N° d'ordre : 03 / STI-STM Année Universitaire : 2009 / 2010

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 : Services de Traitement Multimédias (STM)

par : **DAHOLISON Eugène**

DETECTION DE RECTANGLES BASEE SUR LA TRANSFORMEE DE HOUGH

Soutenu le 21 Avril 2011 devant la Commission d'Examen composée de :

Président :

M. ANDRIAMIASY Zidora

Examinateurs :

M. RAKOTOMALALA Mamy Alain

Mme RAMAFIARISONA Malalatiana

M. RAKOTONDRAINA Tahina Ezéchiel

Directeur de mémoire :

M. RAZAFINDRADINA Henri Bruno

REMERCIEMENTS

Ce travail de mémoire de fin d'études n'aurait pu être réalisé sans la Grâce de Dieu ainsi que la contribution des personnes suivantes auxquelles, j'adresse mes remerciements les plus profonds :

Monsieur ANDRIANARY Philipe, Professeur Titulaire et Directeur de l'Ecole Supérieure Polytechnique d'Antananarivo, pour nos cinq années d'études dans cet établissement.

Monsieur RAZAKARIVONY Jules, Maître de Conférences et Chef du Département Télécommunication à l'ESPA pour les précieuses directives et la qualité de la formation qu'il nous a dispensé pendant nos études universitaires.

Monsieur RAZAFINDRADINA Henri Bruno, Maître de Conférences, Enseignant au sein du Département Télécommunication et Directeur de ce mémoire, qui n'a cessé de me prodiguer de précieux conseils et de me guider dans les grandes lignes de ce travail.

Monsieur ANDRIAMIASY Zidora, Maître de Conférences, Enseignant du Département Télécommunication, qui nous a fait l'honneur de présider les membres de jury de ce mémoire.

Je tiens à témoigner ma gratitude aux autres membres de jury à savoir :

Monsieur RAKOTOMALALA Mamy Alain, Assistant, Enseignant au sein du Département Télécommunication.

Madame RAMAFIARISONA Malalatiana, Doctorante Télécommunications, Enseignante au sein du Département Télécommunication.

Monsieur RAKOTONDRAINA Tahina Ezéchiel, Doctorant Télécommunications, Enseignant au sein du Département Télécommunication, qui ont accepté de juger notre travail ainsi que d'y apporter des remarques et des suggestions visant à son amélioration.

Je tiens également à remercier tous les Enseignants de l'Ecole Supérieure Polytechnique, qui nous ont formés durant ces cinq années passées.

Je n'oublierai jamais ma famille qui m'a soutenu tant moralement que du point de vue matériel durant mes années d'études.

Et pour toute personne qui m'a aidé de près ou de loin à l'élaboration de ce mémoire, je vous en suis reconnaissant et que Dieu avec sa bonté et son éternelle bienveillance soit avec vous.

TABLE DES MATIERES

REMERCIEMENTS ..... **i**

TABLE DES MATIERES ..... **ii**

NOTATIONS ET ABREVIATIONS ..... **vi**

INTRODUCTION **1**

CHAPITRE 1 : ACQUISITION DES IMAGES NUMERIQUES **3**

1.1 Généralités **3**

1.1.1 Historiques ***3***

1.1.2 Introduction ***3***

1.1.3 Quelques définitions ***4***

1.1.4 Domaines d'application du traitement d'images ***5***

1.2 Perceptions visuelles **6**

1.2.1 Le système visuel humain (SVH) ***6***

1.2.2 Perception des couleurs ***7***

1.2.2.1 La lumière 8

1.2.2.2 Notion de couleurs 8

1.2.2.3 Synthèse de couleurs 9

1.3 Acquisition et numérisation de l'image **11**

1.3.1 Principe ***11***

1.3.1.1 Temporelles 11

1.3.1.2 Spatiales 11

1.3.2 Numérisation de l'image ***11***

1.3.2.1 Les étapes de numérisation de l'image 12

1.3.2.2 Caractéristiques des images numériques 14

1.3.2.3 Les différents types de représentation d'image 15

1.4 Les différents types de formats d'images **17**

1.5 Propriétés d'une image numérique 18

1.5.1 Résolution et dimension (ou définition) 18

1.5.1.1 Définition 18

1.5.1.2 Résolution 18

1.5.2 Représentation des couleurs 19

1.6 Amélioration d'image 19

1.6.1 Rehaussement de dynamique 19

1.6.1.1 Extension de dynamique 19

1.6.1.2 Egalisation d'histogramme 20

1.6.2 Réduction du bruit dans les images 20

1.6.2.1 Filtre moyenneur ou lissage par la moyenne 21

1.6.2.2 Filtre de Gauss ou lissage Gaussien 22

1.6.2.3 Filtrage utilisant le Laplacien 24

1.7 La détection de contour 24

1.7.1 Calcul direct des dérivées 25

1.7.2 Segmentation par seuillage 26

1.8 Conclusion 26

CHAPITRE 2 : DESCRIPTION DE CONTOURS 27

2.1 Notions du *Pavage* et *maillage* 27

2.2 Adjacences 29

2.3 Description de contours (caractérisation de la forme par ses contours) 30

2.3.1 Représentation des contours 30

2.3.1.1 Code de Freeman 30

2.3.1.2 Code de Freeman relatif 33

2.3.1.3 Descripteur de Fourier 33

2.3.2 Approximations polynomiales 37

2.3.2.1 Approximation d'un nuage de points par une droite unique 37

2.3.2.2 Approximations polygonales, simplification de contours polygonaux 39

2.4 Conclusion 40

CHAPITRE 3 : TRANSFORMEE DE HOUGH 41

3.1 Introduction 41

3.2 Propriétés de la transformée de Hough 41

3.3 Transformée directe 42

3.4 Transformée inverse 42

3.5 Algorithme de la transformée de Hough 43

3.5.1 Pour l'équation de droite de la forme : $y = ax + b$ 43

3.5.2 Exemple 43

3.5.3 Pour l'équation de droite de la forme : $\rho = x\cos\theta + y\sin\theta$ 46

3.5.3.1 Mise en œuvre 46

3.5.3.2 L'algorithme 47

3.5.4 Exemple 47

3.6 Les différentes transformées de Hough 49

3.6.1 La transformée de Hough standard 49

3.6.2 La transformée de Hough généralisée 49

3.6.3 La transformée de Hough pondérée 50

3.7 Avantages et inconvénients 50

3.7.1 Avantages 50

3.7.2 Inconvénients 50

3.8 Conclusion 50

CHAPITRE 4 : APPLICATION DE LA TRANSFORMEE DE HOUGH A LA DETECTION DES RECTANGLES 51

4.1 Introduction 51

4.2 Hypothèse 51

4.3 Généralité sur le rectangle 51

4.3.1 Propriété d'un rectangle *51*

4.3.2 Exemple d'un rectangle *52*

4.4 Les différentes étapes pour reconnaître un rectangle dans une image **52**

4.4.1 Détection de contours *52*

4.4.2 Extraction de contours *53*

4.4.2.1 La première dérivée d'une image (approche gradient): 55

4.4.2.2 La seconde dérivée de l'image (approche Laplacien) 56

4.4.3 Transformation de l'espace image en espace Hough *56*

4.4.4 Recherche des pics dans l'accumulateur *56*

4.4.5 Recherche des pics qui sont parallèles (pics réguliers) *57*

4.4.6 Recherche des pics prolongés (le pic qui correspond à un rectangle) *58*

4.4.7 Détection d'une droite *59*

4.4.8 Détection des droites qui sont parallèles *63*

4.4.9 Détection des droites qui sont perpendiculaires *65*

4.5 Applications «détection d'un rectangle» **66**

CONCLUSION: **70**

ANNEXE 1 : ***Les fonctions de base*** **71**

ANNEXE 2 : ***Quelques codes de l'interface de simulation*** **72**

ANNEXE 3: ***Utilisation du logiciel*** **80**

BIBLIOGRAPHIE **86**

RENSEIGNEMENTS SUR L'AUTEUR **87**

RESUME **88**

ABSTRACT **88**

NOTATIONS ET ABREVIATIONS

1D	Une dimension
2D	Deux dimensions
3D	Trois dimensions
BMP	BitMaP
CD-ROM	Compact Disc-ROM
CMY	Cyan Magenta Yellow
CMYK	Cyan Magenta Yellow blacK
DPI	Dots Per Inch
GHT	Transformée de Hough Généralisée
GIF	Graphics Interchange Format
HT	Hough Transform
HSL	Hue Saturation Lightness
IR	Infra-Rouge
IRM	Imagerie par Résonance Magnétique
ISO	International Organization for standardization
JPEG	Joint Photographic Experts Group
MPEG	Moving Picture Experts Group
PGML	Precision Graphics Markup Language
PNG	Portable Network Graphics

PPP	Pixels Par Pouce (en DPI : Dots Per Inch)
RdF	Reconnaissance de Formes
RFC	Request For Comments
RGB	Red Green Blue
RMN	Résonance Magnétique Nucléaire
RVB	Rouge Vert Bleu
SHT	Transformée de Hough Standard
SVG	Scalable Vector Graphics
SVH	Système Visuel Humain
TIFF	Tag (ged) Image File Format
UV	Ultra Violet
VML	Vector Markup Language
XML	eXtensible Markup Language
YUV	Y représentant la luminance, U et V deux chrominances orthogonales

INTRODUCTION

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

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 avait pas vraiment eu d'évolution pour l'image jusqu'à la période d'après-guerre.

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 traitement d'images désigne une discipline de l'informatique et des mathématiques appliquées qui étudie les images numériques et leurs transformations, dans le but d'améliorer leur qualité ou d'en extraire de l'information [1].
Il s'agit d'un sous-ensemble du traitement du signal dédié aux images et aux données dérivées comme la vidéo (par opposition aux parties du traitement du signal consacrées à d'autres types de données: son et autres signaux monodimensionnels notamment), tout en opérant dans le domaine numérique (par opposition aux techniques analogiques de traitement du signal, comme la photographie ou la télévision traditionnelles).

On désigne par reconnaissance de formes ou RdF (ou parfois reconnaissance de motifs) un ensemble de techniques et méthodes visant à identifier des *motifs* à partir de *données brutes* afin de prendre une décision dépendant de la catégorie attribuée à ce motif. On considère que c'est une branche de l'intelligence artificielle qui fait largement appel aux techniques d'apprentissage automatique et aux statistiques.

Les *formes* ou *motifs* à reconnaître peuvent être de nature très variée. Il peut s'agir de contenu visuel (code barre, visage, empreinte digitale...) ou sonore (reconnaissance de parole), d'images médicales (rayon X, EEG, IRM...) ou multispectrales (images satellitaires) et bien d'autres. Par exemple, on utilise ce système pour la détection d'un numéro d'immatriculation d'un véhicule, la recherche des cellules cancéreuse, la recherche de numéro sur le cadre rectangle d'un chèque, la détection des panneaux routiers.

La transformée de Hough est une technique de reconnaissance de formes inventée en 1962 par Paul Hough, utilisée dans le traitement d'images numériques [2].
L'application la plus simple permet de reconnaître les lignes d'une image, mais des modifications peuvent être apportées pour reconnaître n'importe quelle forme : c'est la *transformée généralisée de Hough* développée par Richard Duda et Peter Hart en 1972.
Le principe qui sous-tend la transformée de Hough est qu'il existe un nombre infini de lignes qui passent par un point, dont la seule différence est l'orientation (l'angle). Le but de la transformée est de déterminer lesquelles de ces lignes passent au plus près du schéma attendu.
Afin de déterminer que deux points se trouvent sur une même ligne potentielle, on doit créer une représentation de la ligne qui permet une comparaison dans ce contexte.

Le titre de ce mémoire est ***la détection de rectangles basée sur la transformée de Hough.***

Le but de ce mémoire est la détection d'un rectangle dans une image de synthèse. Pour cela il faut extraire les points caractéristiques pour reconnaitre des silhouettes dans une image. Mais avant d'entrer dans le vif du sujet nous allons tout d'abord présenter des notions importantes, des méthodes d'amélioration de qualité d'image (rehaussement dynamique et la réduction du bruit dans les images) qu'il faut savoir en priori dans le traitement d'image, ensuite on va faire une étude comparative entre les différentes approches pour la description de contours. Après on va expliquer les différentes étapes de la transformée de Hough. Et enfin on va passer à la simulation de la détection d'une droite, d'une droite parallèle, d'une droite perpendiculaire et d'un rectangle

CHAPITRE 1 : ACQUISITION DES IMAGES NUMERIQUES

1.1 Généralités

1.1.1 Historiques

On désigne sous le terme d'image numérique toute image (dessin, icône, photographie,…) acquise, créée, traitée ou stockée sous forme binaire (c'est-à-dire composée 0 et 1).

- Acquise par des convertisseurs analogique numérique situés dans des dispositifs comme les scanners, les appareils photo ou caméscopes numériques, les cartes d'acquisition vidéo (qui numérisent directement une source comme la télévision) [3];
- Créée directement par des programmes informatiques, via la souris, les tablettes graphiques ou par la modélisation 3D (ce que l'on appelle par abus de langage les « images de synthèse »).
- Traitée grâce à des outils informatiques. Il est facile de la transformer, modifier en taille, en couleur, d'ajouter ou supprimer des éléments, d'appliquer des filtres variés, etc.
- Stockée sur un support informatique (disquette, disque dur, CD-ROM…)

1.1.2 Introduction

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 délicate, 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 apparait 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 chaine de traitement car elle conditionne la capacité de stockage nécessaire ainsi que la mise en œuvre.

1.1.3 Quelques définitions

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

Définition 1.01 :

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

Définition 1.02 :

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 1.03 :

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 1.04 :

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 1.05 :

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.

Définition 1.06 :

La chrominance, dans le domaine audiovisuel, désigne la différence colorimétrique existant entre une couleur donnée et une couleur de référence (RVB). En informatique, désigne la partie du signal vidéo relative à la couleur de l'image.

Définition 1.07 :

Un bruit (ou parasite) dans une image est considéré comme un phénomène de brusque variation de l'intensité d'un pixel par rapport à ses voisins, il provient de l'éclairage des dispositifs optiques et électroniques du capteur.

Définition 1.08:

Les contours représentent la frontière entre les objets de l'image, ou la limite entre deux pixels dont les niveaux de gris représentent une différence significative.

Les textures décrivent la structure des objets de l'image. L'extraction de contour consiste à identifier dans l'image les points qui séparent deux textures différentes.

1.1.4 Domaines d'application du traitement d'images

On peut distinguer trois grands domaines d'applications faisant appel au traitement d'image [4]:

- Sur l'imagerie aérienne et spatiale :
  - Amélioration des images satellites,
  - Analyse des ressources terrestres,
  - Cartographie automatique, analyse météo (exemple: détection de couverture nuageuse)
- Sur les technologies biomédicales :
  - Images scanner, échographie, RMN,
  - Reconnaissance automatique de cellules ou de chromosomes, comptage (cellule cancéreux par exemple), ostéodensimétrie (exemple prévention de l'ostéoporose par calcul sur image du taux de calcium osseux), etc.
- Dans le domaine robotique :
  - Assemblage de pièces, contrôle qualité (exemple identification de code barre d'une pièce par traitement d'image), robotique mobile
  - Stéréovision (3D) « ou **mesure stéréoscopique** est une méthode de mesure qui consiste à se servir de la prise d'images (photographiques ou numériques) sous différents angles de vue pour déterminer les dimensions, les formes ou les positions d'objets ».

Et aussi :

- L'astronomie, la chimie, la physique nucléaire (identification de la trajectoire de particules), l'armement (guidage de missiles, Reconnaissance Des Formes)
- La télésurveillance (exemple radar automatique : recherche en temps réel d'un véhicule par reconnaissance de son immatriculation parmi un flot de véhicules circulant sur le boulevard périphérique par caméra fixe).

1.2 Perceptions visuelles

Avant d'entrer plus en détail dans l'étude sur des images, commençons par quelques notions sur le Système Visuel Humain. Assez paradoxalement, il s'agit d'une des taches les plus complexes car le fonctionnement du système visuel humain fait intervenir la subjectivité de l'observateur et car, en pratique, il apparait difficile d'inclure la plupart des résultats des études psycho visuelles dans un traitement d'image courant [5].

1.2.1 Le système visuel humain (SVH)

Grace a 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.

De forme approximativement sphérique, l'œil est l'organe de base de la vision. Il comporte un ensemble d'éléments destinés à recevoir le rayonnement incident, former l'image des objets perçus et traiter les informations recueillies.

Comme le montre la figure 1.01, l'œil humain est constitué de :

- ***La conjonctive*** : c'est une solide membrane blanche, opaque aux rayons lumineux, servant à attacher l'œil dans son orbite.
- ***La cornée*** : il s'agit d'une membrane transparente et résistante située sur la face avant de l'œil. Son rôle est de protéger le globe oculaire sur la face avant.
- ***L'iris*** : il fonctionne comme un diaphragme en dosant la quantité de lumière qui pénètre dans l'œil. Son ouverture centrale est la pupille.

- ***Le cristallin*** : il fonctionne comme une lentille à focale variable, grâce à sa capacité de modifier sa courbure.
- ***La rétine*** : c'est sur elle que se forment les images provenant de l'extérieur. La rétine contient deux types de cellules photosensibles : les cônes et les bâtonnets.
- ***La macula*** : appelée également tache jaune, contient en son centre une petite d'expression, la fovéa. Cette dernière est la zone d'acuité maximum de l'œil.
- ***Le nerf optique*** : il conduit les informations au cerveau, en passant par un relais très important, le corps genouillé latéral, chargé d'effectuer une première analyse des données.

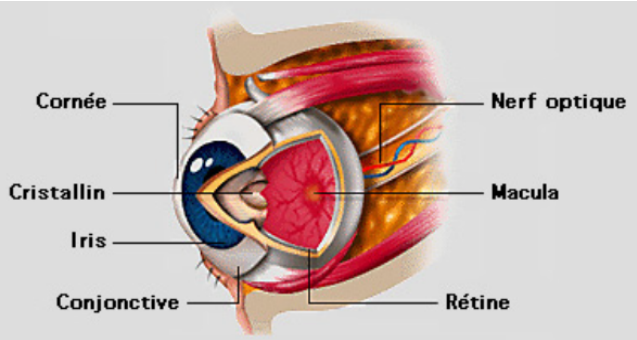


Figure 1.01: *Coupe de l'œil.*

1.2.2 Perception 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.

Nous allons ici faire quelques rappels quant aux relations lumière-matière et donner quelques éléments de compréhension du système visuel humain.

- La source lumineuse utilisée ;
- La géométrie d'observation (angles d'éclairement et d'observation) ;
- La scène et ses caractéristiques physiques ;
- L'œil de l'observateur, avec les qualités et les défauts propres à chaque individu ;
- Le cerveau de l'observateur, dont la capacité de discernement des couleurs évolue en fonction de l'âge et de l'expérience acquise.

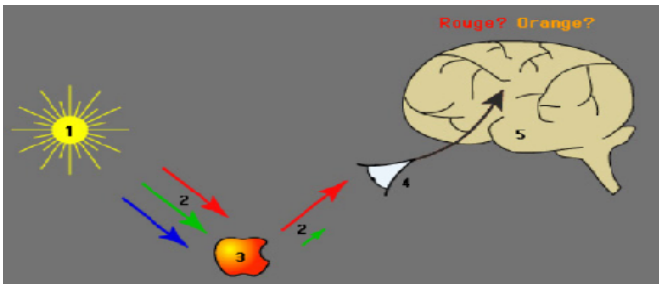


Figure 1.02: *Perception de la couleur*

1.2.2.1 La lumière

La lumière couvre une partie du spectre d'énergie électromagnétique. Un rayonnement électromagnétique est en général constitue d'un certain nombre de longueurs d'onde (ou fréquences) que les dispositifs dispersifs permettent de séparer en un spectre. En un mot, la lumière est une distribution d'énergie émise à certaines fréquences ayant une certaine intensité [5].

1.2.2.2 Notion 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 [6]. La longueur d'onde d'un phénomène oscillatoire est exprimée par la relation (1.01) :

$$\lambda = CT \tag{1.01}$$

- λ désigne la longueur d'onde
- C désigne la célérité de l'onde (pour la lumière $3 \times 10^8\ m.s^{-1}$)
- T désigne la période de l'onde

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.

(Higher Frequency= très haute fréquence, Lower Frequency= plus basse fréquence et

Wavelength = Longueur d'onde en nano-mètre (nm)).

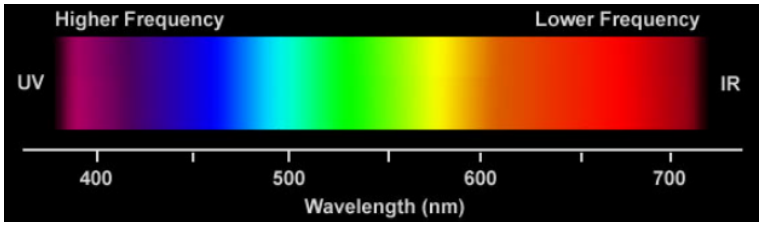


Figure 1.03 : *Les longueurs d'onde associées aux couleurs.*

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 utilise comme système de base sur l'écran informatique et dans le traitement d'image. Ces trois couleurs sont alors dénommées «couleurs primaires ».

1.2.2.3 Synthèse de couleurs

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

a. Synthèse additive :

La synthèse additive consiste à restituer une couleur par addition de trois sources lumineuses rouge, vert et bleu (RVB) [RGB : Red, Green, Blue]. Ce procédé est utilisé dans les tubes cathodiques. 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 combine au bleu donne du cyan
- Le bleu combine au rouge donne du magenta
- Le vert combine au rouge donne du jaune

b. Synthèse 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. Ce procédé est utilisé en photographie et pour l'impression des couleurs.

Dans un système de référence RVB, une couleur C est définie par ses trois composantes :

$$C = rR + vV + bB \tag{1.02}$$

Où r, v, et b représentant les quantités de chaque couleur dans une échelle appropriée 0-100 (ou 0-255 pour une quantification sur un octet par composant).

De la même façon on peut définir une couleur dans le système de référence complémentaire (Cyan, Magenta, Jaune) :

$$C = cC + mM + jJ \tag{1.03}$$

Le passage d'un système à l'autre se fait par les relations : $C = 100 - r; m = 100 - v;$

$$j = 100 - b$$

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 combine avec le cyan donne du bleu
- Le magenta combine avec le jaune du rouge
- Le cyan combine avec le jaune du vert

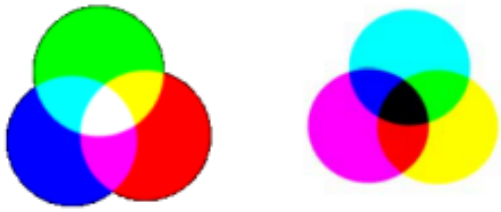


Figure 1.04 : *Synthèses additive et soustractive*

Remarque:

La synthèse soustractive est utilisée pour l'impression sur feuille blanche car la détermination des composantes RVB d'une onde s'opère par addition sur fond noir. L'arrière plan est donc suppose 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).

1.3 Acquisition et numérisation de l'image

1.3.1 Principe

Avec un seul œil, capteur a 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:

1.3.1.1 Temporelles

L'apparition subite d'une forme dans notre champ de vision attire instantanément notre attention.

1.3.1.2 Spatiales

L'œil est instinctivement attire par les transitions brutales de niveau de luminance.

Une transition entre deux zones homogènes sera naturellement accentuée par l'œil.

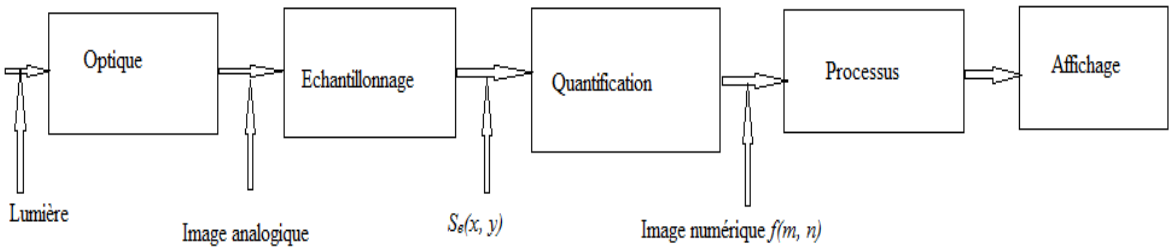


Figure 1.05: *Acquisition d'une image*

1.3.2 Numérisation de l'image

Une image « réelle » va être transformée en une image numérique par différents outils de transformation (camera ou appareil photo numérique, scanner, satellite, …). Tous ces systèmes peuvent être comparés à des capteurs. L'image numérique, est le terme utilisé pour designer 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.

1.3.2.1 Les étapes de numérisation de l'image

a. Echantillonnage :

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 :

$$S_e(t) = S(t)\sum_{-\infty}^{+\infty}\delta(t - nT_e) \tag{1.04}$$

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

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

$$peigne(x, y, \Delta x, \Delta y) = \sum_{-\infty}^{+\infty}\sum_{-\infty}^{+\infty}\delta(x - n\Delta x, y - n\Delta y) \tag{1.05}$$

La fonction représentant l'image échantillonnée :

$$S_e(x, y) = f(x, y)\, peigne(x, y, \Delta x, \Delta y) \tag{1.06}$$

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

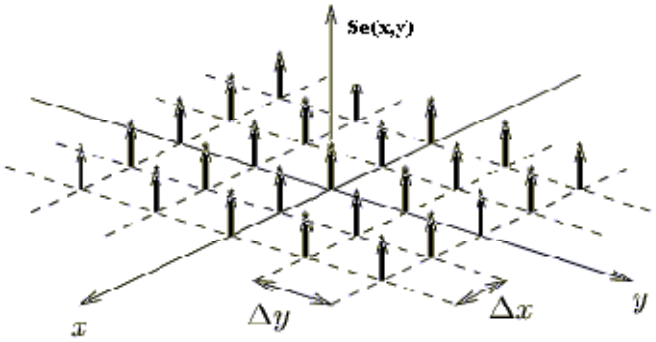


Figure 1.06 : *Image échantillonnée*

Remarque :

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

$$F_e > 2.F_{max} \tag{1.07}$$

Avec

$$F_e = \frac{1}{T_e} \tag{1.08}$$

b. Quantification

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 $\boldsymbol{\propto_{max}}$ désigne la valeur maximale du signal, le pas de quantification q s'écrit alors sous la forme :

$$q = \frac{2.\propto_{max}}{N} \tag{1.09}$$

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

$$x_i - \frac{q_i}{2} < s_e(t) < x_i + \frac{q_i}{2} \tag{1.10}$$

c. Codage

La valeur obtenue a l'issue de la quantification peut alors être codée, sur *n bits* tel que : $\boldsymbol{N = 2^n}$

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 signifie : « transformation d'un message exprime 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.

1.3.2.2 Caractéristiques des images numériques

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

a. La résolution de l'image

Définition 1.09 :

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 utilise. 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 :

$$X \times résolution = x\ pixels$$
$$Y \times résolution = y\ pixels \quad (1.11)$$

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.

Par exemple une image de 1*1 pouce scannée a 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).

b. La dynamique de l'image

La dynamique de l'image correspond a 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 utilise pour stocker l'information sur les teintes de gris ou les couleurs.

1.3.2.3 Les différents types de représentation d'image

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.

a. 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 représentes 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 utilise pour scanner un texte quand celui-ci est compose d'une seule couleur. La figure suivante nous montre un exemple de texte scanne sous forme d'image binaire :

Figure 1.07: *Image binaire*

b. *Image d'intensité :*

C'est une matrice dont chaque élément est un réel compris entre 0 (noir) et 255 (blanc). La dénomination « images à niveaux de gris » est également utilisée car ces valeurs comprises entre

0 et 255 représentent les différents niveaux de gris.

En général, les images en niveaux de gris renferment 256 teintes de gris, elles font aussi partie des images a 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).

c. *Image couleur :*

S'il existe plusieurs modes de représentation de la couleur, le plus utilise pour le maniement de la couleur numérique est l'espace couleur RVB. 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 RVB 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".

d. 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 limite de couleurs. Ces couleurs sont mémorisées dans une table de couleur (appelée *colormap*) qui est une matrice $n \times 3$ (ou 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».

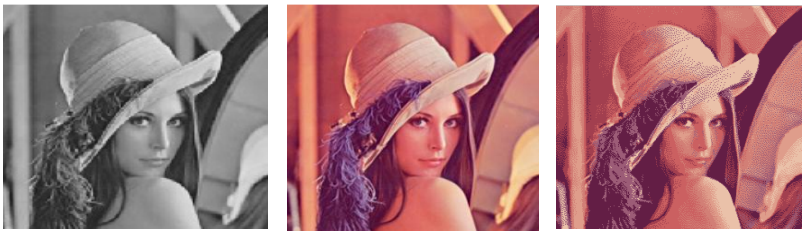


Figure 1.08 : *Image d'intensité, Image couleur, Image couleur indexée*

1.4 Les différents types de formats d'images

- *JPEG :*

La norme JPEG est une norme qui définit le format d'enregistrement et l'algorithme de décodage pour une représentation numérique compressée d'une image fixe [3][5][6].

- *S.V.G :*

SVG est, traduit de l'anglais, signifie «Graphique Vectoriel Adaptable», et couramment abrégé par SVG, est un format de données conçu pour décrire des ensembles de graphiques vectoriels et basé sur XML. Ce format inspiré directement du VML et du PGML est spécifié par le World Wide Web Consortium.

- *Le G.I.F :*

Ce format (Littéralement «format d'échange d'images»), plus connu sous l'acronyme GIF, est un format d'image numérique couramment utilisé sur la Toile.

- *Le TIFF :*

C'est un format de fichier pour images numériques. Adobe en est le dépositaire et le propriétaire initial.

Plus exactement, il s'agit d'un format de conteneur (ou encapsulation), à la manière de avi ou zip, c'est-à-dire pouvant de contenir des données de formats arbitraires.

- *Le PNG :*

C'est un format ouvert d'images numériques, qui a été créé pour remplacer le format GIF, à l'époque propriétaire et dont la compression était soumise à un brevet. Le PNG est un format non

destructeur spécialement adapté pour publier des images simples comprenant des aplats de couleurs. Il a été normalisé par l'ISO (ISO/CEI 15948:2004).

PNG est une spécification pour Internet et l'objet d'un RFC.

1.5 Propriétés d'une image numérique

1.5.1 Résolution et dimension (ou définition)

1.5.1.1 Définition

Une image est définie par le nombre de points la composant. En image numérique, cela correspond au nombre de pixels qui compose l'image en hauteur (axe vertical) et en largeur (axe horizontal) : 206 pixel par 345 pixel par exemple abrégé en «206x346».

1.5.1.2 Résolution

La résolution d'une image est définie par un nombre de pixels par unité de longueur de la structure à numériser (classiquement en PPP). Ce paramètre est défini lors de la numérisation (passage de l'image sous forme binaire), et dépend principalement des caractéristiques du matériel utilisé lors de la numérisation. La résolution d'une image numérique définit le degré de détail de l'image. Ainsi, plus la résolution est élevée, meilleure est la restitution. Cependant, pour une même dimension d'image, plus la résolution est élevée, plus le nombre de pixels composant l'image est grand. Le nombre de pixels est proportionnel au carré de la résolution, étant donné le caractère bidimensionnel de l'image : si la résolution est multipliée par deux, le nombre de pixels est multiplié par quatre. Ceci est illustré à la figure 1.09

Figure 1.09 : *Image voiture de même dimension mais de différentes résolutions (résolution décroissante)*

1.5.2 Représentation des couleurs

Il existe plusieurs modes de codage informatique des couleurs, le plus utilisé pour le maniement des images est l'espace colorimétrique RVB ou RGB. Il existe d'autres modes de représentation des couleurs :

- CMJN ou CMYK utilise principalement pour l'impression, et basé sur une synthèse soustractive des couleurs ;
- TSL ou HSL, où la couleur est codée suivant le cercle des couleurs ;
- Base de couleur optimale YUV, Y représentant la luminance, U et V deux chrominances orthogonales.

1.6 Amélioration d'image

Dans cette partie, nous allons voir d'une part comment modifier l'apparence d'une image pour qu'un observateur puisse plus facilement extraire de l'information et d'autre part, comment supprimer le bruit des images en vue d'un traitement ultérieur.

1.6.1 Rehaussement de dynamique

1.6.1.1 Extension de dynamique

Cette méthode consiste à utiliser au mieux la dynamique de niveaux de gris. Ainsi, si une image possède des niveaux de gris entre G_{min} et G_{max}, on va étendre la plage des niveaux de gris pour ramener à une dynamique comprise entre 0 et 255. Cette étendue réduite de niveaux de gris peut survenir suite à un temps de pose incorrect ou à un éclairage de la scène trop faible.

La transformation mise en place est :

$$I'(x,y) = \frac{I(x,y) - G_{min}}{G_{max} - G_{min}} \tag{1.12}$$

Cette transformation ne fait qu'améliorer la qualité visuelle de l'image, elle ne change pas l'information présente dans l'image, comme on peut le constater dans la figure 1.10

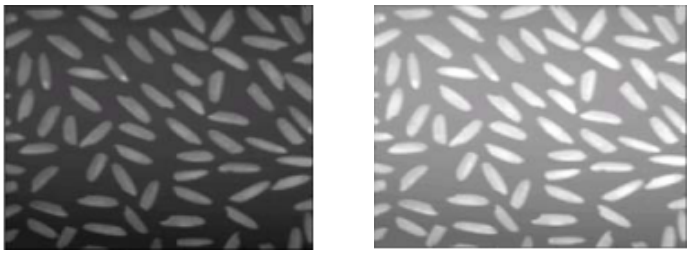


Figure 1.10: *Extension de dynamique d'une image « rice.png» avec* $\boldsymbol{G_{max} = 234}$ *et* $\boldsymbol{G_{min} = 34}$

1.6.1.2 Egalisation d'histogramme

Cette opération a pour but de rendre l'histogramme le plus plat possible. On souhaite ainsi que chaque niveau de gris soit également représenté dans l'image. Soit G, le niveau de gris d'un pixel de départ, le niveau de gris de l'image d'arrivée sera :

$$G' = \frac{255}{Nombre\ de\ pixel} histocumulé(G) \tag{1.13}$$

Cette opération est illustrée par l'équation (1.13) :

Figure 1.11: *Egalisation d'histogramme d'image «Lena.tif»*

1.6.2 Réduction du bruit dans les images

On peut considérer une image comme étant constitué de plusieurs zones homogènes représentant les objets. Dans la réalité, des fluctuations des niveaux de gris sont présentées à cause du bruit. On cherchera donc à diminuer l'amplitude de ces fluctuations. La réduction du bruit dans une image *I* revient à filtrer ce dernier avec filtre *f* appelé aussi masque, soit de convoluer de l'image avec le filtre *f*. si *If* est l'image filtrée alors on l'obtient par :

$$I_f = I * f \tag{1.14}$$

Dans le domaine de traitement de l'image on peut rencontrer différente sorte de filtre.

But :

On peut considérer une image comme étant constituée de plusieurs zones homogènes représentant les objets. Dans la réalité, des fluctuations des niveaux de gris sont présentes à cause du bruit. On cherchera donc à diminuer l'amplitude de ces perturbations, sans toucher aux zones de transitions.

1.6.2.1 Filtre moyenneur ou lissage par la moyenne

L'idée est de réaliser une moyenne des niveaux de gris autour du pixel centrale. Pour cela, on peut utiliser un masque du type :

Voici le masque de la moyenne :

$$\frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{1.15}$$

Exemple :

Soit le morceau d'image originale ci-dessous

30	**50**	**60**
40	**100**	**30**
30	**60**	**10**

Avec ce type de masque :

$$\frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

On aura alors les différentes étapes suivantes :

- La valeur centrale 100 sera remplacée par:

$$(100 \times 1) + (30 \times 1) + (50 \times 1) + (60 \times 1) + (40 \times 1) + (30 \times 1) + (30 \times 1) + (60 \times 1) + (10 \times 1) = 410$$

- La somme est normalisée : X=410/9=45.55
- La valeur 100 de départ, deviendra donc 45.55, et la nouvelle fenêtre image sera :

30	**50**	**60**

40	**45.55**	**30**
30	**60**	**10**

- Une fois ce calcul effectué, le filtre passera à la valeur suivante en (ici 30) et appliquera le calcul par rapport aux voisins immédiats de 30 en ayant bien évidemment prit la valeur d'origine du point précédent et non la valeur filtrée.
- Au bout de l'avant dernière colonne, il ne prend que les valeurs où le filtrage est possible, et donc, une image filtrée possède systématiquement des marges non-filtrés, il passera à la ligne suivante, en occurrence ici, à la valeur 60.

La taille du masque est un paramètre variable Plus la taille du masque est élevée, plus le filtre est actif mais le coût en temps de calcul est élevé.

Bien que cette méthode soit très simple à mettre en œuvre, comme on peut le constater elle possède un inconvénient majeur :

Le filtrage introduit un effet de flou ce qui amène à des contours dégradés.

On peut aussi observer l'apparition de bordure en noir sur les images filtrées, ceci est dû au calcul du produit de convolution. On aura donc à effacer ces excédents de coefficient dans la matrice du résultat de la convolution.

Figure 1.12: *Filtre moyenneur d'image « Lena.tif » avec un masque de la moyenne de taille 5*

1.6.2.2 Filtre de Gauss ou lissage Gaussien

L'usage du filtrage gaussien pour lisser une image est devenu extrêmement populaire. En effet, par rapport au filtre moyenneur, le filtre gaussien accorde une grande importance aux pixels proches du pixel central, et diminue cette importance au fur et à mesure que l'on s'éloigne de celui-ci.

Rappel :

a. courbe de Gauss 1D :

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\ (-\frac{x^2}{2\sigma^2}) \tag{1.16}$$

où σ: est l'écart-type : largeur à mi-hauteur

Voici la courbe qui représente la fonction *g*

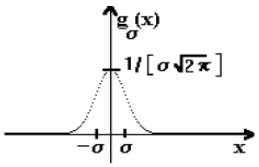


Figure 1.13 *: Courbe représentative de la fonction de Gauss en 1D*

Cette courbe est positive, symétrique (voire Figure 1.13) et possède les propriétés suivantes

$$\int_{-\infty}^{+\infty} g_\sigma(x)dx = 1, \int_{-\infty}^{+\infty} g_\sigma(x)dx = 0.95\ ,\ \int_{-\infty}^{+\infty} g_\sigma(x)dx = 0.999. \tag{1.17}$$

b. La fonction de Gauss en 2D :

On utilise alors la fonction de Gauss pour avoir les éléments du masque à utiliser. L'expression de la fonction de gauss en 2D est donnée par.

$$g(x,y) = \frac{1}{2\pi\sigma^2} exp\ (-\frac{x^2 - y^2}{2\sigma^2}) \tag{1.18}$$

L'intérêt du filtre gaussien est que l'on règle très facilement le degré de filtrage à travers le paramètre σ. En théorie, la gaussienne a une étendue infinie, mais en pratique, on limite cette étendue [-3σ, 3σ].

Le seul inconvénient majeur est qu'il a le même défaut que le filtre moyenneur par le non respect des contours en introduisant un flou. En effet, le filtrage s'accompagne d'un étalement des transitions. La détermination des coefficients du filtre résulte ainsi d'un compromis entre filtrage et dégradation.

Remarquons que comme pour le filtre moyenneur, plus la taille de la fenêtre filtrante est grande plus l'effet du filtrage sera fort.

Figure 1.14: *Filtre de Gauss d'image «Lena.tif» avec* σ = 2

1.6.2.3 Filtrage utilisant le Laplacien

Le Laplacien d'une fonction *f(x, y)*, noté $\nabla^2 f(x,y)$, est définie par :

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} = f(x+1,y) + f(x-1,y) + f(x,y-1) + f(x,y+1) - 4f(x,y) \tag{1.19}$$

Cette expression peut être implémentée à chaque point (*x*, *y*) de la fonction par la convolution de la fonction avec la masque :

$$\begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{-4} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \end{bmatrix} \tag{1.20}$$

Le filtrage d'une image en utilisant le Laplacien est basé sur l'équation :

$$g(x,y) = f(x,y) - [\nabla^2 f(x,y)] \tag{1.21}$$

avec *f(x, y)* est l'image à filtrer, *g(x, y)* est l'image améliorée.

Le Laplacien possède cependant un inconvénient majeur qui est sa grande sensibilité au bruit. En effet cet operateur réalise une dérivée seconde de l'image et est donc instable.

1.7 La détection de contour

Une image est généralement composée de plusieurs zones de niveaux de gris différents correspondants aux différents objets de la scène. Nous allons maintenant trouver une manière d'obtenir ces objets dans l'image, ceci à cause du fait que ces contours d'images sont très utilisés

dans diverses domaines (exemple : processus de reconnaissance des formes traitement d'image satellite).

Deux grandes approches peuvent être envisagées pour extraire les zones pertinentes des images :

- On recherche des zones de niveaux de gris homogènes, c'est l'approche région,
- On recherche les discontinuités dans la scène, c'est l'approche contour.

On s'intéressera à la seconde méthode vu que la première relève beaucoup plus de la logique que du traitement du signal.

L'approche gradient :

La recherche de la transition peut être réalisée en recherchant les maxima locaux du gradient. Dans le cas des images, le vecteur gradient est défini (x, y) par :

$$\overrightarrow{Grad(y,x)} = \begin{pmatrix} \frac{\partial I(y,x)}{\partial x} \\ \frac{\partial I(y,x)}{\partial y} \end{pmatrix} = \begin{pmatrix} Ix(y,x) \\ Iy(y,x) \end{pmatrix} \quad (1.22)$$

Le module du gradient est défini par :

$$G(y,x) = \sqrt{(Ix(y,x))^2 - (Iy(y,x))^2} \quad (1.23)$$

Tandis que son orientation est définie par :

$$\emptyset(y,x) = arctan\left(\frac{Iy(y,x)}{Ix(y,x)}\right) \quad (1.24)$$

Le calcul des dérivées partielles peut être réalisé de plusieurs façons

1.7.1 Calcul direct des dérivées

Il est réalisé de la façon la plus simple qu'il soit :

$$\begin{aligned} Ix(y,x) = I(y,x+1) - I(y,x) = (I * Gx)(y,x)\ et\ I(y,x) \\ = I(y+1,x) - I(y,x) = (I * Gy)(y,x) \end{aligned} \quad (1.25)$$

Ce qui correspond à une convolution de *I* avec les masques :

$$G_x = [-1\ 1]\ et\ G_y = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \tag{1.26}$$

1.7.2 Segmentation par seuillage

Une fois la norme du gradient calculée en chaque point de l'image, il faut seuiller cette norme pour décider si un pixel fait partie ou non d'un contour.

La méthode la plus simple consiste à considérer un seuil fixe S. Tous les pixels possédant une norme supérieure à S sont déclarés appartenir à un contour, mais un seuil trop bas amène à des sur détections, tandis qu'un seuil trop élevé amène à des contours non fermés.

Figure 1.15 : *L'image «Lena.jpg » et l'image après segmentation par seuillage*

1.8 Conclusion

Après avoir définir quelques notions importantes qui s'y trouvent dans l'étude du traitement d'image. Nous avons vu aussi les différents types d'image et de formats d'image. Ainsi on a appliqué différents méthodes de filtrage pour réduire les bruits dans l'image. Mais les méthodes de filtrage créent souvent des images floues. Nous avons utilisé plusieurs méthodes de détection de contours. On a remarqué que ces approches pour détecter des contours séparent les différentes régions sur l'image d'où on peut reconnaître tous les silhouettes dans l'image.

CHAPITRE 2 : DESCRIPTION DE CONTOURS

Dans de nombreuse applications de traitement des images, une fois l'image est segmentée (on peut le faire, de la façon la plus simple, en seuillant l'image), on essaie de reconnaître les divers objets qui la composent à partir de leur seule silhouette. C'est la caractérisation de la forme par ses contours.

2.1 Notions du *Pavage* et *maillage*

Une abstraction mathématique des pixels comme taches lumineuses consiste à considérer qu'ils forment des connexes de points du plan euclidien, de telle sorte que deux pixels voisins ne peuvent s'intersectés que sur leur bord, et que l'ensemble des pixels recouvre le plan. Une telle décomposition du plan s'appelle un *pavage.*

Il n'y a que 3 types de pavages dont les pavages dont les pavés sont des polygones réguliers : triangulaire, carré et hexagonal. Nous les illustrons ci-dessous:

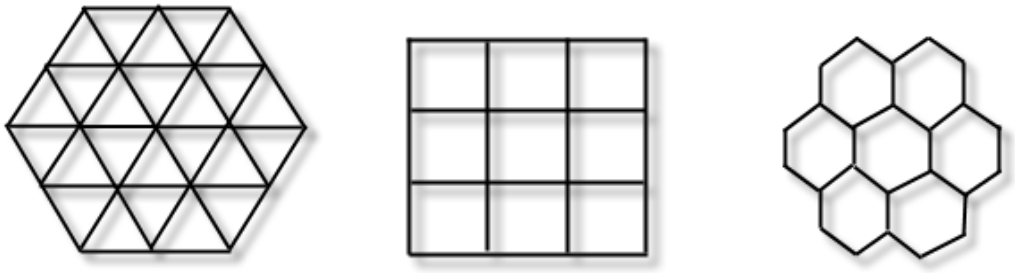


Figure 2.01 : *Les différentes sortes de pavages*

On place un point au centre de chaque pavé, et on joint par une ligne ceux parmi ces points dont les pavés correspondants se touchent par un côté. Cela donne le *maillage* correspondant au pavage, comme illustré ci-dessous.

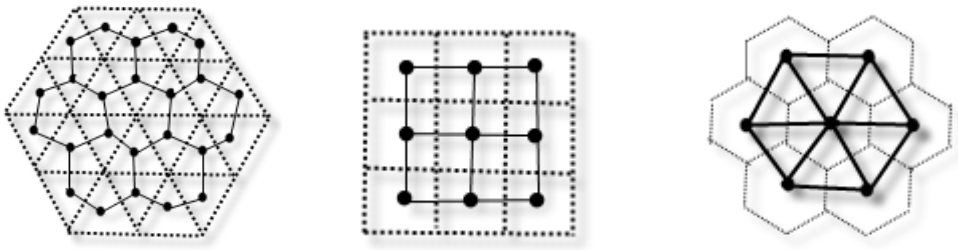


Figure 2.02 : *Maillages correspondants à chaque pavage*

Comme on peut le voir, le pavage carré donne un maillage carré, tandis que le pavage triangulaire donne un maillage hexagonal et vice versa. C'est ce qu'on appelle la *dualité* entre pavages et maillages.

Les centres des pavés seront les points discrets correspondant aux pixels. Pour les pavages carré ou hexagonal, ces points forment un *réseau*, c'est-à-dire, ils coïncident avec les points à coordonnées entières selon deux axes; ces axes sont orthogonaux pour le pavage carré, et forment un angle de 60° ou de 120° pour le pavage hexagonal, comme on le voit ci-dessous.

Figure 2.03 : *Repère dans un maillage*

Pour le pavage triangulaire il est trop difficile de le faire le repère des pixels voisins. Par conséquent, on n'utilise jamais le pavage triangulaire comme modèle de pixels.

Dans le pavage carré, chaque pavé a 8 pavés voisins se répartissant en deux types, à savoir 4 pavés le touchant par un côté et 4 le touchant par un sommet. En d'autres termes, dans le maillage carré, chaque point a 8 voisins, 4 selon les axes et 4 selon les diagonales. Par contre dans le pavage hexagonal, chaque pavé a 6 pavés voisins tous du même type, qui sont les 6 pavés le touchant par un côté. Donc dans le maillage triangulaire correspondant au pavage hexagonal, chaque point a 6 voisins, tous du même type.

Par conséquent le pavage hexagonal a la topologie la plus simple du point de vue mathématique et algorithmique (6 voisins au lieu de 8, et d'un seul type au lieu de deux). De plus, il se prête mieux à la modélisation de phénomènes naturels. Par exemple sur un réseau triangulaire, la modélisation discrète de la dynamique des molécules d'un fluide (liquide, gaz) permet de retrouver à l'échelle macroscopique les lois de la physique des fluides ; ce n'est pas le cas si on prend un maillage carré.

Cependant, on utilise presque toujours le pavage et le maillage carrés, car ils correspondent à nos habitudes cartésiennes. En particulier les points lumineux d'un écran sont toujours disposés suivant un maillage carré.

2.2 Adjacences

Un point du maillage sera appelé *pixel.* Comme le maillage forme un réseau, les pixels sont les points à coordonnées entières selon deux axes. Donc tout pixel se code comme un couple *(i, j)* d'entiers, et l'ensemble des pixels correspond à Z^2. En maillage carré il est d'usage de prendre le premier axe *i* orienté vers le bas et le deuxième axe *j* orienté vers la droite ; ainsi les coordonnées correspondent à la notation matricielle, le pixel *(i, j)* se trouvant à l'intersection de la ligne *i* et de la colonne *j*.

Dans un maillage carré (aussi appelé *grille*), tout pixel a 2 types de voisins, à savoir ses 4 voisins selon les axes, et ses 4 voisins selon les diagonales. La relation de proximité entre deux voisins axiaux est plus forte que celle entre deux voisins diagonaux. Par conséquent nous définissons deux relations d'adjacence sur les pixels de ce maillage : deux pixels *p* et *q* sont dits

- *4-adjacents* s'ils sont voisins suivant un axe,
- *8-adjacents* s'ils sont voisins suivant un axe ou une diagonale.

Les nombres 4 et 8 correspondent au nombre de pixels adjacents à un pixel donné pour le type d'adjacence choisi. Pour le maillage triangulaire (correspondant au pavage hexagonal), il n'y a qu'une seule relation, appelée *6-adjacence.* Les 3 relations d'adjacence sont illustrées ci-dessous.

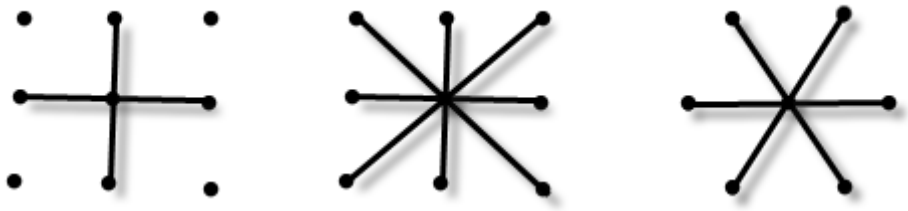


Figure 2.04 : *Adjacence selon les maillages considérés*

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

- (*i*+1, *j*), (*i*-1, *j*), (*i*, *j*+1), et (*i*, *j*-1) pour la 4-adjacence ;
- Les mêmes, plus (*i*+1, *j*+1), (*i*+1, *j*-1), (*i*-1, *j*+1), et (*i*-1, *j*-1) pour la 8-adjacence.

On vérifie les relations suivantes :

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

En maillage triangulaire, les pixels 6-adjacents à (i, j) sont :

- (i+1, j), (i-1, j), (i, j+1), (i, j-1), (i+1, j-1), et (i-1, j+1) pour des axes formant un angle de 60° ;
- (i+1, j), (i-1, j), (i, j+1), (i, j-1), (i+1, j-1), et (i-1, j+1) pour des axes formant un angle de 120°.

2.3 Description de contours (caractérisation de la forme par ses contours)

Une fois que la carte des contours est obtenue sur l'image, il est nécessaire de représenter ceux-ci sous une forme différente qui permette d'arriver à l'objectif voulu qui est la réduction de la quantité des données nécessaires à la description des objets. Le principe est d'isoler chaque contour (chaque ensemble de points connectés), et de décrire chacun d'eux suivant le code

2.3.1 Représentation des contours

2.3.1.1 Code de Freeman

Ce codage est une représentation exacte de chaque pixel du contour et utilise la position relative d'un pixel du contour.

Notions :

a. *Les frontières*

Les pixels d'un écran peuvent être vus comme des points indépendants les uns des autres. Ainsi, une forme est un ensemble de pixels allumés et éteints [7][8]. Ainsi, il est possible de définir 02 types de frontières pour les formes.

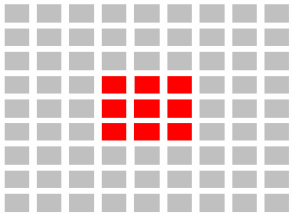
- Une frontière interne qui correspond à une ligne qui passe par tous les pixels éclairés situés à la périphérie de l'objet (en gros les pixels allumés qui marquent le changement entre un état éteint et un état allumé).
- Une frontière externe qui correspond à une ligne qui passe par tous les pixels éteints situés à la périphérie de l'objet.

b. *Les voisinages*

Il y a le voisinage V4 qui définit 4 voisins pour chaque pixel et le voisinage V8 qui définit 8 voisins.

Explications:

- On considère le schéma ci-dessous qui représente un ensemble de pixels de l'écran (en gris le pixels éteints et en rouge les pixels allumés) :



- Nous allons définir la frontière externe de cette forme avec les voisinages V4 et V8. Pour cela nous considérons les pixels éteints autour de la zone rouge. En bleu nous allons étudier le voisinage V4 (en 2 bits et 4 directions ou 4-connexité) et en noir le V8 (en 3 bits et 8 directions ou 8- connexité).

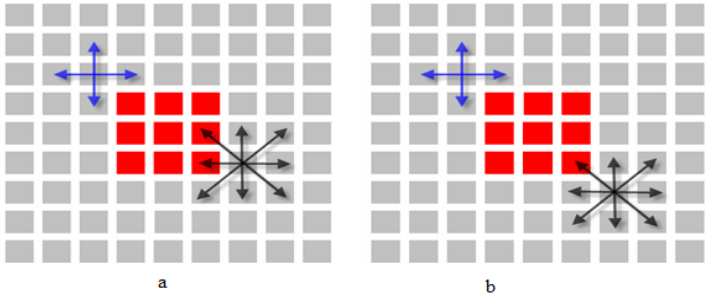


Figure 2.05: *Les frontières et les voisinages*

- Sur la figure (a), nous pouvons voir que le pixel au centre de la croix bleue n'est en contact avec aucun pixel allumé. Ainsi, ce pixel ne fera pas parti de la frontière externe V4. Cependant, le pixel au centre de la croix noire fait parti de la frontière V8.
- Sur la figure (b), nous pouvons remarquer que les 2 pixels considérés font partis de la frontière externe.

Ainsi, en déplaçant comme ceci les croix sur tous les pixels aux alentours de la forme, nous allons pouvoir définir des frontière V4 et V8. Le résultat est affiché ci-dessous :

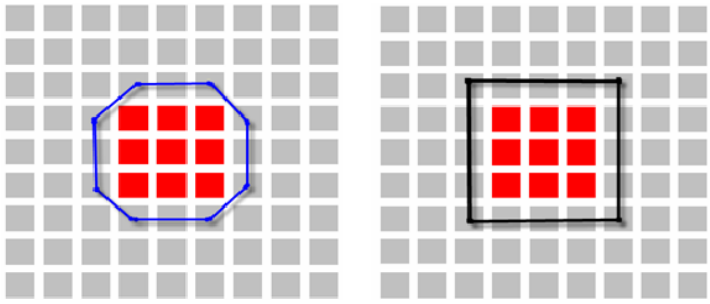
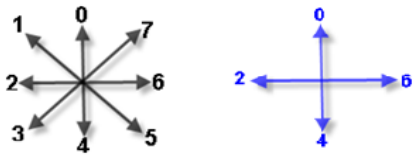


Figure 2.06: *Frontière V4 (en bleue) et frontière V8 (en noir)*



Le code de Freeman se base seulement sur 3 éléments pour caractériser une forme :

- Les coordonnées ***(x, y)*** d'un point appartement au contour de la forme que l'on veut caractériser.
- Un sens de rotation.
- Une chaine de caractères qui code le contour.

Exemple 01:

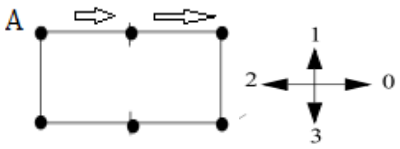


Figure 2.07: *Exemple de la frontière externe en V4*

- Frontière externe en V4 :
  - Coordonnées du point d'origine : $A(x_1, y_1)$: c'est le point de départ dans le sens indiqué par la flèche.
  - Sens de rotation : Trigonométrique
  - Chaine de Freeman : «003221»

Exemple 02:

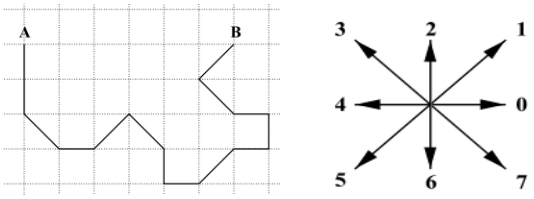


Figure 2.08: *Exemple de la frontière externe en V8*

- Frontière externe en V8 :
  - Coordonnées du point d'origine : $A(x_2, y_2)$: c'est le point de départ et sens de A vers B
  - Sens de rotation : Trigonométrique
  - Chaine de Freeman:«66701760102431»

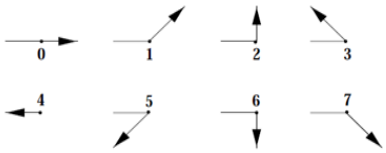
Propriétés :

- Invariant par translation
- Rotation (multiple de 45°)
- Dilatation, rotation, longueur, inversion d'un chemin, simplification d'un chemin

2.3.1.2 Code de Freeman relatif

Code de façon différentielle

- Coordonnées cartésiennes du premier point + 1er déplacement
- Liste des changements de direction



2.3.1.3 Descripteur de Fourier

a. Descripteur de tangente

Dans cette approche on considère le contour comme une courbe continue qui peut être décrite par son abscisse curviligne s à partir d'une origine A choisie (Figure 2.09). On paramètre la courbe par l'angle fait par le vecteur tangent en chaque point et celui au point origine $\emptyset(s)$ et on crée la

variable réduit t qui prend ses valeurs entre 0 et 2π:$t = 2\pi s\ /\ L$, où L est la longueur complète du contour [8]. On construit alors la fonction $\emptyset(t)$:

$$\emptyset(t) = \emptyset\left[\frac{2\pi s}{L}\right] - \frac{2\pi s}{L} \tag{2.01}$$

Le terme correctif prenant en compte l'enroulement de 2π de la tangente pour un tour de contour. La fonction $\emptyset(t)$ est une fonction périodique sur $[0,2\pi[$ qui admet donc une série de Fourier :

$$\emptyset(t) = \sum_{k=0}^{\infty} a_k exp(-ikt). \tag{2.02}$$

On appelle descripteur de Fourier l'ensemble des modules a_k: $\{|a_k|\}$.

Ils bénéficient des propriétés suivantes :

- Ils sont invariants par translation affine,
- Ils sont invariants par changement d'échelle (puisque t est normalisé),
- Ils sont invariants par rotation, puisque l'on a choisi la différence d'angle entre 2 tangentes,
- Ils sont invariants par changement d'origine, car passer d'une origine A à une origine A' revient à :
  - Retrancher $\emptyset(\boldsymbol{t_{A'}})$ à toutes les valeurs $\emptyset(\boldsymbol{t})$,
  - Changer t en $t' - t_{A'}$.

    Donc : $\emptyset_{A'}(t) = \emptyset_A(t) * \delta(t' - t_{A'})$ et: $a_k \longrightarrow a_k \exp(-kt_{A'})\ si\ k \neq 0$

Pour comparer des formes on compare leurs descripteurs par ordre croissant. Si de plus on veut simplifier le contour. Il suffit de supprimer les ordres k élevés dans le développement. Malheureusement, dans cette représentation, si un contour est fermé, le contour obtenu en filtrant les hautes fréquences ne l'est généralement plus [8]. C'est pourquoi on préfère souvent les descripteurs de Fourier par représentation complexe qui n'ont pas ce défaut.

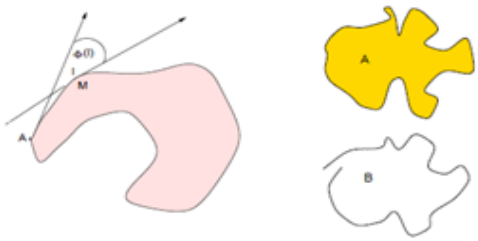


Figure 2.09: *A gauche : descripteur de Fourier par tangente. Le point de départ A est choisi arbitrairement. Son vecteur tangent sert de référence à la paramètrisation de la courbe.*

A droite : après troncature du développement de Fourier, la forme A, initialement fermée devient B, plus régulière, mais non fermée.

b. Représentation complexe:

Dans cette représentation, on décrit la forme par un ensemble $\{M_j\}$ de points de contours, et on représente la forme dans le plan complexe. On attache donc à chaque M_j un nombre complexe $z_j = x_j + iy_j$. On appelle alors descripteurs de Fourier, les coefficients de la transformée en Z :

$$Z_k = \frac{1}{N}\sum_{j=1}^{N} z_j \, exp\,(-2\pi i \frac{jk}{N}) \tag{2.03}$$

Les coefficients Z_k, pour $k \in [-\frac{N}{2}+1, N/2]$, jouissent d'intéressantes propriétés [Bertrand et al, 1982]. Ils possèdent des propriétés suivantes:

- $k = 0, Z_o$ est le centre de gravité de la forme. Si l'on l'omet, la description est invariante par translation.
- Si tous les $Z_k = 0$ sauf pour $k = 1$, la forme est un cercle de rayon Z_1 (ou un polygone régulier à N côtés) donc Z_1 joue le rôle de facteur d'échelle. La normalisation par Z_1 rend la forme invariante par homothétie.
- Les coefficients $Z_{|k|}$ et $Z_{|1-k|}$ (pour $k \neq 0$ et $k \neq 1$) jouent des rôles symétriques (mais opposés) de la façon suivante :
  - L'ordre de k indique le nombre d'actions sur le cercle unité (entre 0 et 2π) :
    - 1 action pour k= 2 et k= - 1,

- 2 actions pour k= 3 et k= - 2,
- 3 actions pour k= 4 et k= - 3, etc.

– Les valeurs de k> 0 indiquent des actions de traction sur la courbe, pour la déformer vers l'extérieur du cercle unité.

– Les valeurs de k< 0 indiquent des actions de pression sur la courbe, pour creuser la courbe vers son centre.

– La phase du nombre complexe Z_k: $\emptyset_k$ exprime le lieu, sur le cercle unité où s'exerce l'action

- Nombre de coefficients :
  – coefficients nombreux⟹ forme complexe
  – coefficients d'ordre élevés ⟹ détails fins sur la courbe.

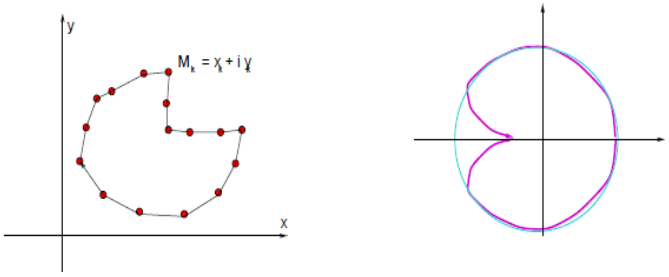


Figure 2.10: *A gauche : descripteur de Fourier par représentation complexe.*

Le point courant $\mathbf{M_k}$ est décrit par ses coordonnées complexes dans le plan image. A droite : le cercle résulte de la prise en compte du seul coefficient $\boldsymbol{Z_1}$, la courbe présentée résulte de l'adjonction d'un coefficient de pression (donc à $< \mathbf{0}$), en 1 seul point (donc $\mathbf{k = 2}$ **ou** $\mathbf{k = 1}, \mathbf{ici\ k = -1}$), de déphasage nul (puisque situé en $\boldsymbol{\pi}$).

On voit que les descripteurs de Fourier par représentation complexe ont le même type de comportement que les descripteurs par tangente. Ils sont mieux adaptés aux formes discrètes puisqu'ils garantissent toujours que la forme demeure fermée après troncature du développement de Fourier. Ils peuvent être invariants par rotation si l'on s'intéresse aux seuls modules des coefficients Z_k.

Parmi leurs inconvénients, il faut noter que l'on ne peut pas aisément garantir qu'un contour de forme simplement connexe ne donnera pas, après troncature, un contour qui s'auto-intersectera.

2.3.2 Approximations polynomiales

Il y a essentiellement 03 types de telles approximations [8][9].

- Les approximations analytiques par ajustements de nuages de points par des polynômes du premier ordre;
- Les approximations polygonales par des critères géométriques obtenues en parcourant la courbe des points ordonnés;
- Les approximations par des polynômes de degré > 1 et en particulier les approximations par des fonctions splines.

Les approximations polygonales des formes décrites par des points de contour sont les représentations les plus employées. Dans les cas les plus simples, on se contente de relier 2 à 2 des points de contour dans un ordre détermine au préalable (donc selon une abscisse curviligne croissante).

Deux problèmes se posent alors :

- Peut-on éliminer d'éventuels points parasites qui n'appartiennent pas au contour?
- Peut-on faire l'économie de certains sommets de la ligne polygonale pour obtenir des formes plus simples ?

Mais avant d'en arriver à cette situation simple, se posent les problèmes difficiles de déterminer quels points appartiennent à un même contour et dans quel ordre les relier lorsque ces points sont issus d'un détecteur de contour nature local. C'est ce que nous abordons tout d'abord.

2.3.2.1 Approximation d'un nuage de points par une droite unique

C'est un problème bien classique mais qui mérite pourtant un peu d'attention. Soit $M_i = (x_i, y_i)$ les points, en nombre N, que l'on cherche à approcher par une droite. Il y a deux façons de prendre le problème.

Approximation par régression linéaire ou approximation par segments de droite:

C'est une approche aux moindres carrés : on recherche le $\Delta: y = a_0 + a_1 x$ qui minimise la distance (voir Figure 2.11)

$$d_1^2 = \sum_i^N [y_i - (a_0 + a_1 x_i)]^2 \tag{2.04}$$

La solution est donnée par :$A = X\#Y = (X^t X)^{-1} X^t Y$, où $X\#$ dénote la matrice pseudo-inverse de la matrice X, où $X, Y\ et\ A$ sont donnés par :

$$\boldsymbol{X} = \begin{bmatrix} \mathbf{1} & \boldsymbol{x_1} \\ .. & .. \\ \mathbf{1} & \boldsymbol{x_N} \end{bmatrix} \tag{2.05}$$

$$\boldsymbol{Y} = [\boldsymbol{y_1}, \boldsymbol{y_2}, .., \boldsymbol{y_N}]^{\boldsymbol{t}}$$

$$\boldsymbol{A} = [\boldsymbol{a_0}, \boldsymbol{a_1}]^{\boldsymbol{t}}$$

Ces formules s'étendent très aisément aux espaces de dimensions supérieures, ainsi qu'aux approximations par des polynômes d'ordre plus élevé. La distance minimisée est celle mesurée selon le seul axe y. C'est donc une mesure généralement mal adaptée en traitement, puisque x et y jouent habituellement un rôle équivalent. On lui préfère donc les méthodes par axe d'inertie.

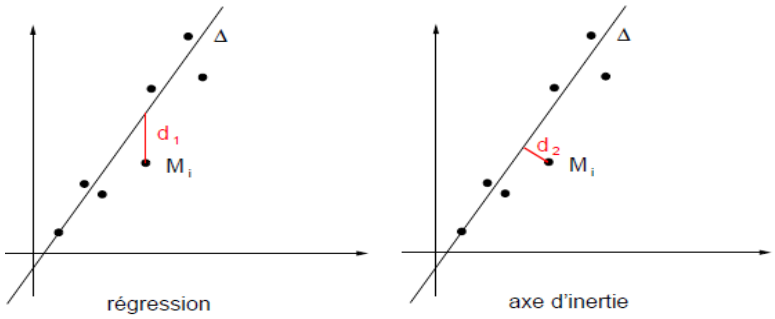


Figure 2.11 : *Deux approximations linéaires aux moindres carrés minimisant des distances différentes : à gauche par régression linéaire, à droite par axe d'inertie.*

Approximation par axe principale d'inertie :

C'est aussi une approche aux moindres carrés, mais on minimise dans ce cas la somme des distances de tous les points à la droite Δ :

$$d_1^2 = \sum_{i=1}^N \frac{[a_0 + a_1 x_i - y_i]^2}{a_1^2 + 1} \tag{2.06}$$

C'est l'équation de l'axe d'inertie des points, qui passe par leur centre de gravité X_g, Y_g et qui est donné comme vecteur propre de plus grande valeur propre de la matrice de forme quadratique :

$$\boldsymbol{S} = \sum_{N} \boldsymbol{V_i V_i^t} = \begin{bmatrix} \sum \boldsymbol{x_i x_i} & \sum \boldsymbol{x_i y_i} \\ \sum \boldsymbol{x_i y_i} & \sum \boldsymbol{y_i y_i} \end{bmatrix} \tag{2.07}$$

Ces équations s'écrivent sans problème en dimensions supérieures (pour estimer des variétés d'ordre variable). Elles se transcrivent beaucoup plus difficilement à des polynômes d'ordre plus grand car on ne sait pas, en règle générale, exprimer la distance d'un point courant à une telle fonction.

Extension de l'estimation à des coniques

Dans le cas où l'on recherche des formes représentées par des cercles ou des ellipses, on peut choisir une paramètrisation matricielle de la forme quadratique les représentant. Chaque conique indicée par i est alors décrite par son centre q_i , son rayon r_i et sa matrice d'ellipticité A_i :

$$[(x - q_i)^t A_i (x - q_i)] = r_i^2 \tag{2.08}$$

Où x est le point courant du plan. La distance d'un point x_k à cette conique s'exprime par :

$$d_i^2(k) = \left[[(x_k - q_i)^t A_i (x - q_i)]^{1/2} - r_i \right]^2 \tag{2.09}$$

2.3.2.2 Approximations polygonales, simplification de contours polygonaux

Lorsque l'on dispose d'une courbe continue ou finement échantillonnée et que l'on souhaite la réduire à une ligne polygonale, on dispose de très nombreux algorithmes qui proposent soit des critères d'approximations soit des mises en œuvre différents. Le plus connu est l'algorithme de la corde

Algorithme de la corde (ou de Ramer)

C'est un processus de subdivision qui peut être entre pris soit de façon récursive (chaque segment crée fait l'objet d'une nouvelle subdivision), soit de façon itérative (la courbe Γ est considérée globalement à chaque étape) [8][9]. C'est cette dernière version que nous examinons sur l'exemple de la figure 2.12. Les sommets du polygone sont choisis successivement comme les

points de Γ les plus éloignés des cordes précédemment tirées. Le processus s'arrête lorsque la nouvelle distance candidate est inférieure à un seuil e fixé. Très employé dans de nombreuses applications par la simplicité de sa mise en œuvre, l'algorithme de la corde n'est pas très rapide. Il ne garantit pas non plus une convergence uniforme vers la courbe finale car, dans son implémentation itérative, il se peut que la distance à l'étape n soit supérieure à celle à l'étape $(n-1)$.

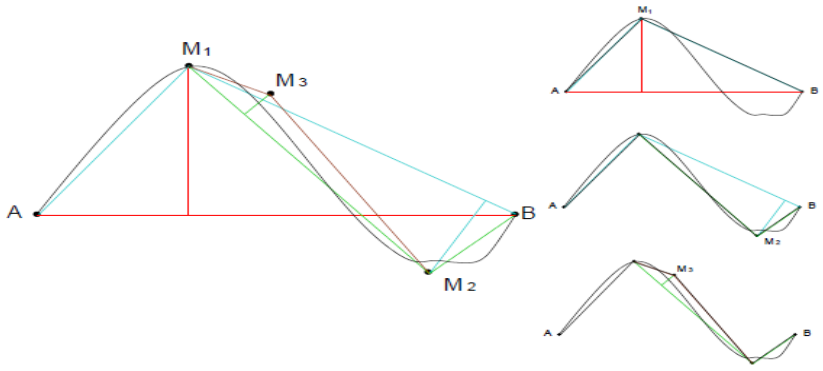


Figure 2.12 : *Algorithme de la corde.*

Les points $\mathbf{M_1}, \mathbf{M_2}$ et $\mathbf{M_3}$ sont successivement sélectionnés pour créer la ligne polygonale représentant la courbe **AB**. Le critère de sélection est la distance maximale à la corde polygonale précédemment obtenue.

2.4 Conclusion

Après avoir expliqué les méthodes pour les descriptions de contours, on a observé que les formes détectées sont surtout obtenus par leurs approximations. Par exemple le code de Freeman nous montre sa limite si au fur à mesure on augmente le nombre de connexité ou voisinage et en plus le calcul est un peu compliqué. Et pour les approximations polynomiales, on rencontre aussi des problèmes sur l'élimination d'éventuels points parasites qui n'appartiennent pas au contour et c'est difficile d'économiser certains sommets de la ligne polygonale pour obtenir des formes simples.

Plutôt que d'approximer les contours par des segments de droite, une approche, dual de la première, consiste à détecter dans une image de contours les appartenant à des segments de droite.

Et on a fait recours à d'autres approches pour pallier ce problème. Comme le but de ce mémoire est détecté des formes géométriques simple comme les droites, les rectangles alors la transformée de Hough s'affiche parmi la méthode plus adaptée.

CHAPITRE 3 : TRANSFORMEE DE HOUGH

3.1 Introduction

La transformée de Hough ou HT est une technique de reconnaissance des formes inventée en 1962 par Paul Hough, utilisée dans le traitement d'images numériques [2][8].

L'application la plus simple permet de reconnaître les lignes d'une image, mais des modifications peuvent être apportées pour reconnaître n'importe quelle forme: c'est la *transformée généralisée de Hough* développée par Richard Duda et Peter Hart en 1972.

Intéressante par son analyse globale de l'image, cette méthode de détection des segments de droite agit sur une image dont on a préalablement extrait les contours. On applique la transformation à chaque point de contour, se plaçant ainsi dans un espace dual, appelé aussi espace des paramètres.

Le principe de la transformée de Hough est qu'il existe un nombre infini de lignes qui passent par un point, dont la seule différence est l'orientation (l'angle). Le but de la transformée est de déterminer lesquelles de ces lignes passent au plus près du schéma attendu.

Afin de déterminer que deux points se trouvent sur une même ligne potentielle, on doit créer une représentation de la ligne qui permet une comparaison dans ce contexte.

Dans la transformée de Hough, dite aussi transformée standard de Hough ou SHT, chaque ligne est un vecteur de coordonnées paramétriques: θ: l'angle, ρ : la norme du vecteur (la longueur du segment perpendiculaire à la droite d'angle θ et passant par l'origine)

En transformant toutes les lignes possibles qui relient un point à un autre, c'est-à-dire en calculant la valeur de ρ pour chaque θ, on obtient une sinusoïde unique appelée *espace de Hough*. Si les courbes associées à deux points se coupent, l'endroit où elles se coupent dans l'espace de Hough correspond aux paramètres d'une droite qui relie ces deux points.

3.2 Propriétés de la transformée de Hough

Dans le plan image J, tout point M est sur la droite $D(\rho, \theta)$ dont les paramètres ρ et θ sont des constantes [10][11].

Dans le plan de Hough (ou espace des paramètres), tout point A de la transformée appartient à une sinusoïde $S(x, y)$ dont les paramètres x et y sont des constantes.

On peut résumer ces propriétés par:

- A un point de (J) correspond une sinusoïde de (H).
- A un point de (H) correspond une droite dans (J).
- Des points de la même droite de (J) donnent des courbes passant par un point commun de (H).
- Des points de la même courbe de (H) donnent des droites passant par un point commun de (J).

Voici la figure qui illustre ces propriétés :

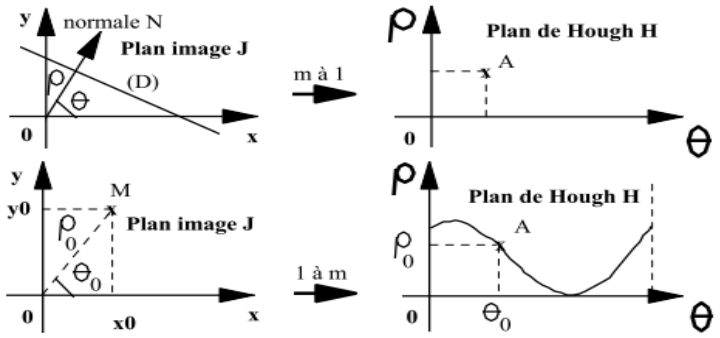


Figure 3.01 : *Propriété de la transformée de Hough*

3.3 Transformée directe

Pour la visualisation de l'image transformée, on choisit de garder le même format que l'image d'origine (256*256) (figure 3.02). Le plan H peut être considéré comme un ensemble de cases qui accumulent les sinusoïdes associées à chaque point caractéristique du plan J.

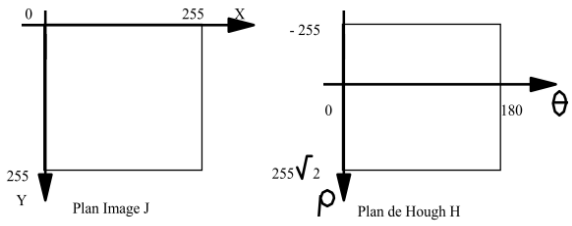


Figure 3.02: *Transformée directe*

3.4 Transformée inverse

La transformée inverse est l'opération de tracer toutes les droites reliant les points alignés détectés dans le plan de Hough. On distingue les étapes suivantes:

Le choix d'un seuil de détection: ce qui représente le nombre de points minimum considérés alignés.

Pour chaque point $(\boldsymbol{\rho}, \boldsymbol{\theta})$ du plan H dépassant le seuil, synthèse de la droite d'équation

$$\boldsymbol{\rho = xcos\theta + ysin\theta}$$

3.5 Algorithme de la transformée de Hough

3.5.1 Pour l'équation de droite de la forme : $y = ax + b$

On traite une image binaire Ic, résultant par exemple d'une détection de contours

- Partitionner l'espace (a, b) sous forme d'un tableau A à 2 dimensions.
- Initialiser le tableau A à 0.
- Pour chaque pixel (x', y') correspondant à un point contour dans l'image Ic, incrémenter toutes les entrées (a, b) de A satisfaisant $b = -x'a + y'$
- Les valeurs élevées de A, par exemple en $(a_0,\ b_0)$ correspondent à l'équation d'une droite $y = a_0\ x\ +\ b_0$ pour laquelle beaucoup de pixels de l'image Ic ont voté.

3.5.2 Exemple

Soit la figure ci-dessous :

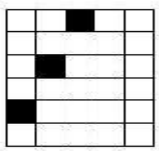


Figure 3.03 : *Les 3 points d'une image bitmap* $\mathbf{5 \times 6}$

La représentation la plus simple d'une ligne est sous forme : $y = ax + b$, où " $\boldsymbol{a}$ " est la pente de la ligne . Par exemple, la figure 3.03 montre les 3 points sur l'image bitmap 5×6. D'après cette image bitmap, on peut définir ces points A(1,2), B(2,4) et C(3,6). Pour chacun de ces points, on peut avoir une famille des lignes qui possèdent la forme: $y = ax + b$. Par conséquent, avec ces trois points dans l'image, A(1,2); B(2,4); C(3,6). On a :

- Pour le point A(1,2):

Pour $y = ax + b$, on change la valeur de x et y par 1 et 2 , et on a: $b = 2 - 1a$

On prend les valeurs entiers ***a*** qui varie de 1 à 5 avec un pas de 1 (car hauteur de l'image bitmap est 5), et les valeurs de b sont obtenues comme suit :

- Quand $a = 1, b = 1$
- Quand $a = 2, b = 0$
- Quand $a = 3, b = -1$
- Quand $a = 4, b = -2$
- Quand $a = 5, b = -3$

Alors on a le tableau de l'accumulateur suivant :

a/b	-6	-5	-4	-3	-2	-1	0	1	2	3
1	0	0	0	0	0	0	0	**1**	0	0
2	0	0	0	0	0	0	**1**	0	0	0
3	0	0	0	0	0	**1**	0	0	0	0
4	0	0	0	0	**1**	0	0	0	0	0
5	0	0	0	**1**	0	0	0	0	0	0

Tableau 3.01: *L'accumulateur pour la droite* $b = 2 - 1a$

Les valeurs de b sont choisit ici arbitrairement. Ces valeurs sont comprises entre -6 et +3

- Pour le point B(2,4) :

Pour $y = ax + b$, on change la valeur de x et y par 2 et 4 , et on a : $b = 4 - 2a$. On fait la même chose pour ce point :

- Quand $a = 1, b = 2$
- Quand $a = 2, b = 0$
- Quand $a = 3, b = -2$
- Quand $a = 4, b = -4$
- Quand $a = 5, b = -6$

Alors on a le tableau de l'accumulateur suivant :

a/b	-6	-5	-4	-3	-2	-1	0	1	2	3
1	0	0	0	0	0	0	0	**1**	**1**	0
2	0	0	0	0	0	0	**2**	0	0	0
3	0	0	0	0	**1**	**1**	0	0	0	0
4	0	0	**1**	0	**1**	0	0	0	0	0
5	**1**	0	0	**1**	0	0	0	0	0	0

Tableau 3.02: *L'accumulateur pour la droite d'équation* $b = 4 - 2a$

Le couple (a, b) = $(2, 0)$ correspond à la valeur 1 dans le tableau 3.01 et on a incrémenté de 1 cette valeur dans le tableau 3.0 afin d'obtenir ce valeur 2. Puisque 2 est la valeur la plus élevée de tous les nombres dans l'accumulateur.

Comme les points A et B donnent le même couple (a, b) qui est égal à (2, 0), donc c'est le pic le plus fort. Alors on peut tracer les lignes reliant entre A et B d'équation: $y = 2x + 0$

- Pour le point C(3,6):

Pour $y = ax + b$, on change la valeur de x et y par 3 et 6, et on a: $b = 6 - 3a$

On fait la même chose pour ce point :

- Quand $a = 1, b = 3$
- Quand $a = 2, b = 0$
- Quand $a = 3, b = -3$
- Quand $a = 4, b = -6$
- Quand $a = 5, b = -9$

Ici $b = -9$ existe sur le tableau (a, b), mais j'avais limité mon tableau entre la valeur -6 et +3.

Alors on a le tableau de l'accumulateur suivant :

a/b	-6	-5	-4	-3	-2	-1	0	1	2	3
1	0	0	0	0	0	0	0	**1**	**1**	**1**
2	0	0	0	0	0	0	**3**	0	0	0
3	0	0	0	**1**	**1**	**1**	0	0	0	0
4	**1**	0	**1**	0	**1**	0	0	0	0	0
5	**1**	0	0	**1**	0	0	0	0	0	0

Tableau 3.03: *L'accumulateur pour la droite* $b = 6 - 3a$

Ici, pour $a = 2, b = 0$, correspond à la valeur 3. On a encore l'équation : $\boldsymbol{y = 2x}$

Inconvénients :

- L'espace des paramètres doit être borné et discrétisé dans une implémentation réelle
- Une droite verticale ne peut pas être représentée ($a = \infty$)

3.5.3 Pour l'équation de droite de la forme : $\boldsymbol{\rho = xcos\theta + ysin\theta}$

3.5.3.1 Mise en œuvre

L'ensemble des droites de paramètres (ρ, θ) passant par le point (x, y) de l'image a pour équation :

$$\rho = xcos\theta + ysin\theta$$

Ceci suggère d'échantillonner le paramètre angulaire θ en valeurs θ_i avec un pas d'échantillonnage Δθ.

Pour chaque point de contour (x, y) de l'image, on calcule pour tous les échantillons θ_i, sa transformée**:** $\rho = xcos\theta_i + ysin\theta_i$ que l'on qualifie en classes**:** $\rho^i \triangleq [\rho_i, \rho_{i+1}]$ de la largeur Δρ

On construit donc un tableau bidimensionnel, constitué de case de dimension Δρ et Δθ, que l'on incrémente afin de comptabiliser les intersections des transformées des points de l'image, intersections qui, on l'a vu, rendent compte de la présence de points colinéaires.

Chaque case (ρ^i, θ_i) du tableau est alors représentative d'un ensemble de points alignés dans une direction perpendiculaire à θ_i, et dont la droite moyenne est repérée à une distance ρ^i à l'origine.

Enfin de traitement, aux éléments du tableau ayant une valeur supérieure à un seuil que l'on fixe en fonction de l'image, correspondent des ensembles de points appartenant effectivement à des segments de droite. Le processus peut donc être interprété comme une élection où chaque point (x_i, y_i) «vote» pour les droites qui le contiennent. La droite réalisant le meilleur score est la droite d'alignement de ces points (s'ils sont bien alignés)

3.5.3.2 L'algorithme

- Appliquer une détection de contours
- Discrétiser le plan des paramètres (ρ, θ)
- Initialiser un accumulateur à 0
- Pour chaque point sur un contour :
  - Déterminer sa droite image dans l'espace de Hough sous la forme de
    $$\rho = x\cos\theta + y\sin\theta$$
  - Incrémenter de 1 l'accumulateur sur les points de cette droite
- Recherche de maxima pour le couple des paramètres (ρ, θ)

3.5.4 Exemple

Soit la figure suivante :

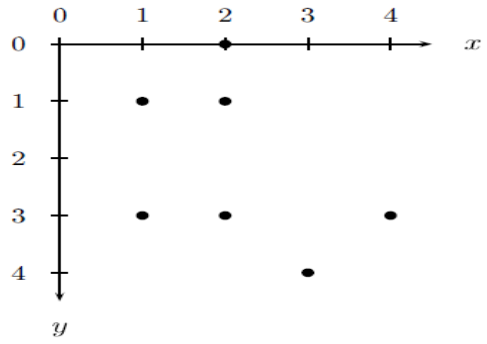


Figure 3.04 : *Un repère orthonormé qui contient six points*

On prend les valeurs entiers de θ variant entre -45°, 0, 45°, 90° et afin d'obtenir tous les valeurs ρ pour chaque point (x, y).

On a le tableau suivant :

(x, y)	*-45°*	*0*	*45°*	*90°*
(2, 0)	1.4	2	1.4	0
(1, 1)	0	1	1.4	1
(2, 1)	0.7	2	2.1	1
(1, 3)	-1.4	1	2.8	3
(2, 3)	-0.7	2	3.5	3
(4, 3)	0.7	4	4.9	3
(3, 4)	-0.7	3	4.9	4

Tableau 3.04: *Un tableau qui illustre tous les valeurs de ρ*

Alors le tableau de l'accumulateur contient les valeurs de (ρ, θ) apparaissent dans le tableau ci-dessus.

θ/ρ	-1.4	-0.7	0	0.7	1	1.4	2	2.1	2.8	3	3.5	4
-45°	1	2	1	2	0	1	0	0	0	0	0	0
0	0	0	0	0	0	2	**3**	0	0	1	0	1
45°	0	0	0	0	0	2	0	1	1	0	1	0
90°	0	0	0	1	2	0	0	0	0	**3**	0	1

Tableau 3.05: *L'accumulateur*

En pratique, on a encore des valeurs plus grandes. Mais dans notre cas, la valeur maximale 3 correspond aux deux coordonnées $(\rho, \theta) = (2, 0°)$ et $(\rho, \theta) = (3, 90°)$. Alors voici les lignes correspondantes : $xcos0 + ysin0 = 2$, ce qui nous donne $x = 2$, de même aussi,

$xcos90 + ysin90 = 3$, alors $y = 3$.

Voici la figure qui illustre ces deux équations dans l'espace d'image :

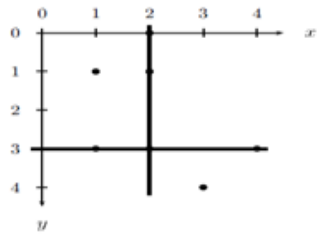


Figure 3.05 : *On a les deux lignes après la Transformée de Hough*

3.6 Les différentes transformées de Hough

3.6.1 La transformée de Hough standard

La transformée de Hough standards (THS) proposé par Paul Hough en 1962 généralement utilisée dans la reconnaissance des formes. La transformée de Hough est largement utilisée en traitement d'images pour détecter les droites, les cercles et les ellipses. Elle comprend deux étapes principales: dans la première, on effectue une transformation des contours de l'image vers une grille ayant habituellement la même résolution que l'image (l'espace des paramètres).Une cellule de l'espace des paramètres est incrémentée d'une unité lorsqu'elle correspond à l'équation de la transformation, pour chaque pixel du contour le vote est performant dans l'espace des paramètres. (Ex. une position possible du centre de l'ellipse).Dans la seconde, on réalise une détection dans cet espace pour localiser la forme recherchée, Le principal inconvénient de la Transformée de Hough Standard (THS) est qu'elle nécessite un énorme espace mémoire. De plus, le nombre d'opérations à effectuer croît exponentiellement avec le nombre de paramètres à détecter, Un autre inconvénient de la THS, est qu'il faut, pour chaque point de contour, incrémenter un ensemble de cellules dans l'espace des paramètres, par exemple les cellules qui correspondent aux centres de toutes les ellipses pouvant passer par ce point. Pour palier à ces problèmes, on notera une intéressante évolution des différentes transformées de Hough.

3.6.2 La transformée de Hough généralisée

La transformée de Hough généralisée est utilisé principalement pour la détection des formes « objet », c'est une méthode robuste au forme altéré complètement ou partiellement et a la présence d'autre objets, il est tolérant aux bruit et capable de détecter plusieurs occurrences avec le même processus. L'inconvenant principal est l'espace mémoire utilisé qui est assez important.

La Transformée de Hough fut tout d'abord introduite pour la détection de droites, puis pour la détection de courbes 2D paramétrisées. Une version généralisée de la Transformée de Hough (GHT) permettant la détection de formes quelconques. Cette méthode crée, lors d'une phase d'apprentissage, une description du contour sous la forme d'une *R-Table.*

3.6.3 La transformée de Hough pondérée

Elle a été proposée par O'Gorman et Clowes pour tenir compte des différences de confiance que l'on porte sur les points M (un pixel donné) de ξ l'ensemble des pixels. Ces points résultent en effet souvent de procédures de détection qui associent à chacun d'eux une note de qualité v_i. La note de qualité est très souvent en fonction de la réponse du point à un détecteur: par exemple l'amplitude d'un gradient pour un détecteur de contours. Principal avantage est qu'elle se prête bien à des implantations rapides, par sa structure parallélisable.

3.7 Avantages et inconvénients

3.7.1 Avantages

- Bonne insensibilité au bruit.
- Elle permet de localiser n'importe quelle courbe qui peut être décrite par une équation analytique.

3.7.2 Inconvénients

- La transformée de Hough nécessite une importante espace de mémoire pour l'exécution
- L'espace des paramètres doit être borné et discrétisé dans une implémentation réelle
- Une droite verticale ne peut pas être représentée dans un espace paramètre

3.8 Conclusion

La transformée de Hough est l'une des méthodes les plus utilisée dans des nombreux domaines que soit dans le traitement du signale ou d'image. Elle offre des méthodes simples à concevoir mais robustes. La transformée de Hough standard permet de détecter des simples formes géométriques comme les droites, cercle et ellipse seulement les paramètres utilisés augments. Par contre la transformée de Hough généralisée permet de détecter des formes altérés complètement ou partiellement et a la présence d'autre objets.

CHAPITRE 4 : APPLICATION DE LA TRANSFORMEE DE HOUGH A LA DETECTION DES RECTANGLES

4.1 Introduction

Ce chapitre présente quelque simulation sous Matlab version 7.8.0 (R2009a) des traitements à effectuer sur la transformée de Hough. Pour cela, on va simuler sur des images de synthèse d'extension .tif, .gif, .jpg.

Tout d'abord on transforme l'image en niveau de gris, ensuite on passe à la détection des points caractéristiques (ou les contours) de l'image. Après la détection de contours, on applique la transformée de Hough sur tous les points caractéristiques trouvés [12][13].

Ce chapitre se divise en 3 parties: la première nous présente un bref détail sur la généralité d'un rectangle, la deuxième nous explique l'approche utilisée pour la détection d'un rectangle, la dernière nous illustre une brève présentation de simulation effectuée sur l'image.

4.2 Hypothèse

L'idée de ce mémoire est de faciliter l'aide à la décision pour la circulation sur une route. On intègre dans un opérateur téléphonique un système de télésurveillance qui commande et traite chaque image. Les images à traiter sont soient captées par des caméras installées sur une route, soient prises par un satellite de surveillance. Chaque portable d'un client de l'operateur peut bénéficier de cette fonction. De ce fait les abonnés peuvent être informés sur l'état de la circulation.

4.3 Généralité sur le rectangle

Un rectangle est une forme géométrique particulière qu'on rencontre dans des nombreuses images numériques. Un rectangle possède une longueur (L) et une largeur (l). Son périmètre est de $2 \times (L + l)$ et sa surface est de $L \times l$.

4.3.1 Propriété d'un rectangle

- Soit un rectangle des sommets A, B, C et D tel que :$\overrightarrow{AB}$ et $\overrightarrow{CD}$ sont colinéaires de côté L, de même aussi pour $\overrightarrow{AD}$ et $\overrightarrow{BC}$ sont colinéaires de côté l, c'est-à-dire :
  $\|\overrightarrow{AB}\| = \|\overrightarrow{CD}\| = L$ et $\|\overrightarrow{AD}\| = \|\overrightarrow{BC}\| = l$.

- Et aussi $\overrightarrow{AB} \perp \overrightarrow{BC}$ *et* $\overrightarrow{DC} \perp \overrightarrow{CB}$ de même pour $\overrightarrow{CD} \perp \overrightarrow{DA}$ *et* $\overrightarrow{CD} \perp \overrightarrow{CB}$.
- On a quatre angles droits tels que :$\widehat{ABC} = \widehat{BCD} = \widehat{CDA} = \widehat{DAB} = 90°$.
- Les deux diagonales sont égales : $\|\overrightarrow{BD}\| = \|\overrightarrow{AC}\|$ et d'après le théorème de Pythagore on a : $(AC)^2 = (AD)^2 + (DC)^2$ et $(BD)^2 = (BC)^2 + (DC)^2$.

4.3.2 Exemple d'un rectangle

Soit le rectangle suivant :

Figure 4.01: *Un rectangle*

4.4 Les différentes étapes pour reconnaître un rectangle dans une image

4.4.1 Détection de contours

Tout d'abord un contour, c'est un changement brusque de l'intensité de l'image. Par définition, c'est la frontière qui sépare deux objets dans une image. Dans notre cas, nous détecterons toutes les lignes marquant des changements d'intensité.

La détection de contours permet de repérer les différents objets qui constituent la scène de l'image. Il existe de nombreuses méthodes pour trouver les contours des objets, la plupart sont basées sur les dérivées premières et secondes de l'image c'est-à-dire du Gradient et du Laplacien.

Voici quelques exemples des contours qu'on rencontre souvent dans un traitement d'image :

- Marche d'escaliers
- Rampe
- Toit

Figure 4.02: *Contours « marche d'escaliers, rampe et toit »*

Problème:

On rencontre des nombreux problèmes sur les contours trouvés dans une image :

- Des contours non fermé
- Des contours fermé qui possèdent des discontinuités
- Des branches pendantes d'un contour fermé
- Le dédoublement de contour.

Tous ces problèmes créent des parasites qui engendrent des fausses droites dans une image.

Solution :

- Pour trouver des rectangles, il faut passer par la suppression des contours non fermé.
- Pour des contours qui possèdent des discontinuités, on procède à la fermeture de contour par extrapolation.
- Et pour le dédoublement des contours, il faut amincir les lignes de contour.

4.4.2 Extraction de contours

Pour palier ces différents problèmes rencontrés sur le contour, voici les étapes qu'il faut mettre en évidence :

- *Lissage préalable pour l'image :*

Pour cela, on utilise le filtre moyenneur, Médian ou encore le filtre de Nagao dans le but de réduire les bruits et d'affiner les zones de transitions (filtre passe-bas).

Figure 4.03 : *Lissage de l'image « Lena.tif ».*

- *La dérivation :*

C'est la détection proprement dite des transitions de l'image (filtre passe-haut). Les bruits occupant des hautes fréquences, la dérivation va amplifier les bruits. (Le Laplacien est encore plus sensible au bruit).

  - *La dérivation du 1^{er} ordre (∇) (Gradient) :*
  - *La dérivation du 2^{nd} ordre (Δ) (Laplacien)*

- *Suppression des discontinuités et amincissement des contours :*
  - *Elimination des non maxima locaux du module du gradient (Gradient)*
  - *Elimination des non « zéros » locaux du module du Laplacien (Laplacien)*
- *Seuillage :*

Ne retenir comme contours que les points de l'image possédant un fort gradient (en module), ou un Laplacien proche de 0 (en module).

  - *Seuillage de l'image « norme de gradient » :*

Sélection des maximas de la norme du gradient pour obtenir des contours extraits.

Figure 4.04 : *Seuillage d'image lissée « Lena.tif » par la norme de gradient*

  - *Passage du zéros par Laplacien (zero cross) :*

Sélection des passages par zéro de la norme du Laplacien (à une tolérance près) afin d'obtenir des contours extraits.

4.4.2.1 La première dérivée d'une image (approche gradient):

La première dérivée de l'image est l'operateur de base pour mesurer les contours dans une image.

$$|\Delta f| = \left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2 \tag{4.01}$$

a. Dérivée discrète

On utilise la première dérivée de l'image pour les contours:

$$\frac{\Delta I}{\Delta x} = \frac{I(x + \Delta x) - I(x)}{\Delta x} \tag{4.02}$$

Approximation simple de la dérivée discrète:$[-1\ 1]\ et\ \begin{bmatrix}-1\\1\end{bmatrix}$ ou encore [−1 0 1] et $\begin{bmatrix}-1\\0\\1\end{bmatrix}$.

b. Filtre de Roberts

En 1965 Roberts fournit une première approximation de la première dérivée d'une image discrète. Le calcul se fait avec deux masques de convolution pour les deux directions de la dérivée.

$$\boldsymbol{G_x = \begin{bmatrix}0 & 1\\-1 & 0\end{bmatrix}\ et G_y = \begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}} \tag{4.03}$$

c. Filtre de Prewitt

On fait le lissage de l'image plus la dérivée de l'image. Voici les masques utilisés:

$$\boldsymbol{\begin{bmatrix}-1 & -1 & -1\\0 & 0 & 0\\1 & 1 & 1\end{bmatrix}\ ou\ \begin{bmatrix}-1 & 0 & 1\\-1 & 0 & 1\\-1 & 0 & 1\end{bmatrix}} \tag{4.04}$$

d. Filtre de Sobel

C'est un filtre moyenneur plus un dérivé d'une image:

$$\boldsymbol{\begin{bmatrix}-1 & -2 & -1\\0 & 0 & 0\\1 & 2 & 1\end{bmatrix}\ ou\ \begin{bmatrix}-1 & 0 & 1\\-2 & 0 & 2\\-1 & 0 & 1\end{bmatrix}} \tag{4.05}$$

4.4.2.2 La seconde dérivée de l'image (approche Laplacien)

Une autre approche pour trouver les contours de l'image est d'utiliser la seconde dérivée de l'image. Pour cela on utilise le Laplacien comme operateur.

$$\nabla^2 I = \frac{\partial I}{\partial x^2} + \frac{\partial I}{\partial y^2} \tag{4.06}$$

Plusieurs approximations discrètes du Laplacien existent :

$$\begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{-4} & \mathbf{1} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \end{bmatrix} \boldsymbol{ou} \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{-8} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \end{bmatrix} \tag{4.07}$$

On utilise une seule matrice d'entre eux pour une matrice de convolution.

4.4.3 Transformation de l'espace image en espace Hough

Comme nous avons mentionné dans la chapitre 3, la transformée de Hough transforme les points caractéristiques dans l'espace image en espace Hough en suivant la formule d'équation :

$\boldsymbol{\rho = xcos\theta + ysin\theta}$.

Prenons par exemple l'image «Sari.jpg». Après l'étape de détection de contours, on utilise la fonction hough.m qui existe déjà sur le logiciel Matlab.

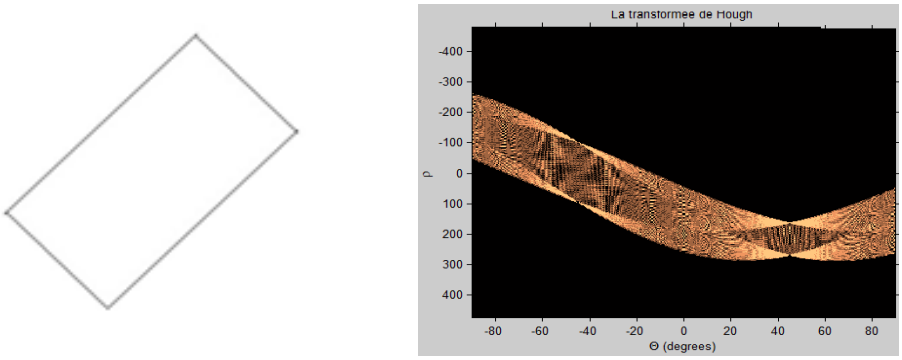


Figure 4.05: *L'image « Sari.jpg »et sa transformée de Hough*

4.4.4 Recherche des pics dans l'accumulateur

Dans cette étape, on cherche tous les pics dans l'espace de Hough, c'est-à-dire, les valeurs parmi les plus grands dans le tableau de l'accumulateur($\boldsymbol{\rho, \theta}$). Pour ce faire, on utilise la fonction houghpeak.m (définie dans Matlab) qui permet de trouver ces valeurs. Le nombre des pics (ou

peaks en anglais) dépend du seuil qu'on veut appliquer.

Voici les pics trouvés sur l'image « Sari.jpg » dans l'espace de Hough avec un seuil égale à $(\mathbf{0.3} \times \mathbf{max}\ (\mathbf{H}(:)))$:

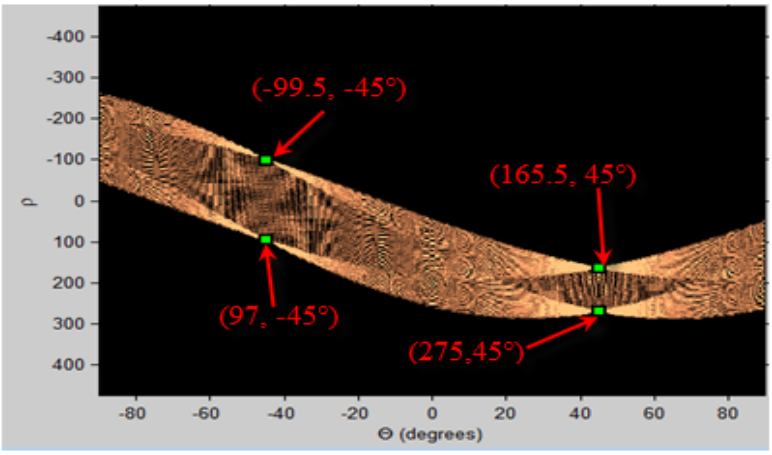


Figure 4.06: *Visualisation des pics en petit carré vert*

Si on le représente sous forme d'un tableau, alors on a :

ρ	θ(dégrée)
275.5	45°
165.5	45°
97	-45°
-99.5	-45°

Tableau 4.01: *Les pics ou votes dans l'espace de Hough*

Les couples (ρ, θ) dans le tableau (4.01) sont choisis en fonction des nombres de votes. Par exemple le couple (275.5, 45°) possède plus de votes par rapport aux autres couples qui le suivent lors de l'élection dans l'espace de Hough ou paramétré.

4.4.5 Recherche des pics qui sont parallèles (pics réguliers)

Soient les sommets d'un rectangle : $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$, $P_3 = (x_3, y_3)$ et $P_4 = (x_4, y_4)$, avec P_1P_2 et P_3P_4 sont en parallèles de côté $\boldsymbol{L}$, et aussi P_2P_3 et P_4P_1 sont en parallèles de côté $\boldsymbol{l}$.

Après la transformée de Hough, on obtient les pics ou coordonnées des points qui ont un maximum d'intersections des sinusoïdes dans l'espace de Hough. Notons les pics par: $H_1 = (\rho_1, \theta_1)$, $H_2 = (\rho_2, \theta_2)$, $H_3 = (\rho_3, \theta_3)$ et $H_4 = (\rho_4, \theta_4)$.

Voici l'algorithme pour détecter un rectangle :

a. Soient les paires: la 1ere paire est formée par les pics H_1 et H_2 avec $\theta = \propto_1$, la 2nd formée par H_3 et H_4 avec $\theta = \propto_0$.
b. Les deux paires sont séparés par $\Delta\theta = 90°$ dans l'axe θ c'est-à-dire $| \propto_1 - \propto_0 | = 90°$
c. La longueur des deux pics appartenant à la même paire sont identiques et représente la longueur de ligne de segment respective c'est-à-dire $C(\rho_1, \theta_1) = C(\rho_2, \theta_2) = L$ et $C(\rho_3, \theta_3) = C(\rho_4, \theta_4) = l$
d. La distance verticale (dans l'axe de ρ) entre deux pics dans chaque paire sont exactement les côtés du rectangle c'est-à-dire $\rho_1 - \rho_2 = l$ et $\rho_3 - \rho_4 = L$.

Donc si on le généralise, on a les conditions suivantes :

Soit $H_1 = (\rho_1, \theta_1), H_2 = (\rho_2, \theta_2),\ H_3 = (\rho_3, \theta_3), \dots, H_m = (\rho_m, \theta_m)$, on les note les m pics de $C(\rho, \theta)$. L'étape suivante est de chercher les quatre pics qui satisfont les conditions listées ci-dessus. Dans le but, tous les pics sont parcourus, et les pics $H_i\ et\ H_j$ sont pairs s'ils satisfont aux conditions suivantes :

$$\Delta\theta = \left|\theta_i - \theta_j\right| < T_\theta, \tag{4.08}$$

$$\left|C(\rho_i, \theta_i) - C\left(\rho_j, \theta_j\right)\right| < T_L \frac{C(\rho_i, \theta_i) + C\left(\rho_j, \theta_j\right)}{2}$$

Où T_θ: C'est le seuil angulaire qui détermine si les segments de lignes correspondant à $H_i\ et\ H_j$ sont parallèles (c'est-à-dire $\theta_i \approx \theta_j$).

T_L: C'est un seuil normal qui détermine si les segments de ligne correspondant à $H_i\ et\ H_j$ ont approximativement la même longueur (c'est-à-dire $C(\rho_i, \theta_i) \approx C\left(\rho_j, \theta_j\right)$).

4.4.6 Recherche des pics prolongés (le pic qui correspond à un rectangle)

Pour chaque paire de $H_i\ et\ H_j$ qui satisfait l'équation (4.08), on génère un pic prolongé:

$$P_k = (\pm\xi_k, \alpha_k), \tag{4.09}$$

Où $\alpha_k = \frac{1}{2}(\theta_i + \theta_j)$ et $\xi_k = \frac{1}{2}\,|\rho_i - \rho_j|$.

Il est noté que P_k code les informations sur les pics $H_i\ et\ H_j$ car $\theta_i \approx \alpha_k, \theta_j \approx \alpha_k, \rho_i \approx -\xi_k$ et $\rho_j \approx \xi_k$.

L'étape final de la technique est de comparer tous les pics prolongés P_k et P_l, de rechercher ceux qui correspondent aux paires orthogonaux de lignes parallèles (par conséquent, relié à un rectangle). Un rectangle est alors détecté si :

$$\Delta\alpha = |\,|\,\alpha_k - \alpha_l| - 90°\,| < T_\alpha \tag{4.10}$$

Où T_α: C'est le seuil angulaire qui détermine si les paires de lignes P_k et P_l sont orthogonaux.

En conclusion, les sommets du rectangle détecté sont obtenus par l'intersection des deux paires de lignes parallèles. L'orientation du rectangle détecté est donnée par α_k et les côtés sont données par $2\xi_k$ et $2\xi_l$.

Après avoir trouvé les pics réguliers, on applique l'équation (4.09) et (4.10). On trouve à la fin un pic prolongé qui satisfait l'équation (4.10) et on peut en déduire qu'il y a un rectangle sur l'image «Sari.jpg».

Voici les pics prolongés:

ρ	θ(dégrée)
53	45
98.25	- 45

Tableau 4.02: *Les valeurs des pics réguliers de l'image «Sari.jpg»*

Donc ici on a un rectangle.

4.4.7 Détection d'une droite

Pour détecter une droite dans une image, plusieurs méthodes peuvent être appliquées, mais il faut savoir choisir laquelle est la bonne. En effet la méthode que j'ai choisi est la fonction de la transformée de Hough. Dans cette méthode chaque point dans l'espace image correspond à une sinusoïde dans l'espace paramétré. Donc on transforme chaque point caractéristique dans l'espace

image en sinusoïde dans l'espace paramétré. Pour la transformation, on utilise l'équation de la droite : $\rho = xcos\theta + ysin\theta$

Le coordonnée du point A dans l'espace image est de ***(5, 5)***, on transforme ce point dans l'espace de Hough :

$x = 5,\ y = 5\ alors\ \rho = 5cos\theta + 5sin\theta$ où $\theta \in [0, 360°]$

Après on trace cet équation pour obtenir la sinusoïde dans l'espace de Hough (voir la Figure 4.10)

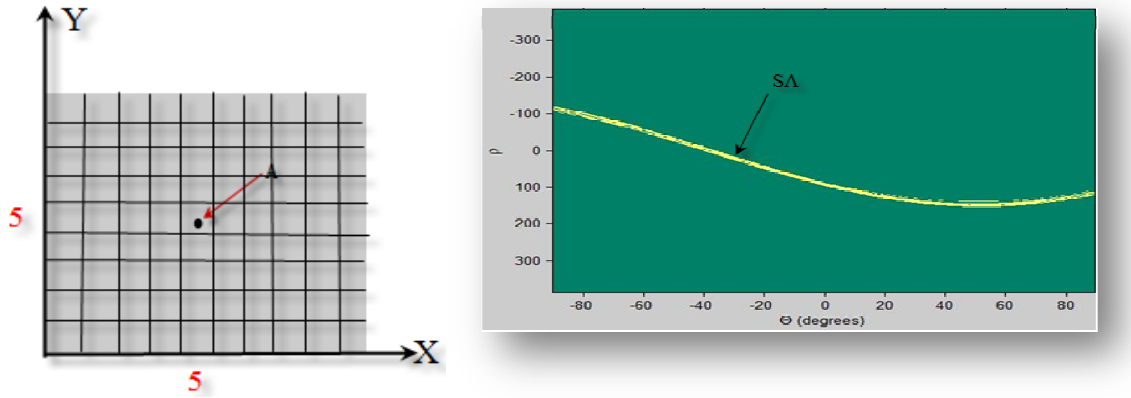


Figure 4.07: *Un point caractéristique dans l'espace image et sa transformée dans l'espace de Hough*

Donc si on augmente le nombre des points dans l'espace image, on observe que les sinusoïdes dans l'espace de Hough aussi augmentent en même temps. Par exemple prenons 3 trois points dans l'espace image:

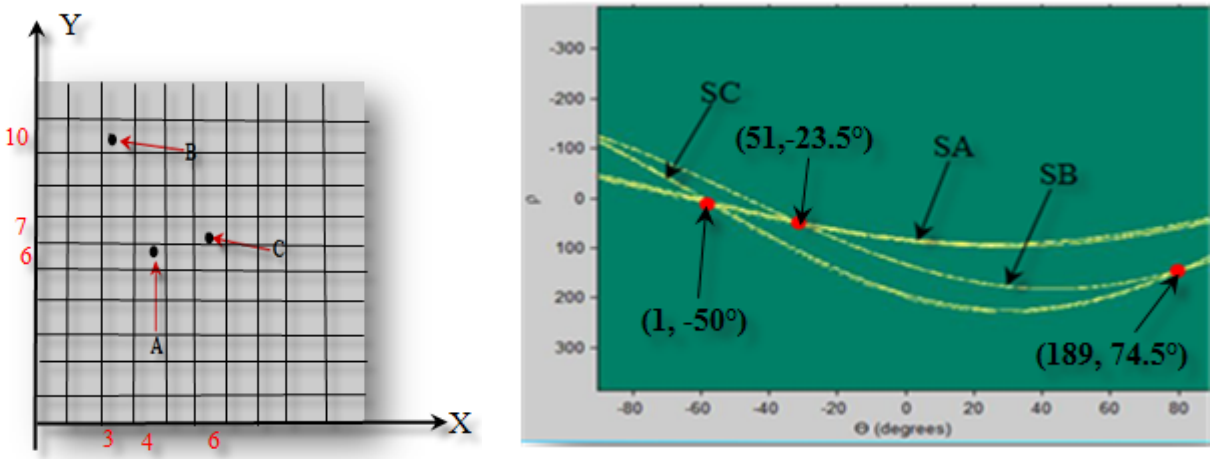


Figure 4.08: *Les 3 points et leurs transformations dans l'espace de Hough*

Dans l'espace image, on a les points A(4, 6), B(3, 10) et C(6,7), après leurs transformation dans l'espace de Hough, on trouve les sinusoïdes SA, SB et SC.

- *Pour le point A:*

$x = 4,\ y = 6\ \ alors\ \ \rho = 4cos\theta + 6sin\theta\ o\grave{u}\ \theta \in [0, 360°]$

- *Pour le point B:*

$x = 3,\ y = 10\ alors\ \ \rho = 3cos\theta + 10sin\theta\ o\grave{u}\ \theta \in [0, 360°]$

- *Pour le point C:*

$x = 6,\ y = 7\ alors\ \ \rho = 6cos\theta + 7sin\theta\ o\grave{u}\ \theta \in [0, 360°]$

Dans l'espace de Hough, on constate qu'il y a trois points d'intersections (en rouge) où chaque point d'intersection correspond à une droite. On appelle ces points d'intersections « pics ». On prend les coordonnées de chaque pic. Voici les coordonnées en fonction de (ρ, θ).

ρ	θ(en dégrée)
1	-50°
51	-23.5°
189	74.5°

Tableau 4.03: *Les pics*

- *Pour* (ρ, θ) *= (1, -50°) :*

$$1 = xcos(-50°) + ysin(-50°) \Longrightarrow y = 0.8x - 1.3$$

- *Pour* (ρ, θ) *= (51, -23.5°) :*

$$51 = xcos(-23.5°) + ysin(-23.5°) \Longrightarrow y = 2.25x - 127.5$$

- *Pour* (ρ, θ) *= (189, 74.5°) :*

$$189 = xcos(74.5°) + ysin(74.5°) \Longrightarrow y = -0.5x + 189$$

Après avoir trouvé ces trois équations de droites, on peut les retracer dans l'espace image.

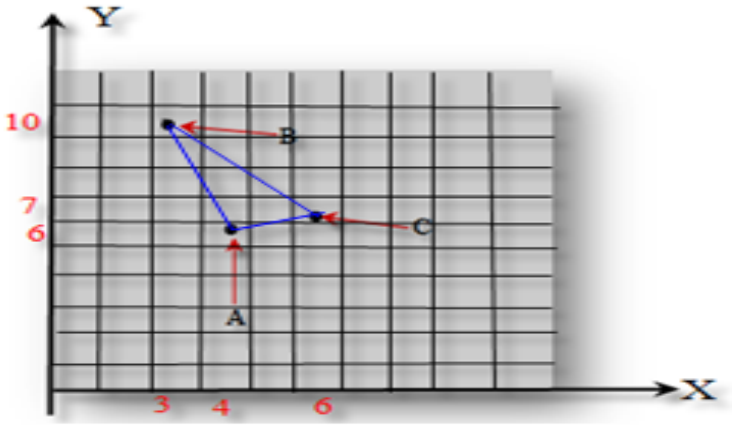


Figure 4.09: *Les droites trouvées et retracées dans l'espace d'image*

Géométriquement, une droite c'est une entité géométrique composée d'un nombre infini de point alignés. Prenons l'exemple d'image synthèse qui contient deux droites.

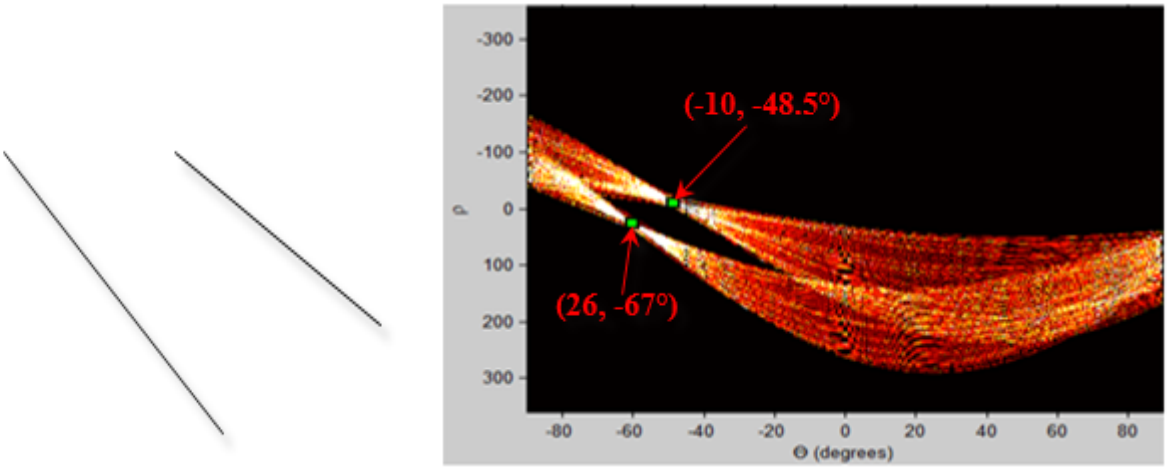


Figure 4.10: *Les deux droites et leurs transformations dans l'espace de Hough*

Dans l'espace de Hough, on observe les deux pics (en vert) qui représentent les deux droites dans l'espace image. Voici les coordonnées de ces deux pics respectifs :

ρ	θ(en dégrée)
-10	-48.5°
26	-67°

Tableau 4.04: *Les deux pics*

Alors les droites correspondantes sont :

(ρ, θ)	$droites$
(-10, -48.5°)	$y = 0.9\ x + 13.5$
(26, -67°)	$y = 0.42\ x - 28.26$

Tableau 4.05: *Les droites correspondantes*

Maintenant on retrace les droites dans l'espace image et on obtient la figure ci-dessous.

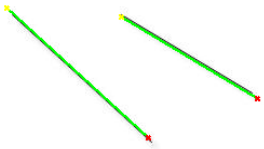


Figure 4.11: *Les droites retrouvées*

4.4.8 Détection des droites qui sont parallèles

Dans cette partie, on joue dans l'espace paramétré pour comprendre si deux droites sont parallèles ou non. Après avoir su comment faire pour trouver des droites dans l'image, on procède la même méthode pour la détection des droites parallèles mais en ajoutant quelques fonctions pour spécifier les droites qui sont parallèles.

Pour plus de précision, prenons par exemple une image de synthèse qui contient deux droites qui sont parallèles.

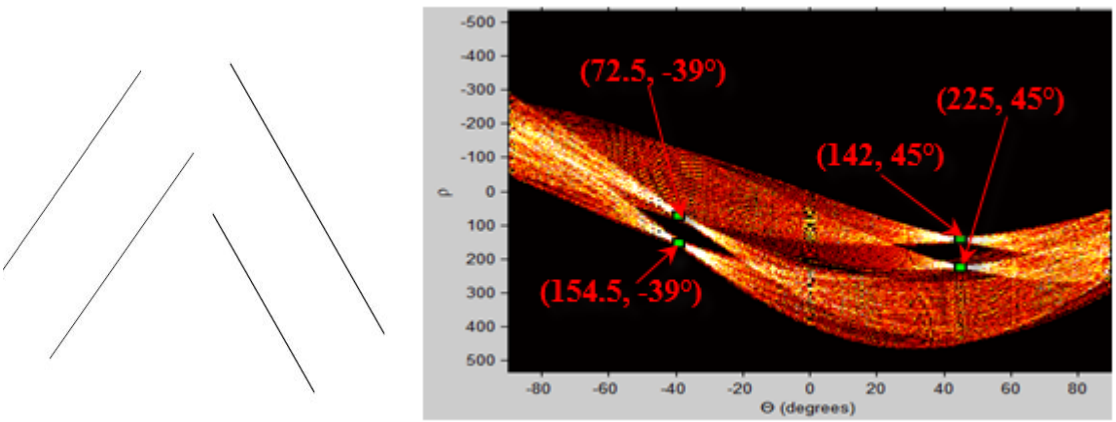


Figure 4.12 : *Les droites parallèles et leurs transformations*

En utilisant la formule (4.08), qui détermine si les segments de lignes correspondant à $H_i \; et \; H_j$ sont parallèles (c'est-à-dire $\theta_i \approx \theta_j$).

Voici les pics trouvés dans l'espace de Hough :

ρ	θ(en dégrée)
225	45
142	45
154.5	-39
72.5	-39

Tableau 4.06: *Les pics correspondants aux quatre droites*

Maintenant on cherche parmi ces pics lesquels sont en parallèles, on fait alors la soustraction deux à deux de l'angle θ pour vérifier s'il satisfait à la formule (4.08).

Ici on a quatre angles, qui sont: 45°, 45°, -39°, -39°. Prenons par exemple le seuil angulaire « $T_\theta = 1$ »

$$\Delta\theta = |\theta_i - \theta_j| < T_\theta$$

- $\Delta\theta 1 = |45 - 45| =$
  $0, alors \; les \; pics \; (\rho 1, \theta 1) \; et \; (\rho 2, \theta 2) \; sont \; parallèles \; c'est$ à $dire \; \alpha 1 = 45°$
- $\Delta\theta 2 = |45 - 39| =$
  $6, alors \; les \; pics \; (\rho 1, \theta 1) \; et \; (\rho 3, \theta 3) \; ne \; sont \; pas \; parallèles \; car \; \Delta\theta 2 > T_\theta$
- $\Delta\theta 3 = |45 - 39| =$
  $6, alors \; les \; pics \; (\rho 1, \theta 1) \; et \; (\rho 4, \theta 4) \; ne \; sont \; pas \; parallèles \; car \; \Delta\theta 3 > T_\theta$
- $\Delta\theta 4 = |45 - 39| =$
  $6, alors \; les \; pics \; (\rho 2, \theta 2) \; et \; (\rho 3, \theta 3) \; ne \; sont \; pas \; parallèles \; car \; \Delta\theta 4 > T_\theta$
- $\Delta\theta 5 = |45 - 39| =$
  $6, alors \; les \; pics \; (\rho 2, \theta 2) \; et \; (\rho 4, \theta 4) \; ne \; sont \; pas \; parallèles \; car \; \Delta\theta 5 > T_\theta$
- $\Delta\theta 6 = |39 - 39| =$
  $0, alors \; les \; pics \; (\rho 3, \theta 3) \; et \; (\rho 4, \theta 4) \; sont \; parallèles \; c'est$ à $dire \; \alpha 2 = 39°$

Donc on a deux pairs de pic qui sont en parallèles, ce qui nous amène à conclure qu'on a trouvé deux droites qui sont parallèles.

4.4.9 Détection des droites qui sont perpendiculaires

Les procédés sont presque les mêmes mais à la différence, on applique la formule (4.10). Deux droites sont perpendiculaires si la différence entre $\theta i \; et \; \theta j$ dans l'espace de Hough sont aux alentours de valeur de 90°.

Prenons l'exemple ci-dessous pour plus de détails :

ρ	θ(en dégrée)
9	-45°
319	26.5°
20.5000	-65.5°
159.0000	45°

Tableau 4.07: *Les pics trouvés*

On applique la formule (4.10) a tous les pics dans le tableau (4.07), posons le seuil angulaire « $T_\alpha = 3$ » , voici la formule :

$$\Delta\alpha = ||\alpha_k - \alpha_l| - 90°| < T_\alpha$$

- $\Delta\alpha 1 = ||-45-26.5| - 90°| = 18.5, alors \; (\rho 1, \theta 1) \; et \; (\rho 2, \theta 2) \; ne \; sont \; pas \; perpendiculaires \; car \; \Delta\alpha 1 > T_\alpha$
- $\Delta\alpha 2 = ||-45+65.5| - 90°| = 69.5, alors \; (\rho 1, \theta 1) \; et \; (\rho 3, \theta 3) \; ne \; sont \; pas \; perpendiculaires \; car \; \Delta\alpha 2 > T_\alpha$
- $\Delta\alpha 3 = ||-45-45| - 90°| = 0, alors \; (\rho 1, \theta 1) \; et \; (\rho 4, \theta 4) \; sont \; perpendiculaires$
- $\Delta\alpha 4 = ||26.5+65.5| - 90°| = 2, alors \; (\rho 2, \theta 2) \; et \; (\rho 3, \theta 3) \; \; sont \; perpendiculaires$
- $\Delta\alpha 5 = ||26.5-45| - 90°| = 71.5, alors \; (\rho 2, \theta 2) \; et \; (\rho 4, \theta 4) \; ne \; sont \; pas \; perpendiculaires \; car \; \Delta\alpha 5 > T_\alpha$
- $\Delta\alpha 6 = ||-65.5-45| - 90°| = 20.5, alors \; (\rho 1, \theta 1) \; et \; (\rho 2, \theta 2) \; ne \; sont \; pas \; perpendiculaires \; car \; \Delta\alpha 6 > T_\alpha$

Donc on a deux pairs de pics qui satisfont les conditions de la formule (4.10), ce qui nous a permet de déduire qu'on a trouvé deux droites qui sont perpendicualires.

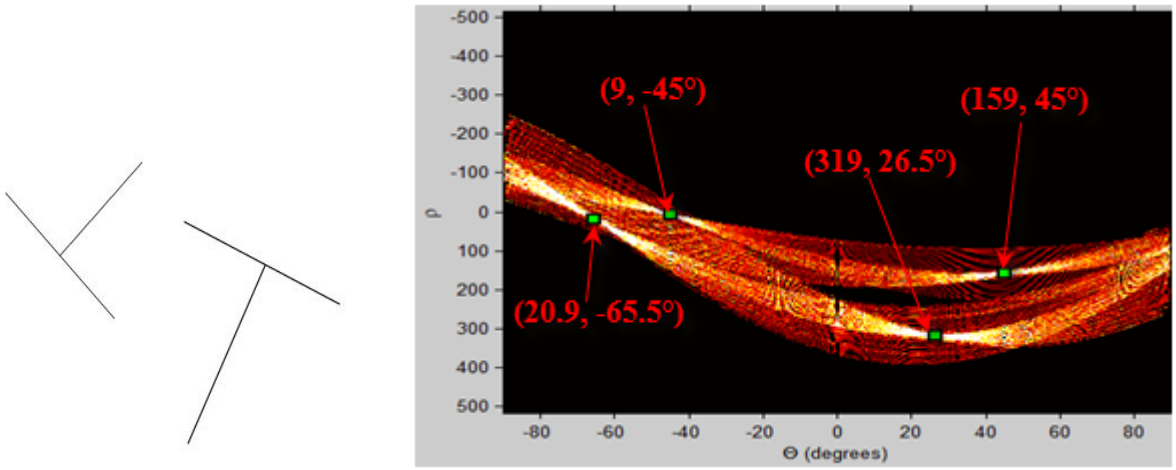


Figure 4.13 : *Les droites perpendiculaires et leurs transformations dans l'espace de Hough*

4.5 Applications «détection d'un rectangle»

Après avoir énuméré l'approche qu'on va appliquer pour détecter un rectangle, passons maintenant à la pratique.

Dans cette partie, on fait l'assemblage des ces trois procédures de détection ci-dessus et on doit chercher les pics prolongées.

Les pics prolongés, ce sont des associations deux à deux des pics réguliers pour n'en former qu'un seul pic qui satisfait la condition donnée. Pour trouver des rectangles dans une image quelconque, il faut toujours commencer par détecter tous les droites ; droites qui sont parallèles et droites qui sont perpendiculaires.

Voici un exemple pour compléter l'explication de tous ce qui est dit:

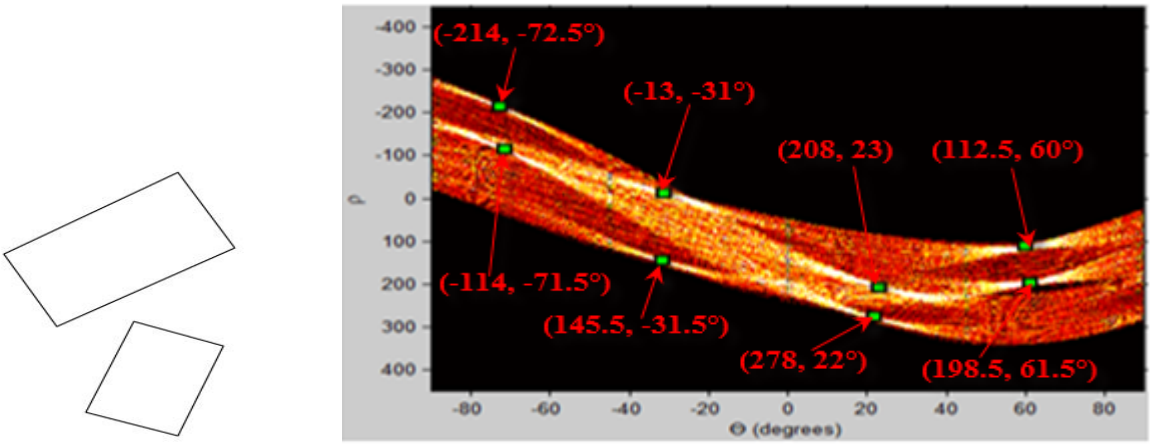


Figure 4.14: *Les deux rectangles et leurs transformations dans l'espace de Hough*

Dans l'espace de Hough, on observe huit pics qui ont été choisi comme étant des droites dans l'espace image. Voici les pics correspondants :

ρ	θ(en dégrée)
112.5	60
198.5	61.5
208	23
-114	-71.5
278	22
-13	-31
145.5	-31.5
-214	-72.5

Tableau 4.08: *Les pics qui correspondent à chaque droite*

Ces pics sont parallèles et perpendiculaires entre eux. On cherche les droites, droites parallèles et perpendiculaires de la même façon qu'on avait déjà abordée auparavant.

Maintenant il faut comparer deux à deux le couple de (ρ, θ) dans le tableau 4.08 pour avoir les pics prolongés. Pour trouver ces pics prolongés, on applique la formule (4.09).

$$P_k = (\pm\xi_k, \alpha_k),$$

Où $\alpha_k = \frac{1}{2}(\theta_i + \theta_j)$ et $\xi_k = \frac{1}{2}\,|\rho_i - \rho_j|$.

Voici les pics prolongés :

$\boldsymbol{\xi}$	$\boldsymbol{\alpha}$
43	60.75
35	22.5
79.25	-31.25
50	-72

Tableau 4.09: *Les pics prolongés*

Donc pour chaque couple de $(\boldsymbol{\xi}, \boldsymbol{\alpha})$ dans les pics prolongés correspondent deux droites qui sont parallèles. Par exemple le couple $(\boldsymbol{\xi}, \boldsymbol{\alpha}) = (43,\ 60.75)$ correspond à deux droites qui sont parallèles et de même pour les trois autres pics.

Et après, on applique la formule 4.10 qui est :

$$\Delta\alpha = |\,|\,\alpha_k - \alpha_l| - 90°\,| < T_\alpha$$

On teste cette formule sur les pics prolongés deux à deux. Donc si le test satisfait cette condition, on peut en conclure qu'on a trouvé un rectangle. Par exemple, on fixe le seuil angulaire « $T_\alpha = 8$ ». Plus précisément, on teste l'orthogonalité de chaque couple du pic prolongé

- $\Delta\alpha 1 = |\,|\,60.75 - 22.5| - 90°\,| = 51.75$

Alors les couples $(\xi 1, \alpha 1)$ et $(\xi 2, \alpha 2)$ ne sont pas orthogonaux donc ne forme pas une rectangle.

- $\Delta\alpha 2 = |\,|\,60.75 + 31.25| - 90°\,| = 2$

Comme $\Delta\alpha 2 < T_\alpha$, les couples $(\xi 1, \alpha 1)$ et $(\xi 3, \alpha 3)$ satisfait la condition (4.10), donc on a un rectangle .

- $\Delta\alpha 3 = |\,|\,60.75 + 72| - 90°\,| = 42.75$

Alors les couples $(\xi 1, \alpha 1)$ et $(\xi 4, \alpha 4)$ ne sont pas orthogonaux donc ne forme pas un rectangle.

- $\Delta\alpha 4 = ||\, 22.5 + 31.25| - 90°| = 36.25$

Alors les couples $(\xi 2, \alpha 2)$ et $(\xi 3, \alpha 3)$ ne sont pas orthogonaux donc ne forme pas un rectangle.

- $\Delta\alpha 5 = ||\, 22.5 + 72| - 90°| = 4.5$

Comme $\Delta\alpha 5 < T_\alpha$, les couples $(\xi 2, \alpha 2)$ et $(\xi 4, \alpha 4)$ satisfait la condition (4.10), donc on a un rectangle .

- $\Delta\alpha 6 = ||\, -31.25 + 72| - 90°| = 49.25$

Alors les couples $(\xi 3, \alpha 3)$ et $(\xi 4, \alpha 4)$ ne sont pas orthogonaux donc ne forme pas un rectangle.

Dans cette explication on a remarqué deux pairs de couple qui ont satisfait la condition (4.10) qui sont les couples « $(\xi 1, \alpha 1)$ et $(\xi 3, \alpha 3)$ » et « $(\xi 2, \alpha 2)$ et $(\xi 4, \alpha 4)$ ».

D'où on a deux rectangles.

Voici les deux rectangles obtenus :

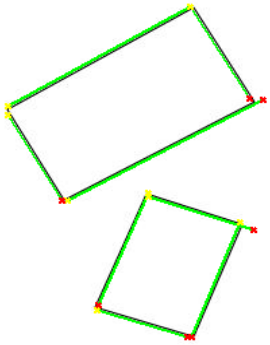


Figure 4.15 : *Les rectangles obtenus*

La détection de forme dans une image s'applique dans des nombreux domaines. La méthode qu'on avait utilisée ici est robuste et facile à intégrer mais l'inconvénient se situe sur le prétraitement d'image, plus précisément la façon d'avoir un contour sans parasite.

Pour cela des nombreuses techniques des prétraitements d'image et de la détection de contours avaient été appliqué pour éliminer des points parasites.

Les applications de cette méthode fonctionnent avec certaines hypothèses.

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.

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 le chapitre 2, on a essayé de trouver des méthodes adéquates pour trouver une droite dans une image. Citons par exemple, le code de Freeman, les approximations polynomiales.

Dans la dernière partie de ce travail, on a traité la détection de contours et on a crée une fonction pour déduire le résultat de la simulation pour chaque fenêtre du Menus sous MATLAB ainsi que le fameux transformée de Hough. 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 la détection des rectangles dans une image de synthèse.

ANNEXE 1 : *Les fonctions de base*

- **Lecture d'une image :**

I=imread('Capture4.jpg');

La fonction imread sert pour afficher l'image en question

- **Détection de contours :**

rgb=rgb2gray(I);// cette fonction « rgb2gray » transforme l'image RGB en gray ou niveau de gris c'est-à-dire des pixels compris entre 0 à 255.

Ca=edge(rgb,'canny');// la fonction « edge » sert pour détecter les points caractéristiques ou le changement brusque de l'intensité de l'image.

- **Transformée de Hough**

[H,Theta,rho]=hough(Ca,'RhoResolution',0.5,'ThetaResolution',0.5);// la fonction hough existe déjà dans le toolbox du Matlab. Cette fonction sert à transformer l'espace image en espace paramétré c'est-à-dire il transforme chaque pixel de l'image dans l'espace image en sinusoïde dans l'espace paramétré.

- **Détection des votes ou peaks**

P=houghpeaks(H,numpeaks,'threshold',ceil(0.37*max(H(:))));
x=Theta(P(:,2));
y=rho(P(:,1));

- **Détections des lignes dans une image**

lines = houghlines(Ca,Theta,rho,P,'FillGap',20,'MinLength',40);// cette fonction houghlines permet de retracer les lignes des droites trouvées.

ANNEXE 2 : *Quelques codes de l'interface de simulation*

Détection d'un rectangle :

```
function varargout = realisation(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @realisation_OpeningFcn, ...
                   'gui_OutputFcn',  @realisation_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
function realisation_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = realisation_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function choix_Callback(hObject, eventdata, handles)
axes(handles.axes1)
cla;
choix_image=get(handles.choix,'value');
img=[];
switch choix_image
   case 1
      %sary1.jpg
```

```
        A=imread('Capture1.jpg');
        imshow(A)
        img=A;
    case 2
        %sary2.jpg
        B=imread('Capture2.jpg');
        imshow(B)
        img=B;
    case 3
        %sary3.jpg
        C=imread('Capture3.jpg');
        imshow(C)
        img=C;
    case 4
        %sary4.jpg
        D=imread('Capture4.jpg');
        imshow(D)
        img=D;
end
save image img
function choix_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function valiny_Callback(hObject, eventdata, handles)
function valiny_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function STx_Callback(hObject, eventdata, handles)
```

```
function STx_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function STa_Callback(hObject, eventdata, handles)
function STa_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function Contour_Callback(hObject, eventdata, handles)
load image
INg=img;
RGB=rgb2gray(INg);
I2=im2bw(RGB);
Icontour=edge(I2,'Canny');
axes(handles.axes1);
cla;
imshow(Icontour)
save Imagecontour Icontour
function Max_Callback(hObject, eventdata, handles)
function Max_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function TH_Callback(hObject, eventdata, handles)
load image
INg=img;
RGB=rgb2gray(INg);
Icontour=edge(RGB,'Canny');
```

```
[H,Theta,rho] = hough(Icontour,'RhoResolution',0.5,'ThetaResolution',0.5);
axes(handles.axes2)
cla;
imshow(imadjust(mat2gray(H)),[],'XData',Theta,'YData',rho,...
      'InitialMagnification','fit');
xlabel('\Theta (degrees)'), ylabel('\rho');
axis on , axis normal,hold on
colormap(copper)
numpeaks=str2num(get(handles.Num,'String'));
Chiffre=str2num(get(handles.Max,'String'));
P = houghpeaks(H,numpeaks,'threshold',ceil(Chiffre*max(H(:))));
x = Theta(P(:,2));
y = rho(P(:,1));
plot(x,y,'rs','LineWidth',2,...
            'MarkerEdgeColor','k',...
            'MarkerFaceColor','g',...
            'MarkerSize',8)

function Pr_Callback(hObject, eventdata, handles)
load image
INg=img;
RGB=rgb2gray(INg);
Icontour=edge(RGB,'Canny');
[H,Theta,rho] = hough(Icontour,'RhoResolution',0.5,'ThetaResolution',0.5);
axes(handles.axes2)
cla;
imshow(imadjust(mat2gray(H)),[],'XData',Theta,'YData',rho,...
      'InitialMagnification','fit');
xlabel('\Theta (degrees)'), ylabel('\rho');
axis on , axis normal,hold on
colormap(copper)
numpeaks=str2num(get(handles.Num,'String'));
```

```
Chiffre=str2num(get(handles.Max,'String'));
P = houghpeaks(H,numpeaks,'threshold',ceil(Chiffre*max(H(:))));
x = Theta(P(:,2));
y = rho(P(:,1));
plot(x,y,'rs','LineWidth',2,...
             'MarkerEdgeColor','k',...
             'MarkerFaceColor','g',...
             'MarkerSize',8)
for i=1:length(x)
   thetas=x(1,i);
   rhos= y(1,i);
   if ~isinf(thetas)& ~isinf(rhos)
       Pr(i,1)=rhos;
       Pr(i,2)=thetas;
   end
end
f = figure;
colnames = {'Rho', 'Theta'};
t = uitable(f, 'Data', Pr, 'ColumnName', colnames, ...
                'Position', [100 4 200 300]);
save peaksreguliere Pr
function Pp_Callback(hObject, eventdata, handles)
load peaksreguliere
Tx=str2num(get(handles.STx,'String'));
for j=1:length(Pr)
   for l=j:length(Pr);
      if l==j
         continue
      end
      deltaTheta=abs(Pr(j,2)-Pr(l,2))
      if (deltaTheta<Tx)
         alpha=(1/2)*(Pr(j,2)+Pr(l,2));
```

```
            sigma=(1/2)*abs(Pr(j,1)-Pr(l,1));
            P1(:,:,l)=[sigma, alpha];
            P2(l,:)=P1(:,:,l);
        end
    end
end
P2
[x2,y2]=size(P2);
for k=1:x2
    while([x2,y2]==size(P2))
        if (P2(k,:)==[0 0])
            P2(k,:)=[];
            [x2,y2]=size(P2);
        else
            P2
            k=k+1
            save P2
            break
        end
    end
    if (k>=x2)
        break
    end
end
load P2
Pp=P2;
f = figure;
colnames = {'Rho', 'Theta'};
r = uitable(f, 'Data', Pp, 'ColumnName', colnames, ...
                'Position', [100 4 200 300]);
save peaksprolongees P2
function Comptagerectangle_Callback(hObject, eventdata, handles)
```

```
load peaksprolongees
Pt=P2;
Talpha=str2num(get(handles.STa,'String'));
compt=0;
for k=1:length(Pt)
   for n=k:length(Pt)
      if (n==k)
         continue
      end

      deltaAlpha=abs(abs(Pt(k,2)-Pt(n,2))-90);
      if (deltaAlpha < Talpha)
         compt=compt+1;
     end
   end
end
set(handles.comptage,'String',compt)
function Num_Callback(hObject, eventdata, handles)
function Num_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function rectObt_Callback(hObject, eventdata, handles)
load Imagecontour
Irect=Icontour;
axes(handles.axes1)
cla;
imshow(Irect), hold on
[H,Theta,rho] = hough(Irect,'RhoResolution',0.5,'ThetaResolution',0.5);
numpeaks=str2num(get(handles.Num,'String'));
Chiffre=str2num(get(handles.Max,'String'));
```

```
P = houghpeaks(H,numpeaks,'threshold',ceil(Chiffre*max(H(:))));
lines = houghlines(Irect,Theta,rho,P,'FillGap',20,'MinLength',40);
for k=1:length(lines)
   xy = [lines(k).point1; lines(k).point2];
   plot(xy(:,1),xy(:,2),'LineWidth',1.7,'Color','green')
   plot(xy(1,1),xy(1,2),'x','LineWidth',1.7,'Color','yellow');
   plot(xy(2,1),xy(2,2),'x','LineWidth',1.7,'Color','red');
end
function initialisation_Callback(hObject, eventdata, handles)
set(handles.STx,'String','')
set(handles.STa,'String','')
set(handles.Num,'String','')
set(handles.comptage,'String','')
set(handles.Max,'String','')
function revenir_Callback(hObject, eventdata, handles)
delete(handles.figure1);
Menus
function quitter_Callback(hObject, eventdata, handles)
exitProg(handles);
function exitProg(handles)
    selection = questdlg(['Voullez-vous fermer l''appliction ?'],...
                        ['TRANSFORMEE DE HOUGH'],...
                        'Oui','Non','Oui');
    if strcmp(selection,'Non')
        return;
    end
    delete(handles.figure1)
```

ANNEXE 3: *Utilisation du logiciel*

Tapez sur le Matlab « intro » et on a cette interface [14][15]

Figure A3.01 : *Fenêtre d'accueil*

Après, on lance le bouton LANCER :

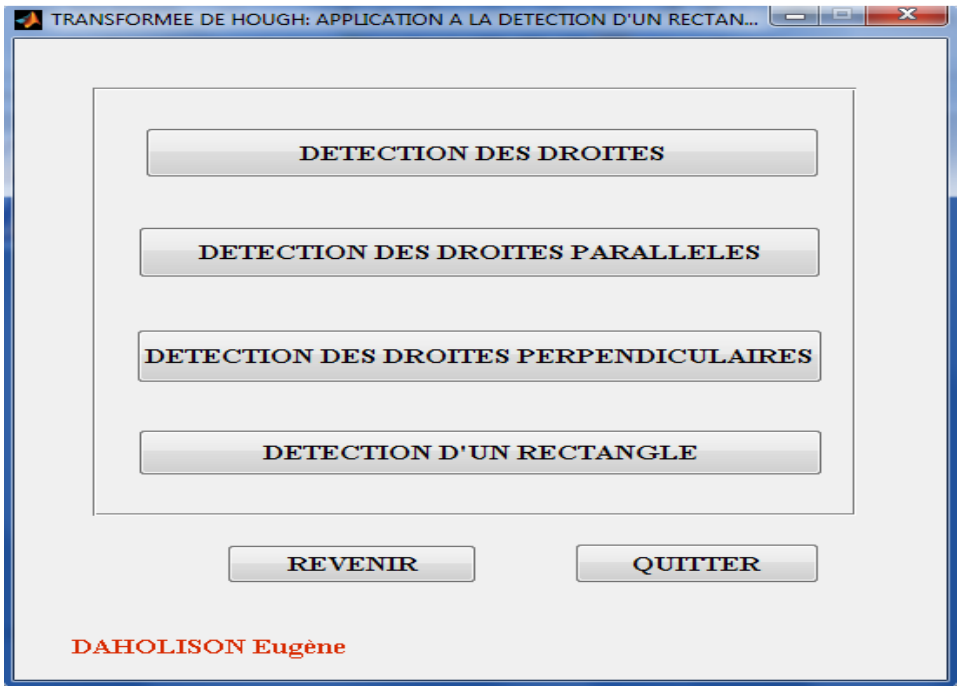


Figure A3.02 : *Fenêtre menus*

On clique sur le bouton « DETECTION DES DROITES », on a la fenêtre ci-dessous :

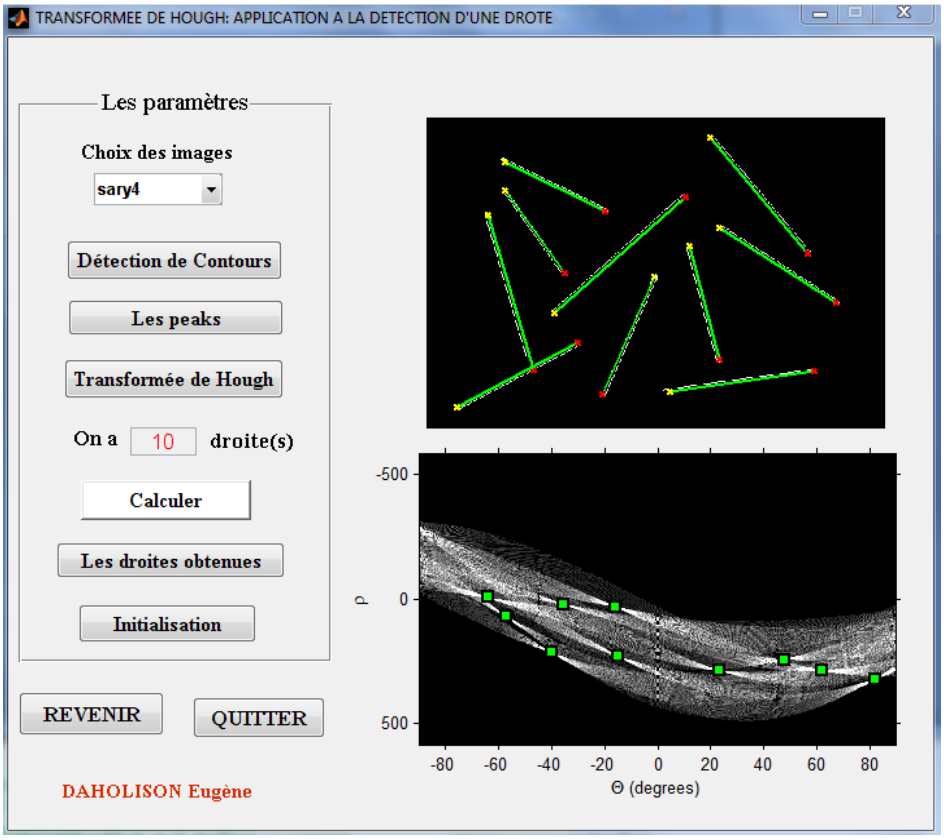


Figure A3.03 : *Détection des droites*

- On fait le choix à l'image (Sray1…4)
  - Par exemple, ici j'ai choisi l'image *« Sary4 »*.
  - Les votes sont dans le tableau A3.04 :

	Rho	Theta
1	243.5000	48
2	34	-16
3	322	82
4	285	62
5	70.5000	-57
6	286	23.5000
7	215.5000	-40
8	231.5000	-15
9	23.5000	-35.5000
10	-6.5000	-64

Tableau 4.10: *Les votes pour l'image « Sary4 »*

Donc pour chaque couple de (Rho, theta) correspond à une droite. Effectivement là on a 10 couples, ce qui revient à dire qu'il y a 10 droites dans l'image *« Sary4 »*

– Et si on clique sur le bouton *Calculer* alors le nombre des droites trouvées seront afficher automatique dans l'éditeur du texte au dessus de ce bouton même

- On clique sur le bouton *« Détection de Contours »*, pour la détection de contours
- *Les peaks (ou votes en français)* c'est pour trouver tous les votes dans le tableau bidimensionnel appelé *l'accumulateur*, il s'affiche dans une figure sous forme de matrice deux colonnes (Rho, theta)
- *La transformée de Hough*, pour tracer tous les sinusoïdes qui correspondent à chaque pixel dans l'image après la détection de contours.
- *Calculer*, pour afficher le nombre des droites trouvées dans l'image après la détection de contours.
- *Les droites obtenues*, c'est pour retracer les droites obtenues en superposant dans l'image après la détection de contours
- *Initialisation*, pour effacer la valeur dans l'éditeur du texte.
- REVENIR, pour revenir à la fenêtre Menus
    - On clic sur le bouton « DETECTION DROITES PARALLELES » dans l'interface Menus et on a :

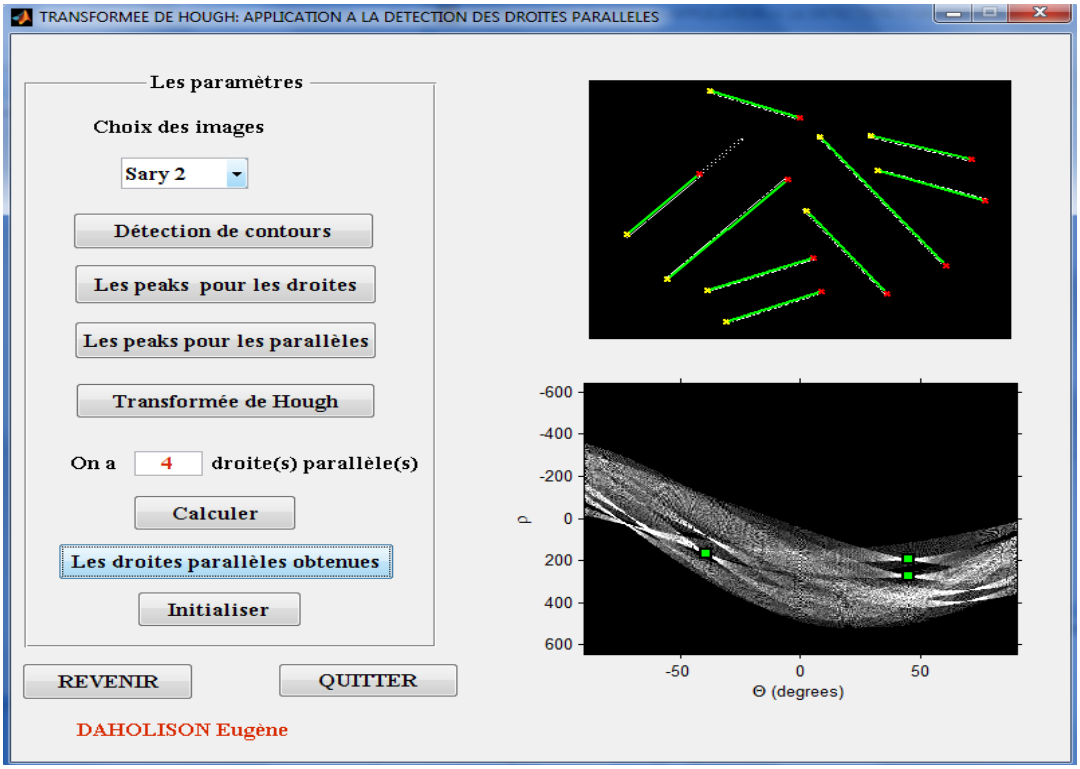


Figure A3.05 : *La détection des droites parallèles*

On effectue le même principe mais la différence se situe au niveau du bouton « *Les peaks pour les parallèles*». Dans ce bouton, on cherche parmi les votes seuls qui sont parallèles deux à deux et après on met dans une matrice à deux colonnes les votes qui correspondent à deux droites parallèles.

Par exemple :

– Voici les votes pour tous les pixels de l'image après la détection de contours :

	Rho	Theta
1	276	45
2	169.5000	-39
3	193	45
4	87.5000	-39
5	-13.5000	-71.5000
6	339.5000	70
7	18	-74
8	392	69
9	37	-70

Tableau 4.11: *Les votes pour toutes les droites trouvées*

– Voici les votes pour les droites sont en parallèles

	Rho	Theta
1	41.5000	45
2	41	-39
3	26.2500	69.5000
4	25.2500	-70.7500

Tableau 4.12: *Les votes pour toutes les droites qui sont en parallèles*

Chaque couple de (Rho, Theta) dans le tableau 4.05 correspond à deux droites qui sont en parallèles

On clic sur le bouton « DETECTION DES DROITES PERPENDICULAIRES » dans la fenêtre Menus, on a :

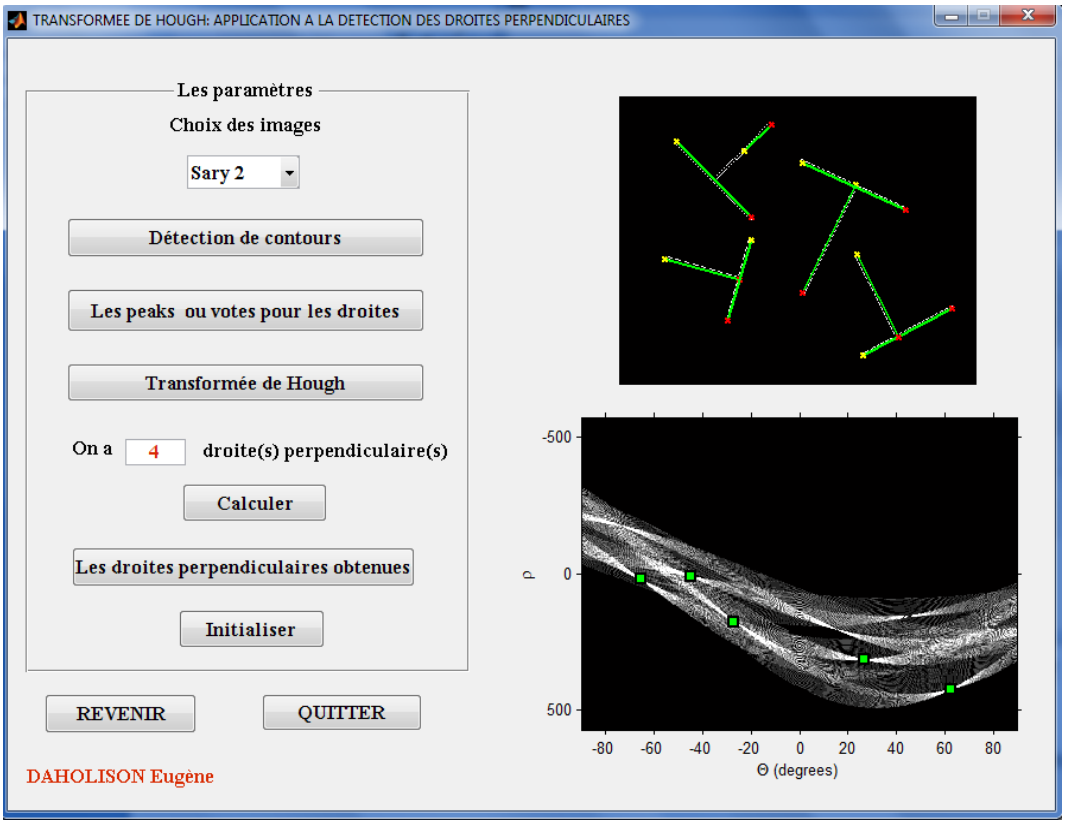


Figure A3.06 : *Détection des droites perpendiculaires*

On effectue la même méthode d'utilisation que la « DETECTION DES DROITES ».

Voici les votes trouvées qui correspondent à chaque droite dans l'image après la détection de contours.

	Rho	Theta
1	12	-45
2	314.5000	26.5000
3	21	-65.5000
4	426.5000	62.5000
5	177.5000	-27
6	159	45.5000
7	-178.5000	-74.5000
8	208.5000	16

Tableau 4.13: *Les votes pour les droites*

Et pour finir, on clic sur le bouton « DETECTION D'UN RECTANGLE », on obtient la fenêtre ci-dessous :

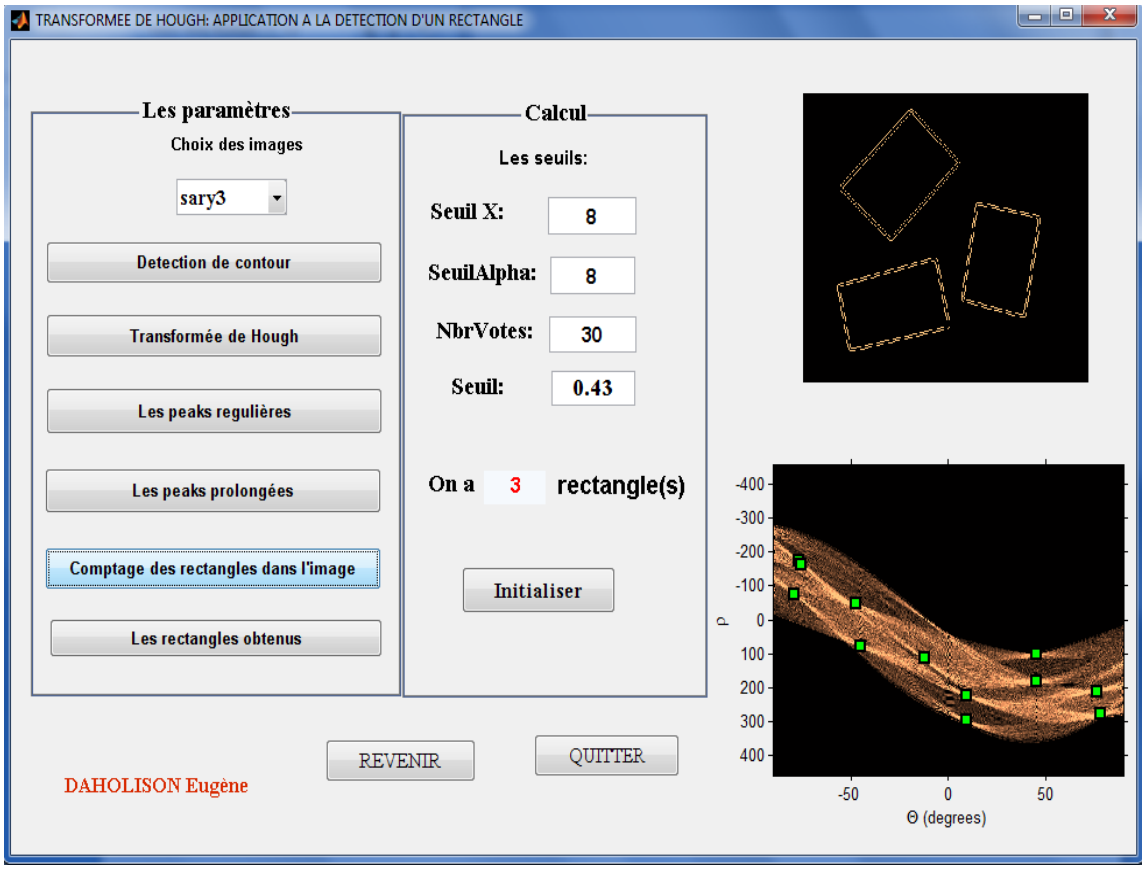


Figure A3.07 : *Détections des rectangles*

On a combiné les deux méthodes précédentes pour obtenir la méthode pour détecter des rectangles

Voici l'étape à suivre :

- Si on choisit une image, voici les valeurs qui correspondes sur les seuils :

➢ Sary1 : (Seuil X=8, SeuilAlpha=8, NbrVotes=30 et Seuil=0.4)

➢ Sary2 : (Seuil X=8, SeuilAlpha=8, NbrVotes=30 et Seuil=0.4)

➢ Sary3 : (Seuil X=8, SeuilAlpha=8, NbrVotes=30 et Seuil=0.43)

➢ Sary4 : (Seuil X=8, SeuilAlpha=8, NbrVotes=30 et Seuil=0.37)

Ces valeurs ont été choisies pour avoir à peu près le nombre exact des rectangles qui se trouve dans une image données.

BIBLIOGRAPHIE

[1] http://fr.wikipedia.org/wiki/Traitement_d'images

[2] http://en.wikipedia.org/wiki/Hough_transform

[3] Andriambololona S. Z., « *DETECTION DE CONTOURS EN TRAITEMENT D'IMAGE* » Mémoire d'Ingénieur ESPA Dép. Télécommunication.-E S P A, année 2004

[4] Almouzni G., «*Traitement numérique des images»,* EISTI traitement d'image 2010-2011.

[5] Brun L., «cours_couleur», *http://www. univ-reims.fr/Labos/LERI/membre/luc/ENSEIGNEMENT/COURS/TR_IMG.*

[6] Gagou Y., « *Cours de Traitement d'Image »,* Université de Picardie Jules Verne Licence de Physique S6 Année Académique 2005-2006.

[7] Bechet P., *«Reconnaissance d'image», RESEAU INFORMATIQUE GALILEO-CPE LYON* en 2005-2006.

[8] Tupin F., «*Techniques du traitement d'images, description de contours et de formes »,* DEA IARFA

[9] Barra V., «*Cours de traitement d'images»*

[10] Cudel C., Colicchio B. et Dieterlen A., *« Extraction dans les images »*, Groupe LAB.EL, Laboratoire MIPS, Université de Haute Alsace, Mulhouse, France.

[11] Maître H, «*Description de contours et de formes»*, *http://www.tsi.enst.fr/TDI/poly_contours.pdf*

[12] Jung C. R., Schramm R., « *Rectangle Detection based on a Windowed Hough Transform»,* UNISINOS - Universidade do Vale do Rio dos Sinos Ciências Exatas e Tecnologicas Av. UNISINOS, 950. Sao Leopoldo, RS, Brasil, 93022-000 {crjung,schramm}@exatas.unisinos.br

[13] Nixon M. S., Aguado A. S., « *Feature Extraction and Image Processing»*

[14] Refaat Y. A. A, Ahmed A. A et Abdulla I. A, « *Introduction to Graphical User Interface (GUI) MATLAB 6.5»,*

[15] Vick B., «*MATLAB Commands and Functions»,* Mechanical Engineering Department Virginia Tech.

RENSEIGNEMENTS SUR L'AUTEUR

Nom : DAHOLISON

Prénom : Eugène

Adresse : ESPA Vontovorona Bloc 3 Porte 89

Tél : +261 32 46 791 94

E-mail : jeneffesoam@yahoo.fr

Titre de mémoire : **DETECTION DE RECTANGLES BASEE SUR LA TRANSFORMEE DE HOUGH**

Nombre de pages : 88

Nombre de tableaux : 13

Nombre de figures : 53

Mots clés :
- Reconnaissance de forme
- Filtrage
- Détection de contours
- Transformée de Hough
- Image de synthèse.

Directeur de mémoire :
- M RAZAFINDRADINA Henri Bruno
- Tel: +261 32 02 174 56
- Email: hbrazafindradina@gmail.com

RESUME

Le traitement d'image est l'une des parties la plus importante pour plusieurs applications, à savoir la reconnaissance des formes et des contours. Il est indispensable que ce domaine soit un domaine de recherche pointu et évolutif à cause de la forte utilisation. Pour cela, il faut alors connaître les différentes techniques et approches à appliquer. Ce mémoire est consacré à l'étude de la transformée de Hough et leurs applications dans le domaine traitement d'image. La connaissance des différentes techniques comme le filtrage, la détection de contours, seuillage et la fameux transformée de Hough nous permis de mieux détecter les droites, les droites qui sont en parallèles, les droites perpendiculaires et enfin les rectangles dans les images après la détection de contours. Et ce qui nous a conduits d'effectuer quelques simulations sous MATLAB. En effet, ce langage permet de faire les manipulations des matrices et les calculs Mathématiques qui paraissent trop difficile pour certain langage

ABSTRACT

The numeric treatment of image is one of the parties the most important for several applications, to know the recognition of forms and edge. It is indispensable that this domain is a pointed and progressive research domain because of strong use. For it, it is then necessary to know different techniques and approaches to be applied. This memory is dedicated to the study of the Hough transform and their applications in domain treatment of image. The knowledge of different techniques as filtering, edge, seuillage and famous Hough transform we allowed to discern better rights, rights which are in parallels, perpendicular rights and finally rectangles in image after the edge. And what led us to perform some simulation under MATLAB. Indeed, this language allows to make the manipulations of the matrices and the Mathematical counting which seem too difficult for sure language