N° d'ordre : 01 /RC Année Universitaire 2002 – 2003

UNIVERSITE D'ANTANANARIVO

ECOLE SUPERIEURE POLYTECHNIQUE

DEPARTEMENT TELECOMMUNICATION

MEMOIRE DE FIN D'ETUDES

en vue de l'obtention

du DIPLOME d'INGENIEUR

Spécialité : Télécommunication
Option : Radiocommunication

par : **ANDRIAMBOLOLONA Soamampianina Zazah**

DETECTION DE CONTOURS EN TRAITEMENT D'IMAGE

Soutenu le Mardi 02 Mars 2004 devant la Commission d'Examen composée de :

Président : M. ANDRIAMIASY Zidora

Examinateurs :

- M. RAZAKARIVONY Jules
- M. RANDRIANTSIRESY Ernest
- M. RATSIHOARANA Constant

Directeur de Mémoire : M. RANDRIAMITANTSOA Paul Auguste

REMERCIEMENTS

Avant tout, loué soit l'Eternel tout puissant de m'avoir conseillée et indiquée le chemin à suivre dans la réalisation de cet ouvrage comme il m'a promise au Psaumes 32.8 :

« Je t'instruirai et te montrerai la voie que tu dois suivre ; Je te conseillerai, j'aurai le regard sur toi ».

Ensuite, il ne serait possible d'aboutir à cette fin sans le concours de plusieurs personnes à qui nous aimerons adresser notre profonde gratitude et vifs remerciements :

- Monsieur **RANDRIANOELINA Benjamin**, Professeur et Directeur de l' Ecole Supérieure Polytechnique d'Antananarivo ;

- Monsieur **ANDRIAMIASY Zidora**, Maître de conférences, pour l'honneur qu'il nous a fait en acceptant de présider le jury de ce mémoire.

- Monsieur **RANDRIAMITANTSOA Paul Auguste,** Professeur et Chef de département Télécommunication ,qui malgré ses multiples obligations, n'a ménagé ni temps ni efforts pour nous avoir guidé et soutenu tout au long de notre travail.

- Messiers :

RAZAKARIVONY Jules, Maître de conférences au sein du département Télécommunication à l'ESPA, Membre de Jury ;

RANDRIANTSIRESY Ernest, Assistant d'Enseignement Supérieur et de Recherche, Membre de Jury ;

RATSIHOARANA Constant, Assistant d'Enseignement Supérieur et de Recherche, Membre de Jury ;
qui ont accepté de sacrifier leur temps pour apprécier ce mémoire ;

- Tous les enseignants et tout le personnel de l' Ecole Supérieure Polytechnique d'Antananarivo, en particulier ceux du département Télécommunication ;

- A toute ma grande famille, mes amis et tous ceux qui de près ou de loin ont contribué à la réalisation de ce travail ;

A tous ! Merci et que Dieu vous bénisse.

TABLE DES MATIERES

NOMENCLATURE

A	:	matrice
A_{uv}	:	matrice de transformation par la transformée de Fourier
B	:	élément structurant
BW,BW1,BW2	:	images binaires
C	:	célérité de l'onde
CMY	:	Cyan Magenta Yellow
C_{mn}	:	matrice de transformation par la transformée en cosinus
d(x,y)	:	direction du gradient
DCT ou TCD	:	Transformée en Cosinus Discrète
D_C	:	qualité de détection de Canny (rapport signal sur bruit)
f(m,n)	:	image numérique
fmt	:	format d'image
G(x,y), Δ	:	opérateur gradient
G(x)	:	fonction gaussienne (filtre gaussien)
h, h_x, h_y	:	réponses impulsionnelles du filtre (filtres),
hgram	:	numérotation de niveaux de gris
H_n	:	matrice de Hadamard
H(u,v), Hd	:	réponses fréquentielles
I	:	image d'intensités
JPEG	:	Joint Pictures Experts Group
L_C	:	qualité de localisation de Canny
med	:	filtre médian
map	:	colormap
n	:	nombre d'opérations
N	:	nombre de niveaux de gris
N_q	:	nombre de niveaux de quantifications

noise	:	bruit
pixel	:	picture element
q	:	pas de quantification
RGB ou RVB	:	Red Green Blue ou Rouge Vert Bleu (image couleur)
$S_e(x,y)$	:	image échantillonnée
s(t)	:	signal temporel
S_d	:	fonction de Derich
s_b, s_h	:	seuil bas et seuil haut
T	:	période de l'onde
T(.)	:	opérateur de gradient
theta	:	angle de transformée de Radon
U_C	:	qualité d'unicité de Canny
X	:	image (image indexée sous Matlab)
S	:	système linéaire
SE	:	élément structurant
Y	:	image générée après morphologie mathématique
λ	:	longueur d'onde
win,win1,win2	:	dimensions du filtre
$\delta(t)$	:	fonction de Dirac
$\Delta(x)$	:	pas d'échantillonnage suivant x
$\Delta(y)$	:	pas d'échantillonnage suivant y
∇	:	opérateur gradient
∇^2	:	opérateur laplacien
σ	:	déviation standard du filtre gaussien

INTRODUCTION

Selon un adage populaire : « une image vaut mille mots ». Cela souligne l'importance des images dès lors qu'il s'agit de faire passer un message ou autre information.

Dès les années 1920, le traitement d'image a commencé à être étudié pour la transmission d'image par le câble sous-marin allant de New York à Londres. Harry G. Bartholomew et Maynard D. McFarland effectuèrent la première digitalisation d'image avec compression pour envoyer des fax de Londres à New York. Le temps de transfert passa ainsi de plus d'une semaine à moins trois heures. Par la suite, il n'y a pas vraiment eu d'évolution pour l'image jusqu'à la période d'après-guerre.

A la fin de la guerre, le traitement du signal prit son essor avec l'arrivé du radar. La prospection pétrolière participa également pour beaucoup au développement des techniques de traitement de signal.

L'essor du traitement d'image en particulier n'a lieu que dans les années 1960 quand les ordinateurs commencèrent à pouvoir travailler sur des images.

Le monde du traitement du signal en général a connu une révolution le jour où la transformation de Fourier rapide a été redécouverte. L'ensemble des manipulations fréquentielles devenait envisageable sur ordinateur.

L'essentiel des recherches portent sur l'amélioration des images et leurs compressions.
Au cours des années 1980, il y a eu un véritable engouement pour le traitement de l'image et surtout pour la compréhension de l'image par des systèmes experts. Les ambitions étaient beaucoup trop grandes. L'échec fut à hauteur.

Les années 1990 ont vu l'amélioration constante des opérateurs. La recherche médicale est devenue un très gros demandeur en traitement d'images pour améliorer les diagnostics faits à partir des nombreuses techniques d'imageries médicales. En même temps, le grand public se familiarise avec le traitement d'image grâce aux logiciels comme Photoshop.

Le traitement d'image constitue actuellement l'une des grandes orientations du traitement de l'information acquise à partir de capteur offrant des capacités de plus en plus importantes. L'essor des potentialités des calculateurs a permis d'envisager des procédures plus complexes pour mettre en évidence ce que l'œil peut voir. Le fossé entre la perception discrète du monde analogique, et la réalité de ce monde, est alors apparu à tous, et il a fallu développer des algorithmes pour appréhender cette perception numérique. L'expérience montre que la vision humaine a dicté plusieurs orientations, et ce, au détriment de la démarche plus méthodologique

mise en œuvre par des traiteurs de signaux. La vision est un processus de traitement de l'information. Elle utilise des stratégies bien définies afin d'atteindre ses buts. L'entrée d'un système de vision est constituée par une séquence d'image. Le système lui-même apporte un certain nombre de connaissances qui interviennent à tous les niveaux. Les connaissances mises en jeu peuvent être de trois types : physiques, géométrique et sémantique. Il est alors possible de mettre en correspondance la représentation extraite de l'image avec les descriptions des objets. Les attributs étudiés correspondent à des points d'intérêt ou à des zones caractéristiques de l'image comme les contours.

La détection de contours est un des problèmes les plus étudiés depuis l'origine des travaux sur les images numériques, elle représente même une des étapes préliminaires de nombreuses applications d'analyse d'image. Ceci est en grande partie due à la nature très intuitive du contour qui apparaît très naturellement comme l'indice visuel idéal dans la plus grande partie des situations.

Les informations que l'on souhaite extraire d'une image, les applications, les types de données, sont si variées qu'il est difficile, lorsque l'on doit traiter une nouvelle application, de savoir à quel modèle recourir et quelle méthodologie suivre.

On a alors recours à de nombreuses recherches afin de donner une bonne compréhension des principes de base du traitement d'images, associée à un certain esprit critique. A travers des traitements de nature différente. On va montrer quels sont les raisonnements et méthodes qui permettent d'obtenir le résultat désiré. nous avons décidé de réaliser ce travail et de lui donner comme titre : **« DETECTION DE CONTOURS EN TRAITEMENT D'IMAGE ».**

Ce mémoire a pour but de présenter les notions de base sur le traitement d'image en général et d'entamer une étude un peu détaillée sur la détection de contour afin de comprendre le principe des logiciels de traitement d'image.

Il est subdivisé en quatre grandes parties :

- la première partie parle de l'image en général ;
- la deuxième partie présente dans une vue assez générale, les notions de base sur le traitement d'images ;
- la troisième partie considère la détection de contours avec une étude qui porte sur les éléments de base, les techniques utilisées.
- enfin, la quatrième et dernière partie constitue une simulation sous MATLAB des notions abordées dans les parties théoriques.

CHAPITRE I:FORMATION, ACQUISITION ET MODELISATION DE L'IMAGE NUMERIQUE

I.1Généralités [1][2][3][4][5]

I.1.1Introduction

La description d'une image est une démarche faussement facile. S'il est vrai qu'une image consiste en un jeu de valeurs représentables, sur un écran par exemple, comprendre la nature d'une image s'avère délicat, d'autant que l'on confond souvent l'image, la visualisation et les traitements effectués sur l'image. Pour aborder la question de formation et de modélisation de l'image, nous distinguons les éléments suivants :

- la perception d'une image: elle s'articule autour des caractéristiques du système visuel humain. Ainsi, il apparaît que l'œil est sensible à certaines fréquences du spectre électromagnétique; ces fréquences représentent la lumière en général. Dans certains cas, le rayonnement électromagnétique à observer se situe en dehors de la plage des fréquences visibles.

- la représentation d'une image: il s'agit de représenter une entité physique sous une forme électrique ou une forme informatique. La représentation joue un rôle essentiel dans une chaîne de traitement car elle conditionne la capacité de stockage nécessaire ainsi que la mise en œuvre.

I.1.2Définitions

Voici quelques termes les plus fréquemment utilisés lorsque l'on parle des images numériques :

Définition I.1 :

Le bit est la plus petite unité d'information utilisée dans un ordinateur. Il peut prendre deux états, ce qui peut être présenté par les chiffres binaires 1/0 ou par *arrêt/marche*. Toutes les informations d'un ordinateur sont présentées au moyen de bits.

Définition I.2 :

L'image peut être définie comme étant la représentation d'un objet ou d'une scène, mais en parlant de numérisation, une image est définie comme étant une matrice M*N pixels.

Définition I.3 :

Le pixel (de l'Anglais :picture element) est un point élémentaire qui constitue une image. Le pixel n'est pas une unité de mesure arbitraire comme le centimètre, c'est le fondement même des images numériques dont c'est la plus petite unité logique. Chaque pixel peut revêtir un certain nombre de variétés de tons allant des différents niveaux de gris aux couleurs. Le nombre de bits

nécessaire est plus ou moins important en fonction du nombre de couleurs ou de niveaux de gris que l'on veut représenter par pixel.

Définition I.4 :

Le niveau de gris est un nombre équivalent compris entre 0 et 255 donnant l'information sur la luminance d'un point élémentaire (pixel).

Définition I.5 :

L'intensité ou luminance est le caractère qui indique l'intensité de lumière perçue indépendamment de la couleur. Elle s'étend du noir au blanc avec toutes les nuances de gris .Elle indique l'expression qualitative de la brillance énergétique.

I.1.3Domaines d'applications du traitement d'image

Il existe plusieurs grandes classes d'applications faisant appel au traitement d'images :

- L'imagerie concernant l'amélioration des images satellites, l'analyse des ressources terrestres, la cartographie automatique, les analyses météorologiques…

- Les technologies biomédicales concernant le scanner, l'échographie, la résonance magnétique nucléaire, la reconnaissance automatique des cellules et des chromosomes…

- La robotique concernant l'assemblage automatique des pièces et des composants, le contrôle de qualité,…

- Les applications multimédia comme la télévision ou l'Internet.

I.2Perceptions visuelles [3][4][5][6]

Avant d'entrer plus en détail dans les études sur l'image, commençons par quelques notions sur le Système Visuel Humain. Assez paradoxalement, il s'agit d'une des tâches les plus complexes car le fonctionnement du système visuel humain fait intervenir la subjectivité de l'observateur et car, en pratique, il apparaît difficile d'inclure la plupart des résultats des études psychovisuelles dans un traitement d'image courant.

I.2.1Le système visuel humain (SVH)

Grâce à la cornée (l'enveloppe translucide de l'œil) et l'iris (qui en se refermant permet de doser la quantité de la lumière), une image se forme sur la rétine. Celle-ci est sensible aux rayonnements électromagnétiques de longueur d'onde comprise entre 380 et 700 nm. Elle est composée de petits bâtonnets et de trois cônes : les bâtonnets permettent de percevoir la luminosité et le mouvement, tandis que les cônes permettent de différencier les couleurs.

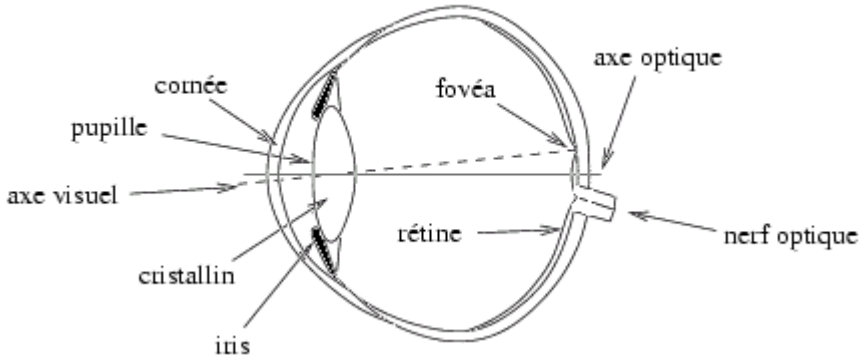


Figure 1.1: Coupe latérale simplifiée de l'œil.

I.2.2Perception des couleurs

C'est par la lumière que la couleur existe. Elle ne réside pas dans l'objet mais dans la lumière qui les éclaire et dans leur propriété à absorber certaines radiations tout en réfléchissant d'autres. La couleur n'est donc qu'une impression, un effet physiologique produit par notre cerveau et dont les causes sont captées par nos sens.

I.2.2.1La lumière

La lumière couvre une partie du spectre d'énergie électromagnétique. Un rayonnement électromagnétique est en général constitué d'un certain nombre de longueurs d'onde (ou fréquences) que les dispositifs dispersifs permettent de séparer en un spectre. La longueur d'onde du spectre visible s'étend approximativement de 380 à 720 nm. En un mot, la lumière est une distribution d'énergie émise à certaines fréquences ayant une certaine intensité.

I.2.2.2Notion de couleurs

La couleur de la lumière est caractérisée par sa fréquence, elle-même conditionnée par la longueur d'onde et la célérité de l'onde. La longueur d'onde d'un phénomène oscillatoire est exprimée par la relation (1.1).

$$\lambda = CT \qquad (1.1)$$

- λ désigne la longueur d'onde
- C désigne la célérité de l'onde (pour la lumière $3x10^8$ m. s^{-1})
- T désigne la période de l'onde

L'œil humain est capable de voir des lumières dont la longueur d'onde est comprise entre 380 et 780 nm.

Les études des trois espèces de cônes et les phénomènes complexes qui permettent de percevoir les sensations colorées aboutissent à dire que l'œil n'est sensible qu'à trois plages de radiation ayant pour maximum :

- 450 nm pour le bleu
- 525 nm pour le vert
- 625 nm pour le rouge

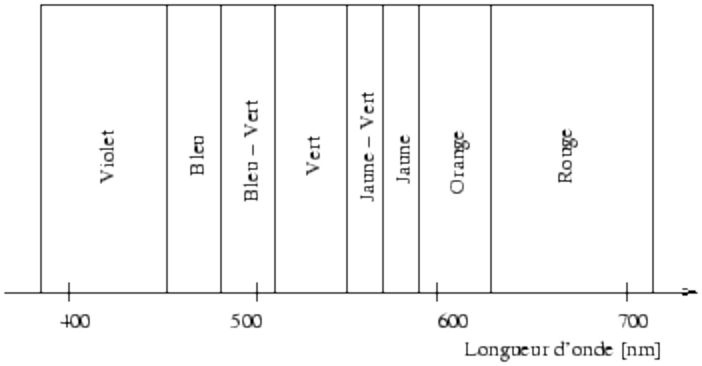


*Figure 1.2***:** Les longueurs d'onde associées aux couleurs.

La nature trichrome de l'image permet de recréer n'importe quelles couleurs avec le mélange du rouge, vert et bleu. Le système RGB (de l'Anglais Red, Green et Blue) ou RVB (Rouge, Vert et Bleu en Français) est alors le plus utilisé comme système de base sur l'écran informatique et dans le traitement d'image. Ces trois couleurs sont alors dénommées « couleurs primaires ».

I.2.2.3Synthèse de couleurs

Dans la chaîne de création d'image, deux méthodes sont utilisés pour couvrir la quasi totalité du spectre visible : la synthèse additive et la synthèse soustractive.

I.2.2.3.1Synthèse additive

La synthèse additive est le fruit de l'ajout de composantes de la lumière et prend le RGB comme système de référence. La combinaison des trois composantes Rouge, Vert, Bleu donne du blanc. L'absence de composante donne du noir. Les couleurs secondaires sont le cyan, le magenta et le jaune car :

- le vert combiné au bleu donne du cyan
- le bleu combiné au rouge donne du magenta
- le vert combiné au rouge donne du jaune

I.2.2.3.2Synthèse soustractive

La synthèse soustractive permet de restituer une couleur par soustraction, à partir d'une source de lumière blanche, avec des filtres correspondant aux couleurs complémentaires : jaune, magenta et cyan. L'ajout de ces trois couleurs donne du noir et leur absence produit du blanc. Les composantes de la lumière sont ajoutées après réflexion sur un objet, ou plus exactement sont absorbées par la matière. Bleu, Rouge, Vert sont devenus les couleurs secondaires car :

- le magenta combiné avec le cyan donne du bleu
- le magenta combiné avec le jaune du rouge
- le cyan combiné avec le jaune du vert

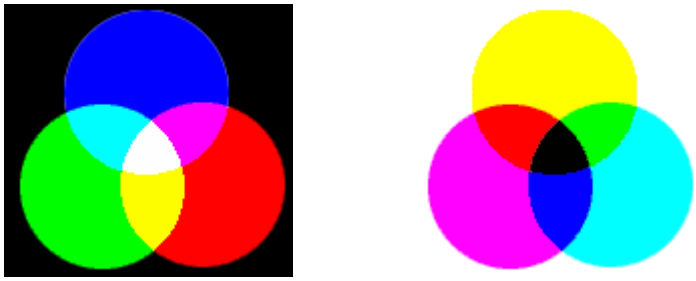


Figure 1.3 : synthèses additive et soustractive

Remarque : La synthèse soustractive est utilisée pour l'impression sur feuille blanche car la détermination des composantes RGB d'une onde s'opère par addition sur fond noir. L'arrière plan est donc supposé absorbant pour toutes les couleurs. Un tel système n'est pas adéquat pour traiter l'impression sur feuille blanche car cette dernière réfléchit l'ensemble des couleurs. Le système utilisé est alors le CMY (Cyan Magenta Yellow).

I.3Acquisition et numérisation de l'image [1][4][5][7]

I.3.1Principe

Avec un seul œil, capteur à deux dimensions, l'homme peut interpréter les trois dimensions dans lesquelles il évolue. Un capteur d'images est donc un appareil sensible aux mêmes types d'informations que celle perçues par l'œil. En fait, l'œil n'est sensible qu'aux variations :

- temporelles : l'apparition subite d'une forme dans nôtre champ de vision attire instantanément nôtre attention.
- spatiales : l'œil est instinctivement attiré par les transitions brutales de niveau de luminance. Une transition entre deux zones homogènes sera naturellement accentuée par l'œil.

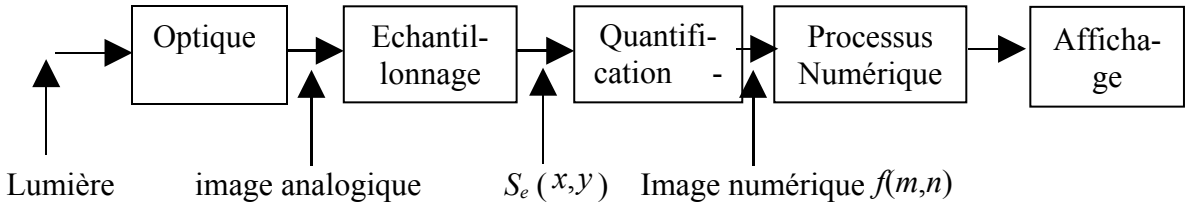


Figure 1.4: Acquisition de l'image

I.3.2 Numérisation de l'image

Une image « réelle » va être transformée en une image numérique par différents outils de transformation (caméra ou appareil photo numérique, scanner, satellite,….). Tous ces systèmes peuvent être comparés à des capteurs. L'image numérique, est le terme utilisé pour désigner une photo, une image existant sous forme de fichier informatique, c'est-à-dire que l'on peut visionner sur un ordinateur.

L'image reçue est fonction des caractéristiques du capteur. Le passage de l'information continue à une information discrète doit s'effectuer avec un minimum de dégradation de l'information. Après, l'information est représentée sous forme de tableau de mots binaires de longueur finie. Autrement dit, numériser une image c'est lui donner une représentation électronique à partir de l'objet réel.

I.3.2.1Les étapes de numérisation de l'image

I.3.2.1.1Echantillonnage

L'échantillonnage des images est la première étape de la numérisation des images. Elle est la restriction d'une fonction d'un espace sur un espace plus petit. Une image échantillonnée est habituellement représentée sous la forme de « pixels », des carrés où l'intensité de l'image est constante. L'échantillonnage détermine la taille de chaque pixel. Cette taille est fonction de la résolution du capteur. Les contraintes physiques imposent ensuite les dimensions maximales de l'image (nombre de pixels en ligne et en colonne).

L'échantillonnage consiste à prélever les valeurs instantanées d'un signal $s(t)$ à des intervalles réguliers séparés par une période d'échantillonnage T_e. Mathématiquement, le signal échantillonné est le résultat du produit du signal avec la « peigne de Dirac ». Il s'écrit :

$$s_e(t) = s(t)\sum_{-\infty}^{\infty} \delta(t-nT_e) \qquad (1.2)$$

Où $\delta(t)$ désigne la fonction de Dirac

D'une manière générale, la fonction peigne à deux dimensions s'écrit :

$$peigne(x,y,\Delta x,\Delta y,) = \sum_{-\infty}^{+\infty}\sum_{-\infty}^{+\infty} \delta(x - n\Delta x, y - n\Delta y) \qquad (1.3)$$

La fonction représentant l'image échantillonnée :

$$S_e(x,y) = f(x,y)\, peigne(x,y,\Delta x,\Delta y)$$

$$= \sum_{n=-\infty}^{+\infty}\sum_{n=-\infty}^{+\infty} f(m\Delta x, n\Delta y)\,\delta(x - m\Delta x, y - n\Delta y) \qquad (1.4)$$

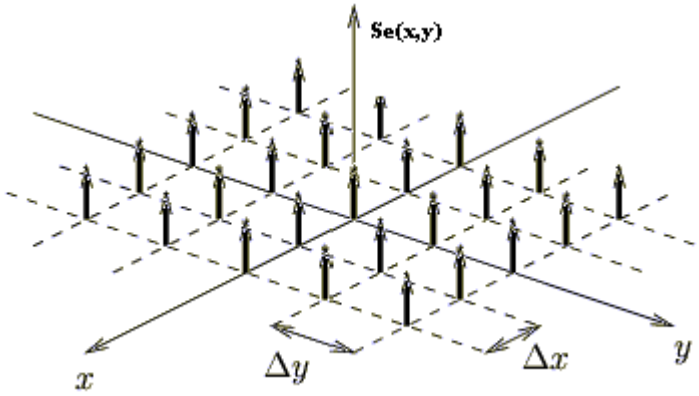


Figure 1.5: image échantillonnée

Illustration :

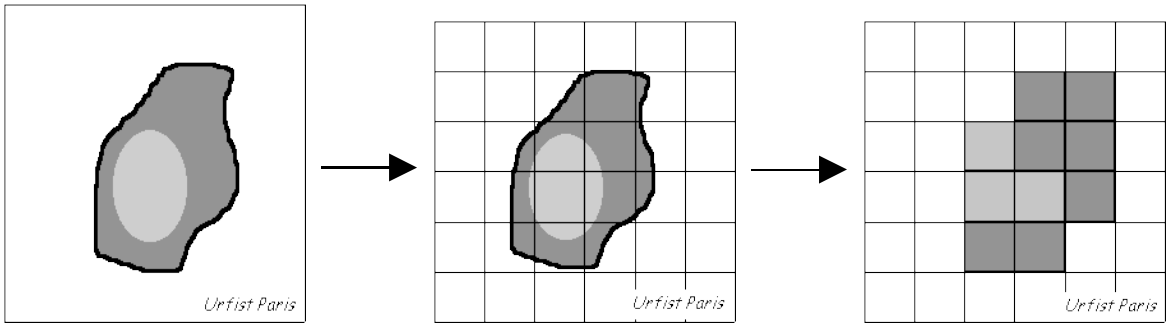


Figure 1.6 : Echantillonnage d'une image

Remarque : Pour que le signal puisse être reconstitué, la fréquence d'échantillonnage doit vérifier le théorème de Shannon, si F_{max} désigne la fréquence maximale du signal, il faut avoir :

$$F_e > 2.F_{max} \qquad (1.5)$$

Avec $F_e = \frac{1}{T_e}$.

I.3.2.1.2Quantification

La phase finale de la discrétisation de l'image est la quantification qui n'est autre que le résultat de la conversion de la mesure en une valeur discrète (entière). Elle fixe le nombre de niveaux possibles pour chaque pixel de l'image, souvent interprétés comme des niveaux de gris. Pour une conversion analogique/numérique, la quantification consiste à représenter toutes les valeurs contenues dans un même intervalle par une même valeur. Si α_{max} désigne la valeur maximale du signal, le pas de quantification q s'écrit alors sous la forme :

$$q = \frac{2.\alpha_{max}}{N} \qquad (1.6)$$

où N est le nombre de niveaux. L'intervalle de quantification est fixé pour une largeur notée q_i et pour une valeur centrale x_i . L'échantillon $s_e(t)$ obtenu vérifie ainsi :

$$x_i - \frac{q_i}{2} < s_e(t) < x_i + \frac{q_i}{2} \qquad (1.7)$$

I.3.2.1.3Codage

La valeur obtenue à l'issue de la quantification peut alors être codée, sur n bits tel que : $N = 2^n$. L'objectif étant de transformer l'image en une suite de mots binaires destinés à faciliter son stockage, son traitement et sa transmission.

D'une manière simple, le mot *codage s*ignifie : « transformation d'un message exprimé en langage clair, suivant des équivalences convenues dans un code ». Appliquée aux images, cette définition conduit aux associations suivantes :

- le « langage clair » est l'image elle-même ;
- le « message » est le contenu de l'image ou encore ce qu'elle nous apprend ;
- les « équivalences » sont les mots binaires dans le cas d'un code numérique.

Pour le codage informatique, il est commode d'exprimer la quantification en octets.

I.3.2.2Caractéristiques des images numériques

Une image numérisée est caractérisée en général par sa résolution et par sa dynamique :

I.3.2.2.1La résolution de l'image

Définition I.6 :

La résolution d'une image est définie par un nombre de pixels par unité de longueur de l'objet physique à numériser, exprimée en *dpi (dots per inches)* ou *ppp (points par pouce)*. Ce paramètre est défini lors de la numérisation et dépend principalement des caractéristiques du

matériel utilisé. Plus le nombre de pixels est élevé par unité de longueur de la structure à numériser, plus la quantité d'information qui décrit cette structure est importante et plus la résolution est élevée. D'une autre manière, la résolution d'une image numérique définit le degré de détail qui va être représenté sur cette image.

Les phénomènes de numérisation dépendent des deux équations suivantes :

$$\begin{aligned} (X \times \textit{résolution}) &= x \textit{ pixels} \\ (Y \times \textit{résolution}) &= y \textit{ pixels} \end{aligned} \qquad (1.8)$$

Où :

X et *Y* représentent la taille (en pouces ou en mètres) de l'objet à numériser, *Résolution* représente la résolution de la numérisation et enfin *x* et *y* représentent la taille (en pixels) de l'image.

Exemple :

Une image de 1*1 pouce scannée à 100 dpi aura une taille x,y de 100 pixels :

(1*100)*(1*100)=100 pixels sur 100 pixels (notons que 1 pouce=2,54 centimètres).

I.3.2.2.2La dynamique de l'image

La dynamique de l'image correspond à l'étendu de la gamme de couleurs ou de niveaux de gris que peuvent prendre les pixels, cette notion est liée au nombre d'octets utilisé pour stocker l'information sur les teintes de gris ou les couleurs.

En informatique, l'image peut être représentée de différentes manières, nous pouvons citer les images binaires, les images d'intensité, les images couleurs RGB et les images couleur indexées.

- *Image binaire :*

C'est une matrice rectangulaire dont les éléments prennent une valeur égale à 1 ou 0. Lors de la visualisation, les 0 sont représentés par du noir et les 1 par du blanc. La 'couleur' est donc représentée par un seul bit. Ce type d'image est par exemple utilisé pour scanner un texte quand celui-ci est composé d'une seule couleur. La figure suivante nous montre un exemple de texte scanné sous forme d'image binaire :

Figure 1.7 : Image binaire

- *Image d'intensité*

C'est une matrice dont chaque élément est un réel compris entre 0 (noir) et 1 (blanc). La dénomination « images en niveaux de gris » est également utilisée car ces valeurs comprises entre 0 et 1 représentent les différents niveaux de gris. Par exemple, 0.2 représente un gris foncé et 0.8 un gris clair. En général, les images en niveaux de gris renferment 256 teintes de gris, elles font aussi partie des images à 256 couleurs, simplement chacune de ces 256 couleurs est définie dans la gamme des gris. Par convention, la valeur zéro représente le noir (intensité lumineuse nulle) et la valeur 255 le blanc (intensité lumineuse maximale).

- *Image couleur*

S'il existe plusieurs modes de représentation de la couleur, le plus utilisé pour le maniement de la couleur numérique est l'espace couleur RGB. Il existe différents types d'images couleurs en fonction du nombre de bits utilisés pour le stockage de l'information couleur. Prenons l'exemple des images en « vraies couleurs » (ou 24 bits).

Il s'agit en fait d'une appellation « trompeuse » car évoluant dans un monde numérique (discret, fini) qui ne peut pas rendre compte de la réalité (infini). Chaque composante est représentée par un octet dans l'espace de représentation des couleurs. Chaque pixel peut prendre une valeur dans le RGB comprise entre 0 et 255 (soit $2^8 \times 2^8 \times 2^8 = 256 \times 256 \times 256 = 16,7$ millions de possibilités environ). L'information couleur de chaque pixel est donc codée par 3 octets, ce qui fait des images en vraies couleurs des images très "lourdes".

-*Image couleur indexée*

L'utilisation de trois matrices pour représenter une image couleur conduit pour les grandes images à l'occupation d'un espace mémoire important. Une manière plus économique de représentation est la représentation indexée qui en contrepartie ne permet de représenter qu'un nombre limité de couleurs. Ces couleurs sont mémorisées dans une table de couleur (appelée colormap) qui est une matrice $n \times 3$ (où n est le nombre des couleurs). L'image est alors une matrice contenant des nombres entiers compris entre 1 et n, chaque entier jouant le rôle d'index relatif à la table de couleur. D'habitude, l'information couleur est codée sur 1 octet, c'est-à-dire, une couleur est représentée sur un octet (8 bits), il y a alors $2^8 = 256$ couleurs possibles. Ce type d'image est également dénommée image "fausses couleurs" ou "à palette ''.

Figure 1.8 : Image d'intensité, Image couleur, Image couleur indexée

I.4Représentations mathématiques [1][4][8]

Il s'agit de représenter l'image sur un autre espace de représentation. L'image transformée est alors de la forme :

$$F = A.I.A^T \tag{1.9}$$

Où I est l'image originale, A la matrice de transformation et A^T sa transposée.

I.4.1Représentation par la transformée de Fourier

Soit *f(x)* un signal analogique converti en N échantillons séparés par Δx, en traitement d'image, on utilise souvent la Transformée de Fourier Discret Unitaire (TFDU) :

$$F(u) = \frac{1}{\sqrt{N}} \sum_{n}^{N-1} f(x) \exp\left(\frac{-2\pi\ jux}{N} \right) \tag{1.10}$$

Et la matrice $N \times N$ correspondante est :

$$F = \left\{ \frac{1}{\sqrt{N}} \exp\left(\frac{-2\pi\ ux}{N} \right) \right\} \quad \text{avec} \quad \begin{cases} 0 \le x \le N-1 \\ 0 \le u \le N-1 \end{cases} \tag{1.11}$$

En deux dimensions :

$$F(u,v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \exp\left(\frac{-2\pi\ jux}{N} \right) \exp\left(\frac{-2\pi\ jvy}{N} \right) \tag{1.12}$$

En notation matricielle :

$$F = \sum A_{uv} g(u,v) \tag{1.13}$$

tel que A_{uv} matrice carrée dont (u,v)^{ème} élément vaut :

$$A(x,u)A(y,v) = \exp\left[\frac{-2\pi\ j}{N} (ux + vy) \right]$$

$$A_{u,v} = \left\{ \frac{1}{N} \bullet \frac{2\pi\ j}{N} (ux + vy) \right\} \quad \text{avec} \quad \begin{cases} 0 \le x, y \le N-1 \\ 0 \le u, v \le N-1 \end{cases} \tag{1.14}$$

posons $W_n = z = \exp\left(\frac{-2\pi j}{N}\right)$, la matrice A est :

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & z & z^2 & z^3 & \dots & z^{(N-1)} \\ 1 & z^2 & z^{2(2)} & z^{2(3)} & \dots & z^{2(N-1)} \\ 1 & z^3 & z^{3(2)} & z^{3(3)} & \dots & z^{3(N-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & z^{N-1} & z^{(N-1)(2)} & z^{(N-1)(3)} & \dots & z^{(N-1)(N-1)} \end{bmatrix} \quad (1.15)$$

I.4.2Représentation par transformée en cosinus

Cette transformée est en fait une variante de la transformée de Fourier discrète qui transforme un signal réel en un signal réel (la notion de phase n'y apparaît pas). Il est possible de coder correctement une image en effectuant la transformation en cosinus de cette image puis en codant judicieusement les hautes fréquences. La reconstitution de l'image se faisant par transformée de Fourier inverse.

Nous faisons l'hypothèse que l'image est carrée de dimension N×N. Soit la matrice de transformation $\{C(m,n)\}$ définie par la relation (1.16):

$$C(m,n) = \begin{cases} \frac{1}{\sqrt{N}} & ;\ m=0\ ;\quad 0 \le n \le N-1 \\ \sqrt{\frac{2}{N}} \cos\left[\frac{(2x+1)m}{N}\right] \cos\left[\frac{(2y+1)n}{N}\right] & ;1 \le m \le N-1; 0 \le n \le N-1 \end{cases} \quad (1.17)$$

Une autre forme pour la transformée en cosinus discrète est donnée par :

$$F(u,v) = \frac{2c(u)c(v)}{N} \sum_{m=0}^{N-1} \sum_{m=0}^{N} f(m,n) \cos\left(\frac{2m+1}{2N} u\pi\right) \cos\left(\frac{2n+1}{2N} v\pi\right) \quad (1.18)$$

Où $u = 0,1, ...,N–1$ et $v = 0,1, ...,N-1$

et la transformée inverse est donnée par :

$$f(m,n) = \frac{2}{N} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} c(u)c(v)F(u,v) \cos\left[\frac{(2m+1)}{2N} u\pi\right] \cos\left[\frac{(2n+1)}{2N} v\pi\right] \quad (1.19)$$

Où $m = 0,1, ...,N-1$ et $n = 0,1, ...,N-1$

$$\text{Avec } c(k) = \begin{cases} \frac{1}{\sqrt{2}} & ;\ k = 0 \\ 1 & \text{sinon} \end{cases} \quad (1.20)$$

I.4.3Représentation de Hadamard

Les matrices de Hadamard ne prennent que les valeurs 1 et -1, d'où son intérêt en traitement d'image.

En général, les matrices de Hadamard sont des matrices carrées (*N*N)* avec $N = 2^n$ donc *N* pair.

$$H_n = \{h_n(i,j)\} \text{ et } h_n(i,j) = \frac{1}{\sqrt{N}}(-1)^{k(i,j)} \text{ où } k(i,j) = k(i,j) = \sum_{i_k, j_k}^{n-1} i_k . j_k \qquad (1.21)$$

avec i_k et j_k sont les représantant binaires de i, j tel que :

$$i = \sum_{k=0}^{n-1} i_k . 2^k \quad \text{et} \quad j = \sum_{k=0}^{n-1} j_k . 2^k \qquad (1.22)$$

Exemple :

Pour *k=3, N=* $2^3 = 8$ *et* $\sqrt{8} = 2\sqrt{2}$

$$H_3 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \qquad (1.23)$$

I.5Formats des images [6]

Définition I.7 :

Le format d'une image est défini par le nombre et l'ordre dans lequel sont rangées les données utiles (concernant le système d'exploitation informatique et les caractéristiques de l'image elle-même), l'algorithme de compression utilisé pour les données d'image et la façon dont sont rangées les données d'images comprimées ou pas.

D'une manière générale, deux types de formats d'images se distinguent:

- les formats matriciels : on parle aussi de bitmap ou de formats adressables dans lesquels sont stockés les informations sur chaque pixel de l'image. L'image est considérée comme une matrice (un tableau) de points (pixels) ayant chacun une couleur. Ils sont utilisés pour stocker des images simples.

- les formats vectoriels : dont le principe est de représenter, autant que possible les données de l'image par des formes géométriques qui vont pouvoir être décrites d'un point de vue mathématique. En d'autres termes, ces formats permettent d'enregistrer différents types d'information permettant de reconstruire l'image (information sur les courbes et les lignes qui la composent), ils permettent de stocker des images complexes et enfin ils permettent un changement d'échelle immédiat ainsi qu'une meilleure qualité d'impression.

Ces notions étant maintenant définies, qu'en est-il du traitement appliqué aux images numériques ? Telle est la question à laquelle nous nous efforcerons d'apporter une réponse dans le chapitre suivant.

CHAPITRE II:TRAITEMENT D'IMAGES NUMERIQUES

II.1Nécessité de traiter les images numériques [4][5][6]

L'acquisition d'images numériques mosaïques (sous forme de matrices de nombres) transforme un signal continu analogique (une radiation lumineuse) en échantillonnage numérique discontinu. Il y a donc des pertes d'information dues aux méthodes d'échantillonnage et d'autres dues aux erreurs d'acquisition (capteur défectueux, couverture nuageuse, défauts d'optique, de parallaxe, etc.). Le premier problème est pris en compte en respectant le principe d'échantillonnage énoncé auparavant. Le deuxième problème est inhérent à la technologie des capteurs et à des facteurs qui échappent en partie au contrôle de l'utilisateur. Ce n'est que par une manipulation des données enregistrées que l'on peut y remédier. D'où les traitements d'images et les filtrages. Dans les paragraphes qui suivent, nous aborderons quelques opérations de base de traitement d'images.

II.2Opérateurs d'images [1][2][3][9]

II.2.1 Introduction

La description d'une forme nécessite un certain nombre d'opérations qui transforment l'image originale en représentation compatible aux traitements ultérieurs. Cette représentation donne les caractéristiques d'une image. Nous allons considérer deux types d'opérateurs d'images : un opérateur morphologique et un opérateur statique.

II.2.2Opérateur morphologique : la morphologie mathématique

II.2.2.1Généralités

Les algorithmes de morphologie mathématique permettent de travailler sur les images 1 bit (noir et blanc) pour mettre en évidence leurs propriétés particulières. Ces algorithmes sont basés sur l'utilisation des "éléments structurants" (notés B) que l'on déplace de façon à ce que son origine passe par toutes les positions de l'image, pour mettre en évidence certaines caractéristiques de l'image, c'est-à-dire des configurations élémentaires de pixels à rechercher dans l'image. Il existe différentes opérations concernant la morphologie mathématique mais les opérations de base sont la dilatation et l'érosion.

Définition II.1 :

Un élément structurant est une forme qui définit une opération morphologique et que l'on peut voir comme un masque.

Pour une meilleure compréhension, illustrons les opérations de base susmentionnées par un exemple.

Soit un exemple d ‘élément structurant :

Figure 2.1 : Un élément structurant

● : Pixel devant appartenir à l’image initiale

○ : Pixel ne devant pas appartenir à l’image initiale

▼ : Origine

· : Pixel indifférent

II.2.2.2Dilatation et Erosion :

II.2.2.2.1.Dilatation

Pour chaque position de B sur l’image X, si un au moins, des pixels de B fait partie de X, alors l’origine de B appartient à l’image Y générée.

Notation : $Y = X \oplus B$.

II.2.2.2.2.Erosion :

Pour chaque position de B sur l’image, si tous les pixels de B font partie de X, alors l’origine de B appartient à l’image Y générée

Notation : $Y = X \ominus B$.

II.2.2.3Ouverture et fermeture

L’ouverture morphologique et la fermeture morphologique résultent de la mise en cascade de l’érosion et de la dilatation.

Soit $\tilde{B}$ la transposée de B, c’est à dire sa symétrie par rapport à l’origine.

Reprenons l’élément structurant précédent.

II.2.2.3.1.Ouverture

But : isoler les surfaces présentes dans l’image, lisser les contours

Notation : $Y = X_B = (X \ominus B) \oplus \tilde{B}$.

On réalise une érosion par B puis une dilatation par la transposée de B.

II.2.2.3.2.Fermeture :

But : recoller des morceaux de surfaces proches de manière à fermer des contours disjoints, lisser les contours. On réalise alors une dilatation par B puis une érosion par la transposée de B.

Notation : $Y = X^B = (X \oplus B)\theta \widetilde{B}$.

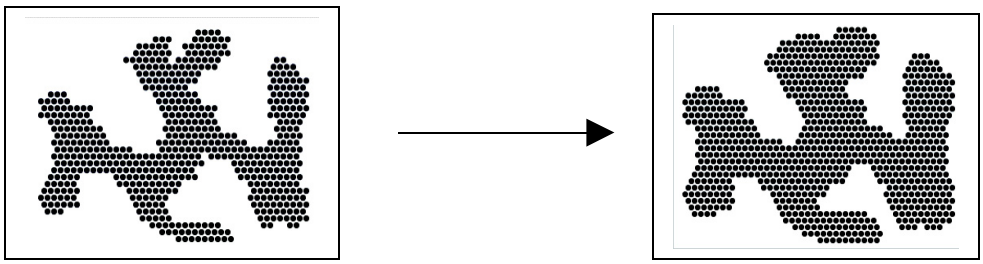


Figure 2.2 : Dilatation

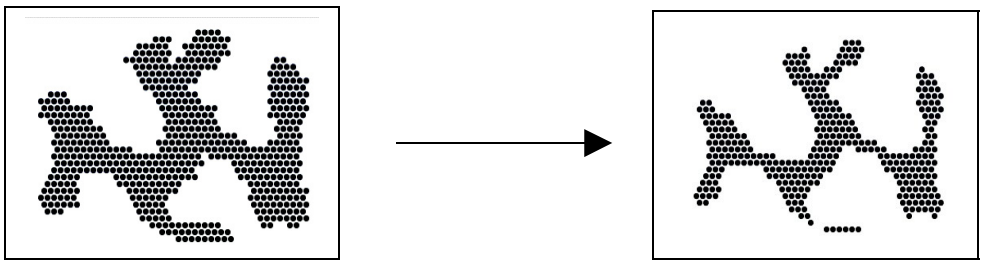


Figure 2.3 : Erosion

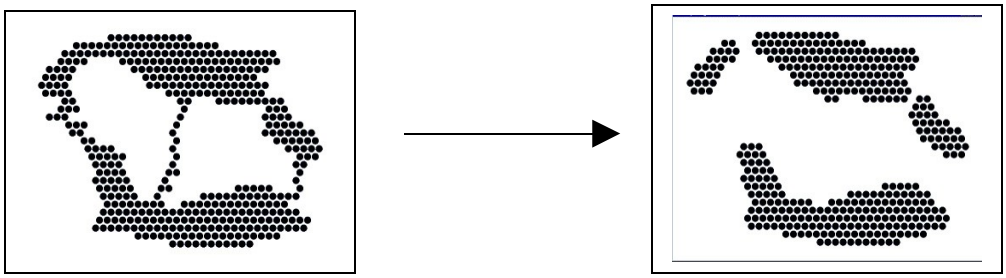


Figure 2.4 : Ouverture

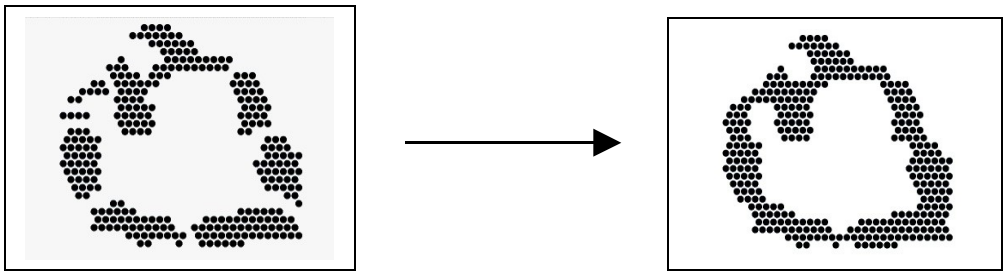


Figure 2.5 : Fermeture

Remarque : La morphologie mathématique est un ensemble de méthodes d'analyse d'images. Elle offre un grand nombre d'outils très puissants de traitement et d'analyse d'images que l'on retrouve sous différentes rubriques dans certains logiciels d'analyse d'images et même de retouche d'images. Les outils proposés ont été développés au départ pour traiter des images binaires : on fait alors de la morphologie mathématique ensembliste. Leur utilisation a été ensuite étendue aux images en niveaux de gris : on parle à ce moment de morphologie mathématique fonctionnelle qui adopte le même principe qu'en morphologie ensembliste mais vu que le domaine considéré est plus vaste, elle joue sur les fonctions. A noter également que certains opérateurs fonctionnent désormais sur des images couleurs.

II.2.3. Opérateurs statiques

II.2.3.1. Histogramme de valeurs

Un modèle statique peut être utilisé pour représenter l'image : l'histogramme qui est une fonction donnant les fréquences d'apparition des niveaux de gris et qui est à la base de nombreux traitements images. *L'histogramme* d'une image est la fonction en escalier qui comptabilise pour chaque niveau de gris le nombre de pixels ayant cette intensité. En d'autres termes, il peut être vu comme une densité de probabilité des niveaux de gris de l'image. En pratique, pour le calculer, on donne un nombre de niveaux de quantification et on compte le nombre de pixels de l'image correspondant à chaque niveau. Ceci peut être illustré dans l'exemple de la figureII.6 où nous avons des images avec les histogrammes correspondants. Sur ceux-ci, les abscisses représentent les niveaux de gris et les ordonnées le nombre de pixels correspondant à chacune des intensités.

II.2.3.2. Egalisation d'histogramme

L'égalisation de l'histogramme consiste à répartir les fréquences d'apparition des pixels sur la largeur de l'histogramme. L'idée est de modifier la répartition des niveaux pour obtenir un histogramme plat étendu à l'ensemble des valeurs possibles. Dans cette opération la dynamique originale [min, max] est étalée à [0,255]. On cherche à affecter le même nombre de pixels à chaque niveau de gris, c'est pourquoi cette opération est appelée équipopulation. L'effet obtenu permet de mieux séparer les valeurs les plus représentées dans l'image et de rapprocher les valeurs marginales.

C'est un outil qui est parfois utile pour améliorer certaines images de mauvaise qualité (mauvais contraste, images trop sombres ou trop claires, mauvaise répartition des niveaux d'intensité,…), pour accentuer les contours, pour renforcer la dynamique de l 'image ou pour réduire le nombre de niveaux pour la visualisation.

Il s'agit de déterminer une transformation *f* des niveaux d'intensité qui rend l'histogramme aussi plat que possible. Si un pixel a l'intensité *i* dans l'image originale, son intensité dans l'image égalisée est *f(i)*.

En général, on choisit pour *f* une fonction en escalier, et on détermine la largeur et la hauteur des différentes marches de manières à aplatir l'histogramme de l'image égalisée.

Dans l'exemple qui suit, l'égalisation de l'histogramme a comme effet d'élever le contraste et de faire apparaître sur la deuxième image des détails qui ne sont pas visibles sur la première. Il n'y a pas création d'information mais mise en évidence de l'information déjà présente.

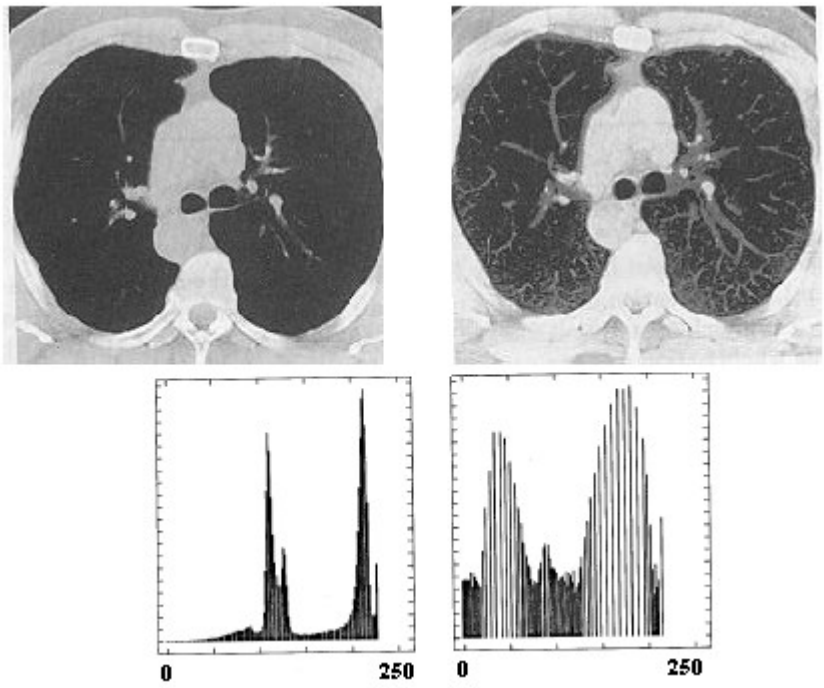


Figure 2.6 : Egalisation d'histogramme

II.3Filtrage [2][3][5][9]

II.3.1Introduction

L'amélioration de l'image est essentiellement obtenue par ce que l'on appelle une opération de filtrage. Le filtrage est utilisé pour modifier ou pour mettre en valeur une image, par exemple, on pourra filtrer une image pour accentuer certains attributs ou pour en supprimer d'autres. L'objectif avoué du filtrage est d'éliminer les perturbations induites par les procédés d'acquisition d'images et aux problèmes de transmission, ainsi de réduire les variations d'intensité au sein de chaque région de l'image tout en respectant l'intégrité de la scène originale comme les

transitions entre régions homogènes, les éléments significatifs de l'image doivent être préservés au mieux.

Différentes méthodes de filtrage ont été développés suivant le type et l'intensité du bruit ou les applications auxquelles on destine l'image (par exemple : détection de contour, segmentation). Les premières et les plus simples sont basées sur le filtrage linéaire, mais les limites atteintes par ces techniques ont conduit au développement des filtres non linéaires. Par la suite, nous ne prendrons qu'un seul exemple pour les filtres non linéaires, ceux-ci n'ayant pas de propriétés communes. Avant d'entamer ces deux classes de filtres, abordons d'abord la notion de filtre idéal.

II.3.2 Notion de filtre idéal

On dira d'un filtre qu'il est idéal s'il multiplie tous les coefficients transformés par 0 ou par 1. Autrement dit, un filtre est idéal si sa réponse est telle que :

$$\forall\ u,v\ ,\mathrm{H}\ (u,v) = 0\ ou\ 1 \qquad (2.1)$$

Un signal préalablement filtré par un filtre idéal n'est pas modifié par application récursive de ce filtre. Cependant, la notion de filtre idéal est purement formelle. En effet, en pratique, un filtre idéal n'a rien de réel, il est irréalisable et ses propriétés le rendent inutilisable.

II.3.3Filtrage linéaire

Le filtrage linéaire d'une image peut donc s'envisager de deux manières. Le filtrage peut tout d'abord se réaliser dans le domaine spatial en effectuant un produit de convolution. Ou alors, il est réalisé dans le domaine fréquentiel en multipliant la transformée de FOURIER de l'image par la fonction de transfert du filtre. L'image filtrée est alors obtenue par transformée de FOURIER inverse. Enfin, un système linéaire invariant par translation peut également être caractérisé par une équation à coefficients. C'est parfois la voie la plus facile pour réaliser le filtrage.

II.3.3.1Filtrage par produit de convolution

- Passage d'une image dans un système linéaire

Comme dans le cas d'un système unidimensionnel, on peut interpréter l'action d'un opérateur bidimensionnel, linéaire et invariant en translation, comme un filtrage. Le problème général du filtrage bidimensionnel consiste à élaborer un filtre qui possède la réponse impulsionnelle voulue et qui soit réalisable efficacement.

Propriétés II.1

Soient : $g_1(x,y)$ et $g_2(x,y)$ les images de sortie du système relatives aux images d'entrée $f_1(x,y)$ et $f_2(x,y)$, un système linéaire vérifie les propriétés suivantes :

- linéarité: toute combinaison linéaire $af_1(x,y) + bf_2(x,y)$ produit en sortie

$ag_1(x,y) + bg_2(x,y)$

- invariance: $f(x-t, y-t)$ donne à la sortie $g(x-t, y-t)$

Un filtre linéaire est un exemple important de système linéaire. Il est utilisé pour modifier le contenu spectral d'une image afin d'en retirer l'information recherchée. Un filtre linéaire bidimensionnel est caractérisé par sa réponse impulsionnelle $h(x,y)$. L'image de sortie $g(x,y)$ résulte du produit de convolution de la réponse impulsionnelle du filtre par l'image d'entrée $f(x,y)$.

Définition II.1

La réponse impulsionnelle d'un système linéaire S est le signal de sortie s(t) de ce système quand on lui applique une impulsion unité (distribution de Dirac) à l'entrée.

$$\begin{cases} \delta(n) = 0 \text{ si } n \neq 0 \\ \delta(n) = 1 \text{ si } n=0 \end{cases} \qquad (2.2)$$

Si $e(t) = \delta(t)$ alors $s(t) = h(t)$ où $e(t)$ est le signal appliqué à l'entrée de *S*. (2.3)

Définition II.2

La convolution est définie par la relation suivante :

$$g(t) = f(t) * h(t) = h(t) * g(t) = \int_{-\infty}^{\infty} f(\alpha,\beta) h(t-\tau) d\tau \qquad (2.4)$$

Une extension dans le deux dimensions s'écrit :

$$g(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(\alpha,\beta) h(x-\alpha, y-\beta) d\alpha\, d\beta \qquad (2.5)$$

En ce qui concerne la convolution dans le domaine du traitement d'image, chaque pixel est remplacé par une combinaison linéaire des pixels qui l'entourent. Un produit de convolution, est un opérateur mathématique que l'on utilise pour multiplier des matrices entre elles. Dans le cas qui nous intéresse, nous mettons en jeu deux matrices très différentes : la matrice image, très grande et une matrice plus petite qu'on appelle le noyau parce que c'est le « cœur » de tous les changements qui vont affecter l'image. Ce noyau représente le filtre, il est connu sous le nom de *« convolution de Kernel »*. Le noyau va donc agir sur chacun des pixels, c'est-à-dire sur chacun des éléments de la matrice « image ».

Pour le cas d'une image numérique (dans le cas discret), la convolution est représentée par les relations (2.6) :

$$g(x,y) = \sum_{u=-N/2}^{u=N/2} \sum_{v=-N/2}^{v=N/2} h(u,v) f(x-u, y-v) = \sum_{u=-N/2}^{u=N/2} \sum_{v=-N/2}^{v=N/2} f(u,v) h(x-u, y-v) \quad (2.6)$$

où (N,N) est la dimension du filtre

- Illustration

Soit le masque 3x3 M:

$$M = \begin{bmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & m_8 \end{bmatrix} \quad (2.7)$$

Soit un voisinage de taille 3x3 de pixels de l'image centré autour du pixel p_4 situé en (x,y), notons par p_4' le résultat de la convolution de l'image par le masque M calculé en (x,y) :

$$p_4' = m_0 * p_0 + m_1 * p_1 + m_2 * p_2 + m_3 * p_3 + m_4 * p_4 + m_5 * p_5 + m_6 * p_6 + m_7 * p_7 + m_8 * p_8 \quad (2.8)$$

En (x,y) , on remplace p_4 par p_4' .

	$x-1$	x	$x+1$
	$\vdots$	$\vdots$	$\vdots$
$y-1$	$\dots p_0$	p_1	$p_2 \dots$
y	$\dots p_3$	p_4	$p_5 \dots$
$y+1$	$\dots p_6$	p_7	$p_8 \dots$
	$\vdots$	$\vdots$	$\vdots$

Figure 2.7 : Voisinage de taille 3x3 d'un point p_4

II.3.3.2Utilisation de la transformée de Fourier

Il est également possible de réaliser le filtrage d'une image dans le domaine fréquentiel. Pour cela, on multiplie la transformée de Fourier de l'image par le conjugué de la réponse fréquentielle du filtre, et on calcule la transformée de Fourier inverse du résultat. En effet, on a vu que le filtrage est lié à la convolution .On notera *I* l'image source, *h* le filtre, *F* l'image filtrée, *I, H et F* les transformées de Fourier correspondantes.

$$F(x,y) = (g * I)(x,y) \Leftrightarrow F(u,v) = G(u;v) I(u,v) \quad (2.9)$$

où $g(x,y) = h(-x,-y)$ et où $G(u,v) = H^*(u,v)$ (2.10)

Pour que le calcul dans le domaine fréquentiel soit aisé à programmer, il faut que la transformée de Fourier soit discrète, c'est-à-dire que les indices (u,v) soient entiers. Ceci sera réalisé si les fonctions *F*, *h*, et *I* sont périodiques. On suppose donc que l'image est périodique, autrement dit, elle se répète identique à elle même hors de son support. On doit également supposer que le filtre est périodique: pour des questions de cohérence, sa période doit bien entendu être la même que celle de l'image. Si la taille de l'image est $N \times M$, on aura donc :

$$I(x+N, y+M) = \square(x,y) \tag{2.11}$$

$$h(x+N, y+M) = h(x,y) \tag{2.12}$$

$$F(x+N, y+M) = F(x,y) \tag{2.13}$$

Les transformées de Fourier discrètes sont données respectivement par :

$$\square(u,v) = \sum_{x=0}^{N-1}\sum_{y=0}^{M-1} I(x,y)\exp\left[-2j\pi\left(\frac{ux}{N}+\frac{vy}{M}\right)\right] \tag{2.14}$$

$$H(u,v) = \sum_{x=0}^{N-1}\sum_{y=0}^{M-1} h(x,y)\exp\left[-2j\pi\left(\frac{ux}{N}+\frac{vy}{M}\right)\right] \tag{2.15}$$

$$F(u,v) = \sum_{x=0}^{N-1}\sum_{y=0}^{M-1} f(x,y)\exp\left[-2j\pi\left(\frac{ux}{N}+\frac{vy}{M}\right)\right] \tag{2.16}$$

Remarques : Selon les fréquences de coupure, trois types de filtres peuvent être distingués :

- Les filtres passe-bas qui ne modifient pas ou peu les basses fréquences et qui atténuent les hautes fréquences. Ils sont caractérisés par une fonction de transfert de la forme :

$$H(u,v) = \begin{cases} 1 \text{ si } \sqrt{u^2+v^2} \le R_0 \\ 0 \text{ si } \sqrt{u^2+v^2} > R_0 \end{cases} \tag{2.17}$$

où R_0 est un seuil ou fréquence de coupure.

- Les filtres passe-haut qui ne modifient pas ou peu les hautes fréquences et qui atténuent les fréquences basses (inférieures à un seuil R_0).

$$H(u,v) = \begin{cases} 1 \text{ si } \sqrt{u^2+v^2} \ge R_0 \\ 0 \text{ si } \sqrt{u^2+v^2} < R_0 \end{cases} \tag{2.18}$$

- Les filtres passe-bande qui atténuent les composantes dont la fréquence est située en dehors d'une plage (R_0,R_1) fixée en avance.

$$H(u,v) = \begin{cases} 1 \text{ si } R_0 \le \sqrt{u^2+v^2} \le R_1 \\ 0 \text{ sinon} \end{cases} \tag{2.19}$$

II.3.4Exemple d'un filtre non linéaire : le filtre Médian

Les filtres non linéaires ont été développés pour pallier aux insuffisances des filtres linéaires, principalement la mauvaise conservation des contours. On a développé une variété de filtres qui ne sont pas linéaires. Tandis qu'ils ne peuvent pas, en général, être soumis à l'analyse de Fourier, leurs propriétés et domaines d'application ont été étudiés intensivement. Un des filtres non linéaires le plus connu est le filtre médian.

Le filtre médian a été proposé par Tukey en 1970 pour l'analyse des séries temporelles. Ce filtre est basé sur la notion du médian des observations où dans une telle approximation on peut présenter nos observations par son médian qui peut être défini comme un élément séparant l'ensemble des observations en deux parties égales. Le médian s'exprime sous la forme :

$$med(x_i) = \begin{cases} x_{(v+1)} & \text{si } i = 2v+1 \\ \frac{1}{2}\left[x_{(v)} + x_{(v+1)} \right] & \text{si } i = 2v \end{cases} \qquad (2.20)$$

Le filtre médian est capable de réduire certains types de bruits en dégradant très peu le contour, il a pour but de supprimer les bruits impulsionnels dans une image. Par bruits impulsionnels ou « speckel », on désigne des points présents dans une image qui sont très différents de leurs voisins.

II.3.4.1Principe

Pour supprimer ce type de bruit, on calcule l'intensité de la lumière sur les points voisins au point que l'on considère. Ensuite, on trie tous les points en fonction de leur intensité lumineuse et on prend le point avec l'intensité médiane (du milieu) comme point résultant du filtrage. Le voisinage V de l'image peut avoir n'importe quelle forme, par exemple une ligne (horizontale, verticale ou oblique) si l'on veut privilégier une direction, ou bien comme c'est très souvent le cas, un carré.

Ceci est illustré par le tableau suivant :

Exemple :	Caractéristiques :	Classement :	Résultat du filtrage
1 9 3 4 8 2 7 5 6	Voisinage 3x 3 autour du point P de niveau de gris 8	1 2 3 4 5 6 7 8 9	La valeur 8 est remplacée par la valeur 5 (position médiane).1 et 9 sont les valeurs extrêmes.

Tableau 1 :filtrage médian

Il n'est pas à proprement parler ici d'un produit de convolution mais sa mise en œuvre sur l'image y est assez similaire puisqu'un noyau est appliqué sur l'image et collecte les valeurs des pixels.

II.3.4.2Propriétés

- Sur une zone homogène (moyenne M) bruitée (moyenne m, variance μ) les niveaux de gris bruités seront rejetés aux extrémités du classement, tandis que les niveaux de gris centrés autour de la valeur moyenne ($M + m$) se trouveront dans la partie médiane. Clairement, le bruit sera éliminé si μ est faible devant $M + m$.
- Au niveau d'une transition abrupte, que l'on peut considérer comme juxtaposition de deux zones localement homogènes, chaque zone homogène bruitée sera filtrée indépendamment de l'autre. Clairement, la transition apparaîtra par juxtaposition de deux zones homogènes lissées et de niveaux fortement différents. Les contours seront donc conservés.

Toutes les manipulations et opérations sur les images qui ont été considérées sont nécessaires pour les différentes applications du domaine de traitement des images. Elles ne sont cependant que des notions de base vu les nombreuses possibilités offertes par la technologie actuelle. Elles permettent toutefois de corriger un grand nombre de défauts ou d'anomalies qui pourraient survenir sur des données images. Ces traitements peuvent également intervenir avant ou après une autre opération essentielle dans ce domaine : la détection de contour. Celle-ci fera l'objet de l'étude que nous réaliserons dans le chapitre suivant.

CHAPITRE III:LA DETECTION DE CONTOURS

III.1Généralités [12][13][14]

D'après l'introduction, les contours constituent des indices riches pour toute interprétation ultérieure de l'image comme les formes d'objets dans une scène. Très schématiquement, les contours sont les lieux de variations significatives de l'information sur les niveaux de gris. Autrement dit, on définit un contour comme une discontinuité locale de l'intensité lumineuse. La détection de contour revient à détecter les points de fortes variations dans l'image. Le principe de la détection de contour repose donc sur l'étude des dérivées de la fonction d'intensité de l'image. La transition d'une zone sombre vers une zone claire est une courbe mono dimensionnelle représentant « une rampe » ou « une marche d'escalier ». Si on considère que la limite entre ces deux zones se trouve là où la courbe est la plus pointue, alors cette limite coïncide avec un extremum de la dérivée première (gradient) ainsi qu'à un « passage par zéro » de la dérivée seconde (laplacien). On peut alors subdiviser les méthodes de détection de contours par ces opérateurs de bases. Il s'agit d'une opération sur la variation de l'intensité dans une image, alors les images prises en considération sont les images à niveaux de gris vue que les images couleurs sont faciles à convertir. Cependant, une image digitalisée n'est pas une fonction continue des variables spatiales mais plutôt d'une fonction discrète, des coordonnées spatiales de nombre entier. En conséquence, les algorithmes que nous nous présenterons peuvent seulement être vus comme approximation aux véritables dérivées spatiales de l'image spatiale continue originale.

En outre, l'opérateur de dérivée (première ou seconde) est linéaire et peut être vu comme un filtre, nous mettons l'accent alors sur l'utilisation des filtres linéaires.

III.2Les différentes méthodes de détection de contours [3][12][13][14][15][16][17]

III.2.1Analyse générale

Les procédés d'acquisition d'images incitent des perturbations qui peuvent être gênantes pour la compréhension et le traitement de la scène ; parmis ces perturbations, on remarque surtout les bruits. Ces bruits induisent des trous ou des coupures sur des formes ou des frontières lors de la détection de contour. Un procédé de débruitage permettra alors d'obtenir une image nette. Une des opérations les plus utilisées est le « lissage » ou le filtrage passe-bas qui consiste en effet à réduire les bruits. La détection de contour peut être alors vue comme une combinaison de pré-filtrage et d'une dérivation par filtrage linéaire. De plus, une image est une fonction de deux variables, il est nécessaire de définir la direction dans laquelle le dérivé est pris. Pour le cas

bidimensionnel nous prenons la direction horizontale $\vec{e}_x$, la direction verticale $\vec{e}_y$ ou une dérivée dans une autre direction peut se ramener à une combinaison des dérivées suivant les axes principaux, il s'agit alors de dérivées *directionnelles*. Il est donc possible de ramener l'hypothèse des filtres séparables

III.2.1.1Séparabilité des filtres

III.2.1.1.1Définitions

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

$$h(x,y) = h_x(x)h_y(y) \tag{3.1}$$

Ce qui se traduit pour le filtrage d'une image par

$$\begin{aligned} I'(x,y) &= h(x,y) * I(x,y) \\ I'(x,y) &= h_y(y).(h_x(x) * I(x,y)) \end{aligned} \tag{3.2}$$

Et pour les dérivées :

$$\begin{aligned} \frac{\partial I'(x,y)}{\partial x} &= I(x,y) * \left(\frac{\partial h_x(x)}{\partial x} h_y(y) \right) \\ \frac{\partial I'(x,y)}{\partial x} &= I(x,y) * \left(h_x(x) \frac{\partial h_y(y)}{\partial h_y} \right) \\ \Delta I(x,y) &= \Delta(x,y) * (\Delta h_x(x)h_y(y) + h_x \Delta h_y(y)) \end{aligned} \tag{3.3}$$

III.2.1.1.2Avantages

Le filtrage séparable comporte plusieurs avantages comme la réduction du temps de calcul d'une convolution $p * p$ de p^2 à $2p$ (pour le cas d'une image de dimension 2) ; possibilité de prise en compte d'un bruit de caractéristiques différentes selon chaque direction ; généralisation directe d'un filtre à une dimension quelconque.

III.2.1.2Les filtres de lissage

D'une manière générale, lisser une image consiste à remplacer la valeur de chaque pixel par la moyenne des valeurs de ses pixels voisins. Le résultat est une image de même nature que l'image d'origine (couleur ou niveau de gris).Le filtrage met à notre disposition un moyen simple mais efficace pour réduire le bruit dans une image. On fait l'hypothèse que le bruit a des dimensions faibles (de l'ordre de 1 pixel) devant les détails de l'image. Autrement dit, on suppose que le bruit est également réparti d'un point de vue spatial et d'un point de vue temporel. En

chaque pixel, le bruit a donc une moyenne nulle sur une infinité d'images.Dans ce cas, la moyenne des pixels avoisinant le pixel considéré sera une bonne estimation de la valeur « réelle » de ce pixel, puisque les pixels parasites, trop différents de leurs voisins, seront atténués.

On peut donc extraire le signal utile en moyennant un grand nombre d'images : sur chacune le signal sera localisé au même endroit alors que le bruit est réparti aléatoirement donc est éliminé par moyennage.

$$image\_res = \frac{1}{n}\sum_{i=1}^{n} image\ brute(i)$$

Avec *image_res=image résultante*

Si on considère que l'on fait la moyenne sur les 9 points qui entourent le point considéré, le calcul ressemble à :

$$pix\_res = \begin{bmatrix} pix(x-1,y-1) + pix(x,y-1) + pix(x+1,y+1) \\ pix(x-1,y) + pix(x,y) + pix(x+1,y) \\ pix(x-1,y+1) + pix(x,y+1) + pix(x+1,y+1 \end{bmatrix} / 9$$

Avec *pix_res*=pixel résultante

Généralement on représente le filtre sous forme d'un masque ou chaque case correspond à la valeur par laquelle il faut multiplier le point correspondant dans l'image source. Ensuite on fait la somme de toutes les multiplications et on divise par la somme des coefficients.

Faire cette opération revient à convoluer l'image par la matrice suivante :

$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$; en une dimension : $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ verticalement ou $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ horizontalement

Et il faudra diviser le résultat par 9 (9*1).

On peut ajuster l'efficacité de ce filtre en pondérant le pixel central :

$\begin{bmatrix} 1 & 1 & 1 \\ 1 & a & 1 \\ 1 & 1 & 1 \end{bmatrix}$ avec $a \geq 1$; en une dimension : $\begin{bmatrix} 1 & a & 1 \end{bmatrix}$ ou $\begin{bmatrix} 1 \\ a \\ 1 \end{bmatrix}$

III.2.2 Méthodes basées sur le gradient

III.2.2.1Schéma bloc de la méthode gradient

En principe, la détection de contours peut être illustré par la figure suivante en rappelant qu'elle est basée sur la variation de l'intensité :

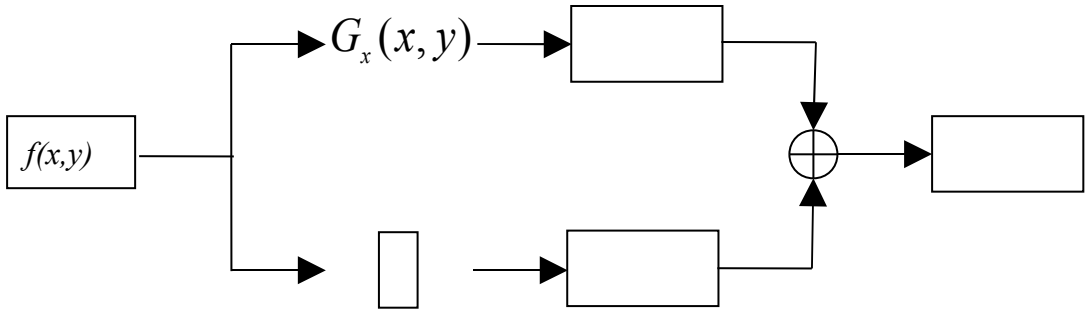


Figure 3.1 : schéma bloc de la méthode par gradient

Remarque :Le choix de l'opérateur T (.) peut avoir une influence importante sur le résultat de la détection de contours : il peut correspondre à une opération simple de seuillages ou à des méthodes rigoureuses telles que extraction des extréma locaux ou seuillage par hystérésis.

III.2.2.2Définitions

En adoptant une notation vectorielle de la dérivée, on définit le gradient d'une fonction *f* par :

$$\nabla f = \frac{\partial f}{\partial x}\vec{e}_x + \frac{\partial f}{\partial y}\vec{e}_y = (h_x \otimes f)\vec{e}_x + (h_y \otimes f)\vec{e}_y = \begin{vmatrix} G_x = \dfrac{\partial f(x,y)}{\partial x} \\ G_y = \dfrac{\partial f(x,y)}{\partial y} \end{vmatrix} \qquad (3.4)$$

Le gradient d'une fonction $f(x, y)$ est une fonction vectorielle. Mais plutôt que de recourir aux composantes en x et y, on peut caractériser un gradient par son amplitude ou son module et sa direction :

-le module ou l'amplitude du gradient :

$$m(x,y) = |\nabla f| = G = \sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2} \qquad (3.5)$$

$$= \sqrt{|G_x|^2 + |G_y|^2} \approx \max(|G_x|, |G_y|) \approx |G_x| + |G_y| \qquad (3.6)$$

-la direction du gradient :

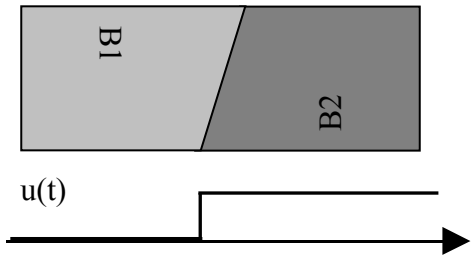
$$d(x,y) = \arctan\left(\frac{\dfrac{\partial f(x,y)}{\partial y}}{\dfrac{\partial f(x,y)}{\partial x}}\right) = \left(\frac{G_y}{G_x}\right) \qquad (3.7)$$

L'expression de la norme du gradient permet d'éviter le calcul de la racine carrée et les élévations au carré afin d'accélérer le calcul, mais ce n'est qu'une approximation qui pourrait être tout

à fait raisonnable lorsqu'un terme dépasse largement l'autre.

III.2.2.3Interprétation géométrique du gradient

Pour mieux comprendre la signification du gradient, considérons une image constituée de deux régions plates séparée par une marche rectiligne :



Pour exprimer analytiquement cette marche :

Soient la fonction marche et la fonction impulsion

$$u(t) = \begin{cases} 1 \text{ si } t \geq 0 \\ 0 \text{ si } t < 0 \end{cases} \quad ; \quad \delta(t) = \begin{cases} 1 \text{ si } t = 0 \\ 0 \text{ sinon} \end{cases}$$

La fonction marche peut être écrite l'intégrale de la fonction de Dirac :

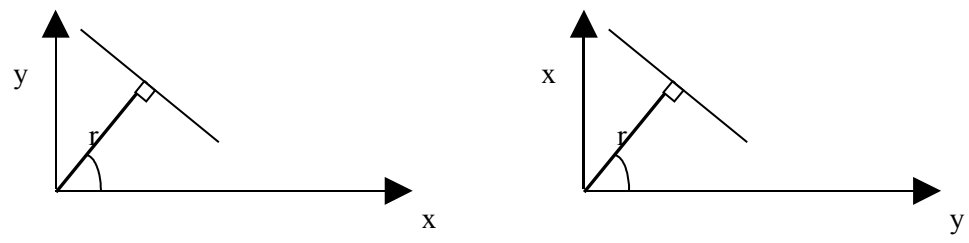
$$u(z) = \int_{-\infty}^{z} \delta(t)$$

La dérivée de u(t) est :

$$u'(t) = \delta(t)$$

L'équation de la droite décrivant la marche dans l'image faisant un angle θ avec l'axe des x est :

$r=x.\cos\theta + y.\sin\theta$



Nous pouvons maintenant écrire la fonction image :

$I(x,y)=B1+(B2-B1)\,u(-x\sin\theta + y\cos\theta + r)$

-Dans la partie claire: u (t) vaut «0 », *I=B1*

-Dans la partie foncée : u (t) vaut « 1 », *I=B1+ (B2-B1).1+B2*

Rappelons que : *u' (f(x))=f'(x) u'(x)*

Les composantes du vecteur gradient au point image *(x, y)* sont donc :

$$G_x(x,y) = \frac{\partial I_f(x,y)}{\partial x} = -\sin(\theta)(B2-B1)u(-x\sin\theta + y\cos\theta + \rho) = \cos\left(\theta + \frac{\pi}{2}\right)(B2-B1)u(-x\sin\theta + y\cos\theta + \rho) \quad (3.8)$$

$$G_y(x,y) = \frac{\partial I_f(x,y)}{\partial y} = \cos(\vartheta)(B2-B1)u(-x\sin\theta + y\cos\theta + \rho) = \sin\left(\theta + \frac{\pi}{2}\right)(B2-B1)u(-x\sin\theta + y\cos\theta + \rho) \quad (3.9)$$

-Le gradient est nul partout dans l'image sauf le long du contour en marche.

-Le vecteur gradient se trouve dans le plan image et il est perpendiculaire à la direction du contour :

$$d(x,y) = \arctan\left(\frac{G_y}{G_x}\right) = \arctan\left(\frac{\sin\left(\theta + \frac{\pi}{2}\right)(B2-B1)\,u(-x\sin\theta + y\cos\theta + \rho}{\cos\left(\theta + \frac{\pi}{2}\right)(B2-B1)u(-x\sin\theta + y\cos\theta + \rho}\right) \quad (3.10)$$

$$d(x,y) = \theta + \frac{\pi}{2}$$

-La norme du gradient est égale à la dérivée de l'image le long de la normale au contour :

$$G = \sqrt{G_x^2 + G_y^2} \quad (3.11)$$

$$G^2 = \left(\cos\left(\theta + \frac{\pi}{2}\right)(B2-B1)u(-x\sin\theta + y\cos\theta + \rho)\right)^2 + \left(\sin\left(\theta + \frac{\pi}{2}\right)(B2-B1)u(-x\sin\theta + y\cos\theta + \rho)\right)^2$$

-Le long de la normale u (t)=1

$$G^2 = \left(\cos\left(\theta + \frac{\pi}{2} \right) (B2 - B1) \right)^2 + \left(\sin\left(\theta + \frac{\pi}{2} \right) (B2 - B1) \right)^2$$

$$G^2 = (B2 - B1)^2 \left(\left(\cos\left(\theta + \frac{\pi}{2} \right) \right)^2 + \left(\sin\left(\theta + \frac{\pi}{2} \right) \right)^2 \right)$$

$$or \quad \cos w^2 + \sin w^2 = 1 \tag{3.12}$$

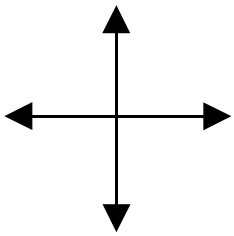
D'où :

$$\begin{aligned} G^2 &= (B2 - B1)^2 \\ G &= B2 - B1 \end{aligned} \tag{3.13}$$

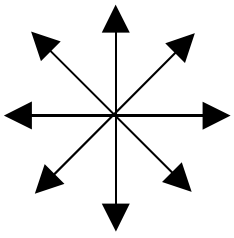
L'approche de la direction du gradient vue en () présente comme principal inconvénient le fait que la direction peut prendre n'importe quelle valeur réelle. Ceci n'est pas en accord avec la nature discrète d'une image. C'est pourquoi on peut adopter un schéma différent adapté à la nature discrète de l'image. Il s'agit alors de calculer le gradient, non plus dans deux directions, mais dans toutes les directions possibles de l'image mais basées sur ces deux directions, par exemple : 0°, 45°, 90°, 135°. On peut se contenter de ces4 directions. On a alors:

$$m(x, y) = Max_{d=0^\circ, 45^\circ, 90^\circ, 135^\circ} m_d(x, y)$$
$$d(x, y) = \arg Max_d m_d(x, y)$$

Où d désigne la direction. En général, on utilise souvent les deux principes d'orientation suivantes:



4 orientations



8 orientations

III.2.2.4Approximation du gradient

En pratique, les images sur lesquelles nous devons calculer la dérivée sont échantillonnées. Il est alors nécessaire d'échantillonner l'opérateur de dérivée première et seconde en les approximant au mieux. Il va apparaître que l'échantillonnage entraîne une erreur dans le calcul de

la dérivée. Le passage le plus simple des expressions analytiques valables pour des images continues à des images échantillonnées s'effectue en considérant un pas d'échantillonnages k identique en x et en y . Par convention, k est choisi égal à 1.Pour cela, nous utiliserons les développements de Taylor :

$$f'(x+k) = f(x) + kf'(x) + \frac{k^2}{2} f''(x) + \cdots + \frac{k^n}{n!} f^{(n)}(x) \tag{3.14}$$

$$f'(x-k) = f(x) - kf'(x) + \frac{k^2}{2} f''(x) + \cdots + (-1)^n \frac{k^n}{n!} f^{(n)}(x) \tag{3.15}$$

Par soustraction membre à membre de ces deux égalités, nous obtenons :

$$\begin{aligned} f(x+k) - f(x-k) &= 2kf'(x) + \frac{2}{3!} f^{(3)}(x) + \cdots \\ &= 2kf'(x) + O(k^3) \end{aligned} \tag{3.16}$$

Après réarrangement :

$$f'(x) = \frac{f(x+k) - f(x-k)}{2k} + O(k^2) \tag{3.17}$$

Le terme d'erreur d'ordre $O(k^2)$ est d'autant plus petit que le pas d'échantillonnage est petit. Pour un même type d'erreur, si on divise k par deux, le terme d'erreur est approximativement quatre fois plus petit. D'une manière générale, l'erreur de discrétisation (k^p) d'un opérateur est d'autant plus faible que l'ordre p de l'erreur est élevé.D'où une approximation de la dérivée première est donnée par :

$$f_a'(x) = \frac{f(x+k) - f(x-k)}{2k} \tag{3.18}$$

L'indice a indique qu'il s'agit d'une approximation. Cette approximation est dite *centrée* car elle utilise des valeurs de la fonction de part et d'autre de x . Les résultats finals de ces calculs dépendent fortement des choix de h_x et de h_y .Un certain nombre de choix possibles pour $(h_x \; h_y)$ sera maintenant décrit :

a)-$[h_x] = [h_y]^t = [1 \quad -1]$ (3.19)

b)-$[h_x] = [h_y]^t = [1 \quad 0 \quad -1]$ (3.20)

Là où « t » dénote la matrice transposez.

III.2.2.5Méthode par différences finies

Cette méthode est basée sur la dérivée première. On tient compte de l'approximation de la formule (3.18), c'est pourquoi, on parle des différences finies. Dès lors, $f(x + k, y)$ représente le pixel situé juste à droite (ou à gauche selon l'orientation de l'axe x) du pixel $f(x, y)$, pour qu'il n'y ait conflit de notation, représentons notre pixel par *I(x,y)*, car nous sommes dans le monde des images d'intensités. L'approximation la plus sommaire de la dérivée
première dans la direction x consiste à prendre la différence entre deux pixels voisins en utilisant le masque de convolution suivant de la formule (3.19):$[h_x] = [1 \ -1]$

Ce qui correspond en fait à l'approximation non-centrée suivante :*f(x+1,y)-f(x,y)* .

Ce masque présente un inconvénient majeur: le résultat est décalé d'un demi pixel par rapport aux points utilisés pour le calcul. On lui préfère généralement le masque étendu suivant :

$[h_x] = [1 \ 0 \ -1]$

Ce qui correspond à l'approximation centrée décrite à la formule (3.20)**.** Ce petit masque s'applique directement sur la grille des pixels de l'image que l'on désire traiter, en plaçant le 0 sur le pixel en cours de traitement. Une approximation de la dérivée première dans la direction y est donnée par :$[h_x]^t$. Trois jeux de filtres les plus utilisés en traitement d'image sont conçus à partir de cette méthode.

III.2.2.5.1Opérateur de Roberts

L'opérateur de Roberts est basé sur la différence en diagonale :

$$R(x,y) = |\nabla I(x,y)| = \left|D_1^2 + D_2^2\right|^{1/2} \quad (3.21)$$

Avec

$$D_1 = I(x,y) - I(x+1,y+1) \qquad D_2 = I(x,y+1) - I(x+1,y) \quad (3.22)$$

Approximativement :

$$R(x,y) = |D_1| + |D_2| \quad (3.23)$$

Les deux filtres de Roberts s'écrivent :

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \text{et} \quad \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \quad (3.24)$$

III.2.2.5.2Opérateur de Prewitt

Les formes de base effectuent un gradient ligne par ligne ou colonne par colonne. Les directions horizontales et verticales sont dès lors privilégiées. Il existe des formes qui permettent de fournir un gradient aux caractéristiques plus isotropes. Pour y parvenir, ces formes s'étendent dans les deux directions. Ces filtres sont indiqués par :

$$[h_x] = \frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \bullet [1 \quad 0 \quad -1] \tag{3.25}$$

$$[h_y] = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \bullet [1 \quad 1 \quad 1] \tag{3.26}$$

h_x et h_y sont séparables. Au delà des implications informatiques sont les implications pour l'analyse du filtre. Chaque filtre prend le dérivé dans une direction en utilisant la formule *(3.20)* et lisse dans la direction orthogonale en utilisant une version unidimensionnelle d'un filtre uniforme comme décrite dans la section (3.2.1.2).

Ce qui nous donne pour une notation plus simplifiée :

$$h_x = s(y) \bullet d(x)$$
$$h_y = d(y) \bullet s(x)$$

s correspond au lissage et *d* à la dérivée.

Autrement dit, le calcul pratique s'effectue en appliquant successivement une convolution par la colonne verticale puis par la ligne horizontale, plutôt qu'en traitant l'image par la matrice carrée.

III.2.2.5.3Opérateur de Sobel

Les filtres de SOBEL constituent une alternative aux filtres de PREWITT mais en pondérant le filtre de lissage. Ils s'expriment comme suit :

$$[h_x] = \frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \bullet [1 \quad 0 \quad -1] \tag{3.27}$$

$$[h_y] = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \bullet [1 \quad 2 \quad 1] \tag{3.28}$$

La dérivation accentue le bruit de l'image. Les opérateurs de Sobel qui effectuent une moyenne locale sur trois pixels en largeur, sont moins sensibles aux bruits .

Encore h_x et h_y sont séparables. Chaque filtre prend le dérivé dans une direction en utilisant les opérateurs de dérivées dans une direction et lisse dans la direction orthogonale en utilisant une version unidimensionnelle d'un décrit dans la section (3.2.1.2)

III.2.2.6Méthode par optimisation : méthode de Canny-Derich

Les dérivations présentées consistent à convoluer l'image par des masques de petites dimensions. Ces approches sont donc dépendantes de la taille des objets traités, elles sont aussi très sensibles au bruit. Un autre type d'approche plus récente repose sur la définition de critère d'optimalité de la détection de contour ; ces critères débouchant sur des filtres de lissage optimaux. L'implantation se fait de manière récursive, c'est-à-dire la valeur de sortie du filtre en un point est déterminée en fonction de celles de ses voisins.

Un des outils considérés comme les plus intéressants pour effectuer le prétraitement des images en vue de la détection de contours est le filtre de Canny que ce dernier a proposé en1983, son étude s'est limitée au cas de la dimension 1, c'est à dire la détection des variations dans un signal bruité. On pourra alors l'interpréter sous une forme séparable dans le cas bidimensionnel: filtrage monodimensionnel ligne par ligne puis colonne par colonne.

Il a le premier formalisé trois critères que doit valider un détecteur de contour :

-Détection : robustesse au bruit

-Localisation : précision de la localisation du point contour

-Unicité : une seule réponse par contour

A chaque critère est associé une formule mathématique.

III.2.2.6.1Modèle de Canny

Canny modélise un contour C comme un seuil d'amplitude S auquel on ajoute un bruit gaussien B de moyenne nulle et de variance n_0^2, ce qui donne :

$$C(x) = Su(x) + B(x)$$

où *u(x)* est la fonction échelon unité :

$$u(x) = \begin{cases} 0 \text{ si } x < 0 \\ 1 \text{ si } x \geq 0 \end{cases}$$

Suivant ce formalisme, le détecteur de contour idéal *h* est celui qui, convolué à *C*, présente un maximum en 0. Cette contrainte n'étant pas suffisante pour déterminer *h*, C'est pourquoi Canny a imposé les trois conditions supplémentaires citées ci-dessus à son détecteur de contour.

a-Critères de Canny

a.1-Détection :

Celle-ci doit être robuste au bruit. La qualité de la détection se mesure quantitativement comme le rapport de la réponse obtenue au point de la réponse moyenne au bruit seul élevé au carré, c'est-à-dire par le SNR (Signal to Noise Ratio) :

$$D_C = \left(\frac{A}{n_0}\right)\gamma \tag{3.29}$$

Où

$$\gamma = \frac{\left|\int_{-\infty}^{0} h(x)dx\right|}{\left[\int_{-\infty}^{+\infty} h^2(x)dx\right]^{1/2}} \tag{3.30}$$

Le terme [ ] ne dépend que du filtre. On l'utilise donc plus que D_C. On cherche le maximum de ce critère.

a.2-Localisation :

Celle-ci doit être aussi précise que possible. La qualité de la localisation se mesure comme l'inverse de la variance [ ] de la position du maximum de I'(x) :

$$L_C = \left(\frac{A}{n_0}\right)\beta \tag{3.31}$$

Où

$$\beta = \frac{|h'(0)|}{\left[\int_{-\infty}^{+\infty} h'^2(x)dx\right]^{1/2}} \tag{3.32}$$

On utilise [ ] plutôt qui ne dépend que du filtre. On recherche la maximisation de ce critère.

a.3-Unicité :

Il faut éviter des réponses multiples au voisinage d'un contour unique. On impose une valeur sur la distance moyenne entre deux maxima de I'(x). Cette valeur est exprimée comme une fraction k du support W de *h* (W est la fenêtre d'étude).

$$U_C = 2\pi\left[\frac{\left(\int_{-\infty}^{+\infty} h'^2\,dx\right)}{\int_{-\infty}^{+\infty} h''^2(x)dx}\right]^{1/2} = kW \tag{3.33}$$

b-Résolution de Canny

La maximisation de ces critères conduit à la résolution d'une équation différentielle dont la solution est le filtre *h* , qui permet la détection du contour, c'est-à-dire que la position du contour

correspond à : $\max(I * h)(x)$ On veut obtenir le maximum de γ , de β sous la contrainte $U_C =$ constante.Canny a proposé de maximiser (γ . β)ce qui n'est pas équivalent mais plus simple.

c-Solution générale :

La formalisation de ces trois conditions impose à la fonction *h* de respecter l'équation différentielle suivante :

$$2h(x) - 2\lambda_1 h''(x) + 2\lambda_2 h^{(3)}(x) + \lambda_3 = 0 \qquad (3.34)$$

dont la solution générale est :

$$h(x) = a_1 e^{\alpha x}\cos(wx) + a_2 e^{\alpha x}\sin(wx) + a_3 e^{-\alpha x}\cos(wx) + a_4 e^{-\alpha x}\sin(wx) \qquad (3.35)$$

Les conditions aux limites imposées à la fonction *h* permettent de fixer les paramètres $\lambda_1, \lambda_2, \lambda_3$:

$h(0) = 0; h(W) = 0; h'(0) = S; h'(W) = 0$

Par analyse numérique, on recherche alors la plus grande valeur de k (k appartient à [0,1]), i.e. garantir l'unicité sur la plus grande portion possible de la fenêtre d'étude.

D'après Canny, ces résultats sont :

k = 0,58 et (γ . β)= 1,12

Malheureusement, ces conditions donnent une fonction *h* très coûteuse à implémenter. Partant de ce constat, Deriche a proposé d'étudier les filtres à support infini (W = inf), ce qui simplifie la solution générale .

III.2.2.6.2Modèle de Deriche

a-Définitions

Soit la fonction de Deriche :

$$s_d = k(\alpha |x| + 1)\exp(-\alpha |x|) \qquad (3.36)$$

elle est choisie de manière à obtenir un filtre discret normalisé soit :

$$\sum s_d(x) = 1 \Leftrightarrow k = \frac{(1 - e^{-\alpha})^2}{1 + 2e^{-\alpha} - e^{-2\alpha}} \qquad (3.37)$$

Ce filtre a été proposé par R.Deriche et sa dérivée est une solution exacte à l'équation de Canny étendue à des filtres infinis.

En effet, ces conditions aux limites proposées par Deriche s'écrivent :

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

D'où la fonction :

$$h = \begin{pmatrix} \square \rightarrow \square \\ x \rightarrow c\exp(-\alpha|x|)\sin x \end{pmatrix} \qquad (3.38)$$

Posons $\alpha = mw$

On obtient :

(a) $m << 1 \qquad \beta = (2\alpha)^{\frac{1}{2}} \qquad \gamma \approx \left(\frac{2}{\alpha}\right)^{\frac{1}{2}} \qquad \gamma\beta \approx 2 \qquad k \approx 0.44$

(b) $m >> 1 \qquad \beta = (2\alpha)^{\frac{1}{2}} \qquad \gamma \approx \left(\frac{\lambda}{m}\right)^{\frac{1}{2}} \qquad \gamma\beta \approx 2m \qquad k \approx 1$

(c) $m = 1 \qquad \beta = (2\alpha)^{\frac{1}{2}} \qquad \gamma = \left(\frac{1}{\alpha}\right)^{\frac{1}{2}} \qquad \gamma\beta = 1.414 \qquad k = 0.58$

(d) $m = (3)^{\frac{1}{2}} \qquad \beta = (2\alpha)^{\frac{1}{2}} \qquad \gamma = \left(\frac{3}{2\alpha}\right)^{\frac{1}{2}} \qquad \gamma\beta = 1.732 \qquad k = 0.5$

Le cas (c) est le cas que Canny a approximé à une gaussienne, le cas (a) présente le meilleur compromis. On peut alors écrire ce filtre optimal de dérivation comme :

$$h(x) = ce^{-\alpha|x|}wx \qquad \text{avec} \qquad k = \frac{\left(1-e^{-\alpha}\right)^2}{e^{-\alpha}} \qquad (3.39)$$
$$h(x) = Cxe^{-\alpha|x|}$$

Rappelons que : pour $x \rightarrow 0; \sin x \approx x$

Le filtre de lissage s'obtient par intégration du filtre de dérivation.L'analyse des critères donnés par Canny montre que les meilleures performances du filtre sont obtenues pour w tendant vers 0.A ce moment, on obtient la formule (3.39) avec $C=wc$. Le paramètre C est un paramètre de normalisation, le paramètre α contrôle quant à lui la sensibilité de l'opérateur de détection de contour. Une augmentation de α favorise la détection au détriment de la localisation et vice-versa. Le paramètre α joue ici exactement le même rôle que le paramètre σ dans la définition du gradient de la gaussienne que nous allons expliquer plus tard.

b-Implémentation de l'opérateur de Dériche :

La détermination de filtres ayant des propriétés intéressantes pour la détection de contour répond à la question : « Que calculer ». Maintenant, il nous faut avec « quel algorithme ». En

d'autres termes, comment implanter efficacement la convolution d'un signal 1-D par une réponse impulsionnelle infinie.

Soit f(m) la fonction d'échantillonnage de f(x) et F(Z) sa transformée en Z

$$F(Z) = \sum_{-\infty}^{+\infty} f(n)Z^{-n} \tag{3.40}$$

On réécrit f(n) comme

$$f(n) = f_{-}(n) + f_{+}(n) \quad \text{où}$$

$$f_{-}(n) = \begin{cases} 0 & n \geq 0 \\ \alpha_1 (p_2)^n + \alpha_1 (p_2^*) & n < 0 \end{cases}$$

$$f_{+}(n) = \begin{cases} \alpha_1 (p_1)^n + \alpha_2^* (p_2^*) & n \geq 0 \\ 0 & n < 0 \end{cases}$$

avec $\alpha_1 = \dfrac{-c}{2ip_1} = e^{-\alpha + iw}$; $p_2 = e^{\alpha + iw}$; α^* = conjugué de α

Les équations précédentes reviennent à séparer la réponse en une partie positive f_{+} et une partie négative f_{-}.

En utilisant la transformée en Z, on reconnaît que chaque facteur $\alpha_i (p_i)^n$ a une transformée en Z égale à $\dfrac{\alpha_i}{(1 - p_i Z^{-1})}$

$$F(Z) = F_{-}(Z) + F_{+}(Z^{-1}) \quad \text{où} \quad \begin{cases} F_{+}(Z^{-1}) = \dfrac{aZ^{-1}}{(1 + b_1 Z^{-1} + b_2 \times Z^{-2})} \\ F_{-}(Z) = \dfrac{-aZ}{(1 + b_1 Z + b_2 Z^{-2})} \end{cases} \tag{3.41}$$

Avec : $a = -c.e^{-\alpha} \sin w ; b_1 = -2.e^{-\alpha} \cos w ; b_2 = e^{-2\alpha}$

Toutes singularités de $F_{+}(Z^{-1})$ (respectivement $F_{-}(Z)$) sont à l'intérieur (respectivement à l'extérieur) du cercle unité pour une valeur de α positive, ces deux transformées en Z

correspondent donc à deux fonctions de transfert rationnelles de filtres récursifs de la gauche vers la droite (F_+) et de la droite vers la gauche (F_-).

On peut montrer que la convolution du signal I par le filtre f peut être obtenue récursivement par

$y(m) = (I \otimes f)(m)$.

$1. y_+(m) = I(m-1) - b_1 y_+(m-1) - b_2 y_+(m-2)$

$2. y_-(m) = I(m+1) - b_1 y_-(m+1) - b_2 y_-(m+2)$

$3. y(m) = a(y_+(m) - y_-(m))$

avec : $C = -(1-k)^2 / k \; ; k = e^{-\alpha} ; a = (1-k)^2 ; b_1 = -2k ; b_2 = -k^2$

Et les conditions initiales :

$y_-(\max+1) = y_-(\max+2) = 0$ et $y_+(0) = y_+(-1) = 0$

III.2.2.7Du gradient au contour (choix de l'opérateur T (.))

Dans le cas d'une approche dérivée première, on dispose donc de la valeur du gradient en tout point de l'image soit de la fonction G ; Dans les premières approches, l'extraction des points de contour s'effectuait par sélection des points de norme de gradient élevée aux deux étapes suivantes :

- calcul de la norme du gradient

$$m(x,y) = \sqrt{G_X{}^2(x,y) + G_Y^2(x,y)}$$

- sélection des points forts gradient :$m(x,y)>s$

s :seuil fixé à priori

Dans le cas d'une image où la norme du gradient aux points de contour varie fortement selon les parties de l'image, cette méthode se révèle inefficace. En effet, il n'existe alors pas de seuil s permettant d'obtenir les vrais contours sans sélectionner aussi ceux dus au bruit. On fixe alors deux seuils (bas et haut) dont le dernier va nous servir de filtre . Un autre moyen de tourner cette difficulté est d'extraire non pas les points de norme de gradient élevé mais les extréma locaux de la norme du gradient. Une méthode efficace consiste à déterminer les maxima de la norme du gradient dans la direction du gradient. Dans une dernière approche, on élimine les points de norme de gradient faible avec un seuillage par hystérésis. Ce type de seuillage permet

l'obtention de points de contour bien connectés entre eux. On obtient donc les traitements suivants :

III.2.2.7.1Détermination de seuils sur l'amplitude :

Il est très difficile de fixer un idéal pour toutes les applications d'extraction de contours. La meilleure méthode reste bien sûr un idéal la connaissance a priori du contenu de la scène. Si celle ci n'est pas accessible, on peut se référer à la technique proposée par Voorhees et Poggio. Dans cette approche, on détermine deux seuils s_h (seuil haut) et s_b (seuil bas) tels que :

- Si m(x,y) < s_b alors on est sûr que (x,y) n'est pas un point contour.
- Si m(x,y) > s_h alors on est sûr que (x,y) n'est pas un point contour.
- Si $s_b \leq$ m(x,y) $\leq s_h$ alors l'appartenance à un contour sera fonction du contexte .

Les seuil s_b et s_h proviennent d'une modélisation de la distribution des amplitudes sous l'hypothèse suivante : soit I l'image, composée d'une structure S et perturbée par un bruit B.

On pose : I(x,y) = S(x,y) + B(x,y) , s_h permet d'éliminer les réponses liés aux bruits en considérant que les plus fortes variations correspondent aux bruits..

III.2.2.7.2Extraction des extréma locaux du gradient :

Soit un point (x, y) de gradient *G(x,y)* et d une distance seuil (par exemple d=1). Soient (x_1, y_1) et (x_2, y_2) deux points de la droite passant par *(x,y)* et de vecteur directeur *G(x,y)* situés à une distance d de *(x,y)* : (x_1, y_1) est pris dans le sens du gradient et (x_2, y_2) dans le sens inverse.
On détermine une approximation du gradient aux points (x_1, y_1) et (x_2, y_2)

Dans la direction du gradient, la réponse à une "variation pure" permet de localiser parfaitement le lieu de cette variation,c'est-à-dire la valeur maximale de l'amplitude du gradient.
Dans la pratique, il est nécessaire de faire en sorte qu'une seule réponse soit détectée. On ne conserve donc que les maxima locaux dans la direction du gradient :
Par exemple (x,y) est conservé si et seulement si :

$$\begin{cases} m(x, y) \geq m(x_1, y_1) \\ m(x, y) > m(x_1, y_1) \end{cases}$$

Le fait d'imposer que le maxima soit strict dans un sens revient à choisir si on localise le point de contour dans la zone de plus faible ou de plus forte valeur de la fonction des niveaux de

gris. Cela revient à se déplacer le long de la normale au contour (approximé par le gradient) et à détecter le point de plus fort gradient.

III.2.2.7.3Seuillage par hystérésis des extréma

Le principe du seuillage par hystérésis est de sélectionner parmi tous les extréma dont la norme du gradient est supérieur à un seuil bas s_b, ceux tels qu'il existe un chemin composé de points dont la norme du gradient est plus élevée que le seuil bas entre l'extremum considéré et un extremum de norme de gradient plus élevé qu'un seuil s_h. En effet, Le seuillage par hystérésis consiste à ne conserver que :

- les points dont la norme du gradient est supérieure au seuil haut s_h.

- les points dont la norme du gradient est supérieure à un seuil bas s_b et appartenant à un bout de contour dont un au moins possède une norme de gradient supérieure au seuil haut avec $s_b < s_h$)

Ce type de seuillage permet de diminuer le nombre de bouts de contour non fermés.

En effet, la reconstitution des points par hystérésis se fait comme suit : partant des éléments essentiels, on extrait les chaînes (ensembles de points contours connexes). La reconstitution des points contours consiste à propager les extrémités des chaînes dans la direction locale du gradient, sous la contrainte que les points agglomérés aient une amplitude de gradient supérieure à s_b.

Variantes :

- on peut fixer une longueur maximale de propagation
- on peut ne pas tenir compte de la direction locale du gradient et propager une extrémité sur celui des voisins immédiats (il ne peut y en avoir plus de 3) ayant le plus fort gradient.

Cette étape est la plus longue car elle met en jeu des algorithmes plus complexes qu'un simple calcul. Sa complexité dépend fortement du nombre d'extrémités.

III.2.3Méthodes basées sur le laplacien

III.2.3.1Schéma bloc

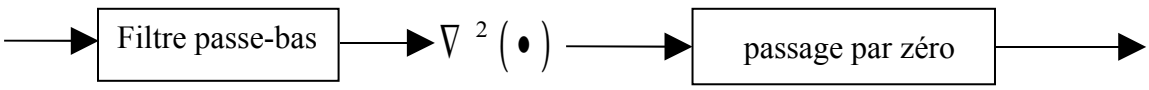


Figure 3.2 : schéma bloc de la méthode par laplacien

III.2.3.2Définitions

Dans le cas d'une image, il n'existe pas une dérivée seconde unique mais 4 dérivées partielles (selon x^2, y^2, xy et yx). En pratique, on lève cette ambiguïté en ayant recours à l'opérateur Laplacien qui fait la somme des deux dérivées partielles principales.

$$\begin{aligned}\nabla^2 f &= \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \\ &= (h_{xx} \otimes f) + (h_{yy} \otimes f)\end{aligned} \tag{3.42}$$

III.2.3.3Approximation du laplacien

Similairement au cas de la dérivée première :

$$f(x+k) + f(x-k) = 2f(x) + k^2 f''(x) + \frac{2}{4!}k^4 f^{(4)}(x) \tag{3.43}$$

D'où $$f''(x) = \frac{f(x+k) - 2f(x) + f(x-k)}{k^2} + O(k^2) \tag{3.44}$$

Même raisonnement que dans les dérivées premières dans le cas de l'erreur d'ordre $O(h^2)$ qui sera d'autant plus petit que le pas d'échantillonnage est petit , on a l'approximation centrée suivante :

$$f_a''(x) = \frac{f(x+k) - 2f(x) + f(x-k)}{k^2} \tag{3.45}$$

Deuxième filtre dérivée de base, ce filtre est indiqué par :

$$[h_{xx}] = [h_{yy}]^t = [1 \quad -2 \quad 1] \tag{3.46}$$

En effet, afin de limiter les réponses dues au bruit de l'image I, elle est préalablement filtrée par un filtre h. On obtient alors, grâce aux propriétés de l'opérateur produit de convolution :

$$\nabla^2 (I \otimes h) = (\nabla^2 I) \otimes h = I \otimes (\nabla^2 h) \tag{3.47}$$

Les opérateurs de filtrage et de dérivation se font donc en une seule étape de calcul.

En effet :

$\nabla^2 = f(x+1,y) + f(x;y+1) + f(x-1,y) + f(x,y-1) - 4f(x,y)$ suivant les 4 orientation,

$\nabla^2 = f(x+1,y) + f(x;y+1) + f(x-1,y) + f(x,y-1) - 4f(x,y)$
$+ f(x+1,y+1) + f(x-1,y-1) + f(x+1,y-1) + f(x-1,y+1) - 4$ suivant 8 orientations

Ainsi :

$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ pour le premier cas et $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ pour le second cas

Remarque :Considérons la dérivée *partielle* de la fonction $f(x, y)$ par rapport à x(horizontal). La transformée de Fourier de cet opérateur vaut :

$$F\left(\frac{\partial f}{\partial x}(x,y)\right) = 2ju\pi\ F(u,v)$$

La dérivée par rapport à x est donc équivaut à multiplier la transformée de FOURIER de $f(x,y)$ par la fonction de transfert $H_x = 2ju\pi$

c'est-à-dire filtrer $f(x, y)$ avec le filtre dont la réponse impulsionnelle est donnée par

$$h_x(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}(2\pi ju)\exp[2\pi j(xu + yv)]dudv$$

De même, on peut définir la réponse impulsionnelle $h_y(x,y)$ correspondant au filtrée dérivée dans la direction verticale. Le module de la fonction de transfert étant égal à $H_x = 2j|u|$ on peut remarquer qu'il y a un effet accentueur des hautes fréquences, d'où le caractère passe-haut de l'opérateur dérivée. C'est pareil pour $h_y(x, y)$:

$$h_y(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}(2\pi jv)\exp[2\pi j(xu + yv)]dudv$$

III.2.3.4Exemple d'opérateur utilisant le laplacien : l'opérateur optimal de Marr-Hildreth

La détection de contour par Marr-Hildreth utilise le filtrage passe-bas de l'image par des filtres gaussiens avec des variances différentes. Ensuite une application du filtre laplacien aux résultats. Les deux premières étapes peuvent être combinées, une autre appellation de cette méthode est la méthode de « laplacian of gaussian » ou tout simplement « Log ».

III.2.3.4.1Filtre gaussien

Le filtre gaussien est un filtre isotrope, c'est-à-dire non directionnel avec des propriétés mathématiques bien précises. Elle est définie par la fonction :

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{x^2}{2\sigma^2}\right) \qquad (3.48)$$

On peut l'interpréter par la figure :

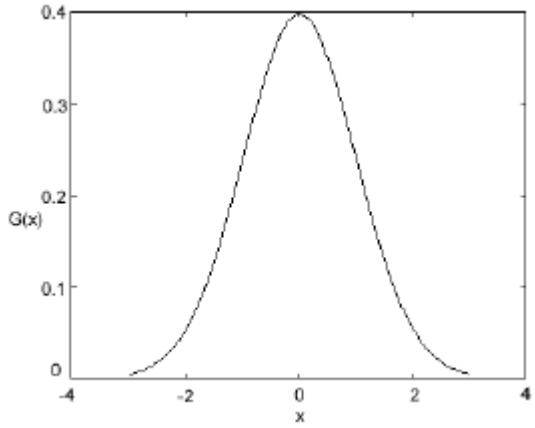


Figure 3.3 : filtre gaussien unidimensionnel

Et donc pour dérivée :

$$G'(x) = -\frac{x}{\sigma^3\sqrt{2\pi}}\exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{3.49}$$

Comme l'image est à deux dimensions :

$$G(x,y) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{\left(x^2+y^2\right)}{2\sigma^2}\right) \tag{3.50}$$

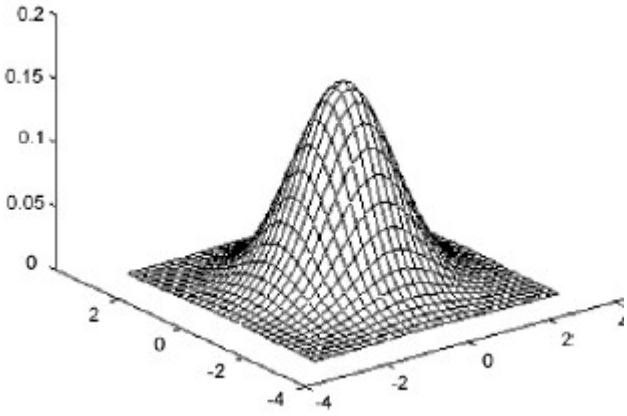


Figure 3.4 : filtre gaussien bidimensionnel

Le paramètre σ s'appelle la déviation standard, et détermine la largeur de la cloche Gaussienne. En général,un filtre gaussien avec un $\sigma < 1$ est utilisé pour réduire le bruit ,et dans la cas contraire ,c'est dans le but de fabriquer une image qu'on va utiliser pour faire un « masque flou » personnalisé. Il faut noter que plus la sigma est grand, plus la cloche Gaussienne est large et plus le flou appliqué à l'image sera marqué.

Les propriétés de réduction de bruit des filtres gaussiens peuvent être utilisées en combinaisons avec d'autres filtres qui au contraire génèrent du bruit, comme les filtres laplaciens.

Canny a montré que la dérivée de ce filtre constitue une approximation de la solution optimale. Ce filtre a été initialement introduit par Marr et Hildreth dans le cadre du calcul du Laplacien (Laplacien of Gaussian Log).Le laplacien du gaussien s'écrit alors :

$$G''(r) = -\frac{1}{\pi\sigma^4}\left(1-\frac{r^2}{2\sigma^2}\right)\exp\left(-\frac{r^2}{2\sigma^2}\right) \quad \text{avec } r = \sqrt{x^2+y^2} \tag{3.51}$$

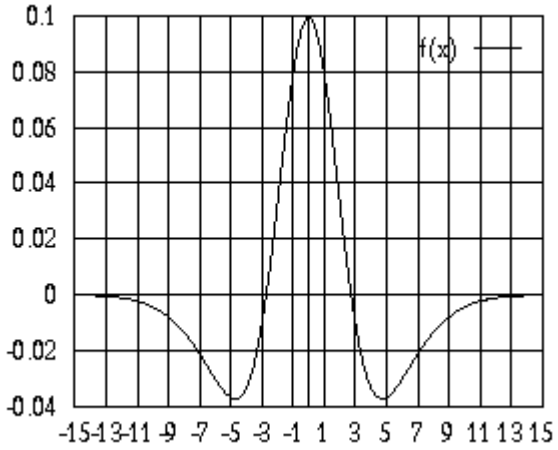


Figure 3.5 : laplacien du gaussien

Plus connu sous le nom de "chapeau mexicain" illustré par la figure (3.5) ; cette fonction sert de modèle au fonctionnement de la cellule ON/OFF, base de notre système visuel. Le paramètre σ sert a régler la résolution à laquelle les contours sont détectés. Son choix est très difficile *a priori* car il est peu probable qu'une valeur unique permette de représenter efficacement toutes les résolutions présentes dans une image.

III.2.3.4.2Implantation du filtre gaussien

Soit l'opérateur de lissage simplifiée gaussien à une dimension : $g(x) = \frac{e^{-x}}{2\sigma^2}$

Les parties positives et négatives de la gaussienne et de ses dérivées peuvent être approximées par un filtre d'ordre quatre avec une erreur très faible de 10^{-6} :

$$h(x) = \left(a_0\cos\left(\frac{w_0}{\sigma}n\right)+a_1\sin\left(\frac{w_0}{\sigma}n\right)\right)\exp\left(-\frac{b_0}{\sigma}x\right)+\left(c_0\cos\left(\frac{w_1}{\sigma}n\right)+c_1\sin\left(\frac{w_1}{\sigma}n\right)\right)\exp\left(-\frac{b_1}{\sigma}x\right) \tag{3.52}$$

On réalise ces approximations de la manière suivante :

$$h(x) = \sum_{i=1}^{m}\alpha_i\exp\left(-\frac{\lambda_i x}{\sigma}\right) \tag{3.53}$$

Nous allons maintenant donner les résultats d'approximation obtenus pour une gaussienne et ses dérivées normalisées d'ordre 1,2,3 que nous noterons :

$$\sum_{-\infty}^{+\infty} g(x)dx = 1 \; ; \sum_{-\infty}^{+\infty} xg_1(x)dx = 1 \; ;$$
$$\sum_{-\infty}^{+\infty} xg_2(x)dx = 0 \text{ et } \sum_{-\infty}^{+\infty} \frac{x^2}{2} g_2(x)dx = 1 \; ; \tag{3.54}$$
$$\sum_{-\infty}^{+\infty} \frac{x^3}{6} g_3(x)dx = 1 \text{ et } \sum_{-\infty}^{+\infty} xg_3(x)dx = 0$$

On définit des variables $\gamma_1, \gamma_2, \gamma_3, \gamma_4$ comme suit :

$$\gamma_1 = \sum_{k=0}^{+\infty} -\frac{k^2}{2\sigma^2} \; ; \gamma_2 = \sum_{k=0}^{+\infty} k^2 - \frac{k^2}{2\sigma^2} \; ; \gamma_3 = \sum_{k=0}^{+\infty} k^4 - \frac{k^2}{2\sigma^2} \; ; \gamma_4 = \sum_{k=0}^{+\infty} k^6 - \frac{k^2}{2\sigma^2}$$

Une fonction $F(\sigma) = \sum_{k=0}^{+\infty} k^\alpha - \frac{k^2}{2\sigma^2}, \alpha \geq 0$ peut être approximé avec une très faible erreur par une

fonction polynomiale $F_a(\sigma) = I(\alpha)\sigma^{\alpha+1}$ avec $I(\alpha) = \int_0^{+\infty} t^\alpha - \frac{t^2}{2} dt$

Nous avons $I(0) = \sqrt{\frac{\pi}{2}}, I(1) = 1$ et $I(\alpha + 2) = (\alpha + 1) I(\alpha)$

Si $\alpha = 0,$ on approxime $F(\sigma)$ avec $F_a(\sigma) = 1.25\sigma - 0.5$ pour $\sigma \geq 0.5$

Ainsi, nous obtenons les approximations suivantes :

$$\gamma_1 \approx 0.5 + 1.25\sigma \; ; \gamma_2 \approx \sigma^3 \sqrt{\frac{\pi}{2}} \; ; \gamma_3 \approx 3\sigma^5 \sqrt{\frac{\pi}{2}} \; ; \gamma_4 \approx 15\sigma^7 \sqrt{\frac{\pi}{2}}$$

Les coefficients de normalisation sont égaux à :

$$\beta_0 = -\frac{1}{2\gamma_1 - 1} \approx \frac{0.4}{\sigma} \; ; \beta_1 = \frac{2\gamma_2}{-2\gamma_2^2 + 2\gamma_3 - \gamma_4} \approx -\frac{0.4}{\sigma^3} \; ; \beta_3 = -\frac{2\gamma_1 - 1}{2\gamma_2} \approx \frac{1}{\sigma^2}$$

$$\beta_4 = \frac{3}{\frac{\gamma_4\gamma_2}{\gamma_3\sigma^2} - \frac{\gamma_3}{\sigma^2}} \approx \frac{1.2}{\sigma^3} \; ; \beta_5 = \frac{3\gamma_2}{\gamma_3\left(\frac{\gamma_4\gamma_2}{\gamma_3\sigma^2} - \frac{\gamma_3}{\sigma^2}\right)} \approx \frac{0.4}{\sigma^5}$$

Nous obtenons les résultats suivants pour les coefficients des filtres d'approximation :

	$g(x) = \beta_0 \exp\left(-\frac{x^2}{2\sigma^2}\right)$	$g_1(x) = \beta_1 \exp\left(-\frac{x^2}{2\sigma^2}\right)$
a_0	0.657/σ	-0.173/σ
a_1	1.979/σ	-2.004/σ
c_0	-0.258/σ	0.173/σ

c_1	-0.239/σ	0.444/σ
b_0	1.906/σ	1.561/σ
b_1	1.188/σ	1.594/σ
w_0	0.651/σ	0.700/σ
w_1	2.053/σ	2.145/σ

Tableau 2 : approximation de la gaussienne

III.2.3.5Du laplacien au contour :recherche du zero crossing

Dans le cas d'une approche dérivée seconde, on dispose donc de la valeur du laplacien en chaque point de l'image soit de la fonction Δ . On considère que les points de contours sont localisés aux passages par zéro du laplacien. Si le calcul du laplacien était exact, il suffirait de sélectionner les points *(x,y)* tels que $\Delta(x,y) = 0$. Mais comme généralement, l'approximation du laplacien est assez bruitée, on détecte les points où il change de signe. Une dernière étape de seuillage est là encore nécessaire afin d'éliminer les points de trop faible gradient. L'extraction de ces passages par zéro s'effectue classiquement en trois étapes :

-Détermination d'une image de polarité

On calcule une image I_P telle que :

$I_p = 0$ si $\Delta(x,y) > 0$

$I_p = 1$ si $\Delta(x,y) < 0$

-Détection des passages par zéro

On calcule une image I_Z telle que :

(x,y)=1 si M correspond à une transition 0-1 ou 1-0

(x,y)=0 sinon

On remarque que le choix de la localisation du passage par zéro au point de laplacien positif ou négatif revient, comme pour l'extraction des extrémas locaux, à définir les points de contour dans la région la plus claire ou la plus foncée.

- Seuillage par passage par zéro

L'élimination des passages par zéro de faible norme de gradient peut s'effectuer par un algorithme de seuillage quelconque. L'algorithme de seuillage par hystérésis décrit pour l'approche dérivée première peut par exemple être utilisé. On peut aussi se servir du fait que les

passages par zéro extraits définissent des lignes fermées délimitant les régions de points connexes où le laplacien est positif ou négatif. Des méthodes reposant sur le suivi de ces frontières et sur un calcul local du gradient peuvent aussi être utilisées.
Cependant, cette dernière approche est rarement utilisée.

CHAPITRE IV:SIMULATION SOUS MATLAB 5.3

IV.1Présentation de Matlab [18]

Matlab(MATrix LABoratory) est un logiciel interactif, développé par Math Works Inc. Et destiné notamment au traitement numérique des données. Il est particulièrement efficace lorsque celles-ci sont présentées sous-forme de vecteurs ou de matrices. Matlab intègre le calcul numérique, la visualisation des résultats et la programmation dans un environnement ouvert aux développement ultérieurs. Un certain nombre de démonstrations sont accessibles en tapant « demo ».

Matlab intègre dans sa version originale les outils mathématiques classiques tels que : calcul matriciel, manipulation de fonction, graphisme…, mais il est possible d'étendre Matlab par l'acquisition d'autres modules, entre autres:

- Simulink pour la simulation des systèmes complexes.

- Toolbox pour les différents types d'applications telles que : automatique, traitement numérique de signal, traitement d'image…

Pour cette simulation nous avons eu accès à la boîte à outils de MATLAB dans sa version récente 5.3.

IV.2Fonctions de Matlab 5.3 pour le traitement d'image

La liste ci-dessous inclut toutes les fonctions de la boîte à outils du traitement d'image elle-même, plus quelques fonctions de la boîte à outils de MATLAB qui sont particulièrement utiles pour le traitement d'image. Ces fonctions sont groupées selon leur utilisation :

IV.2.1Fonctions pour l'affichage d'image

colorbar : affiche le colorbar

getimage : obtient les données à partir des axes

image : crée et affiche l'image objet

imagesc : met à l'échelle les données et les affiche comme image

immovie : faire du film à partir des image indexées multiples

imshow : affiche l'image

montage : affiche les images multiples comme des montage rectangulaires

subimage : affiche les images multiples dans une seule figure

truesize : ajuste la dimension de l'affichage de l'image

warp : affiche l'image comme une surface de texture dressée

zoom : agrandit ou réduit une image ou une graphe à 2-D

IV.2.2Fonctions pour les fichiers d'image d'entrée /sortie

imfinfo : déclare l'information à propos d'un fichier image

imread : lit un fichier image

imwrite : écrit un fichier image

IV.2.3Fonctions pour les opérations géométriques

imcrop : produit une image

imresize : redimensionner une image

imrotate : fait tourner une image

interp2 : interpolation de données à 2-D

IV.2.4Fonctions déclarant les valeurs et les statistiques du pixel :

corr2 : calcule le coefficient de corrélation à 2-D

imcontour : crée un tracé de contour des données image

imhist : affiche l'histogramme des données image

impixel : détermine la valeur de la couleur du pixel

improfile : calcule les valeurs des pixels le long d'un segment de droite

mean2 : calcule la moyenne des éléments de la matrice

std2 : calcule la déviation standard des éléments de la matrice

IV.2.5Fonctions pour l'analyse d'image

edge : détecte les contours dans une image d'intensité.

qtdecomp : exécute la décomposition en arbre quaternaire (quadtree).

qtgetblk : déclare les valeurs du bloc dans la décomposition en arbre quaternaire.

qtsetblk : établit les valeurs du bloc dans la décomposition en arbre quaternaire.

IV.2.6Fonctions pour l'accentuation d'image

histeq : rehausse le contraste en utilisant l'égalisation d'histogramme.

imadjust : ajuste les valeurs de l'image d'intensité ou du colormap.

imnoise : ajoute du bruit à une image**.**

medfilt2 : exécute le filtrage médian à 2-D.

ordfilt2 : exécute le filtrage 2-D d'ordre statistique.

wiener2 : exécute le filtrage adaptatif pour la réduction du bruit à 2-D**.**

IV.2.7Fonctions pour le filtrage linéaire

conv2 : exécute la convolution à 2-D.

convmtx2 : calcule la matrice de convolution à 2-D.

convn : exécute la convolution à N-D.

filter2 : exécute le filtrage linéaire à 2-D.

fspecial : crée des filtres prédéfinis.

IV.2.8Fonctions pour la conception des filtres linéaires 2-D

freqspace : détermine l'espacement de la réponse fréquentielle à 2-D

freqz2 : calcule la réponse fréquentielle à 2-D.

fsamp2 : conçoit un filtre à RIF 2-D en utilisant l'échantillonnage de fréquence.

ftrans2 : conçoit un filtre à RIF 2-D en utilisant la transformation de fréquence**.**

fwind1 : conçoit un filtre à RIF 2-D en utilisant la méthode de la fenêtre 1-D.

fwind2 : conçoit un filtre à RIF 2-D en utilisant la méthode de la fenêtre 2-D.

IV.2.9Fonction pour les transformations d'image

dct2 : calcule la transformée en cosinus discrète à 2-D .

dctmtx : calcule la transformée en cosinus discrète d'une matrice.

fft2 : calcule de la transformée de Fourier rapide à 2-D.

fftn : calcule de la transformée de Fourier rapide à N-D.

Fftshift : renverse en quart de cercle la sortie d'une FFT.

idct2 : calcule l'inverse de la transformée en cosinus à 2-D.

ifft2 : calcule l'inverse de la transformée de Fourier rapide à 2-D.

ifftn : calcule l'inverse de la transformée de Fourier rapide à N-D.

radon : calcule la transformée de Radon .

IV.2.10Fonctions pour le voisinage et le traitement de bloc

bestblk : choisit la dimension du bloc pour le traitement de bloc.

blkproc : exécute différent traitement de bloc pour l'image.

col2im : réarrange les colonnes de matrice en blocs.

colfilt : exécute les opérations de voisinage en utilisant les fonctions du columnwise**.**

im2col : réarrange les blocs d'image en colonnes.

nlfilter : exécute l'opération générale de glissement de voisinage .

IV.2.11Fonctions pour les opérations sur les images binaires

applylut : exécute les opérations de voisinage en utilisant les tables de consultation.

bwarea : calcule l'aire des objets dans les images binaires.

bweuler : calcule le nombre d' Euler dans les images binaires.

bwfill : remplit les régions de fond dans les images binaires.

bwlabel : étiquette les composants reliés dans les images binaires.

bwmorph : exécute les opérations morphologiques sur les images binaires.

bwperim : détermine le périmètre d'objets dans les images binaires.

bwselect : sélectionne un objet dans les images binaires.

dilate : exécute la dilatation sur une image binaire.

erode : exécute l'érosion sur une image binaire.

makelut : construit les tables de consultation (lookup table) pour usage avec *applylut*.

IV.2.12Fonctions sur le traitement des régions de base

roicolor : sélectionne une région d'intérêt basée sur la couleur.

roifill : interpole doucement dans une région arbitraire.

roifilt2 : filtre une région d'intérêt.

roipoly : sélectionne une région d'intérêt polygonale.

IV.2.13Fonction sur la manipulation de Colormap

brighten : éclaircit ou fonce le colormap.

cmpermute : réarrange les couleurs dans le colormap.

cmunique : trouve un unique colormap et l' image correspondante.

colormap : met ou prend une couleur de la table de consultation.

imapprox : rapproche une image indexée par une contre quelques couleurs.

rgbplot : trace des composants de colormap RGB .

IV.2.14Fonctions pour les conversions d'espace de couleur

hsv2rgb : convertit les valeurs de HSV en couleur d'espace RGB.

ntsc2rgb : convertit les valeurs de NTSC en couleur d'espace RGB.

rgb2hsv : convertit les valeurs de RGB en couleur d'espace HSV.

rgb2ntsc : convertit les valeurs de RGB en couleur d'espace NTSC.

rgb2ycbcr : convertit les valeurs de RGB en couleur d'espace YCBCR.

ycbcr2rgb : convertit les valeurs de YCBCR en couleur d'espace RGB.

IV.2.15Fonctions sur les types et conversions d'image

dither : convertit image en utilisant la juxtaposition de points.

gray2ind : convertit l'image intensité en image indexée.

grayslice : crée une image indexée à partir d'une image d'intensité par seuillage**.**

im2bw : convertit une image en image binaire par seuillage**.**

ind2gray : convertit une image indexée en une image d'intensité.

ind2rgb : convertit une image indexée en une image RGB.

isbw : déclare précisément une image binaire.

isgray : déclare précisément une image d'intensité.

isind : déclare précisément une image indexée.

isrgb : déclare précisément une image RGB.

mat2gray : convertit une matrice en une image d'intensité**.**

rgb2gray : convertit une image RGB ou colormap en niveaux de gris.

rgb2ind : convertit une image RGB en une image indexée.

IV.2.16Fonction sur les préférences de la boîte à outils

iptgetpref : obtient la valeur de préférence de la boîte à outils du traitement d'image.

iptsetpref : établit la valeur de préférence de la boîte à outils du traitement d'image.

IV.2.17Fonctions pour les démonstrations

dctdemo : démonstration de compression d' image DCT 2-D **.**

edgedemo : démonstration de la détection de contour.

firdemo : démonstration de filtrage FIR 2-D et de conception de filtre

imadjdemo : démonstration d'ajustement d' intensité et d'égalisation d'histogramme.

nrfiltdemo : démonstration de filtrage de réduction du bruit .

qtdemo : démonstration de décomposition en arbre quartenaire (quadtree).

roidemo : démonstration de traitement de région d'intérêt**.**

IV.2.18Fonctions pour le Diaporama

ipss001 : région étiquetant de grains de l'acier.

ipss002 : caractéristique de base logique

ipss003 : correction d'illumination non uniforme.

IV.3Techniques fondamentales de programmation sous MATLAB [17][19[20]

Dans la partie précédente, nous avons simplement listé les fonctions utilisées en MATLAB. Dans cette partie, nous allons essayer de fournir plus de détails sur les domaines d'opération les plus utilisés afin de nous guider à la formation des programmes destinés au traitement d'image. Supposons que l'on dispose de la boîte à outils du traitement d'image, qui fournit différentes fonctions de manipulation d'images.

IV.3.1Images dans Matlab et la boîte à outils du traitement d'image

La structure de base de Matlab est la matrice, cet objet peut s'adapter naturellement à la représentation des images. Cependant, Matlab ne supporte pas les valeurs complexes. Elle entrepose la plupart des images comme une matrice à deux dimensions dans laquelle chaque élément correspond à un seul pixel sur l'image affichée. Par exemple, on peut sélectionner un seul pixel de la matrice image en utilisant la suscription de la matrice normale : *I(2,5)*.Cette commande déclare la valeur du pixel à la ligne 2, colonne 5 de l'image I .

IV.3.1.1Types de données

Par défaut, MATLAB peut emmagasiner plusieurs données en matrice en double classe. Les renseignements sur ces matrices sont stockés en double précision (64 bits) de nombre de points flottant. Toutes les fonctions et les capacités de Matlab travaillent avec cette représentation.Toutefois, pour le traitement d'image, cette représentation n'est pas toujours idéale. Le nombre de pixels dans une image peut être très large ; par exemple, 1000*1000 image a un million de pixels. Auparavant, chaque pixel est représenté au moins par une matrice élément, cette image nécessitera environ 8 mégabits de mémoire. Afin de réduire la mémoire utilisée, Matlab

supporte stocker une donnée image en matrice par la classe *uint8* .Les données dans ces matrices sont stockées en 8 bits d'entiers non signés.

Le logiciel Matlab supporte les quatre types d'images les plus utilisés : images binaires, images d'intensité, images couleur RGB, images indexées.

IV.3.1.2Travailler avec les données image

Cette section discute des façons de travailler avec les matrices de données qui représentent les images, y compris: la lecture des données image à partir d'un fichier et écriture des données image en dehors du fichier, la conversion des images en d'autres types d'image et le travail avec la matrice *uint8* dans MATLAB et dans la boîte à outils.

IV.3.1.2.1Lecture et écriture des images :imread, imwrite

Supporte les formats graphiques standards dont les plus connus sont les formats jpeg et tiff (fichier*.jpg et *.tiff). Ces fichiers peuvent être crées avec l'instruction imwrite, et ils se lisent avec la fonction imread.

IMREAD lit un fichier image :

Syntaxes:

I = imread (FILENAME, FMT)

[X, MAP] = imread (FILENAME, FMT)

[...] = imread (FILENAME)

IMWRITE écrit un fichier image

Syntaxes :

imwrite (A,FILENAME,FMT)

imwrite (X,FILENAME,FMT)

imwrite (..., FILENAME)

IV.3.1.2.2Conversion d'images en d'autres types : gray2ind, ind2gray, ind2rgb, rgb2ind, rgb2ind

Pour certaines opérations, il est souvent utile de convertir une image en un autre type différent.

GRAY2IND convertit l'image d'intensité en image indexée, elle adapte, ensuite arrondit une image d'intensité afin de produire un équivalent en image indexée.

Syntaxe:

[X, MAP] = gray2ind (I, N)

IND2GRAY convertit une image indexée en une image d'intensité.

Syntaxe:

I = ind2gray (X, MAP)

IND2RGB une image indexée en une image RGB.

Syntaxe :

RGB = ind2rgb (X, MAP)

RGB2GRAY Convertit une image RGB en image à niveaux de gris

Syntaxe :

I = rgb2gray (RGB)

RGB2IND Convertit une image RGB en une image indexée.

Syntaxe :

[X,MAP] = rgb2ind (RGB,N)

IV.3.1.2.3Affichage d'image : image, imagesc, imshow

La boîte à outils présente un nombre de technique d'affichage d'image. Par exemple, la fonction *imshow* affiche n'importe quel type d'image .Ici, seules les techniques d'affichage courantes sont présentées. En Matlab, la première manière d'afficher les images est l'utilisation de la fonction *image* .Cette fonction crée un titre graphique de l'image objet et il inclut la syntaxe pour l'arrangement des diverses propriétés de l'objet.

Matlab comprend également la fonction *imagesc* qui est similaire à la fonction *image* mais qui mesure automatiquement les données entrées. Présenterons une à une ces fonctions :

IMAGE affiche une image

Syntaxe :

image (C)

IMAGESC met à l'échelle les données et les affiche comme image.

Syntaxes :

imagesc

imagesc (...)

IMSHOW affiche l'image.

Syntaxes :

imshow (I,N)

imshow (BW)

imshow (X,MAP)

imshow (RGB)

imshow FILENAME

Remarques : Sur chaque affichage, la fonction *'title'* nous permettra de titrer l'image à afficher

IMSHOW appelle IMREAD pour lire l'image du fichier graphique, mais les données de l'image ne sont pas entreposées dans la mémoire de MATLAB .Le dossier doit être dans le répertoire courant ou sur le chemin MATLAB.

La boîte à outil comprend une fonction IPTSETPREF qu'on peut utiliser pour établir les préférences qui affectent la performance de imshow :

- 'ImshowBorder' contrôle si IMSHOW affiche l'image avec une frontière autour de lui.
- 'ImshowAxesVisible' contrôle si IMSHOW affiche le l'image avec les axes et les étiquettes.
- 'ImshowTruesize' contrôle si IMSHOW appelle la fonction TRUESIZE.

IV.3.2Opérations géométriques : imresize, imrotate, imcrop

Les fonctions d'opérations géométriques modifient les caractéristiques géométriques d'une image .Ces opérations sont basées surtout sur la méthode d'interpolation consistant à échantillonner l'image pour déterminer les valeurs entre des pixels prédéfinis. La boîte à outils fournit trois méthodes d'interpolation : interpolation directe de voisinage ('nearest'), interpolation bilinéaire 'bilinear'), interpolation bicubique('bicubic').

Voici les fonctions exécutant ces opérations :

IMRESIZE rédimensionne une image

Syntaxes :

B = imresize (A,m,n,method)

B = imresize (A,[mrows ncols],method)

IMROTATE fait tourner une image

Syntaxes:

B = imrotate (A,angle,method)

B = imrotate (A,angle,method,'crop')

IMCROP découpe une image

Syntaxes:

I2 = imcrop (I)

X2 = imcrop (X,MAP)

RGB2 = imcrop (RGB)

‘method’ peut être ‘nearest’, ‘bilinear’, ‘bicubic’.

Exemple 4.1 : exemple de découpage d’une image

```
%Leture du fichier image
I=imread (‘ic.tif’);
%Réalisation du découpage
I2=imcrop (I,[60 40 100 90]);
%Affichage
subplot(221),imshow(I),
 subplot(222),imshow(J)
```

Figure 4.1:découpage d’une image (on a découpé la partie entourée)

Exemple 4.2 : exemple de rotation d’une image:

```
%Lecture du fichier image
I=imread (‘ic.tif’) ;
%Réalisation de la rotation
J=imrotate (I,35,’bilinear’) ;
%Affichage
subplot(221),imshow(I), subplot(222),imshow(J)
```

Figure 4.2: rotation d’une image

IV.3.3Filtrage linéaire et concéption de filtre

IV.3.3.1Filtrage linéaire : conv2, filter2

En Matlab, le filtrage linéaire d'une image consiste aussi à mettre en œuvre la convolution à 2-D.Nous disposons deux fonctions permettant de réaliser ce filtrage : *conv2 et filter2.*
CONV2 implémente le filtrage d'image par l'application directe de la convolution.

Syntaxe :

C = conv2 (A,B,'same')

FILTER2 peut engendrer le même résultat que conv2, cependant conv2 prend le molécule de calcul comme un argument d'entrée plutôt que la convolution de Kernel. *Filter2* fonctionne en formant le molécule de calcul à partir de la convolution de Kernel. Il s'agit d'une simple rotation de 180° de la convolution de Kernel. Dans ce cas, on parle plutôt d'une corrélation.

Syntaxe :

B = filter2 (h,A,'same')

La boite à outil possède aussi plusieurs sortes de filtres prédefines par la fonction *fspecial*

Syntaxe :

h = fspecial (type)

'type' est une série de filtre déjà prédéfinis dans Matlab,pour plus d'information voir 'help fspecial'

IV.3.3.2Conception de filtre : ftrans2, fsamp2, fwind1, fwind2

Ici, on fait appel aux fonctions concernant les vecteurs et les matrices en général :
ONES génère une matrice unitaire

Syntaxes :

ones (N) génère une matrice unitaire N*N

ones (N,M) génère une matrice unitaire M*M

ZEROS génère une matrice nulle

Syntaxes :

zeros (N) génère une matrice nulle N*N

zeros (N,M) génère une matrice nulle N*M

Nous disposons plusieurs fonctions destinées à la conception et à la mise en œuvre des filtres linéaires pour les images. On peut travailler dans le domaine fréquentiel pour concevoir ces filtres. La boîte à outils fournit une prise en charge de :

- la méthode de transformation fréquentielle :ftrans2

FTRANS2 conçoit un filtre à RIF 2-D en utilisant la transformation de fréquence**.**

Syntaxes :

h = ftrans2 (B)

h = ftrans2 (B,T)

- la méthode d'échantillonnage fréquentiel basée sur la réponse fréquentielle désirée: fsamp2

FSAMP2 conçoit un filtre à RIF 2-D en utilisant l'échantillonnage de fréquence.

Syntaxes :

h = fsamp2 (HD)

h = fsamp2 (F1,F2,HD,[M N])

- la méthode de fenêtrage exige la multiplication d'une réponse impulsionnelle idéale avec une fenêtre de fonction pour générer un filtre correspondant : fwind1, fwind2

FWIND1 conçoit un filtre à RIF 2-D en utilisant la méthode de fenêtre 1-D

Syntaxes :

h = fwind1 (HD,WIN)

h = fwind1 (HD,WIN1,WIN2)

h = fwind1 (F1,F2,HD,...)

FWIND2 conçoit un filtre à RIF 2-D en utilisant la méthode de la fenêtre à 2-D

Syntaxes :

h = fwind2 (HD,WIN)

h = fwind2 (F1,F2,HD,WIN)

IV.3.3.3Calcul de la réponse fréquentielle d'un filtre 2-D : *freqz2*

La fonction *freqz2* permet de calculer et de visualiser la réponse fréquentielle d'un filtre.

FREQZ2 calcule la réponse fréquentielle à 2-D.

Syntaxes :

[H,F1,F2] = freqz2 (h,[N2 N1])

[H,F1,F2] = freqz2 (h,N1,N2).

[H,F1,F2] = freqz2 (h)

[H,F1,F2] = freqz2 (h,F1,F2)

[...] = freqz2 (h,..., [DX DY])

Exemple 4.3 : exemple de filtrage par une filtre moyenneur

%Lecture du fichier image

```
I=imread (‘zazah.tif’) ;
%Déclaration du filtre moyenneur, il s’agit d’un filtre moyenneur 3*3 :
%h=1/9[1 1 1 ;1 1 1 ;1 1 1]=[0.111 0.111  0.111 ;0.111 0.111 0.111 ;0.111 0.111 0.111]
h=fspecial (‘average’, 3)
%lecture du fichier  image ’zazah.jpg’
I=imread(‘zazah.jpg’) ;
I = double (I)/255
%Filtrage de l’image
F = filter2(h,I) ;
```

image originale

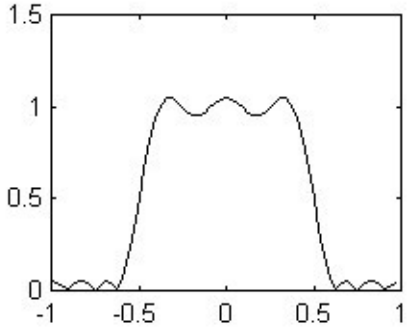
image filtrée

Figure 4.3: filtrage par le filtre moyenneur

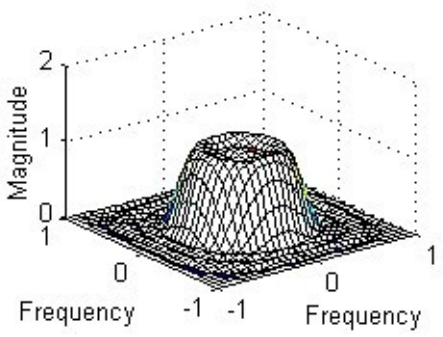
Interprétation : l’image filtrée devient ‘flou’. Le filtre a un effet d’adoucissement de l’image.

Exemple 4.4 : exemple de concéption de filtre par la méthode de transformation fréquentielle:

```
%Conception du filtre
b = remez(10,[0 0.4 0.6 1],[1 1 0 0]);
h = ftrans2(b);
[H,w] = freqz(b,1,64,’whole’);
```



réponse à une dimension



réponse à deux dimensions

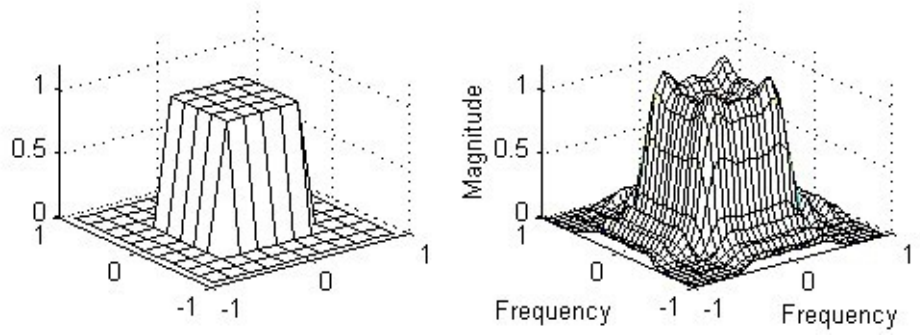
Figure 4.4 : réponses fréquentielles du filtre par la méthode fréquentielle

```
%Visualisation à une dimension
subplot(221),plot(w/pi-1,fftshift(abs(H)))
%Visualisation de la réponse fréquentielle du filtre à deux dimensions
subplot(222),freqz2(h,[32 32])
```

Exemple 4.5 : exemple de conception de filtre par la méthode d'échantillonnage fréquentiel

```
%Choix de la réponse fréquentielle
Hd = zeros(11,11);
Hd(4:8,4:8) = 1;

%Visualisation de la réponse fréquentielle désirée
[f1,f2] = freqspace(11,'meshgrid');
subplot (221),mesh(f1,f2,Hd),axis([-1 1 -1 1 0 1.2]);
%Synthèse du filtre
h = fsamp2(Hd);
%Visualisation de la réponse fréquentielle obtenue
subplot(222),freqz2(h,[32 32]),axis([-1 1 -1 1 0 1.2])
```



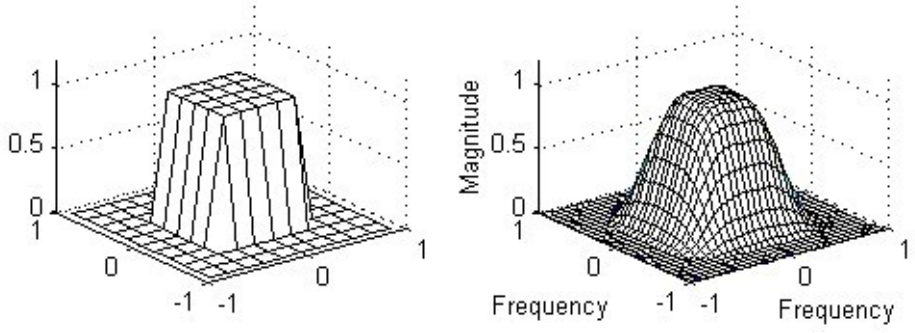
réponse fréquentielle désirée réponse fréquentielle obtenue

Figure 4.5: réponses fréquentielles du filtre par la méthode d'échantillonnage de fréquence

Exemple 4.6 : exemple de conception de filtre par la méthode de fenêtrage

```
%Choix de la réponse fréquentielle
Hd = zeros (11,11);
Hd (4:8,4:8) = 1;
%Visualisation de la réponse désirée
```

```
[f1,f2] = freqspace(11,'meshgrid');
subplot (221)
mesh (f1,f2,Hd),axis ([-1 1 -1 1 0 1.2]);
%Synthèse du filtre
h = fwind1 (Hd,hamming(11));
%Visualisation de la réponse obtenue
subplot (222)
freqz2 (h,[32 32]),axis ([-1 1 -1 1 0 1.2])
```



réponse désirée réponse obtenue

Figure 4.6 : réponses fréquentielles du filtre par la méthode de fenêtrage

IV.3.4Transformations

- Transformée de Fourier discrète : fft2, fftn

FFT2 calcule la transformée de Fourier discrète à 2-D

Syntaxes :

```
fft2 (A)
fft2 (A, MROWS, NCOLS)
```

FFTN calcule la transformée de Fourier discrète à N-D

Syntaxes :

```
fftn (A )
fftn (A,SIZ)
```

- Transformée en cosinus discrète : dct2, dctmtx

DCT2 calcule la transformée en cosinus discrète à 2-D

Syntaxes :

```
B = dct2 (A)
B = dct2 (A, [M N])
```

B = dct2 (A, M, N)

DCTMTX calcule la transformée en cosinus discrète de la matrice

Syntaxe :

D= dctmtx (N)

-Transformée de Fourier inverse : ifft2, ifftn

IFFT2 calcule la transformée de Fourier discrète inverse à 2-D

Syntaxes

ifft2 (A)

ifft (A,MROWS,NCOLS)

IFFTN calcule la transformée de Fourier discrète inverse à N-D

Syntaxes :

fftn (A)

fftn (A,SIZ)

- Transformée en cosinus discrète inverse : idct2

IDCT2 cacule la transformée en cosinus discrète inverse

Syntaxes:

B = idct2 (A)

B = idct2 (A,[M N])

B = idct2 (A,M,N)

- Transformée de Radon : radon

Comme on ne l'a pas détaillé dans la partie théorique, en résumé, la transformée de Radon calcule la projection d'une image suivant une direction spécifiée. Mathématiquement :

$$R_\theta(x') = \int_{-\infty}^{+\infty} f(x'\cos\theta - y'\sin\theta, x'\sin\theta + y'\cos\theta)dy' \qquad (4.1)$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

La fonction qui exécute cette transformée est *radon.*

RADON calcule la transformée de Radon

Syntaxes:

R = radon (I,theta)

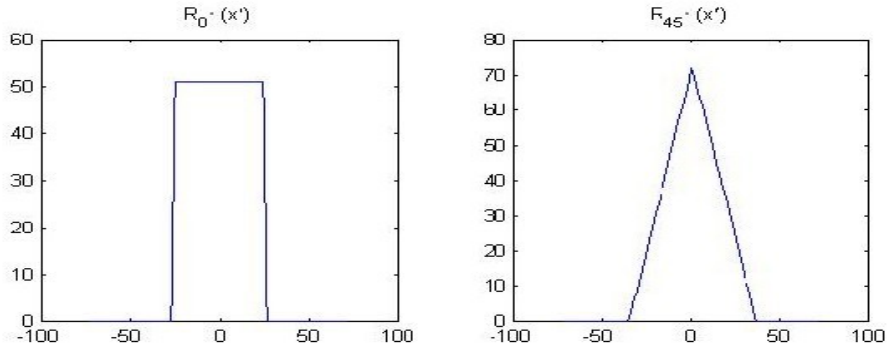
R = radon (I,theta,n)

[R,xp] = radon(…)

Exemple 4.7 : exemple de transformée de Radon

```
%Création de la matrice image
I = zeros (100,100);
I (25:75,25:75) = 1;
Imshow (I)
```

```
%Mise en oeuvre de la transformée de Radon
[R,xp] = radon (I,[0 45]);
%Visualisation pour tetha = 0°(θ =0°)
subplot (121); plot (xp,R(:,1)); title ('R_{0^°} (x\prime)')
%Visualisation pour tetha = 45° (θ =45°)
subplot (122); plot (xp,R(:,2)); title ('R_{45^°} (x\prime)')
```



θ =0° θ =45°

Figure 4.7:transformée de Radon

IV.3.5 Mesures et statistiques du pixel :

Plusieurs fonctions peuvent déclarer l'information sur les valeurs composant une image. Ces fonctions déclarent l'information sur différentes formes de la donnée image incluant :

- la lecture et l'affichage des valeurs du pixel sélectionné : impixel

IMPIXEL détermine les valeurs de la couleur de l'image élémentaire.

Les syntaxes sont les suivantes :

P = impixel (I)

P = impixel (X,MAP)

P = impixel (RGB)

- les valeurs du pixel suivant une direction : improfile

IMPROFILE calcule les valeurs des pixels d'une coupe transversale le long d'un segment de droite. En effet, affiche séparément la variation d'intensités pour chaque couleur de base (RVB)

Syntaxes :

C = improfile

C = improfile (N)

- l'affichage du tracé de contour : imcontour

IMCONTOUR crée un tracé de contour des données image.

Syntaxes :

imcontour (I)

imcontour (I,N)

imcontour (I,V)

imcontour (X,Y,...)

[C,H] = imcontour (...

- histogramme d'une image : imhist

IMHIST affiche l'histogramme d'une image.

Syntaxes :

imhist (I,N)

imhist (X,MAP)

[COUNTS,X] = imhist (...)

- le calcul des statistiques standards : mean2, std2, corr2

MEAN2 calcule la moyenne des éléments de la matrice.

Syntaxe :

B = mean2 (A)

STD2 calcule la déviation standard des éléments de la matrice.

Syntaxe :

B = std2 (A) calcule la déviation standard des valeurs dans A.

CORR2 calcule le coefficient de corrélation à 2-D

Syntaxe :

R = corr2 (A,B)

Exemple 4.8 : exemple d'utilisation de impixel

%Affichage de l'image

imshow satroka.tif

%Déclaration de la fonction

impixel

%cliquer sur la ou les couleurs de l'image dont on voudra la ou les valeurs du pixel, puis sur 'entrée'

Figure 4.8 : la sélection des pixels

ans =

0	0.5373	0.4039	(mur)
0.3961	0.6118	0.6000	(support)
0.2588	0.3216	0.3333	(cheveux)
1.0000	0.9961	0.9882	(nuage)

Exemple 4.9 : exemple d'utilisation de improfile

%Affichage de l'image

I = imshow flowers.tif

%Lorsque l'image s'affiche, on trace une ligne le long de la partie dont on voudra découvrir les variations de l'intensité

Figure 4.9.1 : traçage de la droite

%Exécution de l'opération

improfile

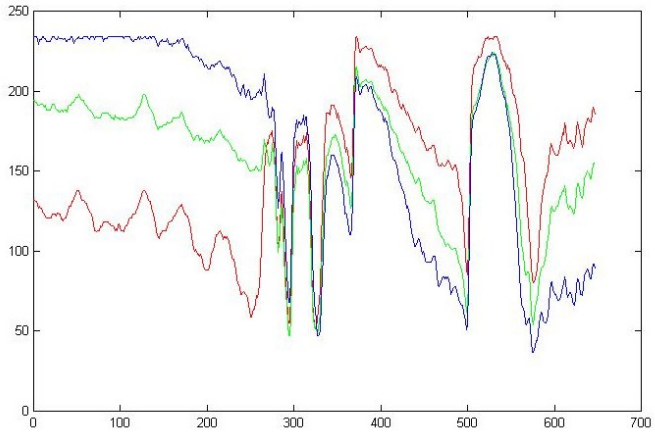


Figure 4.9.2 : visualisation de la variation d'intensités pour chaque couleur de base

IV.3.6Accentuation d'image

Les techniques d'accentuation d'image sont utilisées pour améliorer une image comme : l'ajustement d'intensité et la suppression de bruit

IV.3.6.1Ajustement d'intensité :

L'ajustement d'intensité est une technique pour dresser un graphe d'intensité d'une image évaluée à une nouvelle gamme. Ce processus peut être fait automatiquement par *imadjust* ou par l'égalisation d'histogramme en utilisant *histeq.*

IMADJUST ajuste les valeurs de l'image d'intensité ou du colormap .

Syntaxes:

J = imadjust (I,[low high],[bottom top])

J = imadjust (I,[low high],[bottom top],gamma)

Newmap = imadjust (map,[low high],[bottom top],gamma)

HISTEQ rehausse le contraste d'images en transformant le valeurs dans une image d'intensité, ou les valeurs dans le colormap d'une image indexée. HISTEQ produit une image qui a des valeurs distribuées également partout dans la gamme.

Syntaxe :

J = imhist (I, HGRAM)

J = histeq (I, N)

[J,T] = histeq (I)

NEWMAP = histeq (X,MAP,HGRAM)

NEWMAP = histeq (X, MAP)

[NEWMAP,T] = histeq(X,…)

Exemple 4.10 : exemple d'ajustement d'intensité

```
%Lecture du fichier image
I = imread ('rice.tif');
%Ajustement d'intensité
J = imadjust (I,[0.15 0.9],[0 1]);
Subplot (121), imshow (I)
title ('image originale'),
Subplot (122), imshow (J)
title  ('image ajustée')
```

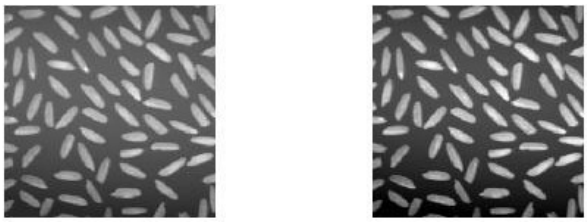


image originale image ajustée

Figure 4.10 : ajustement d'intensité

Exemple 4.11 : exemple d'affichage et d'égalisation d'histogramme :

```
%Lecture du fichier image
I = imread ('z1.jpg');
%Egalisation d'histogramme
J = histeq (I) ;
```

%Le programme suivant permet l'affichage de l'image originale, de l'image égalisée ainsi que les histogramme correspondants :

```
subplot (2,2,1), imshow (I)
title ('image originale')
subplot (2,2,3), imhist (I, 256)
title ('histogramme')
subplot (2,2,2), imshow (J)
title ('image égalisée')
subplot (2,2,4), imhist (J, 256)
title ('histogramme égalisé')
```

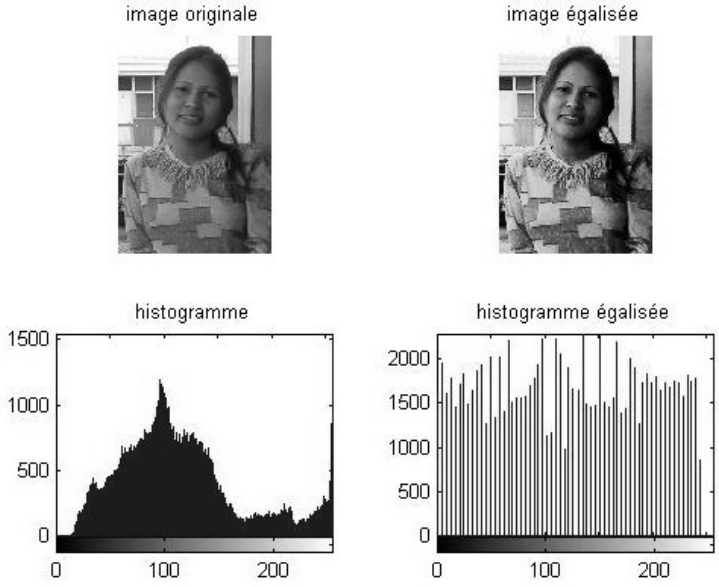


Figure 4.11 : égalisation d'histogramme

IV.3.6.2Réduction de bruit

Toolbox prévoit différentes manières pour supprimer ou réduire le bruit dans une image. Trois jeux de filtres sont disponibles pour ce processus : le filtre linéaire, le filtre médian, le filtre de Wiener.

- Addition de bruit :imnoise

IMNOISE ajoute du bruit dans l'image

Syntaxe :

J = imnoise (I,type)

Matlab propose trois différents types de bruits qu'on rencontre souvent lors du traitement d'image. Ces bruits peuvent être de type :

-‘gaussian’ pour un bruit blanc gaussian

-‘salt and pepper’ pour un bruit de type ‘poivre et sel’

-‘speckle’ pour un bruit multiplicatif

IV.3.6.2.1Filtrage linéaire

On peut utiliser le filtrage linéaire pour supprimer certains types de bruits. Certains filtres comme le filtre moyenneur et le filtre gaussien peuvent être convenables pour cette proposition. Ces deux types de filtres sont déjà prédéfinis dans Matlab :

Syntaxes :

%pour le filtre moyenneur

K = fspecial (‘average’)

%pour le filtre gaussien

K = fspecial (‘gaussian’)

IV.3.6.2.2Filtrage médian

MEDFILT2 exécute le filtrage médian.

Syntaxes:

B = medfilt2 (A,[M N])

B = medfilt2 (A)

IV.3.6.2.3Filtrage adaptatif : wiener2

WIENER2 exécute le filtrage adaptatif pour la réduction de bruit ;

Syntaxes :

J = wiener2 (I,[m,n],noise)

[J,noise] = wiener2 (I,[m,n])

Exemple 4.12: exemple de débruitage par ces quatre types de filter

```
%Lecture du fichier image
I = imread ('eight.tif');
%Addition de bruit
J = imnoise (I,'salt & pepper');
%Filtrage par le filtre moyenneur
K = filter2 (fspecial ('average',3),J)/255;
%Filtrage par le filtre médian
M = medfilt2 (J,[3 3]);
```

```
%Filtrage par le filtre de Wiener
W = wiener2 (J,[5 5]);
%Affichage
subplot (221), imshow (J)
title ('image bruitée')
subplot (222), imshow (K)
title ('image filtrée par le filtre moyenneur')
subplot (223), imshow (M)
title ('image filtrée par le filtre médian')
subplot (224), imshow (W)
 title ('image filtrée par le filtre de Wiener')
```

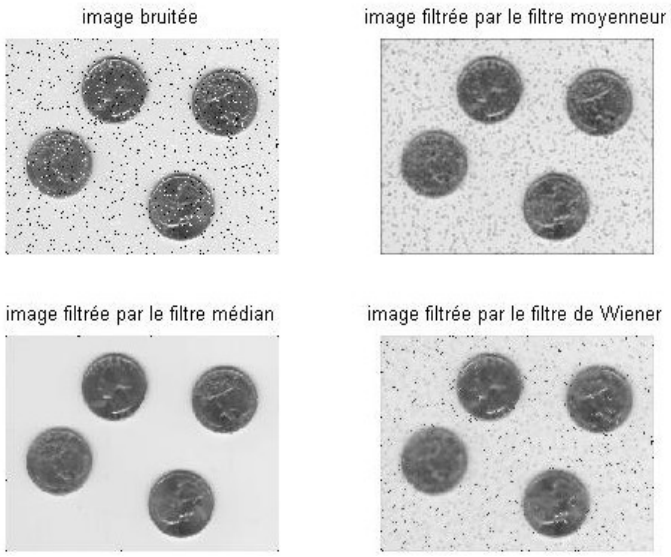


*Figure 4.12 :*débruitage par des différents filtres

Interprétation : on remarque la performance du filtre médian pour le débruitage, le filtre moyenneur ne fait que rendre l'image 'flou'. Ce type de filtre n'est pas très efficace pour les bruits de types « poivre et sel »(salt & pepper). Le filtre de Wiener produit souvent de meilleurs resultats que les filtres linéaires.

IV.3.7Opérations sur les images binaires

Une opération sur les images binaires déclare l'information sur la forme et la structure de l'image binaires seulement. Pour accomplir cette opération à un autre type d'image, il faut la convertir en une image binaire. Un des processus les plus utilisés est la morphologie mathématique.

IV.3.7.1Morphologie mathématique

On a déjà vu en théorie que les opérations de base sont la dilatation et l'érosion .

DILATE exécute la dilatation sur une image binaire

Syntaxe :

BW2 = dilate (BW1,SE)

ERODE excécute l'érosion sur une image binaire

Syntaxe :

BW2 = erode (BW1,SE)

Exemple 4.13: exemple de mise en oeuvre des opérations de dilatation et de l'érosion

```
%Lecture de l'image binaire
BW1 = imread ('circbw.tif');
%Elément structurant
SE = eye (5)
%Mise en œuvre de l'opération
BW2 = erode (BW1,SE); BW3 = dilate (BW1);
%Affichage
subplot (221), imshow (BW1), title (‘image originale’)
subplot (222), imshow (SE), title (‘élément structurant’)
subplot (223), imshow (BW2), title ('image érodée')
subplot (224), Imshow(BW3), title ('image dilatée')
```

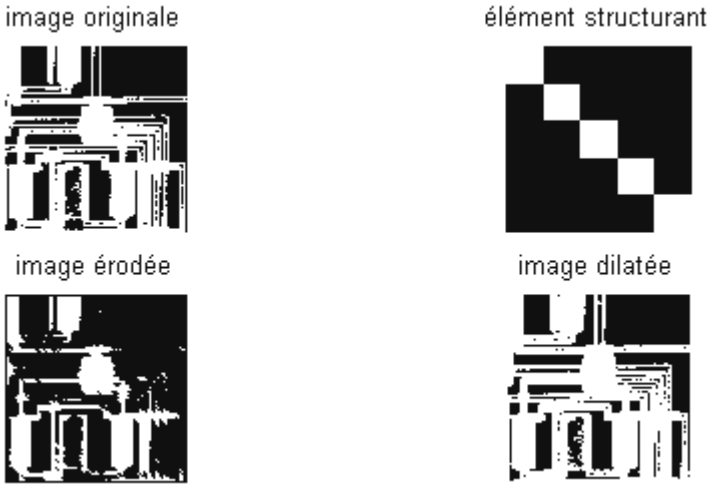


Figure 4.13. : érosion et dilatation

Cependant, il y a certaines opérations qui sont communes et que la boîte à outils fournit comme procédure prédéfinie. Ces opérations sont disponibles à travers la fonction *bwmorpho,* celle-ci fournit au total dix-huit opérations prédéfinies, y compris la dilatation, l'érosion, l'ouverture et la fermeture.

BWMORPHO exécute les opérations morphologiques sur les images binaires

Syntaxes :

BW2 = bwmorpho (BW1,operation)

'Operation' peut prendre les valeurs de ces dix-huits opérations que nous ne pourrons pas toutes présenter dans cet ouvrage. Mais pour de plus amples informations, il est préférable de visiter les préférences de toolbox ou de parcourir les « démo », ou taper simplement « help bwmorph». Supposons que l'on veuille réduire tous les objets dans l'image du circuit sans changer la structure essentielle (topologie) de l'image. Ce processus est connu par le terme : squelettisation. On pourra utiliser *bwmorph* .

Exemple 4.14 : exemple d'utilisation de la fonction *bwmorph*

```
%Lecture du fichier image
BW1 = imread ('circbw.tif');
%Mise en œuvre de la squelettisation
BW2 = bwmorph(BW1,'skel',Inf);
%Affichage
subplot (121), imshow (BW1), title ('image originale')
subplot (122), imshow (BW2), title ('image après squelettisation')
```

Figure 4.14. : squelettisation

IV.3.7.2Extraction de caractéristiques

Lors d'une opération de traitement d'image, il est possible d'obtenir de l'information à propos de certaines caractéristiques qui ont changé lors de cette opération. Par exemple, lorsqu'on

exécute une dilatation, on pourra déterminer combien de pixels ont changé. Cette partie décrit deux fonctions communes pour les mesures sur de images binaires : l'aire et le nombre d'Euler.

- L'aire de l'image peut être définie comme la dimension du premier plan de l'image. En un mot, l'aire est le nombre de pixels dans une image.

BWAREA calcule l'aire dans une image une image binaire.

Syntaxe :

Total = bwarea (BW)

Exemple 4.15 : exemple d'utilisation de bwarea dans une opération de dilatation dont le but est de découvrir la proportion des pixels qui ont subit de changement (noir en blanc) :

```
%Lecture du fichier image
BW1 = imread ('circbw.tif');
%Elément structurant
SE2 = ones (5)
%Dilatation
BW2 = dilate (BW1,SE2);
increase2 = (bwarea (BW2)-bwarea (BW1))/bwarea (BW1);
increase2 = 0.3456 ;
```

Interprétation : la dilatation a agrandi l'aire de l'image environ 35%

```
%En prenant l'élément structurant de l'exemple 4.12 :
SE3 = eye (5)
%Dilatation
BW3 = dilate (BW1,SE3);
increase3 = (bwarea (BW3)-bwarea (BW1))/bwarea (BW1)
increase3 = 0.3207
```

Le nombre d'Euler est la mesure de la topologie de l'image. Il est défini comme le nombre total d'objet dans l'image moins le nombre trous.

BWEULER calcule le nombre d'Euler dans une image

Syntaxe :

Eul = bweuler(BW1)

Exemple 4.16 :

```
BW1=imeard ('circbw.tif') ; eul=bweuler(BW1)
eul =-85
```

Interprétation : Le nombre d'Euler est négatif, indique que le nombre de trous est plus grand que le nombre d'objets. En effet, les trous sont considérés comme des « morceaux » de fond à l'intérieur des objets.

IV.4Simulation de quelques applications en traitement d'image [17][18][19]

Le logiciel que nous allons présenter effectue quelques opérations de base de traitement d'image. Il s'agit dans un premier temps d'une égalisation d'histogramme suivie de procédures de débruitage : application de la morphologie mathématique et utilisation de filtres. En second lieu, nous proposerons quelques méthodes de compression d'images. Les différentes commandes que nous utiliserons visent deux buts distinctes : traiter une image puis afficher le résultat des différentes étapes à l'écran par l'intermédiaire d'une interface graphique permettant l'interaction avec le lecteur. Avant toute chose, notons que les programmes présentés dans le paragraphe précédent sont immédiatement exécutables.

Le programme s'exécute en tapant « accueil » dans la ligne de commande Matlab. Les fenêtres du programme sont représentées par la figure 4.14.

Sur la fenêtre principale du programme se trouve la liste des applications pouvant être exécutées. Nous pouvons y choisir une application en cliquant sur celle désirée.

IV.4.1Notions de bases

Pour commencer, choisissons maintenant entre les opérations de traitement d'image proposées en cliquant sur le bouton radio correspondant ou en sélectionnant l'opération choisie dans le menu « aller à… » de la figure 4.15.

IV.4.1.1Egalisation d'histogramme

La première opération à effectuer est une égalisation d'histogramme, les principaux programmes sont contenus dans les fichiers : *egal et popegal.*

Le but de cette application est de présenter les différentes étapes de l'égalisation d'histogramme ainsi que le mode d'affichage pour les différents résultats. Le problème consiste à choisir une image (figure 4.16), à afficher son histogramme (figure 4.17), puis on essaie d'égaliser son histogramme dans la gamme de 256 niveaux de gris si l'image est une image d'intensité ou dans la gamme de 256 couleur si elle est une image couleur ou indexée. On a déjà vu dans la première partie de cette application la présentation des syntaxes des fonctions utilisées.

Résultat et interprétation : Sur la fenêtre de la figure 4.17 sont affichés : l'image originale, l'image égalisée ainsi que les deux histogramme correspondantes. L'histogramme égalisé présente

moins de variation de niveaux et beaucoup plus de niveaux de gris sont présent. Ceci se traduit sur l'image égalisée par un contraste beaucoup plus marqué par rapport à l'image originale, donnant une impression de meilleure clarté et permettant de distinguer plus de détails.

IV.4.1.2Débruitage par morphologie mathématique

Le deuxième programme est contenu dans les fichiers *morpho et popmorpho*

Un exemple de débruitage par morphologie mathématique s'illustre par la figure 4.18.

On peut interpréter la morphologie mathématique comme suit : l'ouverture permet de détruire certains pixels blancs isolés et la fermeture des pixels noirs isolés. En conséquence, un ouverture suivie 'une fermeture permet de détruire des pixels isolés. De plus, si l'on reprend la définition des opérateurs de base (érosion et dilatation), on peut en tirer que ces opérateurs ne se limitent pas aux images binaires. Cependant, les fonctions de filtrage morphologique proposés par Matlab se limitent à ce type d'image .Si nous voulons appliquer ces opérateurs sur des images d'intensité,

il est donc nécessaire de réaliser nos propres fonctions. Le travail à réaliser est cependant très réduit, grâce à une puissante fonction Matlab : nlfilter. Cette fonction NLFILTER découpe l'image en blocs élémentaires et applique une fonction à chaque bloc.

syntaxe :

E=nlfilter(J,'taille,'eroder')

```
%Lecture de l'image à traiter
I=double(I)/255.0;
%Addition de bruit
J=imnoise(I, 'salt&pepper');
%ouverture
A= nlfilter(J, taille, 'eroder');
B=nlfilter(A, taille, 'dilater');
%fermeture
C=nlfilter(B, taille, 'dilater');
D=nlfilter(C, taille, 'eroder');
%affichage des différents résultats :
subplot(2,2,1),imshow(I),  title('image originale')
subplot(2,2,2),imshow(J),  title('image bruitée')
subplot(2,2,3),imshow(B),  title('image après ouverture')
subplot(2,2,4),imshow(D), title('image obtenue après fermeture')
```

Résultat et interprétation : l'image bruitée est dans nôtre cas une image originale affectée d'un bruit de type sel&poivre (constitué de points noirs et blancs éparpillés sur l'image). On constate à la fin de l'opération sur l'image obtenue qui se voit entièrement débarrassée du bruit qui l'a affectée.

Remarque : Cette procédure a donné les résultats escomptés, il présente néanmoins comme défaut une lenteur d'exécution due à la présence de la fonction *nlfilter.*

IV.4.1.3Filtrage

Les programmes se trouve dans les fichiers *filtrage, filmed, median filwiener et wiener.*

Cette application consiste à débruiter une image avec deux types de filtres : le filtre médian et le filtre de Wiener, dans un premier temps, on choisit une image, puis on lui introduit un des trois types de bruits les plus familiers (gaussien,'poivre et sel et bruit multiplicatifs), ensuite nous introduisons l'image bruitée dans l'opération de filtrage. Un exemple de filtrage médian d'une image bruitée par un bruit impulsionnel est illustré par la figure 4.19 et un autre exemple par le filtrage de Wiener dans la figure 4.20.

Filtrage d'une image par le filtre médian

```
%Lecture de l'image (toujours à choisir)
I=imread(…) ; colormap(jet(64)); I=double(I)/255.0;
%Création d'une image bruitée
Ibruitée=imnoise(I,'bruit')
%Affichage de l'image bruitée dont nous tenons compte comme une image originale
subplot(1,2,1),imshow(Ibruitee),title('image originale')
%Filtrage
F = medfilt2 (Ibruitee);
%Affichage de l'image filtrée
subplot (1,2,2), imshow (F), title ('image filtrée')
```

Filtrage d'image par le filtre de Wiener (même principe que précédemment)

```
%Lecture de l'image (toujours à choisir)
I = imread (…) ; colormap(jet(64)); I=double(I)/255.0;
%Création d'une image bruitée
Ibruitée = imnoise (I,'bruit')
%Affichage de l'image bruitée dont nous tenons compte comme une image originale
```

```
subplot (1,2,1), imshow (Ibruitee), title ('image originale')
%Filtrage
F = wiener2 (Ibruitee);
%Affichage de l'image filtrée
subplot (1,2,2), imshow (F), title ('image filtrée')
```

Résultat et interprétation : Nous pouvons constater que l'image est débarrassée du bruit après filtrage median, par contre le filtre de Wiener n'est adapté à ce type de bruit. En fait, il est normal que deux filtres différents ne présentent pas la même performance sur un même type de bruit vu que les filtres sont généralement conçus pour une utilisation spécifique.

Grâce à cette partie de la simulation, on pourra reprendre chaque filtre en prenant autre type de bruits. Ces différents bruits peuvent s'illustrer mathématiquement dans l'ANNEXE IV.

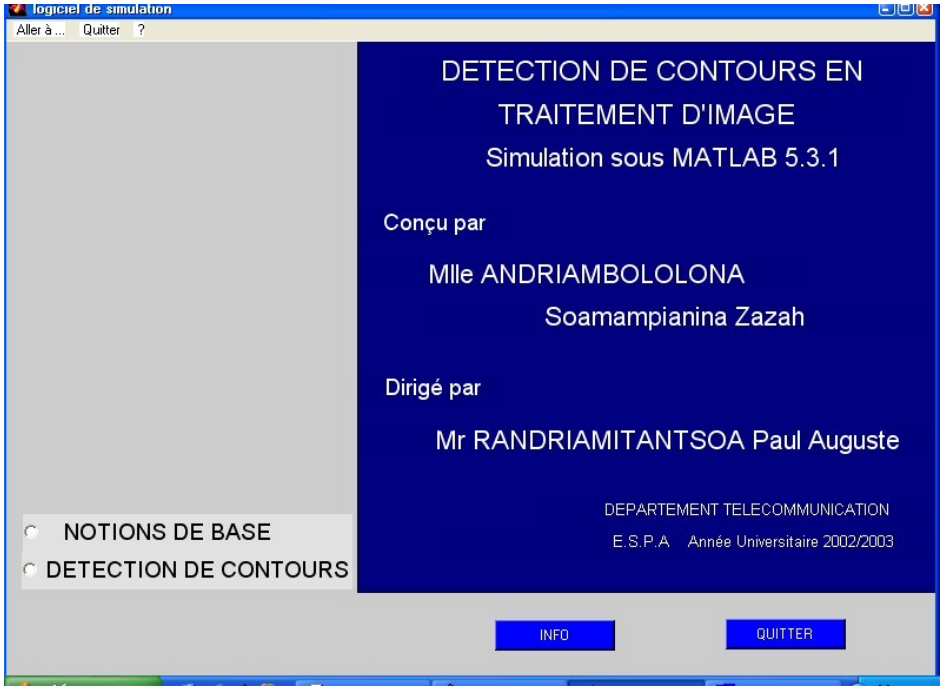


Figure 4.14 : fenêtre d'accueil

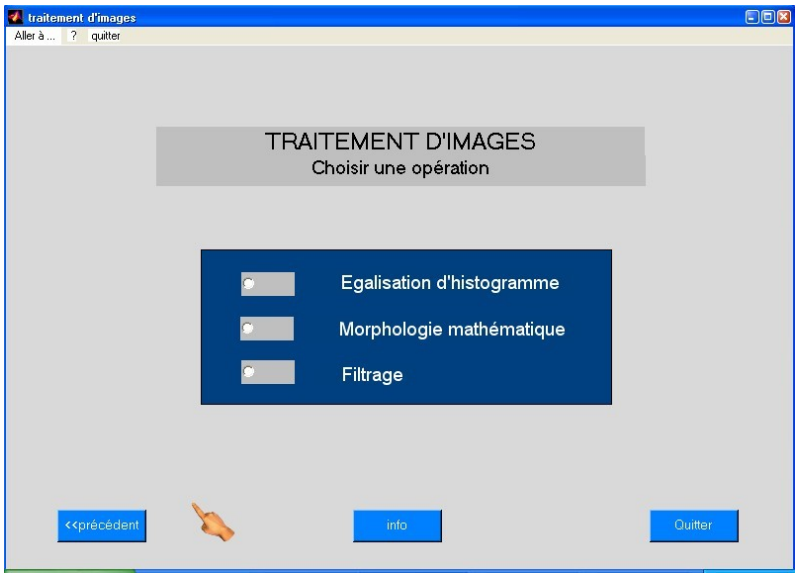


Figure 4.15 : fenêtre principale de notions de base

Figure 4.16 : choix de l'image

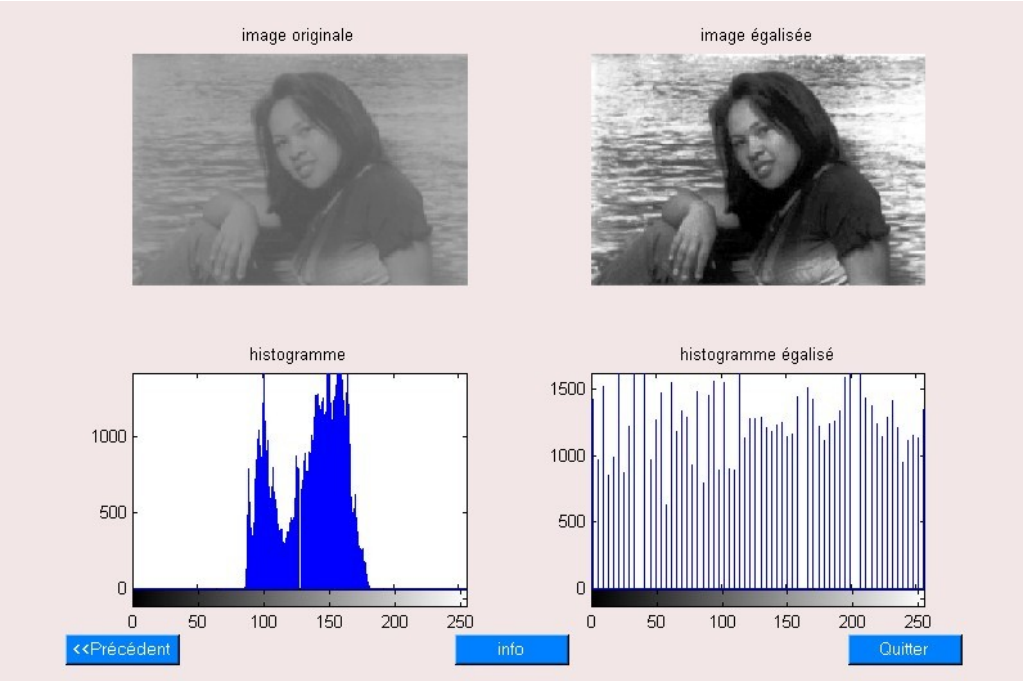


Figure 4.17 : Egalisation d'histogramme

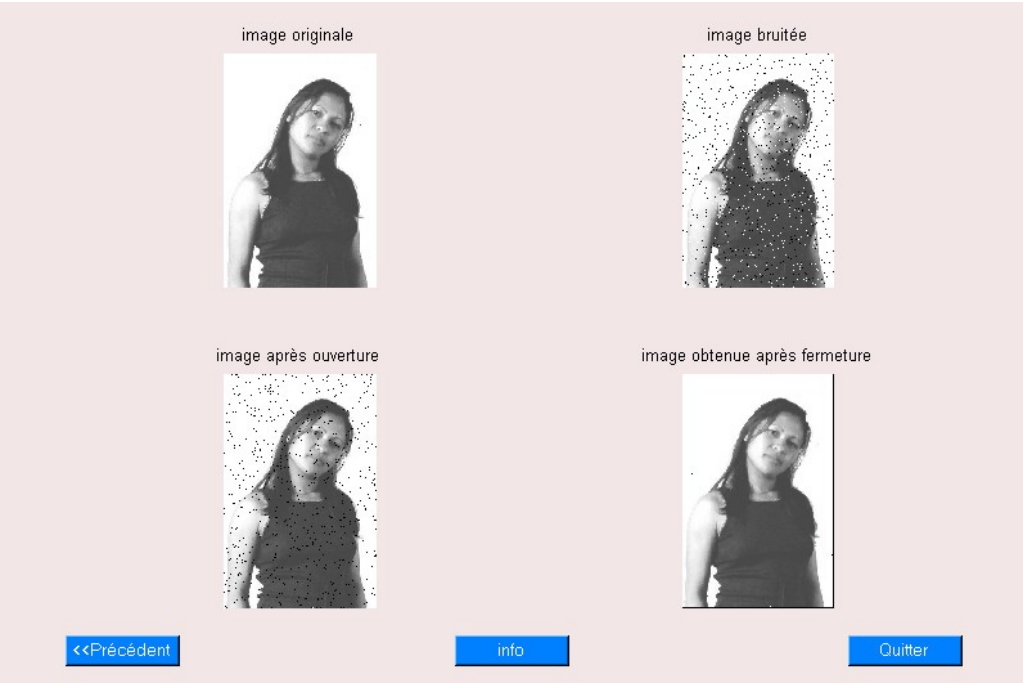


Figure 4.18 : Morphologie mathématique

Figure 4.19 : Filtrage médian

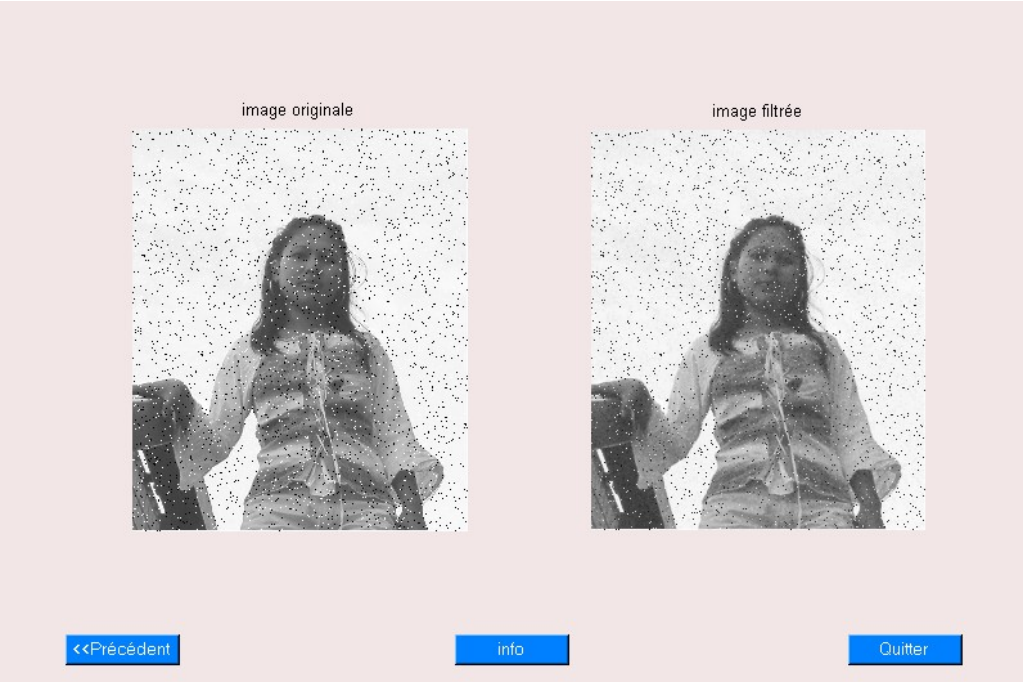


Figure 4.20 : Filtrage de Wiener

Conclusion : le choix que nous avons fait ne sont que des possibilités parmi tant d'autres. Ils démontrent cependant l'efficacité des fonctions que nous avons utilisées et donnent certaine idée de la réalité. Les résultats que nous avons obtenus pour l'égalisation d'histogramme, morphologies mathématiques et le filtrage apportent une nette amélioration de l'aspect visuel de l'image considérée. Un tel résultat permettrait dans la réalité de procéder de manière plus efficiente à d'autres opérations.

IV.4.2Détection de contours

Dans cette partie, on va essayer de simuler quelques exemple détection de contours afin d'en tirer une conclusion. Les contours seront représentés par les noirs, en fait, ils devront s'afficher en blanc mais pour la compréhension, on affichera leurs négatifs, c'est à dire on affiche les contours sur des fond blancs. En revenant dans la fenêtre d'accueil, en cliquant sur le bouton « détection de contour », nous y sommes.

IV.4.2.1Présentation des fonctions de la boîte à outils pour la détection de contours

La fonction edge nous permet de réaliser ce genre de traitement,elle fournit un nombre d'estimation de dérivée. On fait aussi appel surtout à la fonction fspecial, les différents filtres pour la détection de contour sont déjà prédéfinis sous Matlab.

EDGE détecte les contours dans une image d'intensité.

Syntaxes :

- Méthode Sobel :

```
BW = EDGE(I, 'sobel')
BW = EDGE(I, 'sobel',THRESH)
BW = EDGE(I, 'sobel',THRESH,DIRECTION)
[BW,thresh] = EDGE(I, 'sobel' )
```

- Méthode Prewitt

```
BW = EDGE(I, 'prewitt').
BW = EDGE(I, 'prewitt',THRESH)
BW = EDGE(I, 'prewitt',THRESH,DIRECTION)
[BW,thresh] = EDGE(I, 'prewitt'
```

- Methode de Roberts

```
BW = EDGE(I, 'roberts')
BW = EDGE(I, 'roberts',THRESH)
```

[BW,thresh] = EDGE(I, 'roberts',...)

- Laplacian of Gaussian

BW = EDGE(I, 'log')

BW = EDGE(I, 'log',THRESH)

BW = EDGE(I, 'log',THRESH,SIGMA)

[BW,thresh] = EDGE(I, 'log',...)

- Méthode de Canny

BW = EDGE(I, 'canny')

BW = EDGE(I, 'canny',THRESH)

BW = EDGE(I, 'canny',THRESH,SIGMA)

[BW,thresh] = EDGE(I, 'canny',...)

Pour fspecial, elle ne dispose pas tous ces types de filtres par exemple, elle ne pourra pas créer le filtre de Canny-Derich ;

FSPECIAL crée des filtres prédéfinis :

Syntaxes :

- Méthode Sobel :

H=fspecial (‘sobel’)

- Méthode Prewitt

H=fspecial (‘prewitt’)

- Laplacian of Gaussian

H=fspecial (‘log’)

En effet, edge est facile à manipuler puisqu’elle donne tout de suite les résultats mais pour mieux comprendre l’acheminement du programme, utilisons alors fspecial sauf pour le cas de Roberts et de Canny.

IV.4.2.2Applications

Après avoir cliqué sur le bouton poussoir « détection de contour », on nous offre la fenêtre de la figure (4.21) permettant le choix des différentes méthodes pour cette opération.

IV.4.2.2.1 Application 1 : utilisation des filtres par différences finies

Le programme se trouve dans : sarysobel, saryprewitt, saryroberts, popsobel, popprewitt, poproberts, prew, sob, rob.

Il consiste à mettre en œuvre des trois jeux de filtres les plus familiers en détection de contour tels que celui de Sobel, de Prewitt et de Roberts. Les syntaxes permettant le fonctionnement des deux premiers filtres s'exécutent de la même manière. Dans le cas de filtre de Roberts, ce filtre n'est pas difficile à concevoir, on va le réaliser. En fait, la détection de contours se fera par un simple seuillage et avant d'aboutir à la valeur finale, observons comment se fait une détection de contour dans une direction puis dans l'autre. Le programme se fonctionne généralement comme suit :

```
%Lecture de l'image à traiter
I=double(I)/255.0;
%Filtre de Sobel
h=fspecial('sobel');
%Filtre de Prewitt
h=fspecial('prewitt');
%Déclaration des filtres vertical et horizontal
v=-h';
Gh=filter2(h, I);%Horizontal
Gv=filter2(v, I);%Vertical
%Calcule de la norme du gradient
G=sqrt( Gh.*Gh+Gv.*Gv);
%Détermination d'un seuil
Gh=(Gh>seuil);Gh=1.0-Gh; %  ou Gh=Gh<seuil %négatif
Gv=(Gv>seuil);Gv=1.0-Gv; %  ou Gv=Gv<seuil %négatif
Gs=(G>seuil);Gs=1.0-Gs ;  %    ou Gs=Gs<seuil %négatif
%affichage
subplot(221),imshow(I),title('image originale')
subplot(222),imshow(Gh),title('detection de contour horizontalement')
subplot(223),imshow(Gv),title('detection de contour verticalement')
subplot(224),imshow(Gs),title('detection de contour dans les deux directions ')

%Filtre de Roberts
a=[ 1 0;0 -1];
b=rot90(a,-1);
```

```
%filtrage
Ga=filter2(a, I);Gb=filter2(b, I);
%Calcul de la norme du gradient
G=sqrt( Ga.*Ga+Gb.*Gb);
Ga=(Ga>seuil);Ga=1.0-Ga;
Gb=(Gb>seuil);Gb=1.0-Gb;
Gs=(G>seuil);Gs=1.0-Gs ;
%affichage
subplot(221),imshow(I),title('image originale')
subplot(222),imshow(Ga),title('detection dans l''une des diagonales')
subplot(223),imshow(Gb),title('detection dans l''autre')
subplot(224),imshow(Gs),title('detection de contours dans les deux directions')
```

Après avoir choisi une méthode, par exemple, celle de Sobel , une autre petite fenêtre pour le choix du seuil (figure (4.22)) s'affiche.

On tape un seuil, choisit une image, puis exécuter, on a la figure (4.23) pour la méthode de Sobel avec un seuil de 0.12

Interprétation : on a choisit un seuil très inférieur à 1, on détecte plusieurs variations, par contre pour un seuil très proche de 1, les contours sont mal découverts. Ces trois filtres donnent des résultats très proches les uns des autres.

IV.4.2.2.2 Application2 : filtrage optimal par le filtre de Canny

Le programme se trouve dans les fichiers sarycanny, popcanny, canny1

En revenant sur la partie théorique, la méthode de Canny-Dérich dépend du paramètre alpha, est très compliqué puisqu'elle est basée sur le seuillage par hystérésis . Cette simulation nous permettra de choisir alpha. Il n'y a que la fonction edge qui fasse cette opération. L'algorithme est le suivant :

```
%D'abord, lecture de l'image
I=double(I)/255.0;
%Filtrage
C1=edge(I,'canny',[sb sh],alpha1);
C2=edge(I,'canny',[sb sh],alpha2);
C3=edge(I,'canny',[sb sh],alpha3);
```

```
C1=not(C1);% négatif
C2=not(C2); );% négatif
C3=not(C3); );% négatif
%affichage
subplot(221),imshow(I), title('image originale')
subplot(222),imshow(C1),title(pour alpha =alpha1)
subplot(223),imshow(C2), title(pour alpha =alpha2)
subplot(224),imshow(C3), title(pour alpha =alpha3)
```

En tapant sur Canny, on nous offre la fenêtre (4.24) pour le choix de l'image , des deux seuils et trois différentes valeurs de alpha.

Un exemple des résultats se trouve sur la figure (4.25) pour :

- sh= 0.2 et sb=0.05
- alpha1=0.2 ; alpha2=1 ; alpha3=2

Interprétation : Le choix des deus seuils est très aléatoire. Si on choisit un seuil bas supérieur au seuil haut, on n'obtient pas des résultats. On remarque que le meilleur résultat correspond à la valeur de alpha égale à 0.2. Même raisonnement que le filtre gaussien qu'on verra dans la section suivante.

IV.4.2.2.3Application3 : laplacian of gaussian

Le programme se trouve dans sarylog, poplog, log1. Cette application est basée sur le choix de sigma, on pourra même faire une comparaison sur trois différentes valeurs de sigma.

Globalement, le programme s'écrit :

```
%Lecture de limage
I=double(I)/255.0;
%Déclaration du filtre
Ha=fspecial('log',5,sigma1);
Hb=fspecial('log',5,sigma2);
Hc=fspecial('log',5,sigma3);
%Filtrage
La=filter2(Ha,I); La=1.0-La ;%Négatif
Lb=filter2(Hb,I); La=1.0-La ;%Négatif
Lc=filter2(Hc,I); La=1.0-La ;%Négatif
```

```
%Affichage
subplot(221),imshow(I),title('image originale')
subplot(222),imshow(La),title('tag','string',sigma1)
subplot(223),imshow(Lb),title('tag','string',sigma2)
subplot(224),imshow(Lc),title('tag','string',sigma3)
```

L'exemple de la figure (4.27) permet la visualisation des différents résultats pour sigma=0.5, sigma=1, sigma=2 et la fenêtre 4.26 permet le choix de *sigma.*

Interprétation : On remarque même que plus sigma est grand, plus on « perd » les contours, en se référant à la partie théorique, plus sigma est grand, plus la cloche augmente , plus le flou appliqué à l'image devient très important et on perd les contours. Log détecte les contours rigoureusement, pourtant, malgré le raisonnement qu'on vient de découvrir, la valeur de sigma est très aléatoire puisque si on choisit un sigma très inférieur, on confond les vrais contours avec les faux, alors que si on prend un sigma supérieur, on les détecte mal, il faut choisir alors un sigma de valeur moyenne. Malgré tout, cette méthode s'avère la plus meilleure.

Présentons maintenant les différentes fenêtres qui s'afficheront lors de la simulation. En revenant sur la fenêtre d'accueil, puis, en cliquant sur « DETECTION DE CONTOURS », on obtient la fenêtre principale de la détection de contours qui nous permettra de choisir les différentes méthodes.

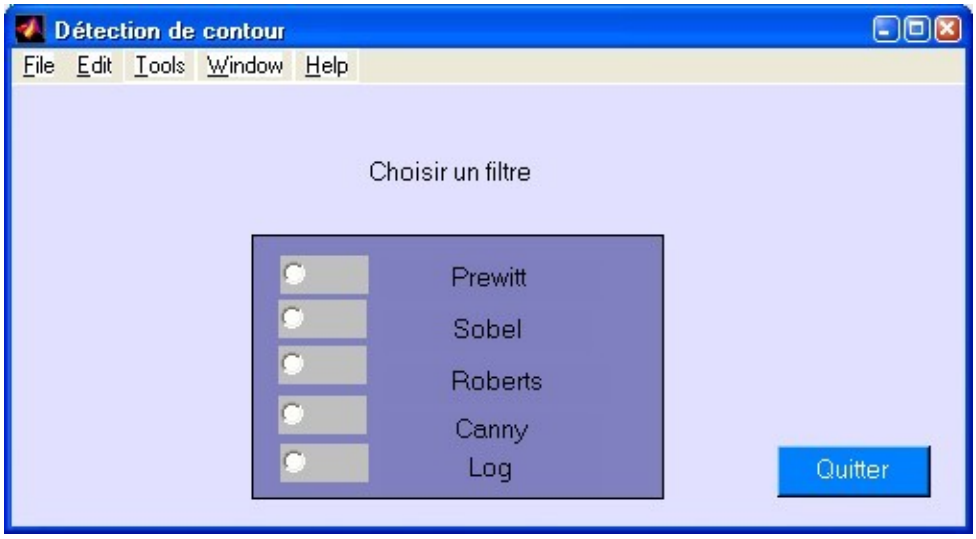


Figure 4.21 : fenêtre principale de la détection de contours

Figure 4.22 : fenêtre pour le choix de seuil de la méthode de Sobel

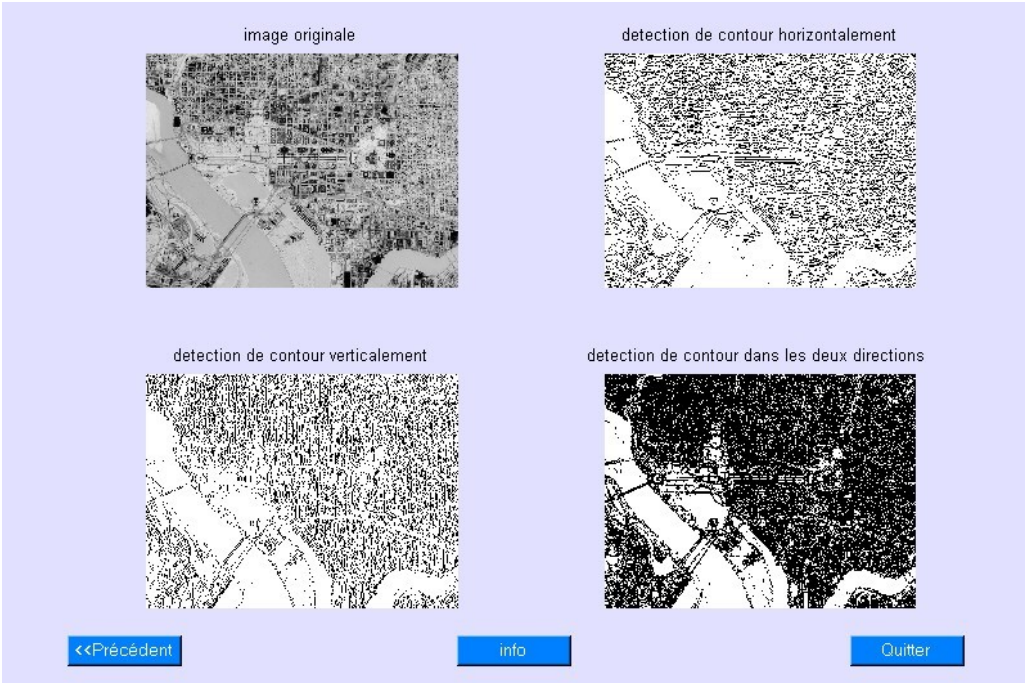


Figure 4.23 : détection de contour par le filtre de Sobel pour un seuil de 0.3

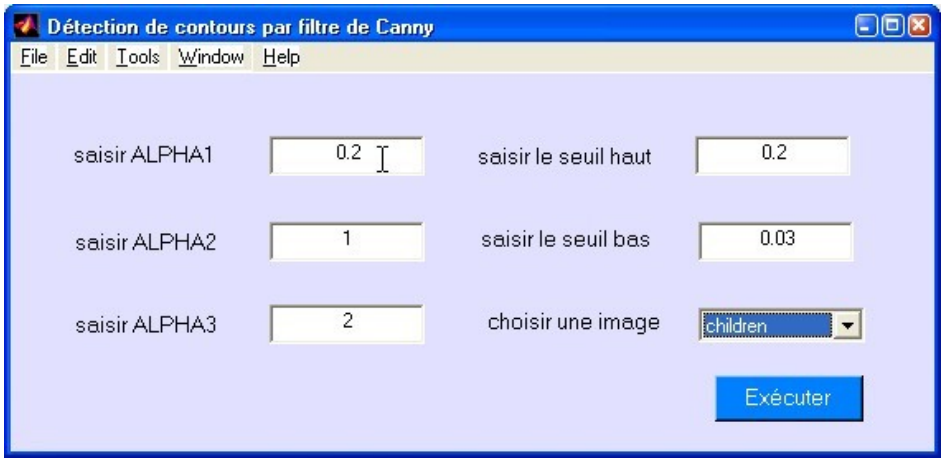


Figure 4.24 : fenêtre de choix des valeurs pour la méthode de Canny-Deriche

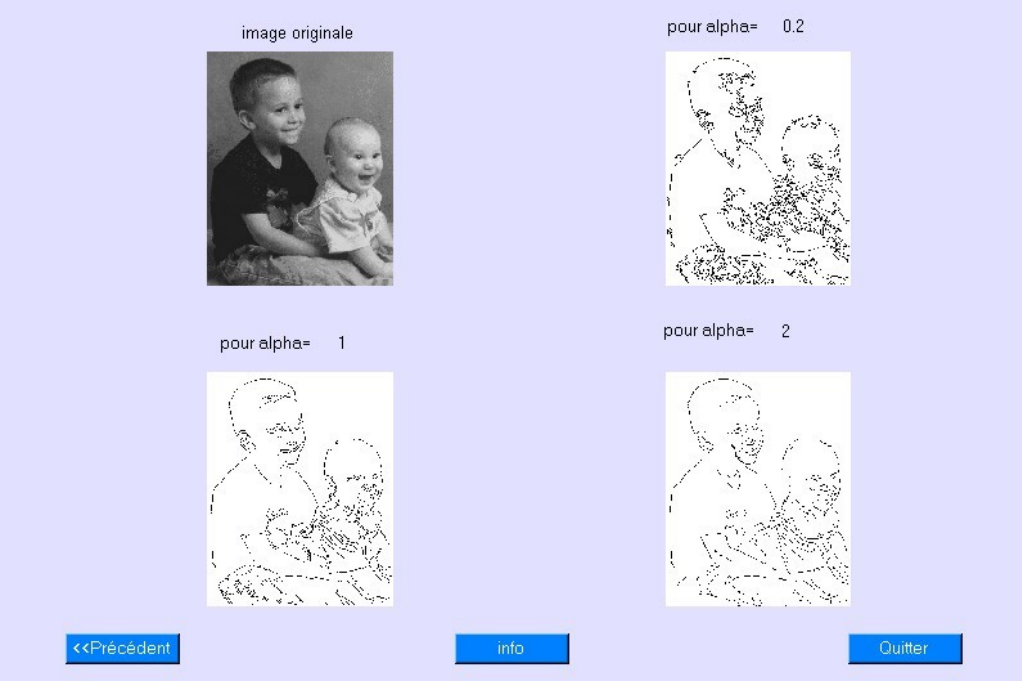


Figure 4.25 : détection de contours par le filtre de Canny-Deriche

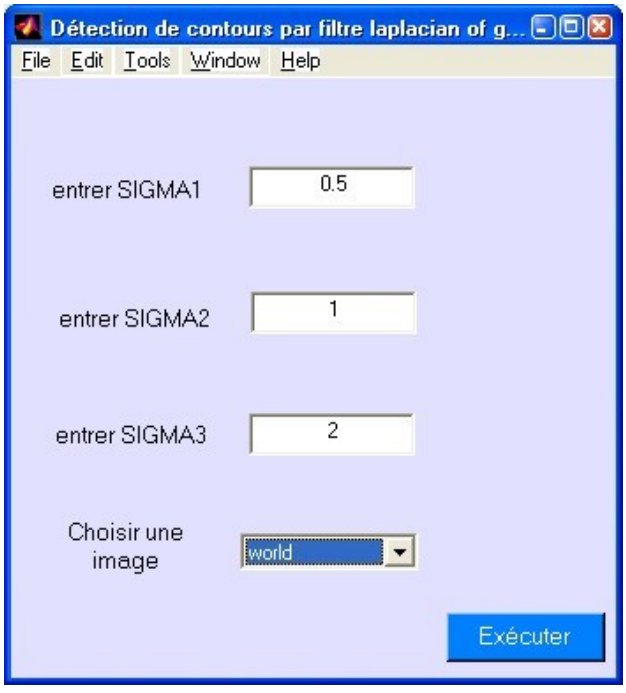


Figure 4.26 : fenêtre pour le choix des valeurs de sigma (Log)

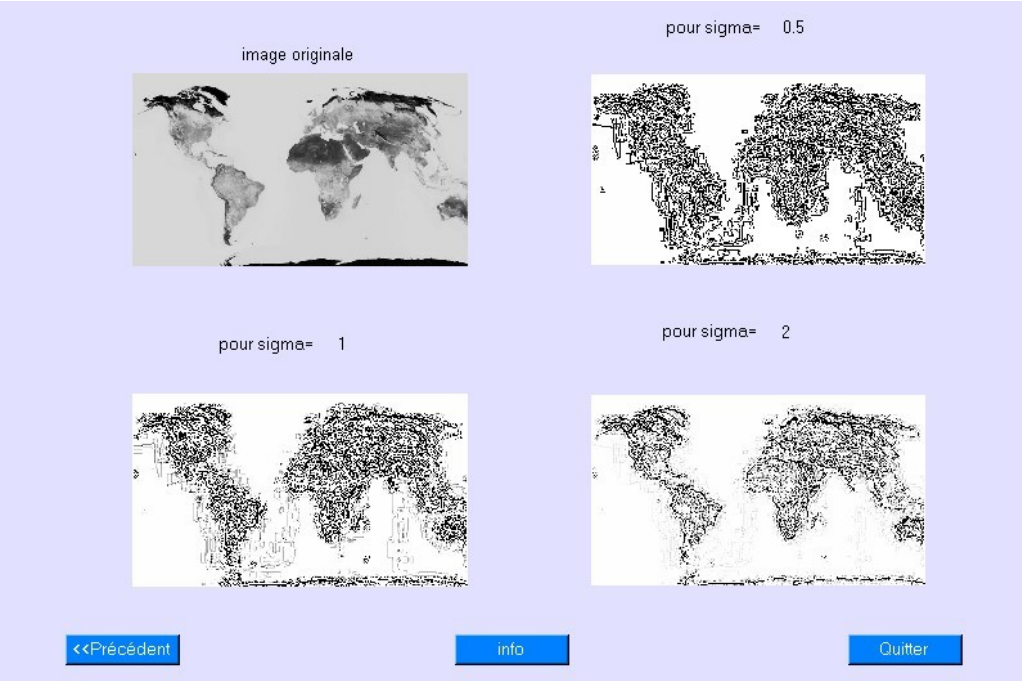


Figure 4.27 : détection de contour par Log

Bref, ainsi se termine notre simulation en rappelant toujours que ce ne sont que des exemples, la detection de contours est un domaine très large et les variables utilisées sont plus ou moins aléatoires.

CONCLUSION

Bref, le traitement d'image désigne un ensemble de méthodes dont l'objectif est soit de transformer des images, soit d'en extraire de l'information. Il s'agit d'un domaine très vaste, qui trouve de nombreuses applications, par exemple dans le domaine industriel, le contrôle automatique par la vision est de plus en plus répandue dans les chaînes de fabrication ; quant au domaine militaire, des dispositifs très performants, capables de détecter et de reconnaître automatiquement leurs cibles voient le jour .Ce ne sont que des exemples.

Par sa diversité même, le traitement d'image est parfois déroutant pour l'étudiant ou l'ingénieur débutant. Les informations que l'on souhaite extraire d'une image, les applications, les types de données, sont si variées qu'il est difficile, lorsque l'on doit traiter une nouvelle application, de savoir à quel modèle recourir et quelle méthodologie suivre.

Pour notre part, ce mémoire nous a permis de mieux cerner le domaine des images numériques. Dans sa première partie nous avons vu les bases théoriques sur le traitement d'image, d'un point de vue plutôt généraliste. Des opérations essentielles destinées à l'amélioration d'image ont été abordées, avec certaines définitions et notions qui ont été nécessaires pour une meilleure compréhension du domaine dans lequel s'inscrit cette étude.

Dans la dernière partie de ce travail, plusieurs processus de traitement d'images ont été simulés sous MATLAB ainsi que les diverses méthodes commodes que rigoureuses de la détection de contours qui nous paraît facile à exécuter après une présentation préalable du logiciel utilisé. D'autres études sont toutefois nécessaires pour une meilleure maîtrise de ce sujet notamment en ce qui concerne les spécificités des nombreux domaines d'applications d'autant plus spécialisés que les évolutions actuelles ouvrent la voie à des possibilités toujours plus larges. La panoplie des techniques de traitement d'images est étendue et variée, les nombreuses recherches passées ou en cours ont justement pour but de trouver de nouvelles méthodes plus performantes ou tout au moins d'apporter des améliorations et d'adapter constamment les méthodes existantes aux évolutions technologiques et face aux exigences de plus en plus nombreuses relatives à l'utilisation des images en télécommunication mais notre étude s'est limitée surtout sur les image d'intensités. Certaines opérations peuvent s'étendre aux images couleurs dont l'une des approches qui est assez traditionnelle consiste à traiter séparément chacune des composantes.

ANNEXE I:FENETRE D'ACCUEIL

```
h0 = figure('BackingStore','off', ...
    'Color',[0.752941176470588 0.752941176470588 0.752941176470588], ...
    'Colormap',mat0, ...
    'FileName','C:\MATLABR11\accueil.m', ...
    'MenuBar','none', ...
    'Name','Logiciel De Simulation', ...
    'NumberTitle','off', ...
    'PaperPosition',[18 180 576 432], ...
    'PaperUnits','points', ...
    'Position',[4 10 795 521], ...
    'Tag','Fig1', ...
    'ToolBar','figure');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'ListboxTop',0, ...
    'Position',[102.75 39 661.5 380.25], ...
    'String','TRAITEMENT D"IMAGE', ...
    'Style','frame', ...
    'Tag','Frame2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'FontSize',17, ...
    'ForegroundColor',[0.250980392156863 0 0.250980392156863], ...
    'ListboxTop',0, ...
    'Position',[189.75 339 330 27.75], ...
    'String','DETECTION DE CONTOUR', ...
    'Style','text', ...
    'Tag','StaticText1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'FontSize',14, ...
    'ForegroundColor',[0.250980392156863 0 0.250980392156863], ...
    'ListboxTop',0, ...
    'Position',[149.25 303 102 18], ...
    'String','CONCU PAR:', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'FontSize',18, ...
    'ForegroundColor',[0.250980392156863 0 0.250980392156863], ...
    'ListboxTop',0, ...
    'Position',[192 277.5 271.5 18], ...
```

```
    'String',' Mlle ANDRIAMBOLOLONA', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'FontSize',18, ...
    'ForegroundColor',[0.250980392156863 0 0.250980392156863], ...
    'ListboxTop',0, ...
    'Position',[279 249 291 26.25], ...
    'String','Soamampianina Zazah', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'FontSize',14, ...
    'ForegroundColor',[0.250980392156863 0 0.250980392156863], ...
    'ListboxTop',0, ...
    'Position',[146.25 227.25 104.25 18], ...
    'String','DIRIGE PAR:', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'FontSize',18, ...
    'ForegroundColor',[0.250980392156863 0 0.250980392156863], ...
    'ListboxTop',0, ...
    'Position',[191.25 168 391.5 24.75], ...
    'String','Mr RANDRIAMITANTSOA Paul AUGUSTE', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'FontSize',10, ...
    'ForegroundColor',[0.250980392156863 0 0.250980392156863], ...
    'ListboxTop',0, ...
    'Position',[279 96 391.5 18], ...
    'String','DEPARTEMENT : TELECOMMUNICATIONS', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'FontSize',10, ...
    'ForegroundColor',[0.250980392156863 0 0.250980392156863], ...
```

```
    'ListboxTop',0, ...
    'Position',[213.75 321 292.5 18], ...
    'String','Simulation sous Matlab 5.3.1', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'FontSize',10, ...
    'ForegroundColor',[0.250980392156863 0 0.250980392156863], ...
    'ListboxTop',0, ...
    'Position',[309.75 77.25 330.75 14.25], ...
    'String','E.S.P.A  2002/2003', ...
    'Style','text', ...
    'Tag','StaticText2');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'Callback','aidaccueil', ...
    'FontSize',12, ...
    'ForegroundColor',[0.250980392156863 0 0.250980392156863], ...
    'ListboxTop',0, ...
    'Position',[17.25 148.5 73.5 24], ...
    'String','INFO', ...
    'Tag','Pushbutton1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'Callback','close all', ...
    'FontSize',12, ...
    'ForegroundColor',[0.250980392156863 0 0.250980392156863], ...
    'ListboxTop',0, ...
    'Position',[16.5 198.75 73.5 23.25], ...
    'String','QUITTER', ...
    'Tag','Pushbutton1');
h1 = uicontrol('Parent',h0, ...
    'Units','points', ...
    'BackgroundColor',[1 1 1], ...
    'FontSize',17, ...
    'ForegroundColor',[0.250980392156863 0 0.250980392156863], ...
    'ListboxTop',0, ...
    'Position',[195.75 366 307.5 24], ...
    'String','TRAITEMENT D"IMAGE', ...
    'Style','text', ...
    'Tag','StaticText3');
```

ANNEXE II:LA TRANSFORMATION DE FOURIER

1. La transformation de Fourier unidimensionnelle

La TF unidimensionnelle est une application F qui associe à une fonction complexe f(t) d'une variable réelle, une fonction complexe d'une variable réelle, notée F(u), appelée « transformée de Fourier de f(t), et définie par :

$$F(u) = \int_{-\infty}^{+\infty} f(t)\exp(-2\pi\ jut)dt$$

Lorsque cette intégrale est définie, quel que soit le réel u. L'ensemble des fonctions f(t) admettant une transformée de Fourier, étant stable par combinaisons linéaires (facile à prouver), est un espace vectoriel.

2. La transformation de Fourier bidimensionnelle (image)

La transformée de FOURIER d'une image *f(x,y)* est définie par :

$$F(u,v) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)\exp[-2\pi\ j(xu+\ yv)]dxdy$$

L'image originale en prenant la transformée inverse :

$$f(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} F(u,v)\exp[2\pi\ j(xu+\ yv)]dudv$$

Pour que la transformée d'une image existe, la condition suivante suffit

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} |f(x,y)|dxdy < +\infty$$

3. Propriétés

- *Séparabilité*

$$F(u,v) = \int_{-\infty}^{+\infty}\left[\int_{-\infty}^{+\infty} f(x,y)\exp(-2\pi\ jxu)\,dx\right]\exp(-2\pi\ jyv)\,dv$$

- *Linéarité*

$TF\{f_1(x,y)\} = F_1(u,v)$ **et** $TF\{f_2(x,y)\} = F_2(u,v)$**,**

$$c_1 f_1(x,y) + c_2 f_2(x,y) = c_1 F(u,v) + c_2 F(u,v)$$

- *Homothétie*

$$TF\{f(ax+by)\} = \frac{1}{|ab|}F\left(\frac{u}{a}\frac{v}{b}\right)$$

-*Dualité*

$$F(u,v) = F(-u,-v)$$

-*Translation spatiale*

$$TF\{f(x-x_o, y-y_o\} = F(u,v)\exp[-2\pi\ j(x_o u+\ y_o v)]$$

-*Translation fréquentielle*

Si $TF\{f(x,y)\} = F(u,v)$, alors $TF\{f(x,y)\exp[2\pi\ j(u_o x, v_o y]\} = F(u-u_o, v-v_o)$

- *Aires*

$$F(0,0) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)dxdy$$

$$f(0,0) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} F(u,v)dudv$$

- *Convolution*

Le produit de convolution de deux fonctions *f(x,y)* et *g(x,y)* est défini par :

$$(f \otimes g)(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(\alpha,\beta)g(x-\alpha, y-\beta)d\alpha\, d\beta$$

Si $TF\{f(x,y)\} = F(u,v)$ et $TF\{g(x,y)\} = G(u,v)$

- *Multiplication*

$TF\{f(x,y)\} = F(u,v)$ et $TF\{g(x,y)\} = G(u,v)$, alors $TF\{f(x,y)g(x,y\} = (F \otimes G)(u,v)$

La multiplication de deux images dans le domaine spatial est transformée dans le domaine fréquentiel en la convolution de leur transformée de FOURIER respective.

4. La transformation de Fourier Discrète

Tout comme les séries de Fourier fournissent le point de départ pour les transformations et l'analyse de fonctions périodiques, la transformée de FOURIER discrète **(*Discrete FOURIER Transform*** ou DFT) est l'équivalent pour l'analyse de vecteurs. Elle effectue un passage du domaine spatial à un domaine spectral**.**

Un vecteur à *N* composantes est calculé comme une combinaison linéaire de *N* vecteurs de base. Les éléments de ces vecteurs de base sont tous des puissances de l'exponentielle imaginaire

$W = \exp[2\pi\, j/N]$

ces vecteurs (lignes) valent

$(1, \vec{W}^{k}, \vec{W}^{2k}, ..., \vec{W}^{(N-1)k}), k = 1,2,...N-1$

A partir de ces expressions, on peut définir la matrice de transformation

$\Phi_{JJ}(k,l) = \frac{1}{J}\exp\left(-j\frac{2\pi}{J}kl\right); k,l = 0,1,...J-1$ **de dimension *J*J***

Soit *f(x)* converti en N échantillons,La TFD est définie par :

$$F(u) = \frac{1}{M}\exp\sum_{x=0}^{M-1} f(x)\exp\left[-2\pi\, j\left(\frac{xu}{M}\right)\right]$$

En 2-D :

$$F(u,v) = \frac{1}{M}\frac{1}{N}\exp\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp\left[-2\pi\, j\left(\frac{xu}{M}+\frac{yv}{N}\right)\right]$$

Et la TFDU s'écrit alors :

$$F(u,v) = \frac{1}{\sqrt{M}}\frac{1}{\sqrt{N}}\exp\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp\left[-2\pi\, j\left(\frac{xu}{M}+\frac{yv}{N}\right)\right]$$

ANNEXE III:TRANSFORMEE EN Z

La transformée en Z est une adaptation de la transformée de Laplace pour l'étude des réponses transitoires des systèmes numériques. La transformée en Z est relative aux suites numériques.La transformée en Z peut s'obtenir de la transformée de Laplace en effectuant de raisonnement suivant :

Soit $F^*(p)$ la transformée de Laplace du signal échantillonné $f^*(t)$ tel que :

$f^*(t) = f_0\delta(t) + f_1(t - T_e) + ...$ avec T_e période d'échantillonnage.

La transformée de Laplace s'écrit :

$$F^*(p) = f_0 + f_1 e^{-pT_e} + f_2 e^{-2pT_e} + ... = \sum_{k=0}^{+\infty} f_k e^{-kpT_e}$$

La transformée de Laplace d'un signal échantillonné s'exprime alors comme une somme de terme en e^{pT_e}. Pour exprimer la transformée en Z, on effectuera simplement le changement de variable $z=e^{pT_e}$, où p est la variable de Laplace, cette variable état en général, considérée comme complexe. Par conséquent :

$$Z(f(t)) = F(z) = \sum_{k=0}^{+\infty} f_k z^{-k}$$

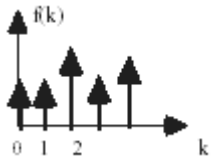
1. Propriétés

- *Linéarités*

Z(af+bg)=aZ(f)+bZ(g)

- *Théorème du retard temporel*

Soient deux signaux discrets f(k) et g(k) tels que :
f(k)=0
g(k)=f(k-n),g(k)=0 pour k<n avec n positif



$G(z) = \sum_{-\infty}^{+\infty} g(i)z^{-i} = \sum_{-\infty}^{+\infty} f(i-n)z^{-i}$ puisque *g(i)=0* sur les premiers échantillons

En posant j=i, il vient :

$$G(z) = \sum_{-\infty}^{+\infty} f(j)z^{-j-n} = z^{-n}\sum_{-\infty}^{+\infty} f(j)z^{-j} = z^{-n}F(z)$$

$$Z(f_{k-n}) = z^{-n}F(z)$$

Retarder un signal causal de n échantillons revient à multiplier sa transformée en Z par z^{-n}

2. Produit de convolution

$$Z(f \otimes g) = Z(f)Z(g)$$

3. Multiplication par e^{-at}

$$Z\left(e^{-akT}.f(k)\right) = F\left(ze^{at}\right)$$

4.Multiplication par α^{t}

$$Z\left(\alpha^{t} f(t)\right) = F\left(\frac{z}{\alpha^{T_e}}\right)$$

5. Multiplication par t^n

$$Z\left(t^n f(t)\right) = -T_e z \frac{d}{dz} Z\left(t^{n-1} f(t)\right)$$

6. Transformée en Z des signaux test *:*

-Dirac

$$Z\left(\delta(T)\right) = 1$$

-Echelon

$$Z\left(\Gamma(t)\right) = \frac{1}{1-z^{-1}} = \frac{z}{z-1}$$

7. Théorème de la valeur finale et de la valeur initiale

-Théorème de la valeur initiale

$$f(0) = \lim_{z \to +\infty} F(z)$$

-Théorème de la valeur finale :

$$\lim_{k \to +\infty} = \lim_{z \to 1}\left((z-1)F(z)\right)$$

ANNEXE IV:BRUITS D'IMAGE

a. Caractéristiques des bruits

a.1. La moyenne

C'est une grandeur de position. Elle est obtenue en calculant la somme des valeurs des pixels de l'image divisée par l'effectif total

a.2. Ecart-type et la variance

Ces deux grandeurs caractérisent la dynamique de la distribution, puisqu'elles expriment le regroupement (ou la dispersion) autour de la valeur moyenne. La variance se calcule en effectuant la somme des carrés de la différence entre chaque valeur et la moyenne, divisée par l'effectif total. L'écart-type est la racine carrée de l'écart-type.

b. Les différents types de bruit

b.1. Modèles de bruit d'image

Le bruit d'image est en toute rigueur considéré comme un champ aléatoire caractérisé par sa densité de probabilité *f* et sa fonction de répartition F.
En général :

$$f(a) = C.\exp(-K|a|^n)$$

avec C et K des constantes de normalisation liées à la variance
Cette modélisation permet de retrouver la nature plu ou moins impulsionnelle du bruit, c'est-à-dire sa tendance à s'écarter de l'espérance mathématique de la région considérée.
Pour n=1, on trouve le bruit exponentiel ou *« impulsionnel »*
Pour n=2, on trouve le bruit *« gaussien »*.

-Le bruit impulsionnel

Dans la pratique, ce type de bruit apparaît durant la transmission d'une image dans un canal de communication. Le bruit impulsionnel est aussi connu sous le nom de bruit de type *« poivre ou sel »* (de l'anglais *« salt (sel) and pepper (poivre) »*).
La densité de probabilité est :

$$f(a) = C.\exp(-K|a|^1)$$

Dans le cas d'une densité de bruit plus élevé, la gêne visuelle devient beaucoup plu important Autrement dit, le bruit « salt and pepper » correspond à la modification de pixel en niveau de gris au départ qui sont une addition de fait transformés en pixels noirs ou pixels blancs (représentant les valeurs extrêmes 0 et 255)

-Le bruit gaussien

Le bruit gaussien apporte un grain à l'image qui pour une variance de 500 est très peu dérangeant pour l'œil.
Dans le cas d'une variance plus élevée, les bruit gaussien devient beaucoup plus gênant pour l'œil.
Sa densité de probabilité s'écrit :

$$f(a) = C.\exp(-K|a|^2)$$

-Le bruit multiplicatif

Le bruit multiplicatif peut être simplement défini de la façon suivante :
Etant donné R une image non bruitée et I la même image avec un bruit multiplicatif B, alors chaque pixel j est caractérisée par la relation :

$$I_j = B_j.R_j$$

BIBLIOGRAPHIE

[1] M.A RAKOTOMALALA, *Traitement d'image,* cours 4ème année, Dep Tél-E.S.P.A, A.U :2001-2002

[2] G.BUREL : *Introduction au traitement d'image, simulation sous Matlab,* Hermès sciences, Lavoisier 2001

[3] http://www.ulg.ac/be/telecom/teaching/notes/totali

[4] http://www.ccr.jussieu.fr/urfist/image_numérique

[5] http://www.tsi.enst/fr/tsi/enseignement/teaching/notes/totali

[6] http://www.commentcamarche.fr

[7] http://etudiant.univ-mlv.fr/~fruitet/trait_image

[8] http://www.ems.fr.pdf

[9] J.P. THIRAN : *Traitement numérique des images médicales,* cours FADES Université d'Antananarivo, Octobre 2003

[10] http://www.esil.univ-mrs.fr.MEM.pdf

[11] http://www.rocq.inria.fr/~fauquet/java/sujet

[12] http://telesun.insa.lyon.fr/~telesun/insa

[13] http://www.univ.reims.fr

[14] http://www.cvc.uab.es.pdf

[15] http://www.essi.fr/~leroux

[16] http://www.epfl.ch.pdf

[17] http://membrs.lycos.fr/merciber

[18] *Matlab, Image processing Toolbox,* User's guide, Version2,The Mathworks Inc

[19] A.QUINQUIS:*Le traitement du signal sous Matlab,* Hermès science, Paris 2000

[20] http://www.mathworks.fr/access

Auteur

Nom	**:**	ANDRIAMBOLOLONA
Prénoms	**:**	Soamampianina Zazah
Adresse	**:**	Lot :39 K III M/3605 - Isada - Fianarantsoa (301)
Titre du mémoire	**:**	**DETECTION DE CONTOURS EN TRAIMENT D'IMAGE**
Téléphone	**:**	032 07 76 234
Nombre de pages	**:**	105
Nombre de tableaux	**:**	2
Nombre de figures	**:**	47
Mots clés	**:**	image lissage contour gradient laplacien
Key words	**:**	image edge smoothing gradient laplacian
Directeur de mémoire	**:**	Monsieur RANDRIAMITANTSOA Paul Auguste

RESUME :

Ce mémoire nous a permis de mieux cerner le domaine des images numériques .Dans sa première partie nous avons vu les bases théoriques sur le traitement d'image, d'un point de vue plutôt généraliste avec une étude un peu approfondie sur la détection de contour. Des opérations essentielles destinées au traitement d'image ont été abordées, avec certaines définitions et notions qui ont été nécessaires pour une meilleure compréhension du domaine dans lequel s'inscrit cette étude.

Dans la dernière partie de ce travail, plusieurs processus de traitement d'images ont été simulés sous MATLAB après une présentation préalable du logiciel utilisé. En effet, la puissance de MATLAB permet de réduire au stricte nécessaire le temps passé en programmation, au bénéfice du temps passé à la compréhension. Ainsi, il sera capable de les adapter facilement à une nouvelle application.

ABSTRACT :

This memory allowed us to better surround the domain of the numeric images. In the first section, we have seen the theoretical bases for image processing, from a rather general point of view and with a more in-depth study on edge detection. We have dealt with some essential operations designed for image processing, with a few definitions and notions that have been useful for a better understanding of this field of study.

In the last part of this research work, several image processing procedures have been simulated of the program used. In fact, MATLAB's power enables us to use the minimum time for the programming thus providing us more time to try to understand it. Therefore, it will be able to adapt them more easily for a new application.