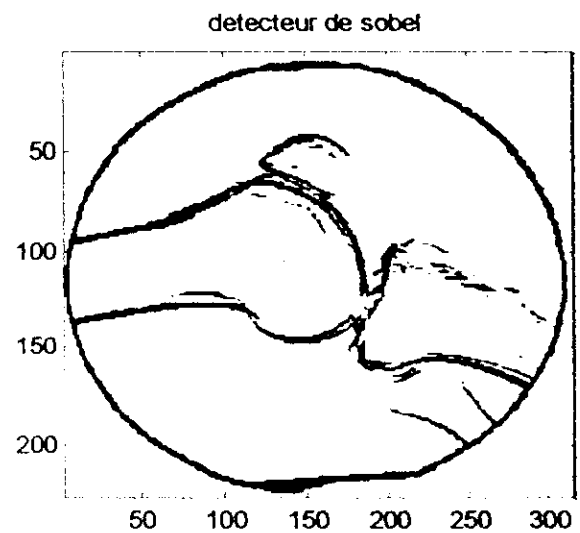




os1 (0.07)

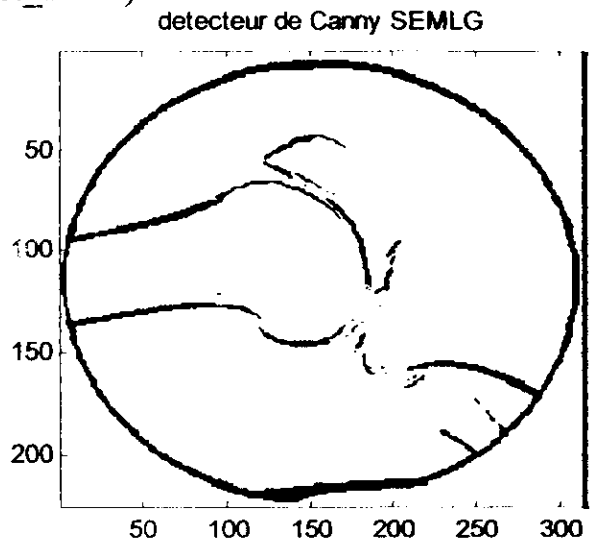

detecteur de Roberts
50
100
150
200
50
100
150
200
250
300

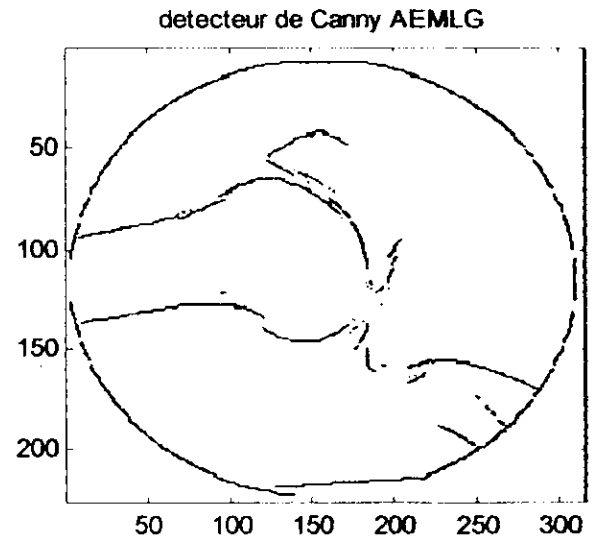

detecteur de prewitt
50
100
150
200
50
100
150
200
250
300


detecteur de sobel
50
100
150
200
50
100
150
200
250
300

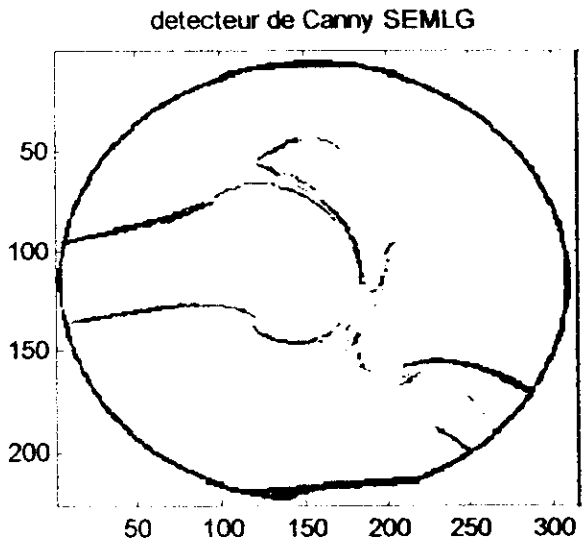
fichier os21.m (os21(s,sig))

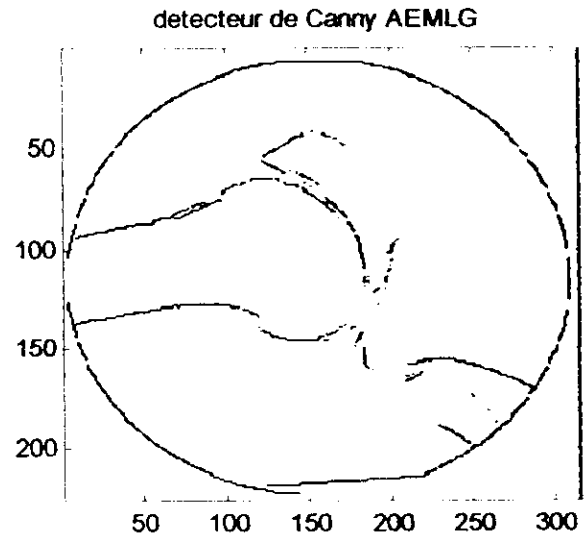
os21 (0.1,0.5) (et_fil =2)





os21 (0.1, 0.07) ,et_fil =1





Interprétation des résultats

A seuil égal, les opérateurs de Sobel et Prewitt détectent plus de contours même s'ils sont plus épais, comparés à l'opérateur de Roberts.

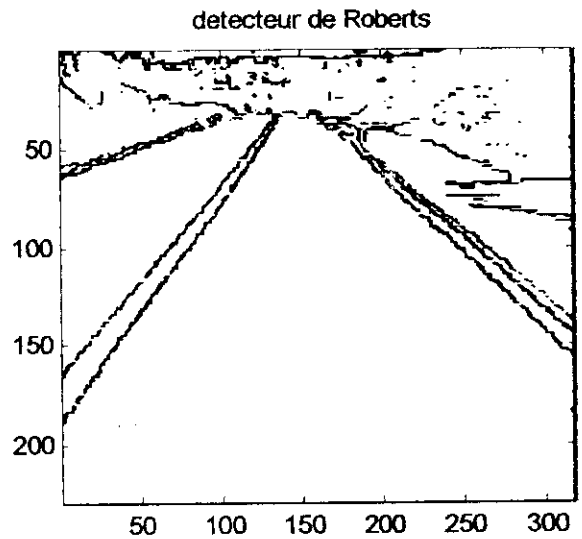
Pour un même seuil, le détecteur de Canny nous permet d'obtenir une image contenant moins de bruit (SELMG).on peut aboutir à des contours plus minces en utilisant l'ELMG.

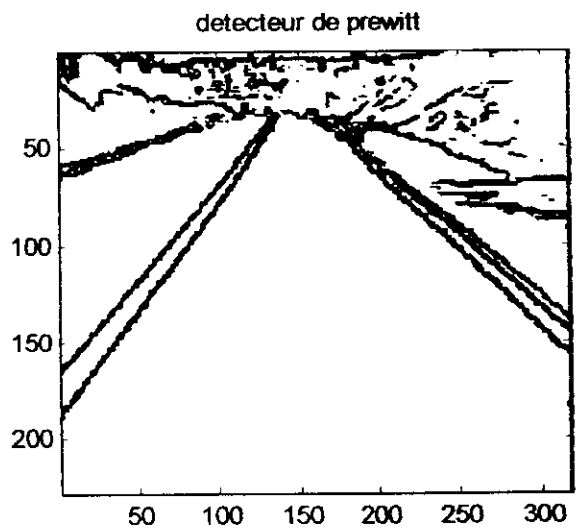
On remarque que les contours disparaissent surtout au niveau de l'os de droite,ceci est du au fait que dans l'image de départ, cette partie est moins contrastée par rapport au fond que l'os de gauche (plus foncé).

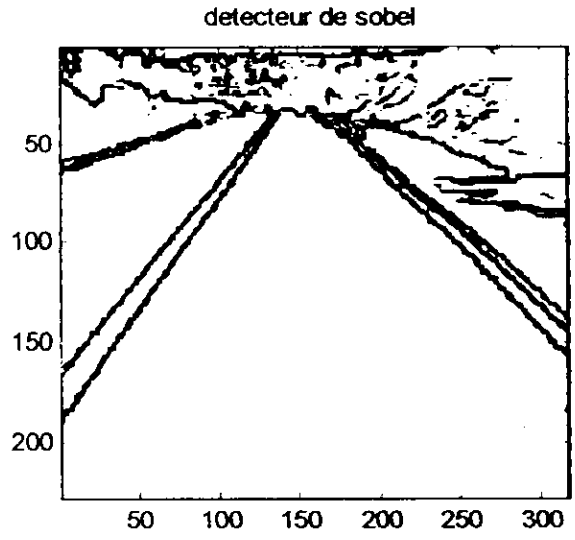
3.1.4. Image 'route1.bmp'

Fichier route1.m (route1(s))

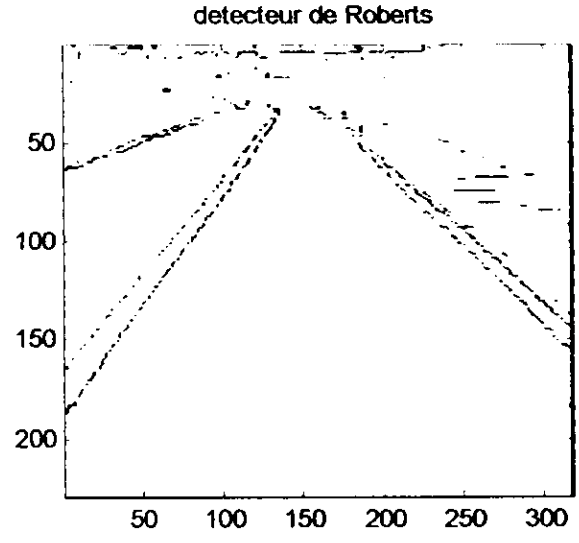
route1 (0.1)

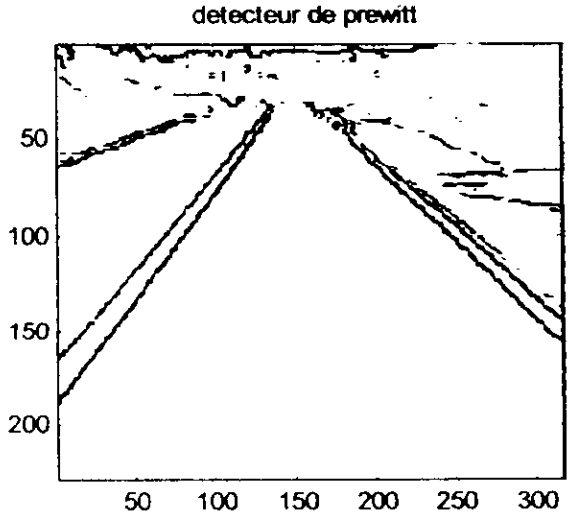


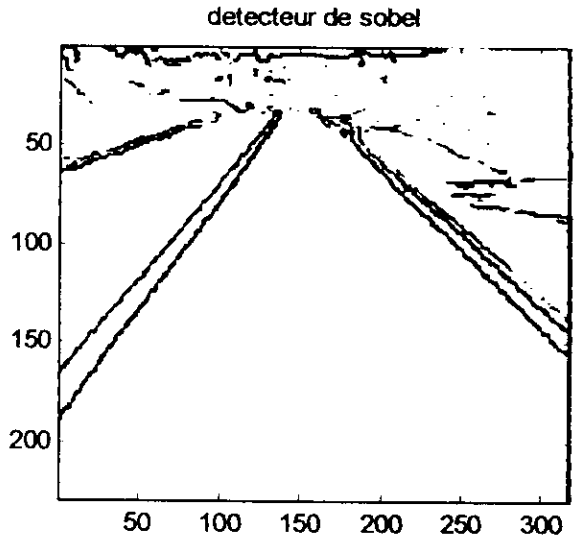



detecteur de sobel
50
100
150
200
50
100
150
200
250
300

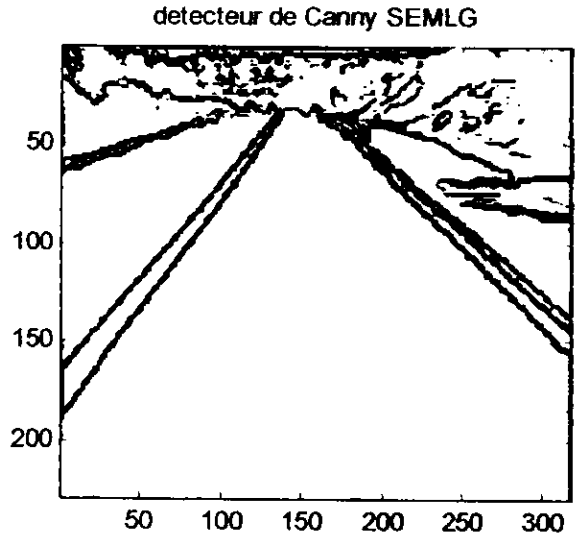
route1 (0.2)

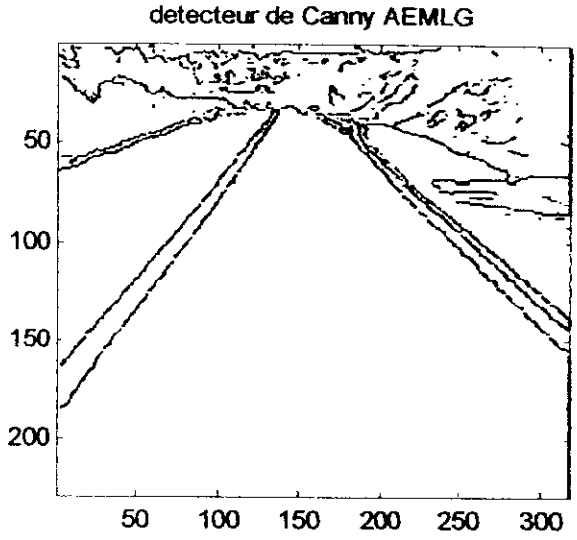

detecteur de Roberts
50
100
150
200
50
100
150
200
250
300


detecteur de prewitt
50
100
150
200
50
100
150
200
250
300



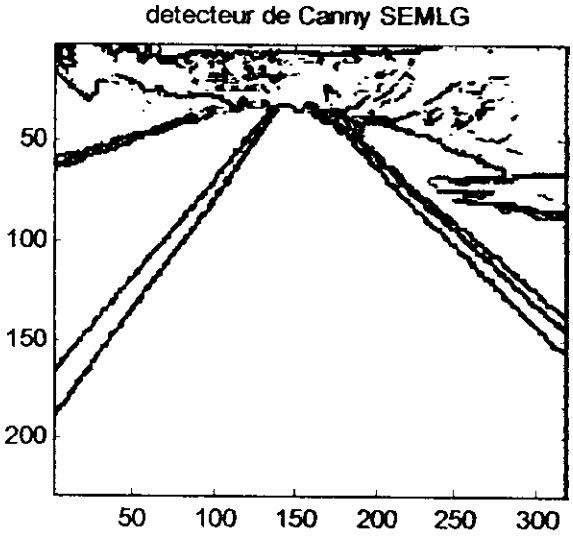
route21 (0.1, 0.5) :et_fil = 2

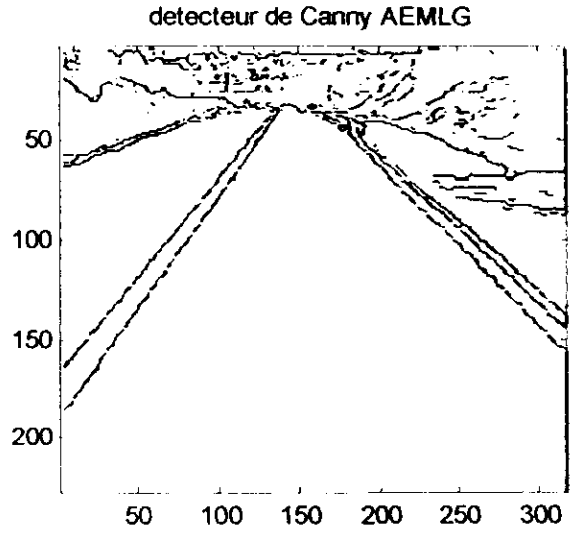




Fichier route21.m (route21 (s,sig))

route21 (0.1, 0.07) : et_fil =1





Interprétation des résultats

Nous pouvons faire les mêmes remarques générales sur les opérateurs que pour l'image 'muscle.bmp' en ce qui concerne l'épaisseur des contours et leur détection.
On remarque cependant, que pour l'image 'muscle.bmp' était texturé ce qui engendrait la présence de bruit à l'intérieur des cellules.
Dans le cas de l'image 'route.bmp', la différence se fait essentiellement au niveau du haut de l'image riche en détails.

3.2. Images avec prétraitement

3.2.1. Image 'chromo.bmp'

Fichier chromo2_bin.m (chromo2_bin(s),s : seuil)

chromo2_bin (210)

image binarisée

image binarisée érodée puis dilatée (deux applications)

image contours

chromo2_bin (230)

image binarisée

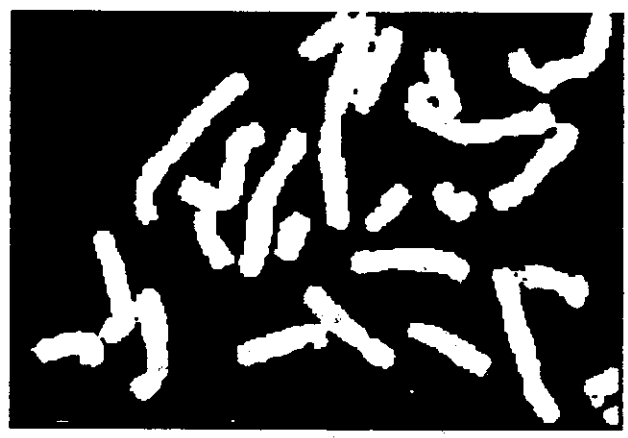
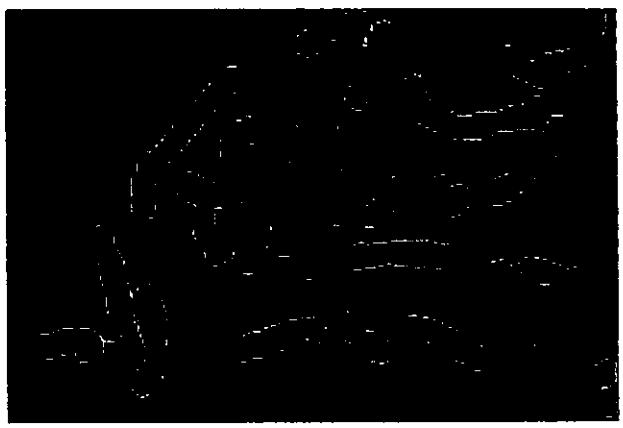


image binarisée érodée puis dilatée (deux applications)

image contours



chromo2_bin (240)

image binarisée

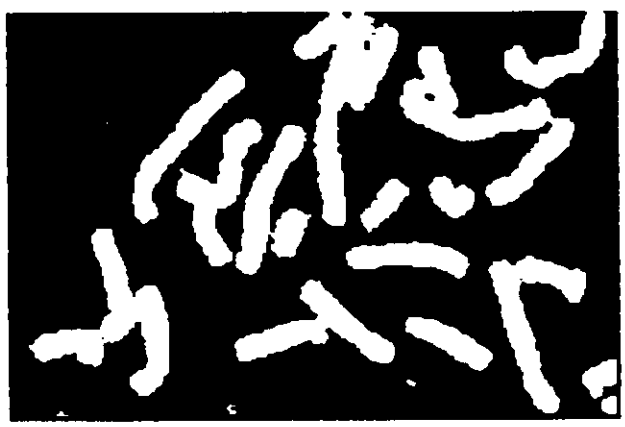
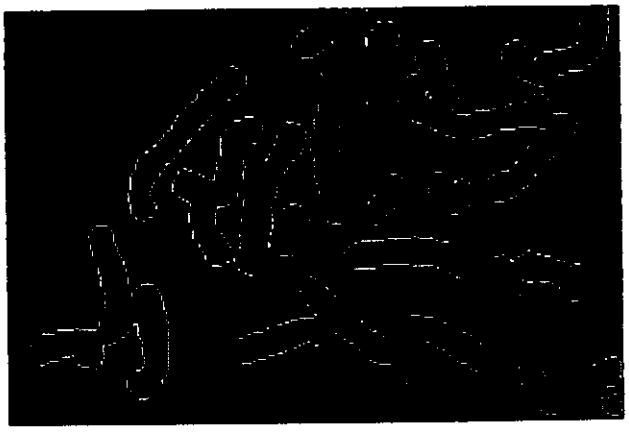


image binarisée érodée puis dilatée (deux applications)

image contours



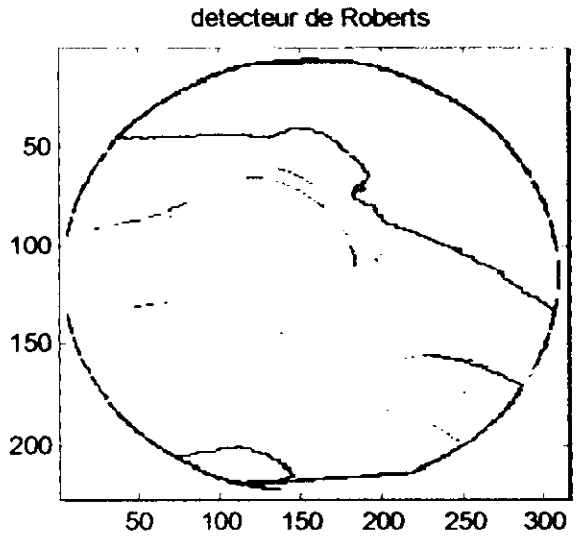
Interprétation des résultats

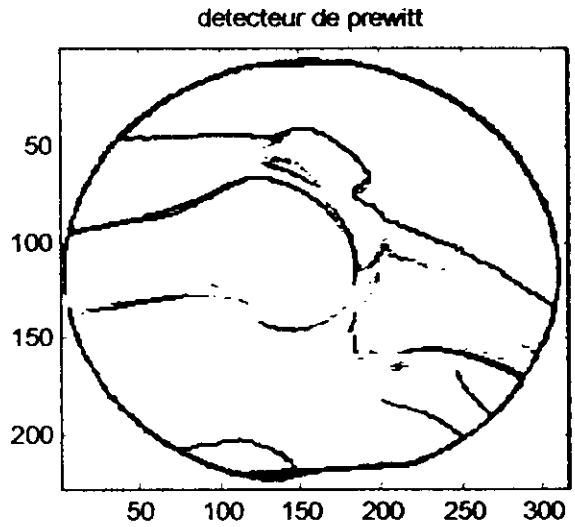
L'application d'une érosion puis d'une dilatation (*2), nous permet d'éliminer une partie du bruit présent dans l'image (du aux taches).
Il en résulte que moins de bruit est détecté lors de la détermination des contours (Totalement disparu pour s=230)

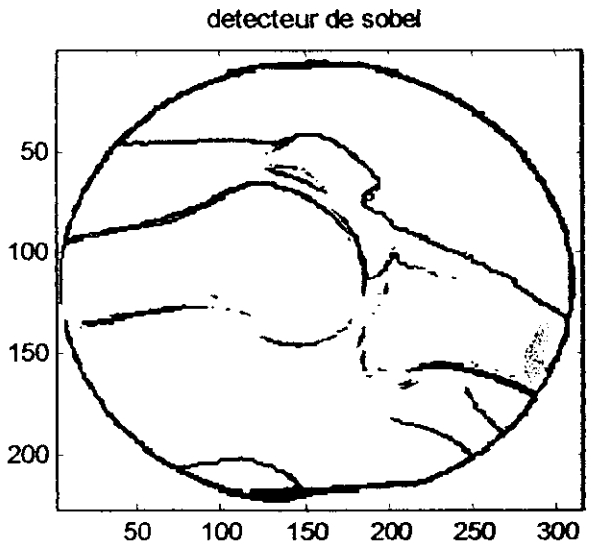
3.2.2. Image 'os.bmp'

Fichier os1_egal.m (os1_egal(s), s: seuil)

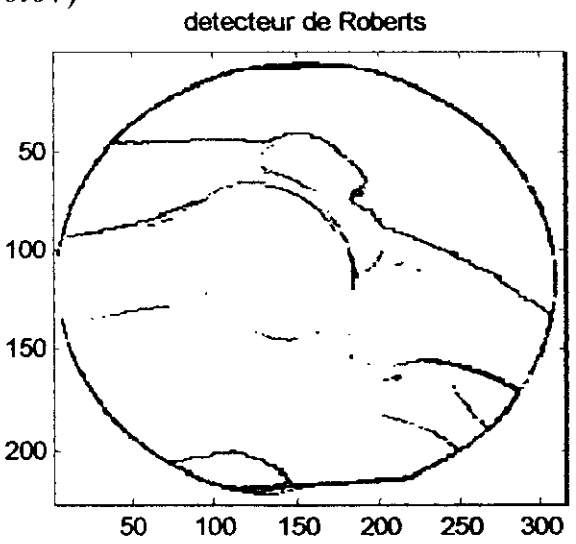
os1_egal (0.1)

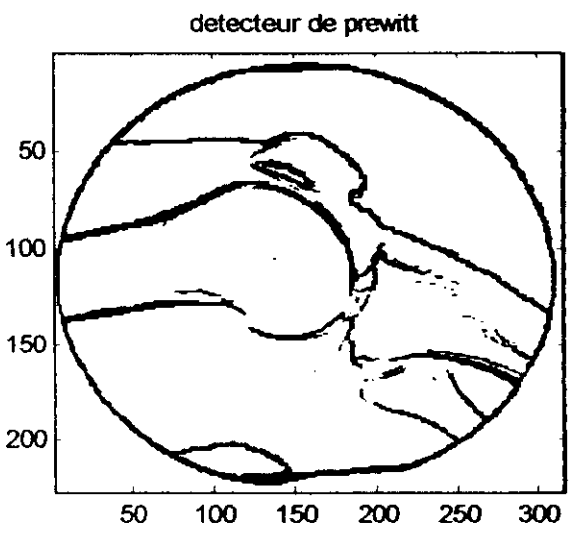


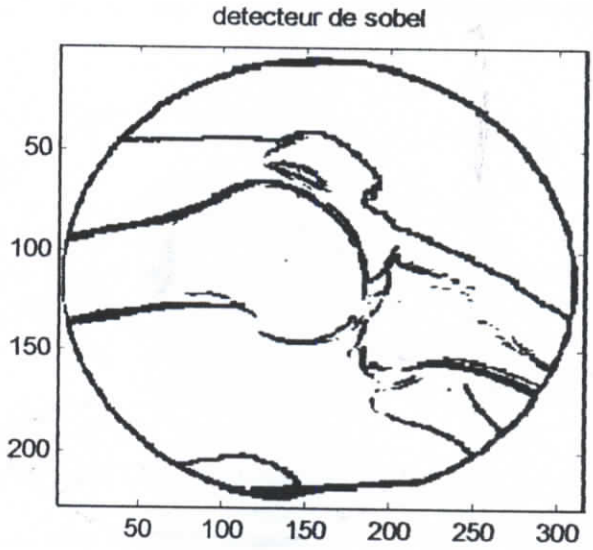


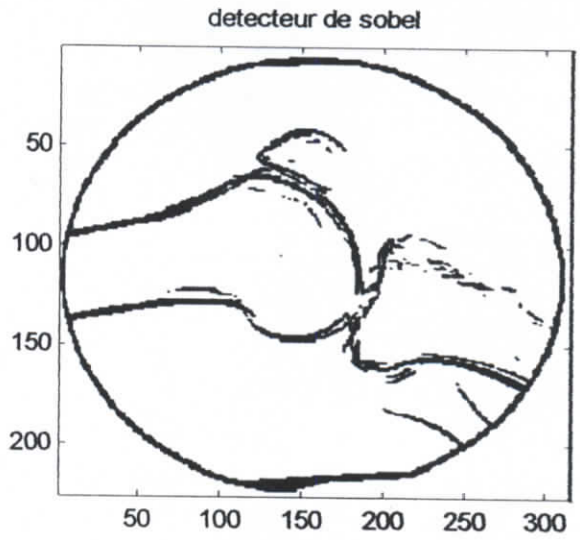


os1_egal (seuil=0.07)



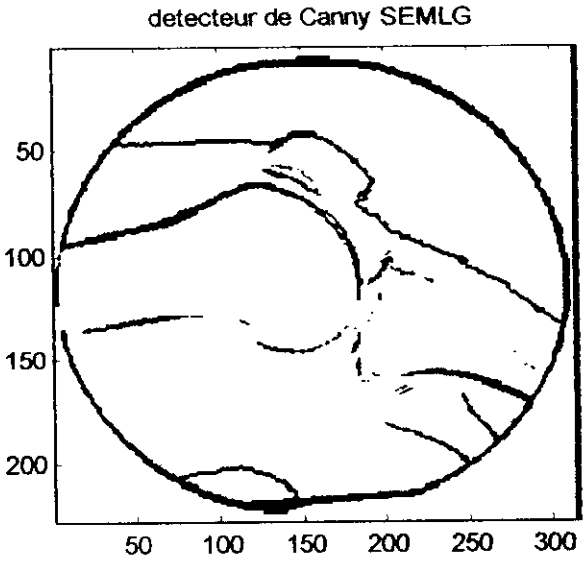


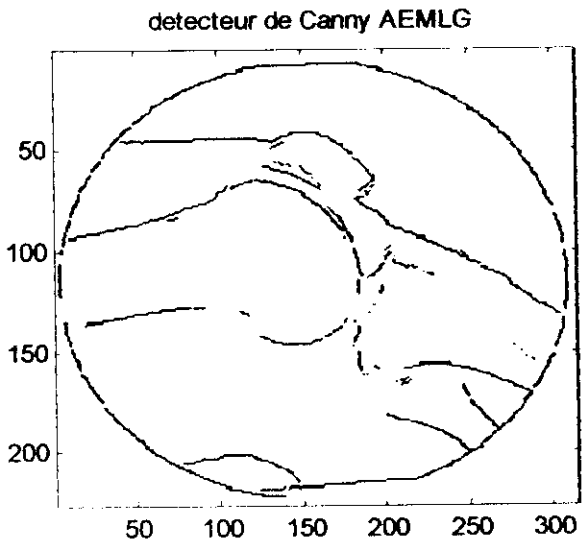

detecteur de sobel
50
100
150
200
50
100
150
200
250
300


detecteur de sobel
50
100
150
200
50
100
150
200
250
300

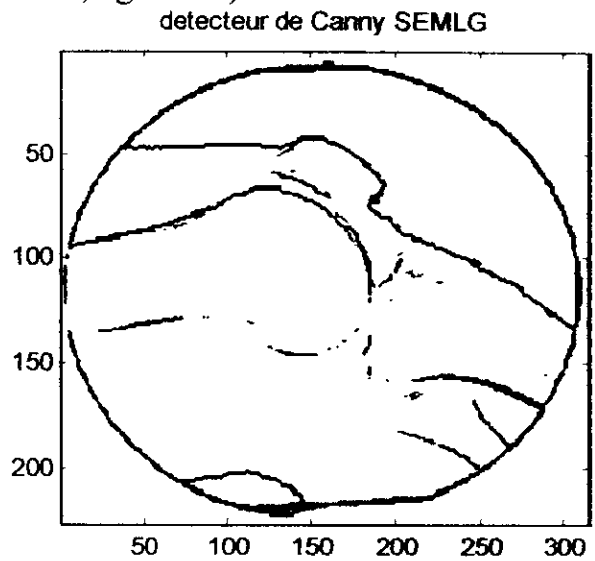
fichier os21_egal.m (os21_egal(s,sig))

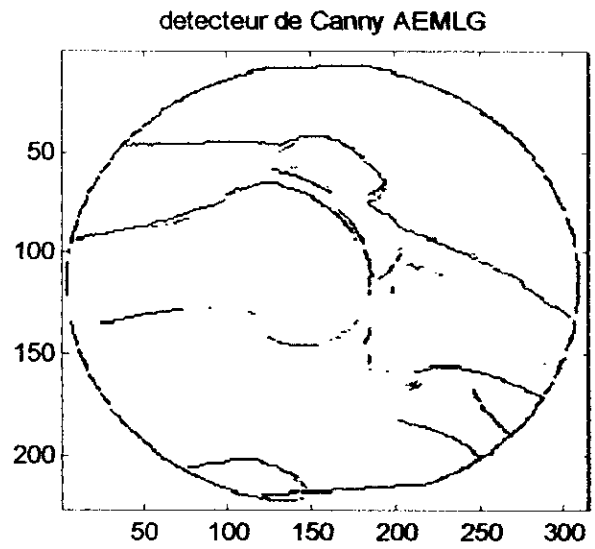
os21_egal (seuil=0.1, sig=0.5)





os21_egal (seuil=0.1,sig= 0.07)





Interprétation des résultats (visuelle)

Pour l'image 'os.bmp', on effectue une égalisation d'histogramme, puis on applique les opérateurs de Roberts, Prewitt, Sobel et Canny.
Cette opération met en valeur des parties de l'image qui n'étaient pas détectées auparavant.
Cette opération bénéficie surtout à la mise en évidence de la rotule.
On remarque que pour seuil=0.07 le bruit détecté par les opérateurs de Prewitt et Sobel à « l'intérieur » de l'os de gauche a diminué.

Conclusion

A partir de programmes faisant appel aux fonctions créées, nous avons essayé de comparer différentes méthodes d'extraction de contours, sur les différentes images dont nous disposions.

Dans une première étape, nous effectuons une détection de contours sur les images originales. Les détecteurs du premier ordre donnent de bons résultats comparés aux résultats de l'ensemble des détections effectuées (visuellement). Pour une même image le critère que l'on fait varier pour ces détecteurs est le seuil d'extraction de la norme du gradient.

Pour l'image choisie 'muscle.bmp', on remarque que les détecteurs du second ordre sont plus sensibles au bruit (points apparaissant à l'intérieur des cellules).Ceci est du en partie à la nature de l'image originale.

Les détecteurs optimaux (Canny et Deriche) sont en général moins sensibles au bruit comparés aux autres détecteurs.

L'extraction de contours, lorsque l'image (ou l'application) nous permet une binarisation de cette dernière nous permet d'aboutir directement à des contours d'une épaisseur d'un pixel, et fermés en général (l'objet de l'image dont il faut définir le contour doit être défini).

Dans une deuxième étape nous avons tenté d'appliquer des prétraitements aux images choisies à cet effet. Pour l'image nous sommes parvenus à éliminer une partie du bruit.

Pour donner des résultats plus objectifs il faudrait effectuer une étude statistique de l'image (d'où l'avantage de disposer des valeurs des matrices images originale et traitée), orientée suivant l'objectif recherché, c'est-à-dire que, par exemple, pour l'image 'chromo.bmp', il pourrait s'agir d'extraire le pourtour extérieur du chromosome ou le pourtour et les bandes caractérisant ce dernier.

. Les interprétations données sont donc essentiellement d'ordre visuel et sont subjectives.

Conclusion générale

Le but de ce travail est de proposer un ensemble d'outils permettant le choix d'un détecteur de contours. Une image est riche en informations, les contours constituent des indices pertinents.

L'application des masques de Prewitt, Roberts, Sobel sont des méthodes classiques et simples à mettre en œuvre et sont moins sensibles au bruit que les détecteurs basés sur la dérivée seconde. Par contre, leur taille est fixe, le recours aux filtres de Deriche et Canny nous permet de remédier à ce problème.

Pour parvenir à des interprétations plus objectives, il conviendrait peut être, d'effectuer une étude statistique plus approfondie de l'image, et n'abordant pas forcément celle-ci d'un point de vue global. Cette étude serait soumise aux objectifs à atteindre et aux moyens (images, matériel) dont nous disposons. Une bonne détection serait relative à la définition des contours (et du bruit) que nous donnons pour l'image.

L'ensemble de ces fonctions nécessiterait donc d'être enrichi en complétant la segmentation par approche frontière, en rajoutant d'autres opérateurs de détection de contours et en ne nous limitant pas à une approche utilisant uniquement les niveaux de gris, mais tenant compte de la texture de l'image qui caractérise beaucoup d'entre elles. Nous pourrions alors réaliser un logiciel complet d'aide à la détection de contours.

Bibliographie

Articles

[1] C. ACHARD-« Cours de traitement d'images-2002/2003 », rédigé par C.ACHARD sur la base du cours de J.DEVARS, 2002
(www.dearesin.cicrp.jussieu.fr/dearesin/module7/cours%20vision.pdf)

[2] L.CHEN et J.Y AULOGE, « Concepts fondamentaux de l'image numérique », 2003
(http://www.deasia/Ana-index.pdf)

[3] L.DELAHOCHE et E.BRASSARD, « Introduction au traitement d'images », 2003

[5] J.FRUITET « Outils et méthodes pour le traitement des images »,2004

[7] C.KADDOUR, « Généralités sur le traitement d'images », 1999.
(http ://iquebec.ifrance.com/kadchakib/chap1/chap1.htm)

[9] C.GIOVANNY, « Détection de contours, détection des points d'intérêt »,2003
(lith.www.epfl.ch/teaching/rdf/exercices03/pdf/Chapitre01_2003.pdf).

[10] S.BOYER, « Détection de contours », 2003.
(http://www.inrialpes.fr/movi/people/Boyer/Teaching/DESS-CI/ASI/c9.pdf).

[11] F.DEVERNAY, « vision par ordinateur: vision pré attentive, détection de contours », 2003.
(http:// Devernay.free.fr/cours/vision/ensg2003.html).

Ouvrages

[4] J.P.COCQUEREZ et S.PHILIPP, « Analyse d'images : filtrage et segmentation », MASSON, Paris, 1995.

[6] A.MARION, « Introduction aux techniques de traitement d'images », EYROLLES, 1987.

[8] P.COTTET, « 1 week end pour comprendre & utiliser l'image et le graphisme », OSMAN EYROLLES MULTIMEDIA, 1999.

1. Formats de fichiers

Pour sauvegarder les données créées, un programme utilise une formule de codage des éléments présents en mémoire qui, une fois écrits physiquement sur le disque, deviendra le format du fichier.

La plupart du temps ce codage vise à réduire l'encombrement pris par l'image sur le disque, donc à faire que son poids soit inférieur à sa taille en mémoire

1.1. BMP

BMP est le format d'image standard Windows pour les ordinateurs compatibles DOS et Windows. Il prend en charge les modes RVB, Couleurs indexées, Niveaux de gris et Bitmap mais pas les couches alpha. Vous pouvez préciser le format Microsoft® Windows ou OS/2® ainsi que la profondeur (résolution) en bits de l'image. Pour les images 4 et 8 bits pour Windows, vous pouvez également choisir la compression RLE.

Enregistrement non compressé de l'image (poids=taille)

1.2. GIF

Le format GIF (Graphics Interchange Format) est le format de fichier le plus souvent utilisé pour afficher des graphiques et des images à couleurs indexées dans des documents HTML sur Internet et d'autres services en ligne. GIF est un format compressé LZW conçu pour minimiser la taille et la durée de transfert des fichiers. Il conserve les zones transparentes dans des images à couleurs indexées mais ne prend pas en charge les couches alpha.

Ce format ne peut coder les images sur 1 à 8 bits

1.3. JPEG

Le format JPEG (Joint Photographic Experts Group) est souvent utilisé pour afficher des photos et des images en tons continus dans des documents HTML sur Internet et d'autres services en ligne. Il prend en charge les modes colorimétrique CMJN, RVB et Niveaux de gris, mais pas les couches alpha. A la différence du format GIF, le format JPEG conserve les informations de couleurs d'une image RVB mais compresse la taille du fichier en éliminant des données de façon sélective.

compressent les images en supprimant des détails. Les techniques les plus courantes sont les suivantes :

• La compression RLE par codage des répétitions est une technique de compression sans perte prise en charge par les formats de fichiers Photoshop et par des formats de fichiers Windows courants.

• La compression LZW (Lemple-Zif-Welch) est une technique de compression sans perte prise en charge par les formats de fichiers TIFF, PDF, GIF et de langage PostScript. Cette technique est particulièrement utile pour compresser des images qui contiennent de grandes zones monochromes, comme des captures d'écran ou des dessins simples.

• La compression JPEG (Joint Photographic Experts Group) est une technique de compression avec perte prise en charge par les formats de fichiers JPEG, TIFF, PDF et de langage PostScript. La compression JPEG donne de meilleurs résultats avec les images en tons continus telles que les photographies.

Si vous utilisez la compression JPEG, vous pouvez spécifier la qualité de l'image en choisissant une option dans la liste Qualité, en déplaçant le curseur Qualité ou en entrant une valeur comprise entre 1 et 12 dans la zone Qualité. Pour obtenir la meilleure impression possible, choisissez une compression de qualité maximale. Les fichiers codés en JPEG peuvent être imprimés uniquement sur des imprimantes PostScript Niveau 2. Ils ne peuvent pas être séparés en plusieurs plaques.

• Le codage CCITT est une famille de techniques de compression sans perte pour les images en noir et blanc. Il est pris en charge par les formats de fichiers PDF et le langage PostScript. CCITT est l'acronyme de Comité Consultatif International Télégraphique et Téléphonique.

• Le codage ZIP est une technique de compression sans perte prise en charge par les formats PDF et TIFF. Comme LZW, la compression ZIP est plus efficace pour des images comportant de grandes zones monochromes.

• (ImageReady) Le codage PackBits est une technique de compression sans perte qui utilise un algorithme de compression de données répétitives. PackBits est pris en charge par le format TIFF dans ImageReady uniquement.

Une image JPEG est automatiquement décompressée à l'ouverture. La qualité de l'image est inversement proportionnelle au niveau de compression. En général, si vous choisissez la qualité maximale, l'image obtenue est identique à l'originale.

Le format jpeg est le standard internet pour les images 24 bits

1.4. TIFF

Le format TIFF (Tagged-Image File Format) permet d'échanger des fichiers entre applications et plates-formes informatiques. TIFF est un format d'image bitmap flexible pris en charge par la plupart des applications de dessin, de retouche d'images et de mise en page. De même, la plupart des scanners de bureau peuvent produire des images TIFF.

Le format TIFF prend en charge les fichiers CMJN, RVB, Lab, à couleurs indexées et en niveaux de gris avec des couches alpha et des fichiers bitmap sans couche alpha. Photoshop peut enregistrer des calques dans un fichier TIFF. Toutefois, si vous ouvrez le fichier dans une autre application, seule l'image aplatie est visible. Photoshop permet également d'enregistrer des annotations, des zones de transparence et de données à structure pyramidale multi-résolution dans un fichier au format TIFF.

1.5. PNG

Le format PNG (Portable Network Graphics) a été mis au point sans brevet pour concurrencer le format GIF. Il permet notamment de compresser des images sans perte et de les afficher sur le Web. Contrairement à GIF, le format PNG prend en charge les images 24 bits et produit une transparence de fond sans bords dentelés. Néanmoins, certains navigateurs Web peuvent ne pas prendre en charge les images PNG. Le format PNG prend en charge les modes de couleur RVB, à couleurs indexées, en niveaux de gris et en mode Bitmap sans couches alpha. Il conserve les zones transparentes dans les images en niveaux de gris et RVB.

2. Compression de fichiers

De nombreux formats de fichiers utilisent des techniques de compression pour réduire le volume occupé sur le disque par les données d'images bitmap. Ces techniques se distinguent selon qu'elles suppriment ou non des détails et des couleurs de l'image. Les techniques *sans perte* compressent les images sans en supprimer des détails ; les techniques *avec perte*

1. Approche de Canny

Soit *A(x)* un signal monodimensionnel représentant un saut d'amplitude *Uo* , noyé dans un bruit blanc stationnaire *N(x)* de moyenne nulle et de densité de puissance(d.s.p) No^2 :

$$A(x) = U_0 U(x) + N(x)$$

Où $\begin{cases} U(x)=1 \ \ si\ x>=0 \\ U(x)=0 \ \ autrement \end{cases}$

Le signal de sortie est donné par l'expression

$$C(x) = A * h(x) = \int_{-\infty}^{+\infty} A(t)h(x-1)dt$$

Il s'agit de déterminer *h(x)* tel que *C(x)* soit maximum au point *x=0* en respectant les 3 contraintes :

- bonne détection
- bonne localisation
- faible multiplicité des maximums dus au bruit.

1. bonne détection

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

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

Le filtre recherché est un dérivateur, *h(x)* est choisie impaire.

On se place ensuite en x=0, on a alors :

$$RSB = \frac{U_0 \int_{-\infty}^{0} h(t)dt}{N_0 \left(\int_{-\infty}^{+\infty} h^2(t)dt\right)^{1/2}} = \frac{u_0}{N_0}\Sigma$$

2. bonne localisation

Ce critère est mesuré par l'inverse de la variance de la distance entre le maximum de la réponse et la position réelle de la transition. Ce critère doit donc être maximisé.

Pour le calculer, on cherche exprimer $E\{x_0^2\}$, x_0 étant la position calculée de la transition du signal de sortie, donc à un passage par zéro de sa dérivée première puisque le filtre étudié est un dérivateur.

On a :

$$\left[\frac{\partial C}{\partial x}\right]_{x_0} = \left[\frac{\partial}{\partial x} A * h(x)\right]_{x_0} = [A * h'(x)]_{x_0} = 0$$

d'où: $$\left[\frac{\partial C}{\partial x}\right]_{x_0} = U_0[u * h'(x)]_{x_0} + [N * h'(x)]_{x_0}$$

$$= U_0 \int_{-\infty}^{+\infty} u(x_0 - t)h'(t)dt + [N * h'(x)]_{x_0}$$

$$= U_0 \int_{-\infty}^{x_0} h'(t)dt + [N * h'(x)]_{x_0}$$

$$= U_0 h(x_0) + [N * h'(x)]_{x_0}$$

En développant l'expression et à l'aide de développement limité de la fonction ***h(x)*** au voisinage de x=0 on aboutit à l'expression :

$h(x_0) \approx h(0) + x_0 h'(0) = x_0 h'(0)$ puisque ***h(0)=0***

Il s'en suit que :

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

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

L'expression ci-dessus devient :

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

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

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

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

Le produit $\sum.\Lambda$ est un critère qui combine une bonne détection et une bonne localisation.

Il ne dépend pas de l'amplitude *Uo* de l'échelon, ni du facteur d'échelle K.

Pour démontrer cette dernière propriété, on pose : $h_K(x) = h\left(\frac{x}{K}\right)$ et on obtient :

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

D'où : $\Sigma_\kappa . \Lambda_\kappa = \Sigma . \Lambda$

3. Non multiplicité des réponses

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

$$d_0 = \frac{1}{\pi}\left(\frac{-R_{BB}''(0)}{R_{BB}(0)}\right)^{\frac{1}{2}} \quad \ldots\ldots\ldots\ldots (**)$$

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

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

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

Nous pouvons écrire :

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

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

$$d_0 = \frac{1}{\pi}\left(\frac{\int_{-\infty}^{+\infty} h''^2(t)\,dt}{\int_{-\infty}^{+\infty} h'^2(t)\,dt}\right)^{\frac{1}{2}} \text{ et on pose : } x_{max} = \frac{1}{d0}$$

Canny a choisi de prendre un filtre à réponse impulsionnelle finie de taille M. Ceci revient à remplacer, dans l'expression des critères, les bornes $-\infty + \infty$ par $-M$ et $+M$. Ensuite, il a imposé comme troisième contrainte que la distance entre 2 maximums consécutifs de la réponse due au bruit seul soit égale à une fraction de M. Ce critère correspond à la limitation du nombre de maximums locaux détectés en réponse à un seul contour. x_{max} est alors donné par :

$$x_{max} = 2.x_{moy} = 2\pi \left(\frac{\int_{-M}^{M} h'^2(t)\,dt}{\int_{-M}^{M} h''^2(t)\,dt} \right)^{\frac{1}{2}} = K_M.M$$

La maximisation du produit $\Sigma . \Lambda$ pour un x_{max} égal à $k_m M$ revient à trouver *h(x)* qui minimise la fonctionnelle :

$$\psi(x, h, h', h'') = h^2 + \lambda_1 h'^2 + \lambda_2 h''^2 + \lambda_3 h$$

La solution optimale *h(x)* satisfaisant les conditions de continuité et de dérivabilité est obtenue par calcul variationnel comme solution de l'équation d'Euler :

$$\psi_h - \frac{\partial}{\partial x}\psi_{h'} + \frac{\partial^2}{\partial x^2}\psi_{h''} = 0 \quad \text{où} \quad \psi_h = \frac{\partial \psi}{\partial h}$$

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

$$2h(x) - 2\lambda_1 h''(x) + 2\lambda_2 h''''(x) + \lambda_3 = 0$$

Qui admet comme solution générale :

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

avec :

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

Pour trouver l'opérateur *h(x)* sous la forme d'un filtre à réponse impulsionnelle finie (RIF) défini sur l'intervalle [-M, M] et présentant une pente S à l'origine Canny a imposé les conditions aux limites suivantes :

$$h(0) = 0; \quad h(M) = 0; \quad h'(0) = S; \quad h'(M) = 0;$$

Ces 4 conditions aux limites permettent de déterminer les coefficients a_1 à a_4. h(x) étant impaire, la solution est étendue aux x négatifs par *h(x)=-h(-x)*.

En utilisant une technique d'optimisation numérique sous contrainte, Canny démontre que l'opérateur le plus performant correspond à un produit $\Sigma . \Lambda$ =1.12.

Une démarche intéressante pour caractériser un filtre dérivateur peut consister à étudier la probabilité d'un maximum erroné dans le voisinage de la position exacte de la transition. Canny a établi la relation suivante :

$$\frac{|h'(0)|}{\left(\int_{-\infty}^{\infty} h''^2(t)\,dt\right)^{\frac{1}{2}}} = K_s \frac{\int_{-\infty}^{0} h(t)\,dt}{\left(\int_{-\infty}^{\infty} h^2(t)\,dt\right)^{\frac{1}{2}}}$$

Où K_s est une constante déterminée par les valeurs de probabilité P_s et P_f qui permet de les situer l'une par rapport à l'autre.

Pour k_s=1, les probabilités sont égales.

Pour l'opérateur dérivée première d'une gaussienne, on obtient k_s=0.51 ; ce qui correspond à une dégradation d'environ 10%.

2. Opérateur de Dériche

2.1. Opérateur monodimensionnel de Dériche

En utilisant la même démarche que Canny, Deriche a cherché une réalisation de l'opérateur sous la forme d'un filtre à réponse impulsionnelle infinie (RII). Il a abouti à la même équation différentielle, seules les conditions aux limites sont différentes $M \rightarrow +\infty$

La solution est alors :

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

La solution est alors :

$$h(x) = c.e^{-\alpha|x|} \sin \omega.x$$

En évaluant, pour cet opérateur, les différentes intégrales intervenant dans le calcul des critères de performance, on obtient les résultats suivants :

$$\Lambda = \sqrt{2\alpha} \qquad \Sigma = \sqrt{\frac{2\alpha}{\alpha^2 + \omega^2}}$$

$$\Sigma.\Lambda = \frac{2\alpha}{\sqrt{\alpha^2 + \omega^2}} \qquad K_s = \sqrt{\frac{\alpha^2 + \omega^2}{5\alpha^2 + \omega^2 + 6\alpha^2\omega^2}}$$

En posant $\alpha = m\,\omega$, on obtient les 3 cas suivants :

a) $m \rangle\rangle 1; \quad \Lambda = \sqrt{2\alpha}\ ; \Sigma = \sqrt{\frac{2}{\alpha}}\ ; \ \Sigma.\Lambda = 2; \quad K_s = 0{,}44$

b) $m = 1; \quad \Lambda = \sqrt{2\alpha}\ ; \Sigma = \sqrt{\frac{1}{\alpha}}\ ; \ \Sigma.\Lambda = \sqrt{2}; \quad K_s = 0{,}58$

c) $m = \sqrt{3}$; $\Lambda = \sqrt{2\alpha}$; $\Sigma = \sqrt{\frac{3}{2\alpha}}$; $\Sigma\Lambda=\sqrt{3}$; $K_s = 0{,}5$

Le premier cas présente le meilleur indice de performance. Il correspond à la limite de l'opérateur de Dériche pour ω tendant vers zéro. Cette limite correspond à l'opérateur *h(x)* donné par

$$h(x) = c.x.e^{-\alpha|x|}$$

Opérateurs bidimensionnels de dérivation et de lissage de Dériche

En utilisant l'opérateur optimal de Deriche, on peut mettre en œuvre les techniques générales énoncées précédemment. On calcule alors le gradient en chaque point de l'image. Pour améliorer l'immunité au bruit on effectue d'abord un lissage.

- **Lissage**

Le filtre utilisé est la combinaison de deux filtres monodimensionnels dans les directions x et y.

Le filtre de lissage monodimensionnel retenu par Dériche est l'intégrale *f(x)* du filtre optimal $h(x) = c.x.e^{-\alpha|x|}$. On a :

$$f(x) = b(\alpha|x| + 1)e^{-\alpha|x|}$$

b est calculé pour donner une réponse constante de valeur 1 pour un signal d'entrée constant de niveau 1 ; on obtient $b = \frac{\alpha}{4}$.

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

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

Si l'image originale est notée A(x, y), on aura pour l'image lissée l'expression suivante :

$$B(x,y) = A * f(x,y)$$

- **Calcul du gradient**

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

Compte tenu des règles de dérivation de l'opération de convolution et de la séparabilité du filtre f(x, y) nous avons par exemple :

$$\frac{\partial B}{\partial x}(x,y) = B_x(x,y) = \frac{\partial\{A * f\}}{\partial x}(x,y) = A * \frac{\partial f}{\partial x}'x,y)$$

Les opérateurs de dérivation suivant x et y se mettent sous la forme :

$$f_x(x, y) = \eta . x . e^{-\alpha|x|}.(\alpha|y| + 1)e^{\alpha|y|}$$
$$f_y(x, y) = \eta . y . e^{-\alpha|y|}.(\alpha|x| + 1)e^{\alpha|x|} \quad (\Lambda)$$

Où η est une constante de normalisation calculée, par exemple pour l'équation de façon à fournir un maximum d'amplitude 1 en réponse à une transition verticale unitaire. On trouve alors que $\eta = -\frac{\alpha^3}{4}$; donc on aura $f_x(x, y) = h(x).f(y)$. Par conséquent, l'image de la dérivée directionnelle en x s'écrit : $B_x(x, y) = (A * h(x)) * f(y)$

Ceci veut dire que la dérivée directionnelle en x est le résultat d'un lissage suivant la direction y, suivi par une dérivation suivant x. les filtres sont monodimensionnels, l'implémentation récursive est aisée.

Pour le filtre décrit par l'équation (Λ), on procède de même avec en entrée une transition horizontale unitaire.

En général, après le calcul de la norme, on procède à une extraction des maximums locaux et un seuillage par hystérésis permettent d'obtenir des contours fins

2.2. Opérateur laplacien de Dériche

On peut calculer le laplacien pour la détection de contours. En effectuant le calcul sur une image lissée, on obtient :

$$\Delta B(x, y) = A * \Delta f(x, y)$$

Comme $\Delta f(x, y) = \frac{\partial f_x}{\partial x} + \frac{\partial f_y}{\partial y}$

L'opérateur laplacien de Dériche s'exprime sous la forme d'un opérateur bidimensionnel dérivée seconde qui est la différence de deux fonctions de transfert séparables :

$\Delta f(x, y) = e^{-\alpha|x|}.e^{-\alpha|y|} - k\alpha|x|e^{-\alpha|x|}.\alpha|y|e^{-\alpha|y|}$, (le premier terme correspond à un filtre de lissage et le second à un filtre dérivateur)

Le calcul du laplacien d'une image *A(x, y)* est obtenu par simple soustraction entre deux images traitées. L'opération peut alors être mise en œuvre de manière récursive.

1. Généralités

1.1. Types de fichiers graphiques

MATLAB accepte les formats de fichiers graphiques suivants :

•BMP (Microsoft Windows Bitmap)

•HDF (Hierarchical Data Format)

•JPEG (Joint Photographic Experts Group)

•PCX (Paintbrush)

•PNG (Portable Network Graphics)

•TIFF (Tagged Image File Format)

•XWD (X Window Dump)

1.2. Types d'images (Types d'image dans la « Boite à outils de traitement d'images » « The Image Processing Toolbox » IPT)

L'IPT supporte 4 types d'images :

- Les images niveaux de gris (d'intensité),
- Les images binaires,
- Les images RGB (en vraies couleurs),
- Les images indexées.

Représentation de chaque type d'images sous MATLAB

1.2.1. Images en niveaux de gris

Il s'agit d'une matrice de données I, chaque coefficient représente la valeur d'intensité dans un intervalle donné. MATLAB stocke ce type d'image comme une matrice dont chaque élément correspond à un pixel de l'image. La matrice peut être de classe **double, uint8, or uint16**. Ce type d'images étant rarement sauvegardé avec une palette de couleurs, MATLAB utilise une palette de couleurs pour les afficher.

Les éléments de cette matrice représentent différentes intensités ou niveaux de gris où la valeur **0** représente le **noir** habituellement, et la valeur **1, 255 ou** 65535 le **blanc**.

1.2.2. Images vraies couleurs (RGB)

Une image RGB (Red, Green, Blue), ou vraies couleurs est stockée sous MATLAB comme un *m*-by-*n*-by-3 tableau de données définissant les composantes rouge, verte et bleue pour chaque pixel.

Ce type d'images n'utilise pas de palette de couleurs. La couleur de chaque pixel est déterminée par la combinaison de s intensités rouge, verte et bleue stockées pour chaque pixel dans chaque plan couleur pour la position de chaque pixel.

Les formats de fichiers graphiques stockent l'image sur 24-bit images, où les composantes rouge, verte et bleue sont codées sur 8 bits chacune, soit un potentiel de 16 millions de couleurs, d'où l'appellation d'« images en vraies couleurs »

Un tableau RGB MATLAB peut être de classe double, uint8, or uint16.

(pour la classe double, chaque composante couleur a une valeur comprise entre 0 et 1)

1.2.3. Images binaires

Dans une image binaire chaque pixel ne peut prendre qu'une des deux valeurs 0 ou 1.

Une image binaire est stockée comme une matrice à 2 dimensions.

Il s'agit d'un cas spécial d'image d'intensité contenant seulement le blanc et le noir ou d'une image indexée contenant 2 couleurs.

Elle peut être de classe **double** ou **uint8**.

1.2.4. Images indexées

Il s'agit de matrices de données X et d'une palette de couleur, map. Map est un tableau de dimensions *m*-by-3 de classe double contenant des valeurs en virgule flottante dans l'intervalle [0,1].

Une palette de couleurs est souvent stockée avec une image indexée et est automatiquement chargée lors de l'utilisation de la fonction **imread.**

1.4. Gestion des bords de l'image sous matlab

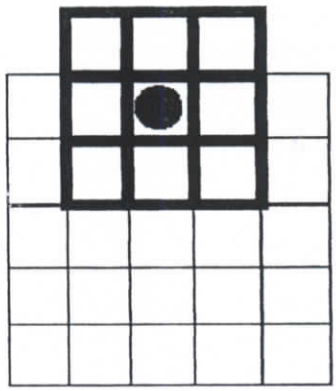


Figure 1: Application d'un masque à un pixel de l'image

Afin de calculer les valeurs de sortie pour les pixels du bord de l'image, la fonction **conv2** rajoute à la matrice image des zéros.en d'autres termes, les valeurs desortie sont calculées en assumant le fait que l'image d'entrée est entourée de lignes et de colonnes additionnelles de zéros.

Dans la figure ci-dessous, les éléments du dessus de l'image sont considérés comme des zéros.

Suivant le traitement à effectuer il s'agira d'ignorer ou de rajouter des pixels à l'image.

3 possibilités existent:

•**'Valid'** – retourne seulement les pixels dont les valeurs peuvent être calculées sans "zero padding" de l'image d'entrée. (Taille de l'image sortie< Taille de l'image d'entrée).

•**'Same'** – retourne le jeu de pixels qui peut être calculé en appliquant le filtre à tous les pixels faisant partie actuellement de l'image d'entrée. Les pixels du bord sont calculés en utilisant le "zero padding", mais le pixel central du masque est seulement appliqué aux pixels de l'image. (Taille de l' image sortie= Taille de l'image d'entrée).

•**'Full'** – retourne toute la convolution, ie que la fonction conv2 retourne tous les pixels pour lesquels chaque pixel du masque recouvre un pixel de l'image, même si le centre du masque est à l'extérieur de l'image d'entrée.

La fonction conv2 retourne la convolution entière par défaut.

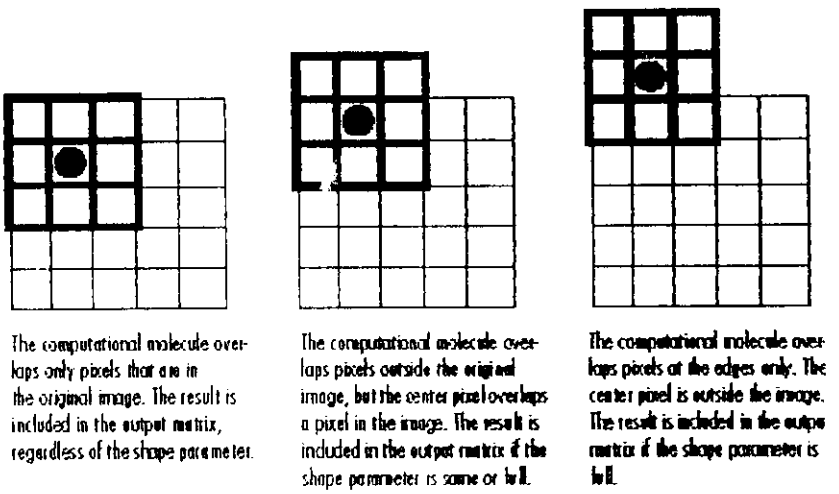


Figure 2: options pour la gestion des bords

1.5. Filtrage linéaire

1.5.1. Fonction 'filter2'

La fonction MATLAB filter2 pour le filtrage linéaire bidimensionnel peut produire les mêmes résultats que la fonction conv2,la première différence réside dans le fait que le masque est pris comme entrée de la fonction l'opération effectuée est la corrélation.

1.5.2. Filtrage linéaire et filtre

Si k est la convolution de kernel, h est la "computational molecule" correspondante, et si A est la matrice image les 2 appels de fonction suivants produiront le même résultat :

- B = conv2(A,k,'same');
- Et B = filter2(h,A,'same');

1.5.3. Système de coordonnées

La position dans une image peut être exprimée à partir de plusieurs systèmes de coordonnées dépendants du contexte. Ces systèmes sont :

•Le système de coordonnées des pixels

•Le système de cordonnées spatiales

Coordonnées des pixels

En général la manière la plus appropriée pour exprimer la position dans une image est l'utilisation du système de coordonnées des pixels. Dans ce système, l'image est traitée comme une grille d'éléments discrets ordonnés du haut vers le bas et de la gauche vers la droite.

Il existe une correspondance entre les coordonnées des pixels et les cordonnées MATLAB des données de la matrice souscrite. La manière d'afficher est plus facile à comprendre.

Cordonnées spatiales

Dans le système de coordonnées de pixels, chaque pixel est une entité discrète identifiable par une paire unique de coordonnées.

Parfois le pixel peut être considéré comme un 'square patch', ayant une surface.

Dans cette perspective, des coordonnées telles que (5.3, 2.2) ont un sens et sont distinctes des coordonnées (5,2).Dans le système de coordonnées spatiales, la localisation dans une image dans un plan. Ils sont décrits en termes de x et y.

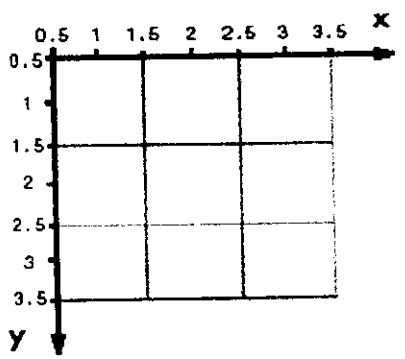


Figure 3: Illustration des coordonnées spatiales

Ce mode de représentation est intéressant (correspondance directe entre le système de coordonnées des pixels et le système de coordonnées spatiales)

Il existe cependant quelques différences telles que l'origine des systèmes et le fait que l'un soit discret et l'autre continu.

Une autre différence importante est la convention de l'ordre dans lequel sont données les coordonnées (inversé).

2. Fonctions matlab utilisées

(Seule une partie de certaines fonctions matlab ont été traduites afin de ne pas modifier l'utilisation de ces fonctions- extrait de [images_tb.pdf])

2.1 Imagesc

scale data et affichage d'une image

Syntaxe

imagesc(C)
imagesc(x,y,C)
imagesc(...,clims)

description

La fonction `imagesc` scales les données image dans l'intervalle entier de la palette de couleur courante et affiche l'image.
`imagesc(C)` affiche `C` comme une image. Chaque élément de `C` correspond à une surface rectangulaire dans l'image. Les valeurs des éléments de C correspondent à des indices dans la palette courante qui déterminent la valeur de chaque 'patch'.
`imagesc(x,y,C)` affiche `C` comme une image et spécifie the bounds des axes *x*- et *y*- avec les vecteurs `x` et `y`.
`imagesc(...,clims)` normalise les valeurs dans `C` dans l'intervalle spécifié par `clims`, et affiche `C` comme une image. `clims` est un vecteur de 2 éléments qui limite l'intervalle des valeurs des données de C
`Remarqu`

`x` et `y` n'affectent pas les éléments dans C, ils affectent uniquement l'annotation des axes.

2.2. Image

Affichage d'une image

```
Syntaxe
image(C)
image(x,y,C)
image(...,'PropertyName',PropertyValue,...)
image('PropertyName',PropertyValue,...)  Formal synatx - PN/PV only
handle = image(...)
```

Description

`image` crée une objet graphique image en interprétant chaque élément de la matrice comme un index dans la figure de la palette ou directement en vraies couleurs en fonction des données spécifiées.
Remarque
Les données del'images peuvent etre indexées ou en vraies couleurs.

Une image indexée stocke les couleurs dans un tableau d'indices de la palette
Une image en vraies couleurs n'utilise pas de palette
*The `imread` function reads image data into MATLAB arrays from graphics files in various standard formats, such as TIFF. You can write MATLAB image data to graphics files using the `imwrite` function. `imread` and `imwrite` both support a variety of graphics file formats and compression schemes.
When you read image data into MATLAB using `imread`, the data is usually stored as an array of 8-bit integers. However, `imread` also supports reading 16-bit-per-pixel data from TIFF and PNG files. These are more efficient storage method than the double-precision (64-bit) floating-point numbers that MATLAB typically uses. However, it is necessary for MATLAB to interpret 8-bit and 16-bit image data differently from 64-bit data. This table summarizes these differences.*

Image Type	Double-precision Data (double array)	8-bit Data (uint8 array) 16-bit Data (uint16 array)
indexed (colormap)	Image is stored as a two-dimensional (`m`-by-`n`) array of integers in the range [1, `length(colormap)`]; colormap is an `m`-by-3 array of floating-point values in the range [0, 1]	Image is stored as a two-dimensional (`m`-by-`n`) array of integers in the range [0, 255] (`unit8`) or [0, 65535] (`uint16`); colormap is an `m`-by-3 array of floating-point values in the range [0, 1]
truecolor (RGB)	Image is stored as a three-dimensional (`m`-by-`n`-by-3) array of floating-point values in the range [0, 1]	Image is stored as a three-dimensional (`m`-by-`n`-by-3) array of integers in the range [0, 255] (`unit8`) or [0, 65535] (`uint16`)

Palette de couleurs

Colormaps in MATLAB are alway `m`-by-3 arrays of double-precision floating-point numbers in the range [0, 1]. In most graphics file formats, colormaps are stored as integers, but MATLAB does not support colormaps with integer values. `imread` and `imwrite` automatically convert colormap values when reading and writing files.

True Color Images

In a truecolor image of class `double`, the data values are floating-point numbers in the range [0, 1]. In a truecolor image of class `uint8`, the data values are integers in the range [0, 255] and for truecolor image of class `uint16` the data values are integers in the range [0, 65535]
If you want to convert a truecolor image from one data type to the other, you must rescale the data. For example, this statement converts a `uint8` truecolor image to `double`,

```
RGB64 = double(RGB8)/255;
```

or for `uint16` images,

```
RGB64 = double(RGB16)/65535;
```

This statement converts a `double` truecolor image to `uint8`.

```
RGB8 = uint8(round(RGB64*255));
```

or for `uint16` images,

```
RGB16 = uint16(round(RGB64*65535));
```

The order of the operations must be as shown in these examples, because you cannot perform mathematical operations on `uint8` or `uint16` arrays.
When you write a truecolor image using `imwrite`, MATLAB automatically converts the values if necessary.

2.3. conv2

Two-dimensional convolution

Syntax

```
C = conv2(A,B)
C = conv2(hcol,hrow,A)
C = conv2(...,'shape')
```

Description

`C = conv2(A,B)` computes the two-dimensional convolution of matrices `A` and `B`. If one of these matrices describes a two-dimensional FIR filter, the other matrix is filtered in two dimensions.
The size of `C` in each dimension is equal to the sum of the corresponding dimensions of the input matrices, minus one. That is, if the size of `A` is `[ma,na]` and the size of `B` is `[mb,nb]`, then the size of `C` is `[ma+mb-1,na+nb-1]`.
`C = conv2(hcol,hrow,A)` convolves `A` separably with `hcol` in the column direction and `hrow` in the row direction. `hcol` and `hrow` should both be vectors.
`C = conv2(...,'shape')` returns a subsection of the two-dimensional convolution, as specified by the *shape* parameter:

`full`	Returns the full two-dimensional convolution (default).
`same`	Returns the central part of the convolution of the same size as `A`.
`valid`	Returns only those parts of the convolution that are computed without the zero-padded edges. Using this

	option, `C` has size `[ma-mb+1,na-nb+1]` when `size(A) > size(B)`.

2.4. Filter2

Two-dimensional digital filtering

Syntax

```
Y = filter2(h,X)
Y = filter2(h,X,shape)
```

Description

`Y = filter2(h,X)` filters the data in `X` with the two-dimensional FIR filter in the matrix `h`. It computes the result, `Y`, using two-dimensional correlation, and returns the central part of the correlation that is the same size as `X`.
`Y = filter2(h,X,shape)` returns the part of `Y` specified by the `shape` parameter. `shape` is a string with one of these values:

- `'full'` returns the full two-dimensional correlation. In this case, `Y` is larger than `X`.
- `'same'` (the default) returns the central part of the correlation. In this case, `Y` is the same size as `X`.
- `'valid'` returns only those parts of the correlation that are computed without zero-padded edges. In this case, `Y` is smaller than `X`.

Remarks

Two-dimensional correlation is equivalent to two-dimensional convolution with the filter matrix rotated 180 degrees. See the Algorithm section for more information about how `filter2` performs linear filtering.

Algorithm

Given a matrix `X` and a two-dimensional FIR filter `h`, `filter2` rotates your filter matrix 180 degrees to create a convolution kernel. It then calls `conv2`, the two-dimensional convolution function, to implement the filtering operation.
`filter2` uses `conv2` to compute the full two-dimensional convolution of the FIR filter with the input matrix. By default, `filter2` then extracts the central part of the convolution that is the same size as the input matrix, and returns this as the result. If the `shape` parameter specifies an alternate part of the convolution for the result, `filter2` returns the appropriate part.

2.5. ordfilt2

Perform two-dimensional order-statistic filtering

Syntax

```
B = ordfilt2(A,order,domain)
B = ordfilt2(A,order,domain,S)
B = ordfilt2(...,padopt)
```

Description

`B = ordfilt2(A,order,domain)` replaces each element in `A` by the `order`-th element in the sorted set of neighbors specified by the nonzero elements in `domain`.
`B = ordfilt2(A,order,domain,S)`, where `S` is the same size as `domain`, uses the values of `S` corresponding to the nonzero values of `domain` as additive offsets.
`B = ordfilt2(...,padopt)` controls how the matrix boundaries are padded. Set `padopt` to `'zeros'` (the default), or `'symmetric'`. If `padopt` is `'zeros'`, `A` is padded with zeros at the boundaries. If `padopt` is `'symmetric'`, `A` is symmetrically extended at the boundaries.

Class Support

`A` can be of class `uint8`, `uint16`, or `double`. The class of `B` is the same as the class of `A`, unless the additive offset form of `ordfilt2` is used, in which case the class of `B` is `double`.

Remarks

`domain` is equivalent to the structuring element used for binary image operations. It is a matrix containing only 1's and 0's; the 1's define the neighborhood for the filtering operation.
For example, `B = ordfilt2(A,5,ones(3,3))` implements a 3-by-3 median filter; `B = ordfilt2(A,1,ones(3,3))` implements a 3-by-3 minimum filter; and `B = ordfilt2(A,9,ones(3,3))` implements a 3-by-3 maximum filter.
`B = ordfilt2(A,1,[0 1 0; 1 0 1; 0 1 0])` replaces each element in `A` by the minimum of its north, east, south, and west neighbors.
The syntax that includes `S` (the matrix of additive offsets) can be used to implement grayscale morphological operations, including grayscale dilation and erosion.

2.6. double

Convert data to double precision

Syntax

```
B = double(A)
```

Description

B = double(A) creates a double-precision array `B` from the array `A`. If `A` is a `double` array, `B` is identical to `A`.

`double` is useful if you have a `uint8` image array that you want to perform arithmetic operations on, because MATLAB does not support these operations on `uint8` data.

Remarks

`double` is a MATLAB built-in function

2.7. imread

Read images from graphics files

Syntax

```
A = imread(filename,fmt)
[X,map] = imread(filename,fmt)
[...] = imread(filename)
[...] = imread(...,idx)    (TIFF only)
[...] = imread(...,ref)    (HDF only)
[...] = imread(...,'BackgroundColor',BG)  (PNG only)
[A,map,alpha] = imread(...)  (PNG only)
```

Description

`A = imread(filename,fmt)` reads a grayscale or truecolor image named `filename` into `A`. If the file contains a grayscale intensity image, `A` is a two-dimensional array. If the file contains a truecolor (RGB) image, `A` is a three-dimensional (`m`-by-`n`-by-3) array.

`[X,map] = imread(filename,fmt)` reads the indexed image in `filename` into `X` and its associated colormap into `map`. The colormap values are rescaled to the range [0,1]. `A` and `map` are two-dimensional arrays.

`[...] = imread(filename)` attempts to infer the format of the file from its content.

`filename` is a string that specifies the name of the graphics file, and `fmt` is a string that specifies the format of the file. If the file is not in the current directory or in a directory in the MATLAB path, specify the full pathname for a location on your system. If `imread` cannot find a file named `filename`, it looks for a file named `filename.fmt`. If you do not specify a string for `fmt`, the toolbox will try to discern the format of the file by checking the file header.

This table lists the possible values for `fmt`.

Format	File type

`'bmp'`	Windows Bitmap (BMP)
`'hdf'`	Hierarchical Data Format (HDF)
`'jpg'` or `'jpeg'`	Joint Photographic Experts Group (JPEG)
`'pcx'`	Windows Paintbrush (PCX)
`'png'`	Portable Network Graphics (PNG)
`'tif'` or `'tiff'`	Tagged Image File Format (TIFF)
`'xwd'`	X Windows Dump (XWD)

Format	**Variants**
BMP	1-bit, 4-bit, 8-bit, and 24-bit uncompressed images; 4-bit and 8-bit run-length encoded (RLE) images
HDF	8-bit raster image datasets, with or without associated colormap; 24-bit raster image datasets
JPEG	Any baseline JPEG image (8 or 24-bit); JPEG images with some commonly used extensions
PCX	1-bit, 8-bit, and 24-bit images
PNG	Any PNG image, including 1-bit, 2-bit, 4-bit, 8-bit, and 16-bit grayscale images; 8-bit and 16-bit indexed images; 24-bit and 48-bit RGB images
TIFF	Any baseline TIFF image, including 1-bit, 8-bit, and 24-bit uncompressed images; 1-bit, 8-bit, 16-bit, and 24-bit images with packbits compression; 1-bit images with CCITT compression; also 16-bit grayscale, 16-bit indexed, and 48-bit RGB images.
XWD	1-bit and 8-bit ZPixmaps; XYBitmaps; 1-bit XYPixmaps

Class Support

In most of the image file formats supported by `imread`, pixels are stored using eight or fewer bits per color plane. When reading such a file, the class of the output (`A` or `X`) is `uint8`. `imread` also supports reading 16-bit-per-pixel data from TIFF and PNG files; for such image files, the class of the output (`A` or `X`) is `uint16`. Note that for indexed images, `imread` always reads the colormap into an array of class `double`, even though the image array itself may be of class `uint8` or `uint16`.

Remarks

`imread` is a function in MATLAB

2.8. Imfinfo

Return information about a graphics file

Syntax

```
info = imfinfo(filename,fmt)
info = imfinfo(filename)
```

Description

`info = imfinfo(filename,fmt)` returns a structure whose fields contain information about an image in a graphics file. `filename` is a string that specifies the name of the graphics file, and `fmt` is a string that specifies the format of the file. The file must be in the current directory or in a directory on the MATLAB path. If `imfinfo` cannot find a file named `filename`, it looks for a file named `filename.fmt`.
This table lists the possible values for `fmt`.

Format	File Type
`'bmp'`	Windows Bitmap (BMP)
`'hdf'`	Hierarchical Data Format (HDF)
`'jpg' or 'jpeg'`	Joint Photographic Experts Group (JPEG)
`'pcx'`	Windows Paintbrush (PCX)
`'png'`	Portable Network Graphics (PNG)
`'tif' or 'tiff'`	Tagged Image File Format (TIFF)
`'xwd'`	X Windows Dump (XWD)

If `filename` is a TIFF or HDF file containing more than one image e, `info` is a structure array with one element (i.e., an individual structure) for each image in the file. For example, `info(3)` would contain information about the third image in the file.
The set of fields in `info` depends on the individual file and its format. However, the first nine fields are always the same. This table lists these fields and describes their values.

Field	Value

- `'salt & pepper'` for "on and off" pixels
- `'speckle'` for multiplicative noise

`J = imnoise(I,type,parameters)` accepts an algorithm type plus additional modifying parameters particular to the type of algorithm chosen. If you omit these arguments, `imnoise` uses default values for the parameters. Here are examples of the different noise types and their parameters:
`J = imnoise(I,'gaussian',m,v)` adds Gaussian white noise of mean `m` and variance `v` to the image `I`. The default is zero mean noise with 0.01 variance.
`J = imnoise(I,'salt & pepper',d)` adds salt and pepper noise to the image I, where `d` is the noise density. This affects approximately `d*prod(size(I))` pixels. The default is 0.05 noise density.
`J = imnoise(I,'speckle',v)` adds multiplicative noise to the image `I`, using the equation `J = I + n*I`, where `n` is uniformly distributed random noise with mean 0 and variance `v`. The default for `v` is 0.04.

Class Support

The input image `I` can be of class `uint8`, `uint16`, or `double`. The output image `J` is of the same class as `I`.

2.10.nlfilter

Perform general sliding-neighborhood operations

Syntax

```
B = nlfilter(A,[m n],fun)
B = nlfilter(A,[m n],fun,P1,P2,...)
B = nlfilter(A,'indexed',...)
```

Description

`B = nlfilter(A,[m n],fun)` applies the function `fun` to each `m`-by-`n` sliding block of `A`. `fun` can be a string containing the name of a function, a string containing an expression, or an inline function object. `fun` should accept an `m`-by-`n` block as input, and return a scalar result:
`c = fun(x)`
`c` is the output value for the center pixel in the `m`-by-`n` block `x`. `nlfilter` calls `fun` for each pixel in `A`. `nlfilter` zero pads the `m`-by-`n` block at the edges, if necessary.
`B = nlfilter(A,[m n],fun,P1,P2,...)` passes the additional parameters `P1,P2,...`, to `fun`.
`B = nlfilter(A,'indexed',...)` processes `A` as an indexed image, padding with ones if `A` is of class `double` and zeros if `A` is of class `uint8`.

`Filename`	A string containing the name of the file; if the file is not in the current directory, the string contains the full pathname of the file
`FileModDate`	A string containing the date when the file was last modified
`FileSize`	An integer indicating the size of the file in bytes
`Format`	A string containing the file format, as specified by `fmt`; for JPEG and TIFF files, the three-letter variant is returned
`FormatVersion`	A string or number describing the version of the format
`Width`	An integer indicating the width of the image in pixels
`Height`	An integer indicating the height of the image in pixels
`BitDepth`	An integer indicating the number of bits per pixel
`ColorType`	A string indicating the type of image; either `'truecolor'` for a truecolor RGB image, `'grayscale'` for a grayscale intensity image, or `'indexed'` for an indexed image

`info = imfinfo(filename)` attempts to infer the format of the file from its contents.

Remarks

`imfinfo` is a function in MATLAB

2.9.Imnoise

Add noise to an image

Syntax

```
J = imnoise(I,type)
J = imnoise(I,type,parameters)
```

Description

`J = imnoise(I,type)` adds noise of type to the intensity image `I`. `type` is a string that can have one of these values:

- `'gaussian'` for Gaussian white noise

A. Prétraitements

1. Réduction de bruit (image 'test')

```
function pretraitem_nagao(d)
d=d;
I=imread('rice.tif');
I2=I(1:64,1:64)
map=palette_nvg256;
figure(1)
imagesc(I2),colormap(map),title('image test ')
I1=imnoise(I2,'salt & pepper',d);
figure(2)
imagesc(I1),colormap(map),title('image test bruitée')
Is1=filtre_moy(I1,3);
figure(3)
imagesc(Is1),colormap(map),title('image test :filtrage moyen 5*5')
[Is2]=filtre_med1(I1,3);
figure(4)
imagesc(Is2),colormap(map),title('image test :filtrage median 5*5')
[Is3,index1]=filtre_nagao(I1);
figure(5)
imagesc(Is3),colormap(map),title('image test :filtrage nagao *1')
[Is4,index2]=filtre_nagao(Is3);
figure(6)
imagesc(Is4),colormap(map),title('image test :filtrage nagao *2')
```

2. Modification d'histogramme: égalisation d'histogramme ('os.bmp')

```
function egal_os

% application image'os.bmp'
map=palette_nvg256;
I=imread('os.bmp');
I1=rgb2gray(I);
[I1]=egal_his1(I1);
```

3. Rehaussement de contraste ('route1.bmp');

```
function reh_route1(k)

k=k;
map=palette_nvg256;
I=imread('route1.bmp');
I1=rgb2gray(I);
hist1(I1);
```

```
reh_contrast(I1,k);
```

B. Détecteurs de contours

1. Image non prétraitée

1.1. Image 'muscle.bmp'

1.1.1. muscle1.m

```
function muscle1(s)

%application image'muscle.bmp'
%Detecteur approche derivee premiere
map=palette_nvg256;
I=imread('muscle.bmp');
I1=rgb2gray(I);
figure(1)
imagesc(I1),colormap(map),title('image en niveaux de gris')
s=s;
%detecteur de Sobel
[N_sob]=detect_sob(I1);
[Is_sob]=seuil_simp(N_sob,s);
figure(2)
imagesc(Is_sob),colormap(map),title('detecteur de sobel')
%detecteur de prewitt
[N_prew]=detect_prew(I1);
[Is_prew]=seuil_simp(N_prew,s);
figure(3)
imagesc(Is_prew),colormap(map),title('detecteur de prewitt')

%detecteur de roberts
[N_rob]=detect_roberts(I1);
[Is_rob]=seuil_simp(N_rob,s);

figure(4)
imagesc(Is_rob),colormap(map),title('detecteur de Roberts')
```

1.1.2. muscle21.m

```
function muscle21(s,sig)

%application image'muscle.bmp'

map=palette_nvg256;
I=imread('muscle.bmp');
I1=rgb2gray(I);

s=s;
sb=0;
```

```
sh=s;
sig=sig;
%detecteur de canny
detect_canny1(I1,sig,0,sh);
```

1.1.3. muscle22.m

```
function muscle22(s,alpha)
%application sur image'muscle.bmp'
%detecteurs optimaux: Deriche
map=palette_nvg256;
I=imread('muscle.bmp');
I1=rgb2gray(I);
s=s;
sb=0;
sh=s;
%detecteur de Deriche
l=s;
alpha=alpha;
[f]=detect_deriche(I1,l,alpha);
```

1.1.4. muscle3.m

```
function muscle3(t,sig)

%application image 'muscle.bmp
%filtrage avec seuil
map=palette_nvg256;
I=imread('muscle.bmp');
I1=rgb2gray(I);
I1=filtre_moy(I1,3);
I1=filtre_med1(I1,3);
t=t; %seuil
sig=sig;
%filtrage
[Is]=filtre_marrhildreth(I1,t,sig);
Is;
ind0=(Is>=0);
ind1=(Is<=-0);
Is(ind0)=0;
Is(ind1)=255;
map=palette_nvg256;
figure(1)
imagesc(Is),colormap(map),title('detecteur marr hildreth')
%
[Is1]=lap_de_gaussienne(I1,t,sig)
Is1
ind0=(Is1>=0);
ind1=(Is1<=-0);
Is1(ind0)=0;
Is1(ind1)=255;
map=palette_nvg256;
figure(2)
```

```
imagesc(Is1),colormap(map),title('détecteur log')
```

1.2. Image 'chromo.bmp'

chromo1_bin.m

```
function [J,Is_cont1]=chromo1_bin(s)
s=s;%seuil de binarisation [0 255]
I=imread('chromo.bmp');
I1=rgb2gray(I);%conversion en niveaux de gris
hist1(I1);
figure(1)
imshow(I1,256),title('image en niveaux de gris')
[J]=bin_im1(I1,s);%binarisation de l'image
figure(2)
imshow(J,256),title('image binarisée')
%conversion des elements de J ds [0 1] pour pouvoir utiliser la fonction 'cont_dil_ibin'
J=double(J(:));
J=J/max(J);
J=reshape(J,size(I1));
[Is_cont1,Is_cont2]=cont_dil_ibin(J);%contours image par dilatation
figure(3)
imshow(Is_cont1,2),title('image contours par dilatation 4-connexité')
```

1.3. Image 'os.bmp'

1.3.1. os1.m

```
function os1(s)

%application image'os.bmp'
%détecteurs approche dérivée première
map=palette_nvg256;
I=imread('os.bmp');
I1=rgb2gray(I);
hist1(I1);
figure(1)
imagesc(I1),colormap(map),title('image en niveaux de gris')
s=s;
%detecteur de Sobel
[N_sob]=detect_sob(I1);
[Is_sob]=seuil_simp(N_sob,s);
figure(2)
imagesc(Is_sob),colormap(map),title('detecteur de sobel')
%détecteur de prewitt
[N_prew]=detect_prew(I1);
[Is_prew]=seuil_simp(N_prew,s);
figure(3)
imagesc(Is_prew),colormap(map),title('detecteur de prewitt')
%détecteur de roberts
[N_rob]=detect_roberts(I1);
[Is_rob]=seuil_simp(N_rob,s);
```

```
figure(4)
imagesc(Is_rob),colormap(map),title('detecteur de Roberts')
```

1.3.2. os21.m

```
function os21(s,sig)

%application image'os.bmp'
%détecteurs approche dérivée première
map=palette_nvg256;
I=imread('os.bmp');
I1=rgb2gray(I);
hist1(I1);
figure(1)
imagesc(I1),colormap(map),title('image en niveaux de gris')
s=s;
s=s;
sb=0;
sh=s;
sig=sig;
%détecteur de canny
detect_canny1(I1,sig,0,sh);
```

1.4. Image 'route1.bmp'

1.4.1. route1.m

```
function route1(s)

s=s;
map=palette_nvg256;
I=imread('route1.bmp');
I1=rgb2gray(I);
hist1(I1);
%détecteur de Sobel
[N_sob]=detect_sob(I1);
[Is_sob]=seuil_simp(N_sob,s);
figure(2)
imagesc(Is_sob),colormap(map),title('detecteur de sobel')
%détecteur de prewitt
[N_prew]=detect_prew(I1);
[Is_prew]=seuil_simp(N_prew,s);
figure(3)
imagesc(Is_prew),colormap(map),title('detecteur de prewitt')
%détecteur de roberts
[N_rob]=detect_roberts(I1);
[Is_rob]=seuil_simp(N_rob,s);
figure(4)
```

```
imagesc(Is_rob),colormap(map),title('detecteur de Roberts')
```

1.4.2. route21.m

```
function route21(s,sig)

%application image'route1.bmp'
%detecteurs optimaux: Canny
map=palette_nvg256;
I=imread('route1.bmp');
I1=rgb2gray(I);
s=s;
sb=0;
sh=s;
sig=sig;
%detecteur de canny
detect_canny1(I1,sig,0,sh);
```

2. Image prétraitée

2.1. Image 'chromo.bmp'

chromo2_bin.m

```
function chromo2_bin(s)
s=s;%seuil de binarisation [0 255]
map=palette_nvg256;
I=imread('chromo.bmp');
I1=rgb2gray(I);%conversion en niveaux de gris
hist1(I1);
figure(1)
imshow(I1),title('image en niveaux de gris')
[J]=bin_im1(I1,s);%binarisation de l'image
figure(2)
imshow(J,256),title('image binarisée')
%conversion des elements de J ds [0 1] pour pouvoir utiliser la fonction 'cont_dil_ibin'
J=double(J(:));
J=J/max(J);
J=reshape(J,size(I1));
%1ere application
[Is_eros11,Is_eros12]=eros_ibin(J);
[Is_dil11,Is_dil12]=dilat_ibin(Is_eros11);
%2eme application
[Is_eros21,Is_eros22]=eros_ibin(Is_dil11);
[Is_dil21,Is_dil22]=dilat_ibin(Is_eros21);
figure(3)
imshow(Is_dil21,2),title('image binarisée erodée puis dilatée deux applications')
[Is_cont1,Is_cont2]=cont_dil_ibin(Is_dil21) %contours image erodée dilatée
figure(4)
imshow(Is_cont1,2),colormap(map),title('image contours')
```

2.2. Image 'os.bmp'

2.2.1. os_egal1.m

```
function os1_egal(s)

%application image'os.bmp'
%détecteurs approche dérivée première
map=palette_nvg256;
I=imread('os.bmp');
I1=rgb2gray(I);
[I1]=egal_his1(I1);
figure(1)
imagesc(I1),colormap(map),title('image en niveaux de gris')
s=s;
%détecteur de Sobel
[N_sob]=detect_sob(I1);
[Is_sob]=seuil_simp(N_sob,s);
figure(2)
imagesc(Is_sob),colormap(map),title('detecteur de sobel')
%détecteur de prewitt
[N_prew]=detect_sob(I1);
[Is_prew]=seuil_simp(N_prew,s);

figure(3)
imagesc(Is_prew),colormap(map),title('detecteur de prewitt')
%détecteur de roberts
[N_rob]=detect_roberts(I1);
[Is_rob]=seuil_simp(N_rob,s);

figure(4)
imagesc(Is_rob),colormap(map),title('detecteur de Roberts')
```

2.2.2. os21_egal.m

```
function os21_egal(s,sig)

%application image'os.bmp'
%détecteurs approche dérivée première
map=palette_nvg256;
I=imread('os.bmp');
I1=rgb2gray(I);
hist1(I1)
[I1]=egal_his1(I1)
figure(1)
imagesc(I1),colormap(map),title('image en niveaux de gris')
s=s;
s=s;
sb=0;
sh=s;
sig=sig;
%détecteur de canny
```