Université Nazi Boni

Unité de Formation et de Recherche en Sciences et Techniques (UFR/ST)

Ecole doctorale Sciences et Techniques

Laboratoire d'Algèbre, de Mathématiques Discrètes et d'Informatique (L.A.M.D.I)

Année académique 2016-2017


# MEMOIRE DE MASTER DE MATHEMATIQUES

Option : Théories Mathématiques et Applications (T.M.A)

---

**THEME : Application de la transformée de Hough rectangulaire à la détection de droites discrètes**

---

Présenté par :

**Cheick Amed Diloma Gabriel Traore**

Directeur de mémoire :

Dr Télesphore Tiendrebeogo

Co-directeur de mémoire :

Dr Abdoulaye Sere

---

Soutenu le 24 juillet 2019 devant le jury composé de :

| | | | |
|---|---|---|---|
| Président : | Théodore Y. M. Tapsoba, | Professeur Titulaire, | Université Nazi Boni |
| Examinateur : | Tiguiane Yelemou, | Maître de conférences, | Université Nazi Boni |
| Directeur de mémoire : | Télesphore Tiendrebeogo, | Maître de conférences, | Université Nazi Boni |
| Co-directeur de mémoire : | Abdoulaye Sere, | Maître assistant, | Université Nazi Boni |

# Remerciements

Le parcours "Mathématiques" de l'Université Nazi Boni est un parcours de titan, en témoigne le nombre de promotionnaires en Master. Nous sommes à présent aux portes de la recherche car nous avons su mettre en pratique les conseils et recommandations de nos professeurs ainsi que du personnel non enseignants de notre unité de formation et de recherche.
A l'issue de la rédaction de ce mémoire, nous sommes convaincus qu'un mémoire est loin d'être un travail solitaire. En effet, nous n'aurions jamais pu réaliser ce travail sans le soutien de certaines personnes dont la générosité, la bonne humeur et l'intérêt manifestés à l'égard de nos recherches nous ont permis de progresser dans cette phase délicate de l'apprenti chercheur.

En premier lieu, nous tenons à remercier nos Directeurs de mémoire, les professeurs Telesphore TIENDREBEOGO et Abdoulaye SERE pour la confiance qu'ils nous ont accordée en acceptant encadrer ce travail. Merci pour tous vos conseils et vos remarques tout au long de ce mémoire. J'ai été honoré de travailler sous votre direction. Trouvez ici l'expression d'une grande reconnaissance pour tout ce que vous avez fait pour moi. Les multiples voyages à vos cotés et les discussions intenses et fructueuses nous ont permis d'être à la hauteur du présent travail. Une fois de plus merci pour votre enthousiasme, votre générosité et votre disponibilité. Trouvez ici l'expression de notre profonde gratitude.

Grand merci aux professeurs B. SANGARE, I. KABORE pour leurs soutiens et conseils permanents, ainsi que au professeur Jean D.W. ZABSONRE dont les conseils nous ont permis de franchir la première année, aux professeurs J. BAYARA, T.M.Y. TAPSOBA, H. SORE et tous les professeurs qui n'ont pas été explicitement cités.
Nous tenons également à remercier tout le personnel de l'Université Nazi Boni de Bobo-Dioulasso pour leurs contributions à notre formation.

Faire de la recherche pour écrire un mémoire de Master en mathématiques est très différent de la rédaction de devoirs ou de projets. Nous avons pu compter sur le soutient sans limite des ainés de nos différents laboratoires de mathématiques que nous remercions tous. En particulier Ousman KOUTOU, Harouna OUEDRAOGO,Moussa BARRO, Thomas OUEDRAOGO, Assane SAVADOGO, Abdoul Hady KONFE, Esaï LANKOANDE, Ernest BOGNINI.
Merci à la famille KOUMARE et mes parents qui mon soutenus, aux amis et à tous ceux qui de près ou de loin ont contribué à ma formation.

# Résumé

Ce mémoire porte sur la reconnaissance de droites discrètes analytiques en dimension 2 à travers la Transformée de Hough Standard Etendue introduite par A. SERE lors de sa thèse de doctorat. L'objectif escompté est de permettre une reconnaissance de droites, optimale en temps. Pour ce faire, un maillage de l'image en grille rectangulaire est effectué. En se basant sur la transformée de Hough qui est une méthode de reconnaissance de formes dans une image bruitée, une sélection de grilles contenant un certain taux de pixels nécessaire à la reconnaissance de droites est faite. L'ensemble des grilles sélectionnées est utilisé dans des algorithmes (9 et 12) de reconnaissances de droites afin de déterminer l'existence de droite(s) dans l'image numérique. Ces algorithmes utilisent soit un accumulateur de données, soit la notion de préimage généralisée. Des simulations via des méthodes de traitement d'images avec des librairies à jours sont effectuées pour illustrer tous nos résultats théoriques.

**Mots-clés :** Maillage, transformée de Hough rectangulaire, reconnaissance de droite, préimage généralisée, géométrie et topologie discrète.

# Abstract

This document deals with the recognition of analytical discrete lines in 2 dimension through the Extended Standard Hough Transform introduced by A. SERE during his doctoral thesis. The objective is to allow optimal recognition in time of straight line. To do this, a mesh of the rectangular grid image is made. Using the Hough transform which is a pattern recognition method in a noisy image, a selection of grids containing a certain pixel rate necessary for the recognition of lines is made. The set of selected grids is used in algorithms (9 and 12) for recognizing lines in order to determine the existence of straight line(s) in the digital image. These algorithms use either a data accumulator or the notion of generalized preimage. Theoretical simulations using image processing methods with current libraries are performed to illustrate all our theoretical results.

**Keywords :** Meshing, rectangular Hough transform, straight line recognition, generalized preimage, geometry and discrete topology.

# Sommaire

# Liste des tableaux

# Liste des Algorithmes

# Table des figures

# Introduction générale

Une image est une représentation visuelle ou mentale de quelque chose (objet, être vivant). Elle peut être naturelle (ombre, reflet), artificielle (sculpture, peinture, photographique), tangible ou conceptuelle. Platon définissait l'image en ces termes [Pla C] : « J'appelle image d'abord les ombres ensuite les reflets qu'on voit dans les eaux, ou à la surface des corps opaques, polis et brillants et toutes les représentations de ce genre. »
Le traitement d'image est une discipline des mathématiques appliquées et de l'informatique qui étudie les images numériques et leurs transformations, dans le but d'améliorer leurs qualités ou d'en extraire de l'information utile dans la prise de décision en imagerie médicale, en contrôle qualité, en contrôle non destructif[1]. Les images numériques sont constituées de petits points discrets appelés pixels en dimension 2 et voxels en dimension 3. La reconnaissance de formes présente dans une image et en particulier la reconnaissance de droites est un problème majeur dans le domaine du traitement d'image[2] qui suscite un grand engouement des chercheurs. Ce sujet a connu plusieurs approches de résolutions en majeure partie basée sur la transformée de Hough [Hou62] que SERE et al [SSA13, SOZ18] ont étendue puis amélioré. La question de la détection optimale est primordiale dans cette ouvrage. Plusieurs interrogations aux quelles nous répondrons naissent alors :

— qu'est ce qu'une droite discrète ?
— Comment reconnaitre une droite discrète dans une image ?
— Comment optimiser la détection de droite(s) discrète(s) dans une image ?

Les mathématiques à travers la géométrie discrète fournissent un ensemble d'outils adaptés à la nature discrète des images ; l'informatique avec des langage et librairie tels que python et opencv[3] permet de corroborer ou d'infirmer des résultats théoriques.

Le véritable problème de détection de droites avec la transformée de Hough est une implémentation gourmande en temps et en ressource. Dans un soucis d'optimisation en temps et en ressource de la détection de droite avec la transformée de Hough rectangulaire [SSA13] nous réalisons tout d'abord un maillage rectangulaire de l'image, ce qui permettra un parcours plus rapide de l'image maille par maille. Les images numériques étant en générale constituées de grande zones ne contenant pas de données devant être utilisées pour la reconnaissance nous effectuerons une sélection des mailles ayant un certain taux de pixels utiles à la détection. Puis nous appliquerons aux mailles sélectionnées un algorithme de reconnaissance de droite (algorithme 8) qui est notre contribution. Pour terminer nous implémenterons nos algorithmes.

1. Aide au pilotage automatique
2. Voir le nombre d'articles sur academia.edu
3. Open Computer Vision

**Organisation du mémoire**

Le présent mémoire de master est constitué de trois chapitres.
Le premier chapitre fait des rappels de géométrie discrète afin que nous soyons dans les meilleurs dispositions pour aborder la suite du document. Il est constitué d'une part de définitions et propriétés relatives aux espaces discrets, aux primitives discrètes, aux modèles de discrétisations. D'autre part constitué d'un état de l'art de la transformée de Hough et de l'algorithme 2 de reconnaissance de droites.
Dans le second chapitre nous mettrons en œuvre la transformée de Hough rectangulaire en créant un maillage rectangulaire de l'image. L'algorithme de création de cette maille nécessitera l'utilisation d'algorithmes déterministes de factorisation et de primalité d'entier. Puis nous choisirons les mailles ayant un certain taux de pixels "utiles", le tout dans le but de détecter des droites dans l'image grâce aux algorithmes 2 et 8.
Le dernier chapitre consistera à programmer des algorithmes du chapitre 2 et à effectuées des tests de ces algorithmes sur des images numériques.
Une conclusion et des perspectives à ces travaux clos le mémoire, nous avons fait une annexe comportant les codes python de détections des contours dans une image numérique avec le filtre de Canny et de l'algorithme 12.

Chapitre 1

# Etat de l'art



## Introduction

Une image numérique obtenue de quelques manière que ce soit contient des formes variées. Cette image peut être étudiée et les formes détectées grâce à des outils mathématiques tels que la géométrie discrète et la transformée de Hough.
Dans ce chapitre nous énumèrerons et étudierons les propriétés des éléments de géométrie et de topologie discrète dont nous aurons besoin dans les chapitres suivants. Puis nous ferons un état de l'art de la transformée de Hough principalement celles standard et standard étendue. Nous terminerons par un algorithme de reconnaissance de forme (hyperplan).

FIGURE 1.1 – Discrétisation d'une image, les points noirs représentent des points discrets

## 1.1 Eléments de géométrie discrète

La géométrie discrète est une branche des mathématiques distincte de la géométrie euclidienne ou continue. Une image numérique est constituée de points lumineux formant une certaine structure permettant de distinguer des sous ensembles d'images. La géométrie discrète permet de définir et de manipuler les points lumineux dans les images qui sont des ensembles de points à coordonnées entières formant des objets discrets.

### 1.1.1 Concepts de bases

$\forall k \in \mathbb{N}^*, [\![1, k]\!]$ représente le sous ensemble des entiers naturels compris entre 1 et $k$.
$E(x)$ représente la partie entière de $x$. $B'_d(a, r)$ est la boule fermée de centre $a \in \mathbb{R}^n$ et de rayon $r \in \mathbb{R}_+$ pour la distance $d$ de $\mathbb{R}^n$.

**Définition 1. ($\beta$-Hypercube [Dex06])** *L'hypercube (ou cube de dimension n) de centre $(c_1, \ldots, c_n) \in \mathbb{R}^n$ et de taille $\beta \in \mathbb{R}_+$ est l'ensemble des points $p = (p_1, \ldots, p_n) \in \mathbb{R}^n$ vérifiant*

$$\forall i \in [\![1, n]\!],\ c_i - \frac{\beta}{2} \leq p_i \leq c_i + \frac{\beta}{2}$$

Un hypervoxel est un hypercube de taille unitaire ($\beta = 1$). En particulier un pixel est un hypervoxel de dimension 2 ; un voxel est un hypervoxel de dimension 3.

**Définition 2. (Point discret)** *Soit $d$ une distance sur $\mathbb{R}^n$. Les points d'un ensemble $X \subseteq \mathbb{R}^n$ sont dits isolés si pour tout point $x \in X$, il existe $\epsilon \in \mathbb{R}^*_+$ tel que la boule fermée $B'_d(x, \epsilon)$ ne contient que $x$. Un point isolé de $\mathbb{Z}^n$ est appelé point discret de dimension n.*

**Définition 3. (Pavage, espace discret, pavé, objet discret [Mon03])**

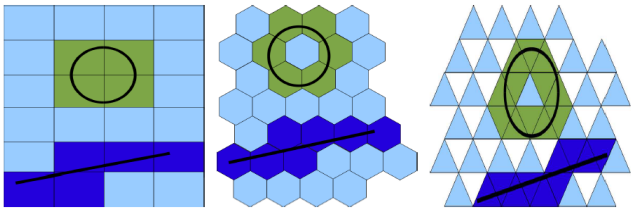
*(i) Un pavage est une partition dénombrable de compactes (pavés) d'intérieurs non vide dans un espace euclidien ;*
*(ii) un espace discret est un ensemble de pavages de l'espace euclidien $\mathbb{R}^n$ ;*
*(iii) un pavé est une cellule (composant élémentaire) d'un espace discret ;*
*(iv) un objet discret est un ensemble de pavés.*

**NB :** Un point lumineux de l'image est un point discret dans l'espace discret.
La discrétisation est le fait de décomposer un espace (une image) en points discrets (figure [1.1]).

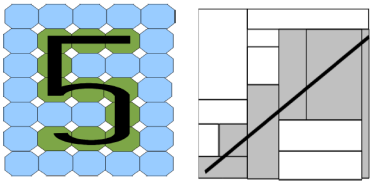
Une grille régulière est un espace discret où les pavés sont tous identiques. Dans cette grille on peut concevoir un point discret comme un élément de $\mathbb{Z}^n$ ils sont alors équidistants pour préserver la régularité.

Une grille irrégulière est un espace discret où les pavés ne sont pas identiques par exemple les pavages isothétiques 1.2b, la de figure à droite. Les pavés isothétiques sont des rectangles ayant leurs côtés parallèles aux axes du repère de l'espace euclidien en dimension 2. Ils améliorent les grilles carrées en fournissant une représentation, segmentation multi-échelle d'images. Les algorithmes s'appliquant aux grilles régulières peuvent donc s'appliquer sur les parties des grilles isothétiques présentant des pavés uniformes.

Dans ce mémoire on travail dans une grille régulière de dimension 2, notre espace discret est



(a) Grilles régulières : carré(pixel), hexagonal, triangulaire



(b) Grilles irrégulières

FIGURE 1.2 – Différents types de grilles [Ser13]

alors pavable dans $\mathbb{R}^n, n = 2$. En effet une image numérique est vue ici comme un espace discret. Tout traitement effectué dans l'image correspond à un traitement dans l'espace discret.

#### 1.1.1.1 Techniques de pavage

A tout ensemble de points isolés peut être associé un pavage par l'ensemble des points de $\mathbb{R}^n$ les plus près de ces points. Ce type de pavage est appelé diagramme de Voronoi.
La triangulation de Delaunay, graphe dual du diagramme de Voronoi s'obstient en reliant les points isolés si leurs pavés du diagramme de Voronoi sont adjacents.

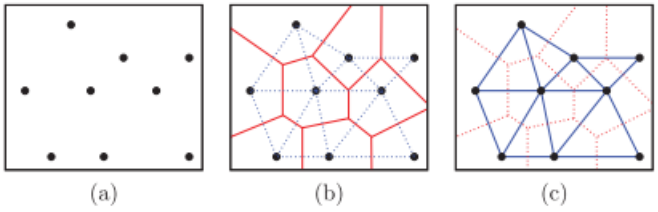



FIGURE 1.3 – (a) Nuage de points isolés. (b) Diagramme de Voronoï. (c) Triangulation de Delaunay.[Rod11]

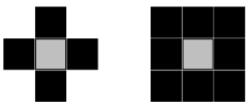
### 1.1.2 Topologie discrète

Nous désignons par $\mathbb{D}^n$ l'espace discret quelconque de dimension $n, n \in \mathbb{N}^*$.
Dans cette section nous introduisons des propriétés relatives aux pavés, parmi lesquels les notions de k-voisinage ($k \in [\![0, n-1]\!]$) et de connexité entre points discrets. Ces propriétés permettent d'établir des relations d'adjacence et d'incidence entre pavés et de comprendre la structure interne d'un objet discret ou d'un contour discret. Dans le discret le k-voisinage est l'équivalent de la notion de voisinage dans le continu, servant à définir la notion de continuité d'une courbe discrète.
La définitions 4 présente les notions de voisinage dans les espaces discrets classiques et la définition 5 la généralise aux espaces discrets non classiques.

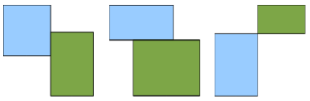
**Définition 4. (k-voisinage [And00])** *Deux points discrets $p = (p_1, ..., p_n) \in \mathbb{Z}^n$ et $q = (q_1, ..., q_n) \in \mathbb{Z}^n$ sont dits k-voisins avec $k \in [\![0, n-1]\!]$ si :*

$$\forall i \in [\![1, n]\!],\ |p_i - q_i| \leq 1 \text{ et } k \leq n - \sum_{i=1}^{n} |p_i - q_i|$$

En dimension 2 deux points discrets $p, q$ sont :
— 0-voisins si les pixels $P, Q$ de centre $p, q$ ont au moins un sommet en commun ;
— 1-voisins si $P, Q$ ont au moins une arête commune.



(a) Un pixel gris et ses 4 1-voisins puis le pixel gris et ses 8 0-voisins [Rod11]



(b) Voisinage de pavé isothétique [Ser13]

FIGURE 1.4 – Voisinage

**Définition 5. (Généralisation du k-voisinage [Dex06])** *Deux points discrets $p$ et $q$ d'un espace discret $\mathbb{D}^n$ sont dits k-voisins si leurs pavés associés $P$ et $Q$ sont adjacents par au moins une cellule topologique de dimension k.*

**Remarque 1.** *Deux points discrets k-voisins sont aussi k-adjacents d'où l'équivalence des notions de k-voisinage et de k-adjacence.*
*En dimension 2 si :*
*$p$ et $q$ sont 0-voisins sans être 1-voisins alors $d(p, q) = \sqrt{2}$ unité ;*
*$p$ et $q$ sont 0-voisins alors $d(p, q) = 1$ unité.*

La définition 5 nous permet de déduire les notions de courbes, de conexité et d'objets discrets.

**Définition 6.** ***(k-chemin, k-arcs,k-courbe, k-connexité, k-objet)***
*(i) Une séquence de pavé adjacents, k-voisins forment un k-chemin.*
*(ii) Soient $\mathcal{C} = \{p_1, ..., p_m\} \subset \mathbb{D}^n$, $m \in \mathbb{N}^*$ une séquence de points discrets, $\mathcal{C}$ est un k-arc si $\mathcal{C}$ est un k-chemin tel que $\forall i \in [\![2, m-2]\!]$, $p_i$ a exactement deux k-voisins. Si $p_1$ et $p_m$ sont k-voisins alors $\mathcal{C}$ est une k-courbe.*
*(iii) Soit $O$ un ensemble de points discrets. $O$ est dit k-connexe si et seulement si il existe un k-chemin entre deux éléments quelconques de $O$, dans ce cas $O$ est un k-objet.*

Les objets discrets ont des propriétés différentes de celles des objets continus. La notion de connexité discrète permet de déterminer si un objet discret est en un ou plusieurs "morceaux". En effet un objet qui n'est pas connexe est subdivisé en plusieurs k-composantes (ensembles maximal k-connexes de l'objet discret).

**Définition 7. (Objet k-séparant)** *Soient $O_1$ et $O_2$ deux objets discrets tels que $O_1 \subseteq O_2$. Si $O_2 \backslash O_1$ n'est pas k-connexe alors $O_2$ est un k-séparant de $O_1$. Un objet 0-séparant sera simplement dit séparant.*

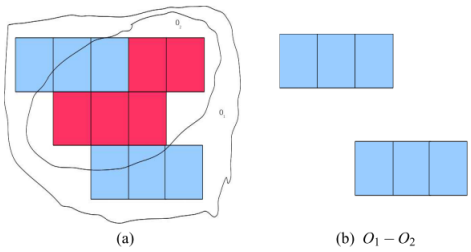



FIGURE 1.5 – Objet k-séparant. $O_2$ en rouge.[SSA13]

**Définition 8. (k-tunnel)** *Soient $O_1, O_2$ deux objets discrets avec $O_2 \subseteq O_1$ et $O_2$ non k-séparant de $O_1$. S'il existe un point discret $p \in O_1$ tel que $O_2 \cup \{p\}$ soit un k-séparant de $O_1$ le décomposant en deux composantes connexes, il existe alors un k-chemin reliant ces deux composantes passant par p. On dit que $O_2$ présente un k-tunnel.*

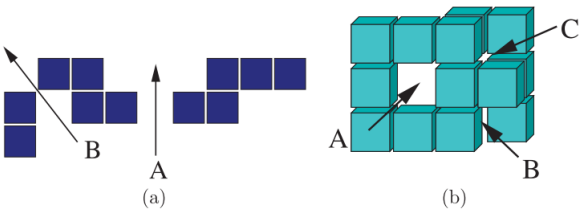



FIGURE 1.6 – k-tunnels. (a) en dimension 2 : un 1-tunnel (A) et un 0-tunnel (B). (b) en dimension 3 : un 2-tunnel (A), un 1-tunnel (B) et un 0-tunnel (C).[Rod11]

**Définition 9. (Point k-simple, objet k-minimal [Rod11])** *Soit $O_2$ un k-séparant de $O_1$ tel que $O_1 \backslash O_2$ a exactement deux k-composantes. Un point discret $p \in O_2$ est dit k-simple si $O_2 \backslash \{p\}$ est un k-séparant de $O_1$.*
*Un objet discret k-séparant est k-minimal s'il ne possède aucun point k-simple.*

La description d'objets discrets peut se faire par une représentation en extension, par énumération des pavés constituants l'objet ou par une représentation analytique en décrivant l'objet par un système d'inéquations qui doit être vérifié par tous les points de l'objet discret.

**Définition 10. (Image discrète, résolution)** *Une image discrète est une fonction totale (fonction dont le domaine de définition est l'ensemble de départ) entre un espace discret quelconque et un ensemble de valeurs (appelées couleurs).*
*La résolution définit le nombre de pixel par unité de longueur (pouce ou centimètre) et s'exprime en PPI (Pixel Per Inch).*

**NB :** Un pixel n'a pas de taille bien définie, elle dépend de la résolution de l'image discrète considérée.

Une image discrète peut être représentée vectoriellement grâce à la description analytique des objets discrets (primitives). Chacune de ces primitives analytiques étant décrite individuellement, indépendamment de la taille et de la résolution de l'image.
Les images matricielles travaillent directement sur les pixels tandis que les objets discrets décrits analytiquement définissent une zone offset. L'objet discret est donc formé de tous les points discrets situés dans la zone offset. Dans les lignes qui suivent nous définissons les hyperplans analytiques discrets et présentons des modèles de discrétisation analytique.

## 1.2 Discrétisation et hyperplans analytiques discrets

Une droite discrète est un hyperplan analytique en dimension 2. En géométrie euclidienne, un segment de droite ou une droite est constitué d'une infinité de points tandis qu'une droite discrète (autre que verticale ou horizontale) en dimension 2 dans un espace discret classique ressemble à un escalier dont les marches (paliers) non pas nécessairement la même longueur. Pour un nombre $\alpha$ la suite $\beta_k(\alpha) = E[(k+1)\alpha] - E[k\alpha]$ donne la taille du $k^{\text{ième}}$ palier d'une droite discrète de pente $\alpha$.

**Définition 11. (Droite arithmétique analytique discrète [Rev89])** *Une droite discrète de paramètres $(a,b,\mu)$ et d'épaisseur $w$ est définie comme l'ensemble des points entiers $(x,y)$ vérifiant la double inégalité $\mu \leq ax + by < \mu + w$ , $(a,b,\mu,w) \in \mathbb{Z}^4, pgcd(a,b) = 1$. La droite est notée $\mathcal{D}(a,b,\mu,w)$ et $\frac{a}{b}$ en est la pente. (Figure 1.7)*

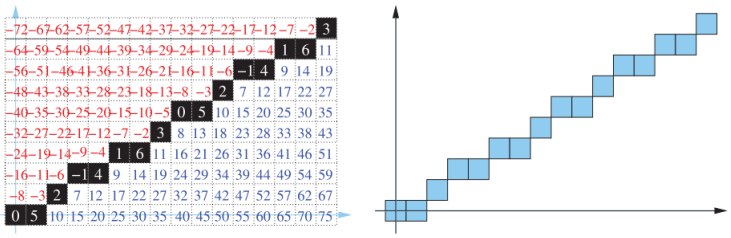



FIGURE 1.7 – Droite discrète arithmétique analytique $\mathcal{D}(5,-8,-1,8)$ les inégalités sont $-1 \leq 5x - 8y < 7$. (a) chaque point discret $(x,y)$ est étiqueté par $5x - 8y$. (b) la droite discrète obtenue.[Rod11]

La définition précédente fut généralisée par J-P. Reveillès pour donner celle des hyperplans analytiques.

**Définition 12. (Hyperplan analytique discret [Rev91])** *En dimension $n, n \in \mathbb{N}^*$, l'hyperplan analytique discret de paramètres $A = (a_1, ..., a_n) \in \mathbb{R}^n, \mu \in \mathbb{R}$ et $w \in \mathbb{R}$ est l'ensemble des points discrets $X = (x_1, ..., x_n) \in \mathbb{Z}^n$ vérifiant*

$$\mu \leq \sum_{i=1}^{n} a_i x_i < \mu + w \tag{1.1}$$

*Si l'inégalité de droite est large, on parle d'hyperplan analytique discret fermé.*

Pour construire un objet discret puis déterminer sa nature il est nécessaire de déterminer sa région offset, qui est l'ensemble des pavés touchés par l'objet continu. Ces pavés de divers natures (carrés, hypercube) donnent plusieurs modèles de discrétisation.

**Définition 13. (Discrétisation d'un objet [Rod11])** *Soit $E$ un objet continu et $d$ une distance, la discrétisation de $E$ notée $\mathcal{D}_d(E)$ associée à la distance $d$ est définie par :*

$$\mathcal{D}_d(E) = \left\{ p \in \mathbb{Z}^2 / d(p, E) \leq \frac{1}{2} \right\} \tag{1.2}$$

Une définition équivalente à la précédente fait intervenir des éléments structurants : boules $B'_d(\frac{1}{2}) = B'_d(x, \frac{1}{2}), \forall x \in E$. La discrétisation de E devient :

$$\mathcal{D}_d(E) = \left( E \bigoplus B'_d\left(\frac{1}{2}\right) \right) \cap \mathbb{Z}^2$$

Où $X \bigoplus Y = \{x + y, x \in X, y \in Y\}$ est la somme de Minkowski.

**Remarque 2.** *$E \oplus B'_d\left(\frac{1}{2}\right)$ est la région (zone) offset de l'objet discret.*
*La définition (1.2) est très générales et en fonction de la distance utilisée on définit les modèles de discrétisation :*

- *Naïf [Rev91] [And00] : basé sur la distance Manhattan $d_1$*
  *où* $(x_1, ..., x_n), (y_1, ..., y_n) \in \mathbb{R}^n$, $d_1((x_1, ..., x_n), (y_1, ..., y_n)) = \sum_{i=1}^{n} |x_i - y_i|$ ;
- *Pythagoricien : basé sur la distance euclidienne $d_2$*
  *où* $(x_1, ..., x_n), (y_1, ..., y_n) \in \mathbb{R}^n$, $d_2((x_1, ..., x_n), (y_1, ..., y_n)) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$ ;
- *Supercouverture et standard : basé sur la distance infinie $d_\infty$*
  *où* $(x_1, ..., x_n), (y_1, ..., y_n) \in \mathbb{R}^n$, $d_\infty((x_1, ..., x_n), (y_1, ..., y_n)) = \max_{1 \leq i \leq n} |x_i - y_i|$ .

### 1.2.1 Modèle naïf

Pour ce modèle, l'élément structurant est un carré de côté $\frac{\sqrt{2}}{2}$ et de diagonales unitaires alignées avec les axes ; cela conduit à un pavage de l'espace avec des losanges en dimension 2. Les pavés sont des boucles $B'_{d_1}(\frac{1}{2})$ centré en un point de $\mathbb{Z}^n$.

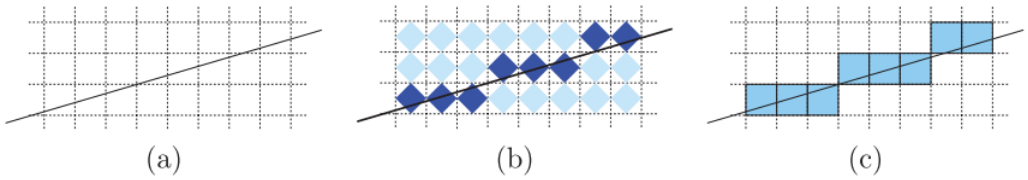



FIGURE 1.8 – Exemple de discrétisation d'une droite euclidienne en dimension 2 . (a) une droite euclidienne. (b) pavage de l'espace par des boules $B_{d_1}(\frac{1}{2})$. (c) Droite discrète naïve obtenue [Rod11]

Un objet décrit avec le modèle naïf est un objet naïf.

**Définition 14. (Hyperplan naïf)** *L'hyperplan naïf de paramètres* $A = (a_0, ..., a_n) \in \mathbb{R}^{n+1}$ *est l'ensemble des points* $X = (x_1, ..., x_n) \in \mathbb{Z}^n$ *vérifiant :* $-\frac{\max\limits_{1\leq i\leq n} |a_i|}{2} \leq a_0 + \sum\limits_{i=1}^{n} a_i x_i \leq \frac{\max\limits_{1\leq i\leq n} |a_i|}{2}$

Pour ce modèle on a pris : $\begin{cases} \mu = & -a_0 - \frac{\max\limits_{1\leq i\leq n} |a_i|}{2} \\ w = & a_0 + \max\limits_{1\leq i\leq n} |a_i| \end{cases}$ dans (1.1)

### 1.2.2 Modèle supercouverture

Pour ce modèle l'élément structurant est un hypercube de taille unitaire dont les côtés sont alignés avec les axes. Les pavés sont des boules $B_{d_\infty}(\frac{1}{2})$ centrées en un point de $\mathbb{Z}^n$.
Il peut être appliqué aux espaces discrets non régulier, ce qui fut établit par D. Cœurjolly dans [Coe05] dans le cadre des grilles irrégulières isothétiques.
Un objet décrit avec le modèle supercouverture est un objet supercouverture.

**Définition 15. (Hyperplan supercouverture [And00])** *L'yperplan supercouverture de paramètres* $A = (a_0, a_1, ..., a_n) \in \mathbb{R}^{n+1}$ *est l'ensemble des points* $X = (x_1, ..., x_n) \in \mathbb{Z}^n$ *vérifiant :*

$$-\frac{\sum\limits_{i=1}^{n} |a_i|}{2} \leq a_0 + \sum_{i=1}^{n} a_i x_i \leq \frac{\sum\limits_{i=1}^{n} |a_i|}{2} \tag{1.3}$$

Pour le modèle supercouverture : $\begin{cases} \mu = & -a_0 - \frac{\sum\limits_{i=1}^{n} |a_i|}{2} \\ w = & a_0 + \sum\limits_{i=1}^{n} |a_i| \end{cases}$ dans (1.1) (1.4)

L'hyperplan supercouverture présente parfois des k-bulles. Lorsque l'hyperplan euclidien passe par un pointel, tous les voxels adjacents à ce sommet sont considérés, formant ainsi un amas de point 0-simples appelé k-bulle.

Les points simples des k-bulles vérifies : $\sum\limits_{i=1}^{n} a_i x_i = -a_0 - \sum\limits_{i=1}^{n} |a_i|$ ou $\sum\limits_{i=1}^{n} a_i x_i = -a_0 + \sum\limits_{i=1}^{n} |a_i|$

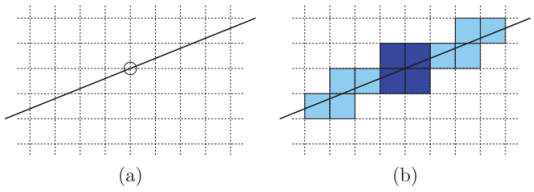



FIGURE 1.9 – Discrétisation supercouverture d'une droite euclidienne en dimension 2 . (a) la droite euclidienne. (b) Droite discrète supercouverture obtenue présentant une 2 -bulle (en foncé) [Rod11]

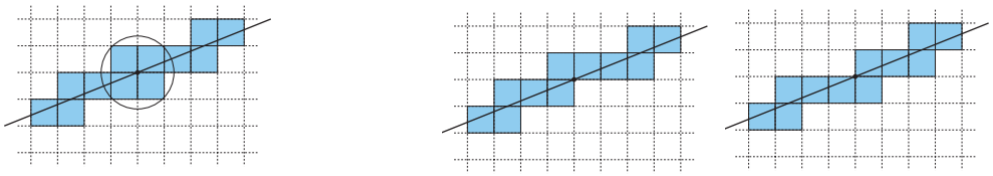
Les k-bulles représentent une insuffisance du modèle supercouverture, pour y palier E. Andrès dans [And03] présente le modèle standard.

### 1.2.3 Modèle standard

Il est une amélioration du modèle supercouverture dans le but d'éliminer les k-bulles. Il possède les mêmes éléments structurant que le modèle précédent, les points d'un hyperplan standard vérifient 1.3. De plus le modèle standard est un 0-séparant, (n-1)-connexe n'ayant ni point simple, ni tunnels.
La suppression des k-bulles se fait en transformant les égalités de (1.4) en inégalité en fonction de l'orientation de l'espace. Ainsi on aura par exemple :

$$\sum_{i=1}^{n} a_i x_i > -a_0 - \sum_{i=1}^{n} |a_i| \quad \text{ou} \quad \sum_{i=1}^{n} a_i x_i < -a_0 + \sum_{i=1}^{n} |a_i|$$

De ce fait il existe plusieurs modèles standards en fonction du point de la k-bulle supprimé.



(a) Discrétisation supercouverture avec une 2-bulle

(b) Droites standard obtenue par suppression d'un des points simples de la 2-bulle

FIGURE 1.10 – Du modèle supercouverture à celui standard [Rod11]

## 1.3 Transformée de Hough Standard Etendue et reconnaissance de primitives discrètes

Les méthodes de reconnaissance des primitives peuvent être classées en deux grandes catégories : les méthodes déterminant si des points discrets appartiennent à un hyperplan discret et celle fournissant en sus les paramètres des primitives reconnues. Nous nous intéressons au deuxième mode de reconnaissance.

La reconnaissance de primitives discrètes dans l'espace image est basée sur la transformée de Hough linéaire [Hou62] permettant d'abord de reconnaitre les lignes dans l'image, puis à l'aide de sa généralisation [DH72] de détecter d'autres formes géométriques (cercles, ellipse, parabole, ...). H. MAITRE écrit un panorama de la transformation de Hough en 1985 [Mai85] dans lequel on y trouve une définition unificatrice de la transformée de Hough et un apperçu de ses différentes variantes. Depuis 1985 de nombreux auteurs travaillent à améliorer la transformée de Hough tels que A. SERE et al [SSA13] qui étendent la transformée de Hough standard à la détection de lignes standards ou naïves dans une image bruitée. Un espace appelé, espace de paramètre est utilisé pour représenter les objets géométriques

[DS84, McI84] tels que les pixels et voxels. Cette représentation géométrique est appelée préimage.

### 1.3.1 Espace de paramètres

Les espaces de paramètres sont utilisés dans le domaine du traitement d'image pour reconnaitre des formes paramètriques présentent dans une image. Ces espaces sont définis à partir de transformations associant un point $P$ de l'espace image à un objet géométrique $O$ (droite, courbe, sphère, ...) dans l'espace de paramètres appelé préimage de $P$. Chaque point de cet objet est alors associé par le biais d'une deuxième transformation à une primitive de l'espace image.
Notons $\mathcal{P}_n = (0_\mathcal{P}, Y_1, ..., Y_n) \subseteq \mathbb{R}^n$ l'espace de paramètres, $\mathcal{I}_n = (0_\mathcal{I}, X_1, ..., X_n) \subseteq \mathbb{R}^n$ l'espace image (euclidien). $\mathcal{P}_n$ et $\mathcal{I}_n$ sont appelés espaces duaux, chacun étant le dual de l'autre. Dans la suite du chapitre on travaillera en dimension 2, c'est à dire $n = 2$.

### 1.3.2 Transformée de Hough classique et standard

La transformée de Hough permet de détecter des droites, des cercles ainsi que d'autres formes paramétriques ou non paramétriques. Il est question dans ce document de la détection de droites.

**Définition 16.** ***(Transformée de Hough)*** *Soit $M(X, Y)$ un point de l'espace image $\mathcal{I}_2 \subset \mathbb{R}^2$. La transformée de Hough de $M$ est un ensemble de points $(a, b)$ dans l'espace de paramètres $\mathcal{P}_2 \subset \mathbb{R}^2$ vérifiant $Y = b - aX$. Donc la transformée de Hough d'un point est une ligne (figure 1.11a, 1.11b).*

Si $A(X_1, Y_1), B(X_2, Y_2), C(X_3, Y_3)$ sont trois points d'une droite euclidienne (de l'espace image) définis par l'équation $y = ax + b,\ (a, b) \in \mathbb{R}^2$. Alors les coordonées de $A, B$ et $C$ vérifient les équations

$$b = Y_1 - aX_1 \tag{1.5}$$

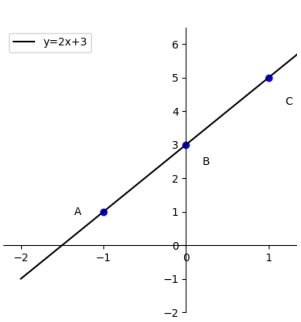
$$b = Y_2 - aX_2 \tag{1.6}$$

$$b = Y_3 - aX_3 \tag{1.7}$$

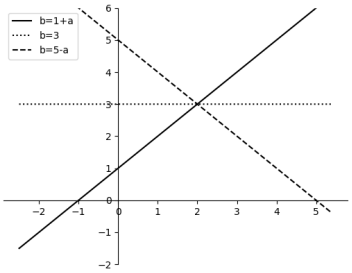
Pour déterminer le couple $(a, b)$ de paramètre il faut résoudre le système d'équation constitué des équations (1.5), (1.6), (1.7). De la figure 1.11b le point de coordonnées $(2, 3)$ représente les paramètres de la droite euclidienne dans l'espace de paramètre.
La Transformée de Hough est définie dans un espace image continu et ne peut donc pas être directement appliquée à la reconnaissance de ligne droite discrète qui sont une séquence de pixels définient dans $\mathbb{Z}^2$.
M. Dexet [Dex06] étend la transformée de Hough à partir de la généralisation d'espace dual pour reconnaitre des hyperplans analytiques discrets.

**Définition 17.** ***(Dual d'un point [Dex06])*** *Soit $E$ un ensemble, $\gamma(E)$ l'ensemble des parties de $E$.*

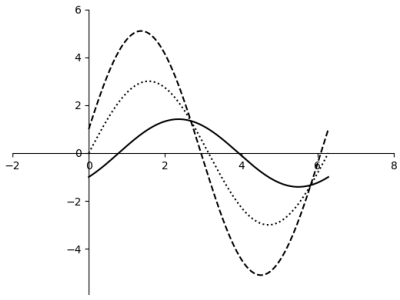
$$\begin{aligned} \mathcal{D}_\mathcal{I} : \quad \mathcal{I}_2 &\longrightarrow \gamma(\mathcal{P}_2) \\ (x_1, x_2) &\longmapsto \{(y_1, y_2) \in \mathcal{P}_2 \mid y_2 = x_2 + x_1 y_1\} \end{aligned}$$




(a) Points




(b) Transformée de Hough et préimage des 3 points



(c) Préimage des 3 points de la droite

FIGURE 1.11 – Transformée de Hough et préimage de 3 points de la droite

*La fonction $\mathcal{D}_{\mathcal{I}}$ associe un point de $\mathcal{I}_2$ à un hyperplan de $\mathcal{P}_2$.*

$$\begin{aligned} \mathcal{D}_{\mathcal{P}} : \quad \mathcal{P}_2 &\longrightarrow \gamma(\mathcal{I}_2) \\ (y_1, y_2) &\longmapsto \{(x_1, x_2) \in \mathcal{I}_2 \mid x_2 = y_2 - x_1 y_1\} \end{aligned}$$

*La fonction $\mathcal{D}_{\mathcal{P}}$ associe un point de $\mathcal{P}_2$ à un hyperplan de $\mathcal{I}_2$, c'est la transformation inverse de $\mathcal{D}_{\mathcal{I}}$. Lorsqu'il n'y aura pas d'ambiguïté on les notera indifféremment Dual.*

**Propriété 1.** *Soient $p$ un point de $\mathcal{I}_2$ ou $\mathcal{P}_2$, et $p'$ un point de $Dual(p)$. Alors $Dual(p')$ est un hyperplan contenant $p$.*

**Preuve :** Soit $p = (p_1, p_2)$ un point de $\mathcal{I}_2$ (le cas ou $p$ est un point de $\mathcal{P}_2$ se démontre de manière similaire). Soit $p' = (p'_1, p'_2)$ un point de $Dual(p)$. Puisque $p' \in Dual(p) = \mathcal{D}_{\mathcal{I}}(p)$, nous avons $p'_2 = p_2 - p_1 p'_1$. D'où $p_2 = p'_2 + p_1 p'_1 \implies p \in Dual(p')$. □

**Définition 18. (Dual d'un objet, préimage généralisée [Dex06])**

*(i) Soit $O$ un ensemble de point de $\mathcal{I}_2$ (resp. $\mathcal{P}_2$). Le dual de $O$, noté $Dual(O)$ est un objet de l'espace de paramètres $\mathcal{P}_2$ (resp. de l'espace image $\mathcal{I}_2$) défini par :*

$$Dual(O) = \bigcup_{p \in O} Dual(p)$$

*(ii) Soit $\mathcal{P} = P_1, \ldots, P_k, k \in \mathbb{N}^*$ un ensemble de $k$ $n-$polytopes (pixels). La préimage généralisée $\mathbb{G}_p(\mathcal{P})$ de l'ensemble de polytopes (pixels) $\mathcal{P}$ est définie par*

$$\mathbb{G}_p(\mathcal{P}) = \bigcap_{i=1}^{n} Dual(P_i)$$

*La préimage est un objet de l'espace de paramètres tel que chaque point de cet objet est associé à un un hyperplan coupant tous les polytopes données.*

**Propriété 2.** *Soient $O_1$ et $O_2$ deux objets de $\mathcal{I}_2$ ou $\mathcal{P}_2$.*

**(i)** $O_1 \subseteq O_2 \implies Dual(O_1) \subseteq Dual(O_2)$

**(ii)** $Dual(O_1 \cup O_2) = Dual(O_1) \cup Dual(O_2)$

**(iii)** $Dual(O_1 \cap O_2) \subseteq Dual(O_1) \cap Dual(O_2)$

**Preuve :** Regarder dans [Dex06]

L'inconvénient de la transformée de Hough est qu'il est impossible de détecter des droites verticales ($a \to \infty$ ou $b \to \infty$) car cela engendrerait un espace de paramètres trop grand. L'algorithme de reconnaissance consiste à déterminer les couples $(a, b)$ de l'espace de paramètre. Pour combler cette insuffisance DUDA et HART ont proposé la transformée standard de Hough [DH72] qui consiste en une représentation sinusoïdale (dans l'espace de paramètres) des points de l'espace image.

**Définition 19.** ***(Transformée de Hough standard)*** *Soient $\mathcal{I}_2 \subset \mathbb{R}^2$ l'espace image c'est à dire une image de largeur $l$, de hauteur $h$ et $(x, y)$ un point de $\mathcal{I}_2$ de paramètres $(\theta, r)$ dans $\mathcal{P}_2$. La transformée de Hough standard de $(x, y)$ est la courbe sinusoïdale $S(x, y)$ dans $\mathcal{P}$ définie par $S(x, y) = \{(\theta, r) \in [0, \pi] \times [-\sqrt{l^2 + h^2}, \sqrt{l^2 + h^2}] / \ r = x\cos\theta + y\sin\theta\}$ .*

Pour $A(1{,}4; 2{,}6) \in \mathcal{I}_2$, la transformée de Hough standard du point $A$ est

$$S(1{,}4; 2{,}6) = \{(\theta, r) \in [0, \pi] \times [-\sqrt{l^2 + h^2}, \sqrt{l^2 + h^2}] / r = 1{,}4\cos\theta + 2{,}6\sin\theta\}$$

Reconnaitre une droite euclidienne consiste à déterminer dans l'espace de paramètres les

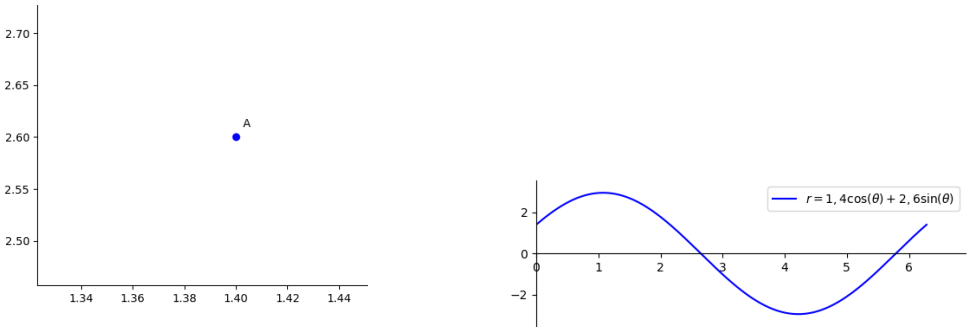



FIGURE 1.12 – Point $A$ et son dual

points d'intersections des duaux des points de la droite dans l'espace image (figure 1.11c). Une droite verticale à pour équation

$$\begin{aligned} y = b,\ b \in \mathbb{R} \quad &\Longrightarrow \quad r = x\cos\theta + y\sin\theta \\ &\Longrightarrow \quad r = y \quad \text{car } \theta = \tfrac{\pi}{2} \text{ modulo } 2\pi \end{aligned}$$

La détection de droite verticale est donc possible.

**Remarque 3.** *Une droite de $\mathcal{I}_2$ est un ensemble de points de $\mathcal{I}_2$.*
*La représentation paramétrique de paramètres $\theta,\ r$ d'équation $r = x\cos\theta + y\sin\theta$ est le produit scalaire entre les vecteurs $\overrightarrow{V}\begin{pmatrix}\cos\theta \\ \sin\theta\end{pmatrix}$ et $\overrightarrow{OM} = \begin{pmatrix}x \\ y\end{pmatrix}$. Donc les points $M$ d'une même droite se projettent sur un même vecteur de coordonnées polaires $(r,\ \theta)$.*

## 1.4 Transformée de Hough standard étendue

Le terme dual désigne ici la transformée de Hough standard et diffère complètement de celui de la définition 17. Nous conservons cependant la définition 18 du dual d'un objet et les propriétés qui en découlent (propriété 2).

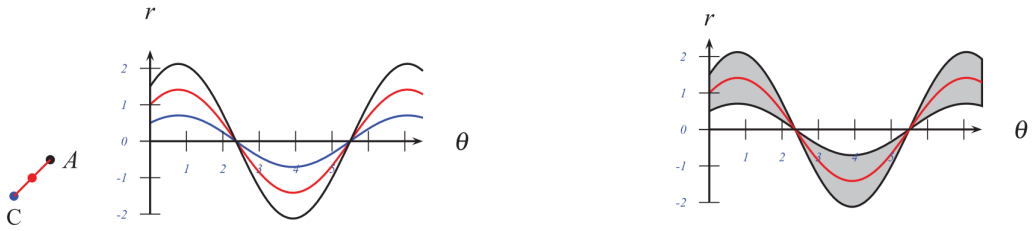
**Définition 20.** ***(Dual d'un pixel [SSA13] figure 1.15b)*** *Soit $p$ un pixel de centre $(p_1, p_2)$ dans l'espace image $\mathcal{I}_2$. Le Dual de $p$ est l'ensemble des points dans l'espace de paramètre $\mathcal{P}_2$ défini par*
$Dual(p) = \{(\theta, r) \in [0, \pi] \times [0, \sqrt{l^2 + h^2}] / \forall (\alpha, \beta) \in [-\frac{1}{2}, \frac{1}{2}]^2, r = (p_1 + \alpha)\cos\theta + (p_2 + \beta)\sin\theta\}$
*Où $l$ est la largeur de l'image et $h$ sa hauteur.*

En effet de la définition 1 un pixel $p$ de centre $(p_1, p_2) \in \mathbb{Z}^2$ est l'ensemble de points :
$\{(x, y) \in \mathbb{R}^2 \ / \ |x - p_1| \leq \frac{1}{2}, \ |y - p_2| \leq \frac{1}{2}\}$

**Théorème 1.** *Ce théorème issus de [SSA13] permet de déterminer de manière optimale le dual d'un segment ou le dual d'un pixel.*

1. *Le dual d'un segment est l'aire délimité par le dual de ses extrémités (figure 1.13).*
2. *Le dual d'un pixel est l'union du dual de ses diagonales (figure 1.14).*

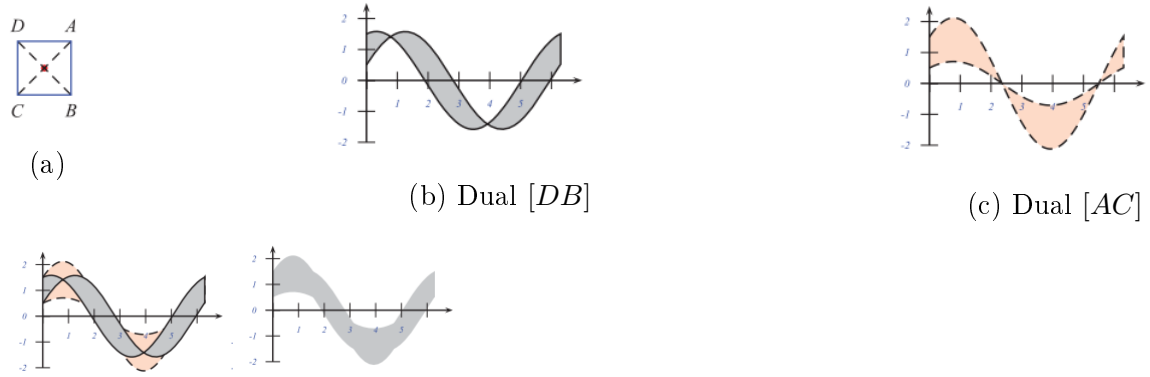
**Preuve :** Regarder dans [SSA13].




(a) $[AC]$ et les duals de ses extrémités et de sa médiatrice

(b) Le dual du $[AC]$

FIGURE 1.13 – Dual d'un segment [SSA13]




(a)

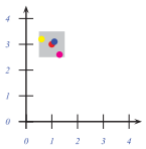
(b) Dual $[DB]$

(c) Dual $[AC]$

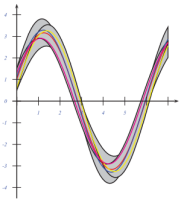
(d) Union des duaux

FIGURE 1.14 – Dual du pixel centré en $(1, 1)$ [SSA13]

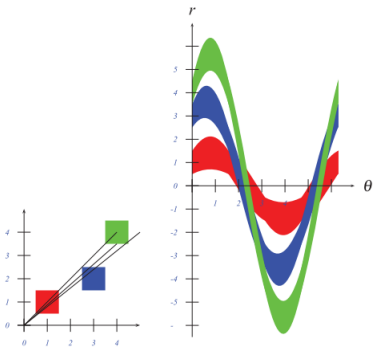
Les figures 1.15a et 1.15b présentent comment calculer le dual d'un rectangle ou d'un carré.



(a) Pixel $(1, 3)$



(b) Dual du pixel $(1, 3)$




(c) Préimage de pixels

FIGURE 1.15 – Transformée de Hough standard étendue de pixels [SSA13]

### 1.4.1 Reconnaissance de droites discrètes

La détection de droites avec la transformée de Hough standard est gourmande est opérations de calculs. Pour limiter la charge de calcul, l'image originale est préalablement limitée aux contours des objets puis rendue binaire. Si des lignes sont présentes dans une image, elles seront alors présentent dans les contours présent dans l'image. Il y a deux méthodes de détections :

— Utilisation d'un accumulateur de données ;
— utilisation de la préimage généralisée.

#### 1.4.1.1 Algorithme de l'accumulateur de donnée

Pour cette méthode chaque axe de l'espace de paramètre est discrétisé selon un pas. L'accumulateur de donnée est une matrice A associée à l'espace de paramètres $(\theta,\ r)$ discrétisé. La matrice d'accumulation est remplie selon l'algorithme 1. Les maxima locaux de cette matrice correspondent à des droites.

**Variables :** A : matrice nulle;
c : image numérique
**début**
  Lire l'image pré-traitée;
  c←contours de l'image pré-traitée;
  p ← pas de discrétisation;
  **pour** *Chaque pixel* $(x, y)$ *de contours* **faire**
    **pour** $\theta$ *de 0 à 180 par pas de* $p$ **faire**
      $r = x\cos\theta + y\sin\theta$;
      $r$=E($r$);
      A$[r, \theta]$= A$[r, \theta]$+1 ;
    **fin**
  **fin**
**fin**

**Algorithme 1 :** Accumulateur de données

Après l'algorithme si A$[r, \theta]$=$\alpha$ signifie que $\alpha$ points de l'image sont alignés sur la droite de paramètre approximatifs $(\theta,\ r)$.

#### 1.4.1.2 Algorithme de la préimage

Le dual d'un polytope $\mathcal{P}$ étant un polytope on déduit de la définition 18 que la préimage $\mathbb{G}_p(\mathcal{P})$ d'un ensemble de polytope est soit vide, soit un polytope.
Si $\mathbb{G}_p(\mathcal{P}) = \emptyset$, l'ensemble de pavé n'appartient pas à un hyperplan discret. Sinon l'ensemble de pavés appartient à un hyperplan discret et $\mathbb{G}_p(\mathcal{P})$ fournit alors l'ensemble des hyperplans euclidiens solutions.
La préimage nous permet d'obtenir un algorithme de reconnaissance d'hyperplans discrets dans les espaces discrets réguliers (indépendamment du modèle de discrétisation utilisé) et isothétiques.

---

**Données :** Un ensemble $S$ de $n$ pixels $p_i$
**début**
  $\mathbb{G}_S \longleftarrow Dual(p_1)$;
  $i \longleftarrow 2$ ;
  **tant que** $\mathbb{G}_S \neq \emptyset$ *et* $i \leq n$ **faire**
    $\mathbb{G}_S \longleftarrow \mathbb{G}_S \cap Dual(p_i)$;
    $i \longleftarrow i + 1$ ;
  **fin**
  **si** $\mathbb{G}_S \neq \emptyset$ **alors**
    $S$ appartient à la droite discrète;
  **sinon**
    $S$ n'appartient pas à la droite discrète;
  **fin**
**fin**

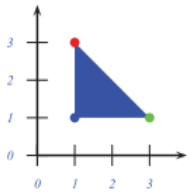
**Algorithme 2 :** Reconnaissance de droite discrète

### 1.4.2 Application au triangle

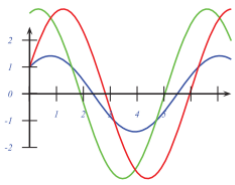
Dans un espace image discrétisé en triangles pour reconnaitre une droite il est nécessaire de savoir calculer le dual d'un triangle.

**Théorème 2.** ***([SSA13])*** *Le dual d'un triangle est l'union du dual de deux de ses côtés adjacents.*

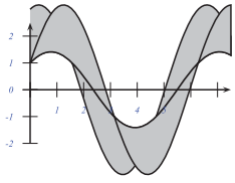
**Preuve :** Regarder dans [SSA13].
Les figures 1.16a à 1.16c présentent comment calculer le dual d'un triangle.



(a) Triangle



(b) Dual des segments du triangle



(c) Dual du triangle

FIGURE 1.16 – Transformée de Hough standard étendue d'un triangle [SSA13]

# Conclusion

La géométrie discrète nous permet de concevoir une image numérique comme un sous espace de $\mathbb{R}^n$. En fonction de la disposition des points lumineux dans l'image on obtient des grilles régulières ou irrégulières. La discrétisation d'un objet dans une grille régulière donne les modèles naïf, standard, supercouverture engendrant chacun un type de droite(hyperplan) particulier.
La reconnaissance de forme dans l'image se fait avec un algorithme de reconnaissance implémenté grâce à la transformée de Hough standard étendue mais nécessite un grand temps de calculs. Comment optimiser cette détection de droites en temps ?

Chapitre 2

# Transformée de Hough rectangulaire



## Introduction

Ces dernières années, la numérisation de la société s'est intensifiée. Parmi les outils les plus concernés par cette révolution numérique, l'image, qui prend une place prépondérante. Les images numériques sont en générales munies de pavés (pixels) qui possèdent une valeur de luminosité qui après pré-traitement grâce aux outils de opencv [1] (voir chapitre 3) par exemple permettent de reconnaitre des objets discrets dans l'image.

Dans ce chapitre il est question de reconnaitre des droites discrètes à l'aide de la transformée de Hough rectangulaire. Pour ce faire, en premier lieu nous définirons les images numériques et donnons quelques propriétés remarquables de celles ci. En second lieu nous proposons un maillage de l'image en pavés (grilles) rectangulaires. Nous terminons avec un algorithme de reconnaissance de droites discrètes à l'aide de la transformée de Hough standard étendue et en fonction d'une certaine densité de pixels allumés dans les pavés rectangulaires définit précédemment.

1. Open Computer Vision

## 2.1 Couleur, image et image numérique matricielle

Pour pouvoir représenter de manière formelle les images, l'homme est partie du modèle de représentation des images dans l'œil humain qu'il a ensuite amélioré au file des besoins et des avancées technologiques. Dans cette section il sera question des images numériques et analogiques, de couleurs et de représentation matricielle des images pour leurs études.

### 2.1.1 Couleur

L'œil est l'organe de sens humain chargé de la vue. Afin que nous voyons les différents éléments (objets, personnes) qui nous entourent la lumière entre dans l'œil et les cônes de la rétine permettent la perception des couleurs. En effet la lumière peut être vue telle un flux de photons (particules lumineuse élémentaires) mais aussi comme une onde, dont la longueur d'onde variant entre 400 et 700 nanomètres, détermine la couleur de l'élément (objet, personne) vu par l'œil.

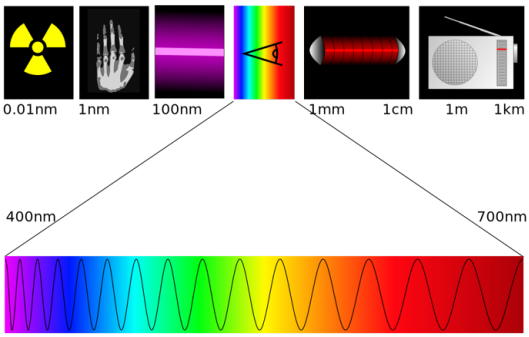



FIGURE 2.1 – Longueur d'onde du spectre de la lumière visible au sein de la gamme d'onde

Pour identifier la couleur, les cônes sont de 03 types (on parle de vision trichromie) et spécialisés pour être sensibles à une plage différente de longueurs d'ondes de la radiation lumineuse. La couleur est alors reconnue par combinaison (R,V,B) de radiation reçue par chacun de ces 03 types de cônes :

- Les cônes (B) sensibles aux longueurs d'ondes courtes (dans les bleus, autour de 437 nm) ;
- Les cônes (V) sensibles aux longueurs d'ondes moyennes (dans les verts, autour de 533 nm) ;
- Les cônes (R) sensibles aux longueurs d'ondes grandes (qualifiés idéalement de rouges mais qui sont plus précisément dans les jaunes, autour de 564 nm).

Les cônes (B) sont les moins nombreux (moins de 10% des cônes), ensuite viennent les cônes (V) puis (R).
Toutes les couleurs n'étant pas représentables par la décomposition (R, V, B) d'autres espaces couleurs ont été formalisés ; pour le passage de la télévision noir-blanc à la télévision couleur l'espace (Y, U, V) fut créé, tel que :

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0,299 & 0,587 & 0,114 \\ -0,14713 & 0,28886 & 0,436 \\ 0,615 & -0,51499 & -1,0001 \end{pmatrix} \begin{pmatrix} R \\ V \\ B \end{pmatrix} \Leftrightarrow \begin{pmatrix} R \\ V \\ B \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1,13983 \\ 1 & -0,39465 & -0,5806 \\ 1 & -0,0025498 & 0 \end{pmatrix} \begin{pmatrix} Y \\ U \\ V \end{pmatrix}$$

### 2.1.2 Image

Nous distinguons deux grands types d'images : les images numérique et analogique.
Dans le jargon du traitement d'image, numérique signifie discret et analogique signifie continue, une image numérique est constituée de petits rectangles dénombrables contenant des pixels.

**Définition 21.** ***(images numériques et analogiques)*** *Une image numérique n'est liée à aucun support et est un ensemble fini de points à valeurs entières appelés pixel .*
*Une image analogique est une image liée au support matériel qui la contient telle qu'un tableau d'art, un pigment de peinture sur une toile.*

#### 2.1.2.1 Echantillonnage d'image numérique

L'échantillonnage est le pavage de l'image numérique en une mosaïque de pixels, qui peuvent être de formes carrée ; rectangulaire etc. Il discrétise tout simplement l'image continue en une suite de pixels élémentaires, utile pour les calculs et traitement sur l'image. Si on agrandit l'image continue on apperçoit les pixels (petits rectangles à la figure 2.2a). L'échantillonnage d'image numérique nous permet d'utiliser les définitions et propriétés du chapitre 1. Si on connait la suite de pixel de l'image numérique on peut facilement la stocker, la transmettre, la dupliquer sans dégradation ; les copies d'une image analogique sont dégradées par rapport à l'image originale du fait de la dépendance de support.
Dans la suite il ne sera question que d'images numériques.

La définition et la résolution d'une image numérique sont deux caractéristiques très précises mais qui sont souvent mal comprises ou confondues. Cette sous-section définie ces deux notions, la définition d'une image sera utilisée dans l'algorithme 9.

**Définition 22.** ***(Définition d'une image)*** *La définition d'une image s'exprime en pixels : c'est simplement le nombre de pixels contenus dans l'image. C'est le produit du nombre de* pixels *sur la hauteur par le nombre de* pixels *sur la largeur.*

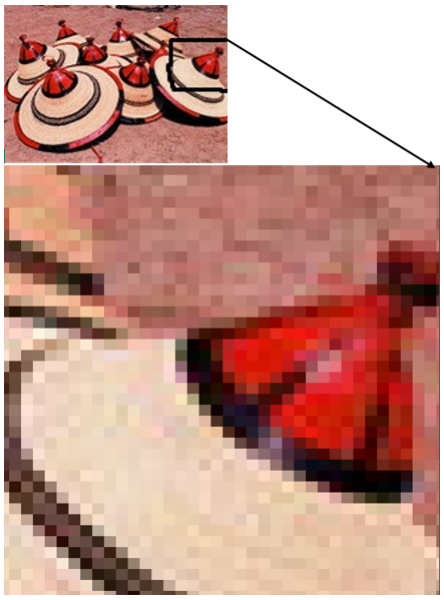
Par exemple un capteur d'appareil photo qui produit 4000 *pixels* sur la ligne de l'image et 3000 sur la colonne aura donc une définition de $4\,000 \times 3\,000$ =12 millions de pixels. Il est bien évident que la taille des fichiers (en kilo-octets) augmente avec le nombre de pixels. Ce n'est pas forcément proportionnel car la plupart des formats graphiques, tels que le JPG, intègrent une compression.

**Définition 23.** ***(Résolution d'une image)*** *La résolution est un nombre de pixels par pouce. Elle s'exprime en dpi (Dots Per Inch), en ppi (Points Per Inch) ou, pour les français en ppp (Points Par Pouce). Ces trois unités sont équivalentes. Sachant qu'un pouce est égal à 2,54cm, on pourrait facilement calculer une résolution en pixels par centimètre mais ce n'est pas dans les habitudes.*

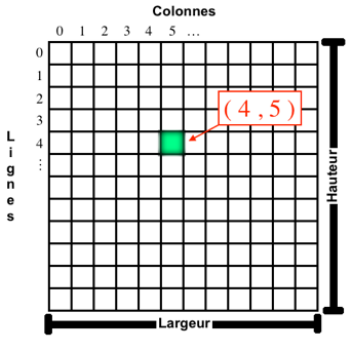
La résolution ne prend vraiment du sens que lorsque l'image est imprimée. Il est inutile de définir ce paramètre dans les paramètres de la photo : la résolution ne sera que la conséquence de la taille d'impression.
Par exemple, si une image de 24 millions de pixels ($6\,000 \times 4\,000$) est imprimée sur du papier de $30 \times 20$ pouces, elle le sera avec une résolution de 200 dpi ($6\,000/30$). Si on veut agrandir le format et imprimer sur du $60 \times 40$ pouces, la résolution sera plus faible et égale à 100 dpi.

### 2.1.3 Image numérique matricielle

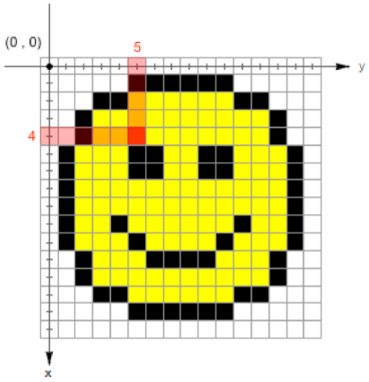
Il existe deux sortes d'images numériques : les images vectorielles et matricielles.
Dans une image vectorielle les données sont représentées par des formes géométriques simples qui sont décrites d'un point de vue mathématique. Par exemple, un cercle est décrit par une information du type (cercle, position du centre, rayon). Ces images sont essentiellement utilisées pour réaliser des schémas ou des plans. Ces images présentent 02 avantages : elles occupent peu de place en mémoire et peuvent être redimensionnées sans perte d'information.
Une image matricielle est formée d'un tableau de points ou pixels , appelé *matrice image*. Plus la densité des points est élevée, plus le nombre d'informations est grand et plus la résolution de l'image est élevée. Par suite la place occupée en mémoire et la durée de traitement seront d'autant plus grandes. Si on agrandit l'image numérique, on peut percevoir les pixels, (figure 2.2a). Les images vues sur un écran de télévision ou une photographie sont des images matricielles. Le nombre de ligne de la matrice est la hauteur de l'image et le nombre de colonne de la matrice, la largeur de l'image (figure 2.2b).



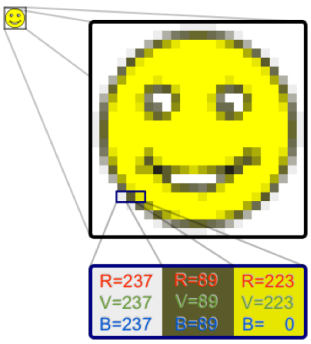
(a) Discrétisation d'une image numérique




(b) Représentation d'un pixel hauteur/largeur




(c) Repère associé à la matrice image




(d) Couleur dans une image numérique

FIGURE 2.2 – Matrice image et couleurs sources Wikipédia

#### 2.1.3.1 Coordonnées de pixels dans la matrice image et repère associé

Dans les matrices images l'indexage des éléments démarrent à 0. Pour désigner la position d'un pixel dans cette matrice, par exemple en parlant de pixel de coordonnées $(4, 5)$, on s'inspire des mêmes conventions que pour les matrices mathématiques :

- La première coordonnée est un indice de ligne, lu sur la hauteur de l'image ;
- La deuxième coordonnée est un indice de colonne, lu sur la largeur de l'image.

Ce pixel est donc celui situé ligne 4, colonne 5 (figure 2.2b).
A cette matrice on peut associer un repère du plan afin de pouvoir utiliser les notions d'abscisses et d'ordonnées tout en conservant l'habitude informatique de l'origine $(0, 0)$ en haut à gauche de l'image. Pour représenter le point $(x, y)$ nous utiliserons la convention ou $x$ est sur la hauteur et $y$ sur la largeur. Ainsi le repère $(x, y)$ est direct nous permettant d'effectuer les opérations mathématiques sans ambiguïté et le point de coordonnées $(4, 5)$ reste bien celui de ligne 4 colonne 5 (figure 2.2c).

#### 2.1.3.2 Représentation des couleurs dans l'image numérique

Les pixels sont les constituants de base de l'image numérique, leurs valeurs (scalaires ou vectorielles) déterminent la couleur de l'image (figure 2.2d).
• La couleur du pixel peut être choisie parmi les intermédiaires entre ces extrêmes, cela donne une image en niveaux de gris. Classiquement la valeur est mémorisée sur un octet, ce qui autorise des valeurs de 0 à 255. Cela permet de représenter l'intensité lumineuse sur 256 niveaux de gris, 0 étant le noir, 255 le blanc, et toute valeur entre les deux représentant un gris intermédiaire. La matrice image est alors une matrice scalaire ;
• Quand chaque pixel est choisi soit noir soit blanc, cela donne une image monochrome. La couleur pourrait donc être mémorisée avec des 0 et des 255, mais par économie on préfère ne pas représenter chacune de ses deux uniques couleurs par un octet (0 ou 255), mais par un seul bit (0 ou 1). La matrice image est alors une matrice scalaire binaire ;
• La couleur du pixel peut être choisie parmi les couleurs visibles par notre œil, cela donne une image en couleurs. la représentation usuelle des couleurs y est directement liée au système visuel humain. On mémorise la quantité de rouge, de vert et de bleu. Comme pour les niveaux de gris, ces quantités respectives sont mémorisées par 3 octets, représentant chacun respectivement le niveau de rouge, le niveau de vert et le niveau de bleu d'un pixel. La matrice image est alors une matrice vectoriel (R,V,B).
Dans les lignes suivantes nous déterminerons comment reconnaitre des droites dans une image 2D de manière optimale en proposant un maillage de l'image numérique matricielle en rectangles et à l'aide de la transformée standard de Hough (point-courbe).

## 2.2 Algorithmes de la transformée de Hough rectangulaire

Dans cette section on va détecter des droites discrètes dans des grilles rectangulaire à l'aide de la transformée standard de Hough.

**Définition 24.** ***(Dual d'un rectangle)*** *Le dual d'un rectangle est le dual des pixels contenus dans ce rectangle. A l'aide de la transformée de Hough standard étendue [SSA13]*

*le dual d'un rectangle se résume à l'aire de la surface contenue entre les duals des extrémités des diagonales du rectangle (figure 1.14).*

IL est possible de détecter des droites avec la transformée de Hough en calculant le dual de tous les pixels allumés dans l'image. Dans un soucis d'optimisation, nous établirons un maillage de l'image en rectangles (plus ou moins grand). Puis nous calculerons la transformée standard de Hough des différents rectangles de l'image en fonction de la proportion de pixels allumés dans les rectangles. En effet cela permet d'optimiser la reconnaissance de droite(s) car au lieu de calculer le dual de tous les pixels allumés dans l'image nous ne calculerons que le dual des pixels contenus dans les rectangles ayant un certain pourcentage de pixels allumés. Nous terminerons avec la reconnaissance de la droite à l'aide de l'algorithme 2 ou d'un autre que nous présenterons.

L'algorithme de la transformée de Hough rectangulaire (algorithme 3) s'implémente en 04 grandes étapes après avoir effectuer certains pré-traitement de l'image (détection de contours, lissage,...), qui permet d'éliminer d'éventuels bruits dans l'image, d'optimiser encore plus les calculs.

---

**Données :** Une image numérique pré-traitée

**début**

1) Générer la grille rectangulaire;
2) sélectionner les rectangles qui contiennent au moins $\alpha\%$ pixels allumés ;
3) reconnaitre la(les) droite(s) dans l'image en fonction des rectangles sélectionnés ;
4) identifier les droites analytiques.

**fin**

**Algorithme 3 :** Transformée de Hough rectangulaire

---

*Opencv* permet d'obtenir le nombre de lignes ou de colonnes d'une image numérique.
Notons Nl le nombre de lignes et Nc le nombre de colonnes de l'image.
Chaque ligne de commentaire dans les algorithmes sera précédé du symbole "%"

### 2.2.1 Génération de la grille rectangulaire

Là nous créons une grille rectangulaire en subdivisant l'image numérique en plusieurs rectangles. Du fait que Nl ou Nc soit un nombre entier on ne peut le diviser par un nombre quelconque comme on le fait par exemple en calcul d'aire, en calcul intégral. Pour effectuer ce maillage on pourra être amené à travailler avec des nombres premiers, ce qui conduira aux algorithmes de primalité d'un entier.

#### 2.2.1.1 Primalité

Le problème de la preuve de primalité d'un nombre entier est très vieux, les mathématiciens écrivent de beaux algorithmes théoriques [Ber04, Ezo13] mais difficiles à implémenter. Les plus rapides de ces algorithmes sont probabilistes [APR83]. Ayant besoin de connaitre avec exactitude si Nl ou Nc est premier ou composé, nous utiliserons l'algorithme déterministe d'Eratosthène [Lir11].

```
Fonction erast(n : entier) : entier
    Pré-condition : n>1
    Variables : r, k : entiers
    k ← 2;
    e← E(√n) ;
    tant que k ≤ e faire
        r ← n  mod k;
        si r = 0 alors
            retourner 0 ;          % k divise n, n n'est donc pas premier
        fin
        k ← k + 1 ;
    fin
    retourner 1 ;                  % n est alors premier
fin
```

**Algorithme 4 :** Fonction d'Eratosthène de primalité d'un entier

**Commentaire sur l'algorithme 4**
Dans cet algorithme on recherche un diviseur inférieur ou égale à la partie entière de la racine carrée du nombre dont on étudie la primalité, s'il admet un diviseur alors il est composé sinon il est premier. Notons $\mathbb{P}$ l'ensemble des nombres premiers.

#### 2.2.1.2 Factorisation d'entiers

Un nombre composé s'écrit comme le produit de ses facteurs premiers. Pour obtenir Nl=$k_1 \times k_2$ nous décomposerons Nl en facteurs premiers grâce au crible Eratosthène puis nous déterminerons ses diviseurs maximaux qui sont $k_1$ et $k_2$ avec l'algorithme 5. Cette fonction appliquée à Nc nous permet de factoriser Nc en Nc $= k_3 \times k_4$.

**Fonction** *factorisation(Nl : entier) : couple d'entiers*
**Pré-condition :** $Nl \notin \mathbb{P}$
**Variables :** $k,\ k_1,\ k_2$ : entiers
tab : tableau d'entiers
**Sorties :** $k_1,\ k_2$ : entiers
$k \leftarrow 2$;
e$\leftarrow E(\sqrt{Nl})$ ;
**tant que** $k \leq e$ **faire**
**si** $Nl \mod k = 0$ *et erast(k)=Vrai* **alors**
ajouter $k$ dans tab % tab est ordonné et contient les diviseurs premiers de $Nl$
**fin**
$k \leftarrow k + 1$;
**fin**
%on divise par 2 pour ne pas considéré un diviseur premier trop petit ou trop grand de $Nl$
$k \leftarrow E(longueur(tab)/2)$;
$k_2 \leftarrow tab[k]$;
$k_1 \leftarrow Nl/k_2$;
**retourner** $k_1,\ k_2$ ; % $Nl = k_1 \times k_2$, $k_1$ *et* $k_2$ n'ont nécessairement premier
**fin**

**Algorithme 5 :** Fonction de factorisation d'un entier composé

**Commentaire sur l'algorithme 5**
La fonction "factorisation" prend en entrée le nombre de lignes Nl. Dans le cas où Nl est un entier premier Nl=1×Nl, sa factorisation est donc connue et la fonction ne sert à rien. De ce fait nous prenons en pré-condition Nl non premier.
La variable $k$ parcourt l'ensemble des entiers inférieurs à la partie entière de la racine carrée de Nl. On initialise $k$ à 2 car 1 est diviseur de n'importe quel entier et 0 d'aucun entier.
Le bloc d'instruction "si" nous permet de déterminer les diviseurs $k$ de Nl (Nl mod $k$=0) avec $k$ premier (erast($k$)=Vrai). Dans "tab" on range par ordre croissant les diviseurs de Nl trouvés .
A la fin de la boucle "tant que" dans le but d'obtenir un maillage ayant un nombre de grilles $k_2$ assez grand on prend $k_1$ comme étant l'élément situé à l'indice : partie entière de la moitié du nombre d'éléments de tab. Pour obtenir $k_2$ on divise Nl par $k_1$, Nl est ainsi subdivisé en $k_1$ partie de longueur $k_2$.
Dans le cas de Nc= $k_3 \times k_4$, $k_3$ est le nombre de subdivision de Nc et $k_4$ est le nombre de colonnes par subdivision.

**Exemple 1 (utilisation de l'algorithme 5)**
Déroulement de l'algorithme pour deux valeurs de $Nl$.

| | |
|---|---|
| On prend $Nl = 100$ | On prend $Nl = 462$ |
| e=$E(\sqrt{100}) = E(10) = 10$ | e=$E(\sqrt{462}) = E(21,49) = 21$ |
| Les diviseurs de 100 inférieurs à 10 sont 2 et | Les diviseurs de 462 inférieurs à 21 sont 2, 3,7 |
| 5 donc tab=[2,5] | et 11 donc tab=[2,3,7,11] |
| $k = E(longueur(tab)/2) = 1$ | $k = E(longueur(tab)/2) = 2$ |
| $k_2 = tab[1] = 5$ | $k_2 = tab[2] = 7$ |
| on retourne 20 et 5 c'est à dire 100=20×5 | on retourne 66 et 7 c'est à dire 462=66×7 |

On sait que l'ensemble des entiers pair et impaire forment une partition de l'ensemble des entiers naturels. Si on considère donc le premier élément du tableau "tab" on aura dans 50% des cas 2 comme facteur de $Nl$ par notre fonction de factorisation. Si on considère le dernier élément du tableau "tab" on pourrait obtenir dans la fonction de maillage (algorithme 6) des mailles trop grandes. Pour mieux contrôler la taille des grilles rectangulaire issus de algorithme 6 on effectué les instructions suivant la boucle "tant que".

---

**Fonction** *maillage(Nl, Nc : entiers) : deux tableaux d'entiers*
- **Variables :** tab : tableau de 4 entiers, tabool : tableau de 2 booléens
- **Sorties :** tab[1] (tab[3]) : largeur (hauteur) du rectangle de notre maillage
  tab[0] (tab[2]) : nombre de grille sur la ligne (la colonne)
  tabool[0] (tabool[1]) : primalité de Nl (Nc)
- *tabool[0]←erast(Nl);*
- **si** *tabool[0]* **alors**
  - *tab[0], tab[1]←factorisation(Nl-1) ;*
- **sinon**
  - *tab[0], tab[1]←factorisation(Nl) ;*
- **fin**
- *tabool[1]←erast(Nc);*
- **si** *tabool[1]* **alors**
  - *tab[2], tab[3]←factorisation(Nc-1) ;*
- **sinon**
  - *tab[2], tab[3]←factorisation(Nc) ;*
- **fin**
- **retourner** *tab,tabool ;*

**fin**

**Algorithme 6 :** Fonction de maillage rectangulaire

**Commentaire sur l'algorithme 6**
Dans les blocs "si", Nl est premier (Nl$\in \mathbb{P}$) d'où Nl$-1 \notin \mathbb{P}$ et a donc deux facteurs maximaux $k_1$ et $k_2$. L'instruction "tab←factorisation(Nl)" fournit $k_2$ et $k_1$ de sorte qu'on ait exactement $k_1$ hauteurs de longueur $k_2$ stockées respectivement dans tab[0] et tab[1] et une ligne comme résidu qui sera considérée lors de la reconnaissance de droite. Pareille pour Nc mais les valeurs sont stockées cette fois dans tab[2] et tab[3] avec une colonne comme résidu qui sera

considérée lors de la reconnaissance de droite.
Dans les "sinon", "tabool[0]=Faux" alors Nl est non premier (Nl$\notin \mathbb{P}$), il existe donc des entiers $k_1$ et $k_2$ tels que Nl=$k_1 \times k_2$. L'instruction "tab←factorisation(Nl)" fournit $k_2$ et $k_1$ de sorte qu'on ait exactement $k_1$ largeurs de longueur $k_2$ stockée respectivement dans tab[0] et tab[1]. Pareille pour Nc mais les valeurs sont stockées cette fois dans tab[2] et tab[3].

**Exemple 2 (utilisation de l'algorithme 6)**
Réalisons notre maillage rectangulaire pour une image où $Nl = 100$ et $Nc = 462$.
100 n'est pas premier donc tabool[0]=0, on obtient alors tab[0]=20 et tab[1]=5.
Sur la hauteur de l'image on aura 20 subdivisions de longueur 5 chacune. 462 n'est pas premier donc tabool[1]=0, on obtient alors tab[2]=66 et tab[3]=7.
Sur la largeur de l'image on aura 66 subdivisions de longueur 7 chacune.

**Remarque 4.** *Chaque grille issus du maillage de l'algorithme 6 à pour :*
- *Largeur, $h_1$=tab[1] ;*
- *hauteur, $l_1$=tab[3] ;*

*Dans le cas de l'exemple 2 on a : $h_1$=5 ; $l_1$=7.*
*La matrice image sans résidus issues de ce maillage à pour :*
- *Largeur, h=tab[0]×tab[1] ;*
- *hauteur, l=tab[2]×tab[3].*

### 2.2.2 Sélection des rectangles et reconnaissance de droite(s)

Possédant une image munie de la grille rectangulaire de la sous-section 2.2.1 nous sélectionnerons les rectangles ayant un certain taux $\alpha$ de pixels allumés ou éteint avec la fonction "comptage" (algorithme 7) ; par exemple au moins $\alpha$ avec $\alpha$ saisie par l'utilisateur. Ce qui nous permettra à l'étape suivante de détecter la présence de droite(s) soit en utilisant un accumulateur de données [SOZ18] soit en calculant le dual de ces rectangles avec l'algorithme 2. Il ne sert en effet à rien de calculer le dual de tous les rectangles définies par notre maillage dans l'image car une région sans pixels allumés traduit l'absence de formes paramétriques. De plus la transformée de Hough étant robuste au bruit si nous ne déterminons pas le dual d'un rectangle ayant une faible résolution cela n'aura pas d'incidence sur les résultats attendus.

**Remarque 5.** *Nous savons qu'un rectangle peut être repéré par :*
- *Son centre et ses dimensions dans ce document la hauteur et la largeur ;*
- *la position du coin supérieur gauche et du coin inférieur droit ;*
- *la position de l'un de ses coins et ses dimensions.*

*Dans les algorithmes 9 et 12 les rectangles sont repérés par la position du coin supérieur gauche et du coin inférieur droit.*

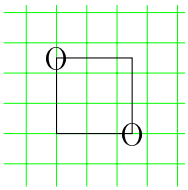


FIGURE 2.3 – Modèle de rectangle utilisé dans les algorithmes 9 et 12

```
Fonction comptage(A, C : points du rectangle, n : entier, img : image) : réel
    Variables : k, l : entiers
    k ←0;
    pour x entre x_A et x_C faire
        pour y entre y_A et y_C faire
            si img[x,y]=n alors
                k ← k + 1 ;        %on compte les pixels ayant la valeur n
            sinon
                l ← l + 1 ;        %on compte les pixels n'ayant pas la valeur n
            fin
        fin
    fin
    retourner k/(k+l) ;
fin
```

**Algorithme 7 :** Fonction de comptage du nombre de pixels ayant la valeur $n$

**Commentaire sur l'algorithme 7**
Le point $A(x_A, y_A)$ est le point supérieur gauche et $C(x_C, y_C)$ le point inférieur droit du rectangle considéré.
Le paramètre $n$ représente la valeur des pixels dans la matrice image, par exemple dans une image binaire si l'on souhaite compter les pixels allumés on prend $n = 1$, les pixels éteints $n = 0$. Pour une image en niveaux de gris $n \in \{0,\ 255\}$. Il est mis pour rendre la fonction la plus générale possible.
Le paramétre img représente une image qu'elle soit binaire, en niveau de gris ou en couleur.
Cette fonction retourne le taux de pixels allumés dans un rectangle.

#### 2.2.2.1 Méthode de la préimage

Dans l'algorithme 9, on met en œuvre ce qui vient d'être dit en parcourant la grille obtenue à partir des sorties de la fonction "maillage" (algorithme 6) tout en nous intéressant au cas particuliers des résidus. L'algorithme 8 permet de reconnaitre des droites dans une image à partir de pixels ou d'objets discrets en utilisant la notion de préimage généralisée $\mathbb{G}_p$.

**Commentaire sur l'algorithme 8**
S est un ensemble de n objets discret (rectangles ou pixels résiduels). Si les points sont ajouter dans n'importe quel ordre dans S, l'algorithme ne permet pas de détecter toutes les droites dans l'image ; par contre mis dans un ordre judicieux, il le fait. L'algorithme dépend donc de l'ordre d'ajout des rectangles.

**Données :** S : Ensemble de $n$ objets
**Post-condition :** Si U contient un rectangle alors ce rectangle est une droite
**Variables :** d, i : entiers ; U : ensemble, $\mathbb{G}_p$ : objet discret ;
**début**
  $\mathbb{G}_p \longleftarrow Dual(S_1)$;
  U$\leftarrow\{S_1\}$, $i \leftarrow 1$ ; % i compte le nombre d'objets discrets traités
  d$\leftarrow$0 ; % d compte le nombre de droites détectées
  **tant que** $i \leq n$ **faire**
    $\mathbb{G}_p \longleftarrow \mathbb{G}_p \cap Dual(S_i)$;
    **si** $\mathbb{G}_p = \emptyset$ **alors**
      d$\leftarrow$d+1;
      $\mathbb{G}_p \longleftarrow Dual(S_i)$;
      Traiter(U) ; % par exemple retourner la position, colorer les rectangles concernés
    **fin**
    U$\leftarrow$U $\cup \{S_i\}$;
    i$\leftarrow i+1$ ;
  **fin**
  **retourner** *d ;*
**fin**

**Algorithme 8 :** Extension de l'algorithme 2 de Reconnaissance de droites

---

**Données :** $\alpha$ : réel, n : entier, img : image numérique binaire
**Pré-condition :** $0 \leq \alpha \leq 1$
**Variables :** $x$, $y$, h, l, d : entiers ; A,B,C,D : couple d'entiers
  tab : tableau de 4 entiers ; tabool : tableau de 2 booléens
  S : ensemble
Nl $\leftarrow$ nombre de lignes de l'image ; Nc $\leftarrow$ nombre de colonnes de l'image;
tab, tabool=maillage(Nl,Nc) ; S $\leftarrow \emptyset$ ; % tab[i]=$k_{i+1}, \forall i \in \{0,1,2,3\}$;
% l et h représentent les produits des facteurs maximaux de Nl et Nc
%d représente la définition d'une grille de notre maillage
h $\leftarrow$ tab[0]$\times$ tab[1], l $\leftarrow$ tab[2]$\times$ tab[3], d $\leftarrow$ tab[1]$\times$ tab[3];
**début**
  %Parcours de la matrice image sans résidus
  **pour** *x entre* 0 *et h-1 par pas de tab[3]* **faire**
    **pour** *y entre* 0 *et l-1 par pas de tab[1]* **faire**
      A=$(x, y)$ ; C=$(x + tab[3], y + tab[1])$
      **si** *comptage(A, C,img)*$\geq \alpha$ **alors**
        B=$(x,\ y + tab[1])$;
        D=$(x + tab[3],\ y)$;
        Mettre(ABCD) dans S ; % On sélectionne le rectangle ABCD
      **fin**
    **fin**
  **fin**

```
    % parcours et traitement des éventuels pixels résiduels
    % à la sortie x = h − 1 et y = l − 1
    si tabool[0]=1 et tabool[1]=0 alors
        % Nl ∈ ℙ il y a une ligne de plus à traiter
        % on est la dernière hauteur traitée
        x ← h;
        pour y entre 0 et l-1 par pas de 1 faire
            si img[x,y]=n alors
                Mettre le pixel (x,y) dans S ;
            fin
        fin
    fin
    si tabool[0]=0 et tabool[1]=1 alors
        % Nc ∈ ℙ il y a une colonne de plus à traiter
        % on est la dernière largeur traitée
        y ← l;
        pour x entre 0 et h-1 par pas de 1 faire
            si img[x,y]=n alors
                Mettre le pixel (x,y) dans S ;
            fin
        fin
    fin
    si tabool[0]=1 et tabool[1]=1 alors
        % Nl ∈ ℙ, Nc ∈ ℙ il y a une colonne et une ligne de plus à traiter
        y ← l;
        pour x entre 0 et h-1 par pas de 1 faire
            si img[x,y]=n alors
                Mettre le pixel (x,y) dans S ;
            fin
        fin
        x ← h;
        pour y entre 0 et l-1 par pas de 1 faire
            si img[x,y]=n alors
                Mettre le pixel (x,y) dans S ;
            fin
        fin
    fin
    reconnaissance_droite(S) ;                    % Reconnaissance de droites
fin
```

**Algorithme 9 :** Dual des rectangles ayant au moins $\alpha$% de pixels allumés

**Commentaire sur l'algorithme 9**
La boucle "pour" parcours les grilles de l'image à la recherche de celles ayant un taux de pixels atteignant au moins la valeur de $\alpha$ et met ces grilles dans l'ensemble S.
Les trois structures conditionnelles "si" traitent les cas où le nombre de lignes ou de colonnes de l'image est un nombre premiers dans le but de sélectionner les pixels ayant pour valeur "n" avec n$\in \{0, 1\}$ pour les mettre dans l'ensemble S. Pour n=0, on a une image binaire à fond blanc et les droites qu'on recherches sont en noires ; c'est l'inverse pour n=1.
La fonction se termine par un appelle à l'algorithme de reconnaissance de droites (algorithme 8), qui cette fois sera implémentée avec des objets. Cela n'affectera en rien le résultat, on y calculera la transformée standard de Hough des différents rectangles sélectionnés.

#### 2.2.2.2 Calcul de l'accumulateur de données

L'utilisation d'un accumulateur de données permet de détecter toutes les droites présentent dans une image, son inconvénient majeur étant son importante taille et la lenteur de son exécution une fois programmer. SERE et al dans [SOZ18] l'on optimisé, dans ce mémoire on réduira surtout sa taille à l'aide du maille rectangulaire effectué.
L'algorithme de reconnaissance de droites discrètes avec l'accumulateur de données utilisera l'algorithme de comptage de pixels allumés ou éteints dans une grille de notre maillage et les algorithmes ci-dessous.

##### 2.2.2.2.1 Fonction de définition de l'accumulateur

Cette fonction a pour but de créer une matrice nulle servant d'accumulateur de données. Elle prend en compte tous les cas de figure concernant la primalité du nombre de lignes et de colonnes de la matrice image. Le nombre de lignes et de colonnes de l'accumulateur de données est fonction du maillage rectangulaire de la section 2.2.1 effectué sur l'image et prend en entrée les paramètres retournée par la fonction de maillage et retourne la matrice d'accumulation, le nombre de lignes et de colonnes de cette matrice. Pour une matrice M, $n \times m$ signifie que M a $n$ lignes et $m$ colonnes.

---

**Fonction** *DefAccu(tab, tabool)*
| **Pré-condition :** Exécuter tab, tabool←maillage(Nl,Nc)
| **Variables :** acc : matrice d'entiers
| **si** *tabool[0]=0 et tabool[1]=0* **alors**
| | **retourner** *Une matrice nulle tab[0]×tab[2], tab[0], tab[2] ;*
| **sinon si** *tabool[0]=1 et tabool[1]=0* **alors**
| | **retourner** *Une matrice nulle (tab[0]+1)×tab[2], tab[0]+1, tab[2] ;*
| **sinon si** *tabool[0]=0 et tabool[1]=1* **alors**
| | **retourner** *Une matrice nulle tab[0]×(tab[2]+1), tab[0], tab[2]+1 ;*
| **sinon**
| | **retourner** *Une matrice nulle (tab[0]+1)×(tab[2]+1), tab[0]+1, tab[2]+1 ;*
| **fin**
**fin**

**Algorithme 10 :** Fonction de définition de l'accumulateur de donnée

**Commentaire sur l'algorithme 10**
La fonction DefAccu retourne une matrice nulle et ses dimensions.
En considérant le cas où tabool[0]=0 et tabool[1]=0 la fonction DefAccu retourne une matrice nulle de tab[0] lignes, tab[2] colonnes ainsi que ses dimension tab[0] et tab[2]
Pour Nl=100 et Nc=462, la fonction DefAccu retourne une matrice nulle 20×66 c'est à dire de 20 lignes et 66 colonnes.

#### 2.2.2.2.2 Fonction de recherche de maxima dans une matrice

Les droites dans l'accumulateur de données correspondent aux maxima de cette matrice d'accumulation, il est alors nécessaire de déterminer les maxima de la matrice d'accumulation d'autant plus que cela nous permettra de représenter les droites détecter si nécessaire. Pour ce faire on annule les valeurs de la matrice d'accumulation inférieur à un seuil donné, les valeurs non nulles représenteront les maxima de l'accumulateur de données.

---

**Données :** seuil : entier, accu : matrice
**Pré-condition :** 0 < seuil
**Variables :** i, j, Nl, Nc :entier
Nl← nombre de lignes de accu;
Nc← nombre de colonnes de accu;
**pour** *i entre 0 et Nl-1 par pas de 1* **faire**
  **pour** *j entre 0 et Nc-1 par pas de 1* **faire**
    **si** *accu[i, j] < seuil* **alors**
      accu[i, j]←0
    **fin**
  **fin**
**fin**

**Algorithme 11 :** Recherche des maxima dans l'accumulateur

**Commentaire sur l'algorithme 12**
**(Voir l'algorithme à la page suivante)**
La boucle "pour" parcours les grilles de l'image à la recherche de celles ayant un taux de pixels atteignant au moins la valeur de $\alpha$ et calcul le dual des pixels contenus dans ces grilles. Cela nous permet de ne pas considérer le facteur de division de la longueur du tableau tab comme un paramètre à l'algorithme 5.
Les trois structures conditionnelles "si" traitent les cas où le nombre de lignes ou de colonnes de l'image est un nombre premiers dans le but de sélectionner les pixels ayant pour valeur "n" avec n$\in \{0, 1\}$ afin de calculer leurs dual.

**Pré-condition :** $0 < \alpha < 1$, 0<seuil
**Données :** $\alpha$ : réels, n : entier,seuil :entier, img : image numérique
**Variables :** x, y, h, l, d,Nl, Nc, tab, tabool : entiers;
acc : matrice d'entiers;
A, C : couple de points;
theta, r, theta_max, r_max, itheta, ir : réels;

**début**

Nl←nombre de lignes de l'image ; Nc←nombre de colonnes de l'image ;
tab, tabool←maillage(Nl, Nc) ;
h←tab[0]×tab[1] ; l←tab[2]×tab[3] ; d←tab[1]×tab[3] ;
theta_max=$\pi$ ; $r = \sqrt{Nl^2 + Nc^2}$ ; $\%(theta, r) \in [0, \pi] \times [-\sqrt{l^2 + h^2}, \sqrt{l^2 + h^2}]$
% Création de la matrice d'accumulation (algorithme 10) et affectation de se dimensions
acc, r_dim, theta_dim←DefAccu(tab, tabool) ;
% Parcours de toutes les grilles de l'image sans tenir compte d'éventuels résidus
**pour** *x entre 0 et h-tab[1]-1 par pas de tab[1]* **faire**
  **pour** *y entre 0 et l-tab[3]-1 par pas de tab[3]* **faire**
    A=(x, y) ; C=(x+tab[1], y+tab[3]) ;
    **si** *comptage(A, C, n, img)*$\geq \alpha$ **alors**
      %Calcul du dual de tous pixels contenus dans le rectangle vérifiant la condition sur $\alpha$
      **pour** *z entre x et x+tab[1] par pas de 1* **faire**
        **pour** *k entre y et y+tab[3] par pas de 1* **faire**
          % Mise à jour de l'accumulateur et traçée du dual des pixels ayant la valeur de luminosité $n$ du rectangle
          **pour** *itheta entre 0 et theta_ dim-1 par pas de 1* **faire**
            **si** *img[z,k]* $\neq n$ **alors** Ne rien faire;
            theta← $\dfrac{itheta \times theta\_max}{theta\_dim}$;
            r← $z\cos(theta) + k\sin(theta)$ ;
            ir← $\dfrac{r\_dim \times r}{r\_max}$;
            ir←E(ir) ; itheta←E(itheta) ;
            acc[ir, itheta]←acc[ir, itheta]+1 ;
          **fin**
        **fin**
      **fin**
    **fin**
  **fin**
**fin**

% parcours et traitement des éventuels pixels résiduels
% à la sortie $x = h - 1$ et $y = l - 1$
**si** *tabool[0]=1 et tabool[1]=0* **alors**
  % Nl $\in \mathbb{P}$ il y a une ligne de plus à traiter
  x=h;
  **pour** *y entre 0 et l-1* **faire**
    % Sélection et traitement des pixels ayant la valeur n
    **si** *img[x,y]=n* **alors**
      % Mise à jour de l'accumulateur de données
      **pour** *itheta entre 0 et theta_ dim-1 par pas de 1* **faire**
        theta$\leftarrow \dfrac{itheta \times theta_max}{theta_dim}$;
        r$\leftarrow x\cos(theta) + y\sin(theta)$ ;
        ir$\leftarrow \dfrac{r_dim \times r}{r_max}$;
        ir$\leftarrow$E(ir) ; itheta$\leftarrow$E(itheta) ;
        acc[ir, itheta]$\leftarrow$acc[ir, itheta]+1 ;
      **fin**
    **fin**
  **fin**
**fin**
**si** *tabool[0]=0 et tabool[1]=1* **alors**
  % Nc $\in \mathbb{P}$ il y a une colonne de plus à traiter
  y=l;
  **pour** *x entre 0 et h-1 par pas de 1* **faire**
    % Sélection et traitement des pixels ayant la valeur n
    **si** *img[x,y]=n* **alors**
      % Mise à jour de l'accumulateur de données
      **pour** *itheta entre 0 et theta_ dim-1 par pas de 1* **faire**
        theta$\leftarrow \dfrac{itheta \times theta_max}{theta_dim}$;  $r \leftarrow x\cos(theta) + y\sin(theta)$ ;
        ir$\leftarrow \dfrac{r_dim \times r}{r_max}$;
        ir$\leftarrow$E(ir) ; itheta$\leftarrow$E(itheta) ;
        acc[ir, itheta]$\leftarrow$acc[ir, itheta]+1 ;
      **fin**
    **fin**
  **fin**
**fin**

**si** *tabool[0]=1 et tabool[1]=1* **alors**
x←h;
% Nl ∈ ℙ, Nc ∈ ℙ il y a une colonne et une ligne de plus à traiter
**pour** *y entre 0 et l par pas de 1* **faire**
% Sélection et traitement des pixels ayant la valeur n
**si** *img[x,y]=n* **alors**
% Mise à jour de l'accumulateur de données
**pour** *itheta entre 0 et theta_ dim-1 par pas de 1* **faire**
theta← $\frac{itheta \times theta\_max}{theta\_dim}$; $r \leftarrow x\cos(theta) + y\sin(theta)$ ;
ir← $\frac{r\_dim \times r}{r\_max}$;
ir←E(ir) ; itheta←E(itheta) ;
acc[ir, itheta]←acc[ir, itheta]+1 ;
**fin**
**fin**
**fin**
y←l;
**pour** *x entre 0 et h-1 par pas de 1* **faire**
% On prend x entre 0 et h-1 pour ne pas traiter 2-fois le pixel situé à la position (Nl, Nc) dans la matrice image
% Sélection et traitement des pixels ayant la valeur n
**si** *img[x,y]=n* **alors**
% Mise à jour de l'accumulateur de données
**pour** *itheta entre 0 et theta_ dim-1 par pas de 1* **faire**
theta← $\frac{itheta \times theta\_max}{theta\_dim}$; $r \leftarrow x\cos(theta) + y\sin(theta)$ ;
ir← $\frac{r\_dim \times r}{r\_max}$;
ir←E(ir) ; itheta←E(itheta) ;
acc[ir, itheta]←acc[ir, itheta]+1 ;
**fin**
**fin**
**fin**
**fin**
Etape de recherche des maxima dans la matrice d'accumulation (algorithme 11) ;
Traçée des droites (si souhaité) ;
**fin**

**Algorithme 12 :** Reconnaissance de droites discrètes avec accumulateur

# Conclusion

Dans ce chapitre nous sommes entrés en plein dans notre sujet en parlant des images à travers les définitions des 02 grands types d'images. Pour ensuite nous restreindre au cas des images numériques dont nous avons donner des définitions, propriétés et algorithmes dans le but d'optimiser la détection de droites dans une image numérique.

Les algorithmes de ce chapitre permettent de se rendre compte de certaines insuffisances de l'algorithme 2 de reconnaissance de droite, qui nous ont conduit à écrire l'algorithme 8. La transformée de Hough étant gourmande en ressource lors de son implémentation nous avons réalisé un maillage rectangulaire de l'image afin d'optimiser la détection de droites avec cette transformée.
Pour corroborer nos résultats théoriques présentés dans ce chapitre il sera fait dans le chapitre 3 l'implémentation ainsi qu'une simulation des différents algorithmes de ce chapitre en langage python.

Chapitre 3

# Programmation, simulation numérique



## Introduction

Dans ce chapitre nous programmerons, simulerons les algorithmes du chapitre 2 dans le but de vérifier si nos résultats théoriques sont programmables et donne des résultats en temps satisfaisant sans grandes influences des paramètres des algorithmes.
Nous utiliserons pour ce faire le langage de programmation python et le logiciel de traitement d'image GIMP dont un apperçu des différentes librairies et fonctionnalité sera donné. Puis nous évoquerons quelques pré-traitements d'images nécessaires dans la réduction de lenteur des calculs de la transformée de Hough. Nous terminerons par l'implémentation de l'algorithme de reconnaissance de droites 12 tout en ayant l'œil sur l'influence des différents paramètres sur le résultat.

## 3.1 Outils de programmation ou de simulation

Cette section vise à établir l'exactitude des résultats des chapitres précédents et leurs applicabilités dans le monde réel au moyen de tests sur des images. Pour y parvenir on utilisera la programmation scientifique qui sert à résoudre des problèmes mathématiques à l'aide d'algorithmes. Les implémentations seront faites en python 3 et avec la librairie de traitement d'image opencv pouvant être implémentée en python ou c++. Les différentes implémentations et simulations seront effectuées en python 3.6 avec un ordinateur portable de caractéristiques :

— Processeur : AMD A8-6410 APU with AMD Radeon RS Graphics 2.00GHz
— Mémoire Ram utilisable : 4,94 GB
— Type du système : système d'exploitation 64 bits, processeur ×64.

### 3.1.1 Python

Python est le deuxième langage le plus utilisé dans le monde[1], portable, interprété, placé sous licence libre fonctionnant sur la plupart des plates-formes informatiques telles que les smartphones (Android, iOS), les ordinateurs (Windows, Unix, macOS) et conçu pour optimiser la productivité des programmeurs en offrant des outils de haut niveau et une syntaxe simple à utiliser. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet.
C'est un langage pédagogiquement apprécié car sa syntaxe simple et clairement séparée des mécanismes de bas niveaux permet une initiation aisée aux bases de la programmation.
La communauté python est très active et permet d'obtenir une multitude de librairie telles que time, scipy, numpy, mathplotlib qui effectuent des taches de haut niveau.

#### 3.1.1.1 Module time et l'éditeur de texte python spyder

Le module time de python permet de connaitre la durée d'exécution en seconde d'un programme.

L'éditeur de texte python, spyder avec son interface et un terminal régulièrement mis à jour facilite encore plus la programmation. Nous l'utiliserons pour nos différentes simulations.

#### 3.1.1.2 Numpy

Numpy est une librairie de calcul numérique hautement optimisée de python et est la base de scipy. Elle contient un grand nombre de matrices et de tableaux multidimensionnels munit de fonctions mathématiques (cosinus, sinus,...) optimisée et opérant sur ces tableaux.

#### 3.1.1.3 Scipy

La librairie scipy est un fruit du projet scipy visant à unifier et fédérer un ensemble de bibliothèques python à usage scientifique. Scipy utilise les tableaux et matrices du module numpy et offre des possibilités avancées de visualisation ou de tracée de courbes grâce au

1. https://jeremymouzin.com/blog/quel-langage-de-programmation-choisir-en-2019/, vu le 22/06/2019

module matplotlib. Elle contient par exemple des modules pour le traitement du signal, le traitement d'images, l'algèbre linéaire, les statistiques, l'optimisation, calcul d'intégrale. Des fonctions de scipy seront utilisées par exemple pour déterminer les maxima locaux de l'accumulateur.

#### 3.1.1.4 Mathplotlib

Mathplolib est une librairie python permettant tout aussi bien de visualiser rapidement des données que de fournir des graphiques de grandes qualités dans divers formats. C'est l'un des modules python les utilisés pour la représentation de graphiques en 2D.

### 3.1.2 Opencv

opencv pour open computer vision est la librairie la plus optimale en traitement d'image et en vision par ordinateur. C'est une librairie initialement conçue en langage $C++$ mais vite traduite en python pour gagner en simplicité d'utilisation et efficacité. Elle contient pratiquement toutes les fonctions nécessaires au pré-traitement et au traitement d'image numérique et de résolution des problèmes de vision par ordinateur. L'utilisation d'opencv nécessite une bonne connaissance de numpy car pour opencv les images sont des matrices ; sa fonction d'affichage d'image est moins perfectionnée que celle de matplotlib, on utilisera donc celle de mathplolib. Par ailleurs les images sont lues au format (B,V,R) et non (R,V,B).

### 3.1.3 GIMP

GIMP pour GNU Image Manipulation Program est un logiciel d'édition et de retouche d'image sous licence gratuite. Il permet d'effectuer de manière optimale les pré-traitement d'image effectué avant la reconnaissance de formes dans une image numérique. Certains pré-traitements d'images de ce mémoire tel que l'image de la figure 2.2a ont été effectué avec GIMP.

## 3.2 Analyse des algorithmes du chapitre 2

Comparer deux algorithmes du point de vue de leurs complexités temporelles est couramment utilisé mais nous sommes confrontés à l'utilisation de fonctions mathématiques dans nos algorithmes pseudo-codes. Dans [SOZ18] est fait une étude comparative de la complexité de l'accumulateur de données amélioré de l'algorithme 12 et de celui utilisé dans la transformée de Hough classique, qui montre qu'ils sont d'une même classe de complexité.

Par ailleurs nous utilisons un langage de programmation évolué conçut avec des bibliothèques optimisées pour ces taches. De plus l'algorithme de maillage du chapitre 2 est tributaire des fonctions de primalité et de factorisation qui pourront être améliorées en fonction des besoins ou de l'institution l'utilisant. Dans le calcul de la complexité de la fonction de factorisation (algorithme 5) l'instruction "$k \leftarrow E(longueur(tab)/2)$" conduit à la détermination du nombre de diviseur du nombre $n$, ce qui qui est intrinsèquement lié à $n$. Nous nous limiterons donc à la comparaison des temps d'exécution de ces différents algorithmes, d'autant plus que le choix d'utilisation d'un programme est surtout fonction de son temps d'exécution.

Se lancer donc dans la comparaison de l'algorithme 12 et de la transformée de Hough standard étendue du point de vue de la complexité des algorithmes est à notre sens inutile vue tous les faits logiques appuyés par l'écart de temps d'exécution des deux algorithmes sur un même ordinateur. L'algorithme 12 est sans aucun doute plus rapide que celui classique.

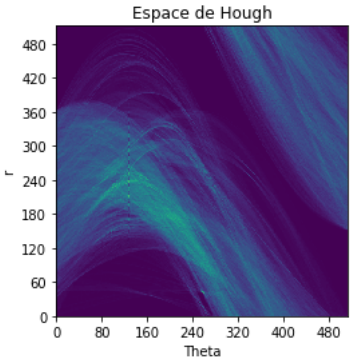
L'algorithme 9 de reconnaissance de droites utilisant la préimage n'a pas été programmé mais le même raisonnement logique montre qu'il devrait être plus rapide que l'algorithme classique.

### 3.2.1 Avantages de la méthode

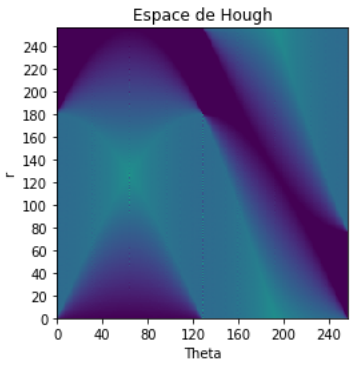
Notre algorithme 12 de reconnaissance de droites discrètes a deux grands avantages qui sont :

- L'utilisation de l'algorithme 6 dans la fonction de reconnaissance de l'algorithme 12 engendre des calculs supplémentaires mais combiner à l'algorithme 7 permet de calculer le dual d'un groupe de pixels et de ne pas calculer pour certains groupes, de ce fait d'aller plus vite que la méthode habituelle de reconnaissance de formes ou on calcul le dual de beaucoup plus de pixels. L'espace de Hough ou de paramètre engendré par ce maillage à des dimensions bien plus petites mais contient les mêmes droites que la matrice image et est donc optimale en utilisation ;
- permet la reconnaissance de droites discrètes minces et épaisses (sous section 3.3.2.1.4) ;
- avec un paramètre $\alpha$ judicieusement choisi le plan d'intersection des courbes dans l'espace de Hough est plus nette (figure 3.1).

(a) Image avec contours




(b) Espace de Hough sans notre maillage




(c) Espace de Hough avec notre maillage rectangulaire et $\alpha = 0$

FIGURE 3.1 – Différents espaces de Hough de l'image 3.3c

### 3.2.2 Limites de la méthode

L'algorithme 12 de reconnaissance de droites discrètes à cependant des limites telles que :

- La détection dépend du choix du paramètre $\alpha$ et du seuil ( voir section suivante), pour certaines valeur de $\alpha$ on peut ne rien détecter ;
- en fonction de la valeur de $\alpha$ ou du seuil on peut ) être en présence de droites détectées plus d'une fois ;
- l'algorithme de reconnaissance utilisant la préimage n'est pas programmable pour le moment du fait qu'il faille programmer l'intersection de demi-plan.

(a) 5-droites (b) 10-droites

FIGURE 3.2 – Droites dans une image avant et après pré-traitement

**NB :** le nombre de lignes qu'on peut voir dans une image peut être différents de celui détecté par un l'algorithme. Sur la figure 3.2a on apperçoit 5 droites et 10 droites après pré-traitement à la figure 3.2b.

### 3.2.3 Autres travaux

Nous introduisons dans cette sous-section une autre fonction de maillage rectangulaire (algorithme 13) de l'image dans le but de montrer que la fonction de maillage de l'algorithme 6 n'est pas unique et que l'essentiel pour une détection est d'avoir un bon contrôle sur l'ensemble des paramètres entrée lors la conception des algorithmes et de plus, ouvrir la réflexion sur le sujet.

---

**Fonction** *maillage(Nl,Nc,$k_1$,$k_3$)*

**Variables :** $r_1$,$r_2$ : entiers

tab : tableau de 4 entiers

$r_1 \leftarrow Nl \mod k_1$;

$r_2 \leftarrow Nc \mod k_3$;

% détermination du nombre de grilles sur la hauteur et la largeur de l'image;

$k_0 \leftarrow \frac{Nl-r_1}{k_1}$, $k_2 \leftarrow \frac{Nc-r_2}{k_3}$;

tab$\leftarrow \{k_0, k_1, k_2, k_3\}$ ; % $Nl = k_0 \times k_1 + r_1$ et $Nc = k_2 \times k_3 + r_2$

**retourner** *tab ;*

**fin**

**Algorithme 13 :** Fonction de maillage 2

**Commentaire sur l'algorithme 13**

Cette fonction prend en entrée les nombres de lignes "$Nl$", de colonnes "$Nc$" de l'image et les dimensions "$k_1$" (sur la hauteur), "$k_3$" (sur la largeur) des grilles de notre maillage.

"$r_1$" et "$r_2$" représentent les résidus sur les lignes et sur les colonnes obtenus après maillage avec $0 \leq r_1 < k_1$, $0 \leq r_2 < k_3$.
On retourne tab pour permettre aux autres algorithmes d'avoir accès à toutes les données pour usage ultérieur.

Avec la fonction de maillage de l'algorithme 13 on maille directement l'image sans passer par les fonctions d'Eratosthène et de factorisation mais à cependant deux grands inconvénients :

— l'ajout de deux paramètres "$k_1$" et "$k_3$" pour les pas de hauteurs et de largeur de l'image ;
— des résidus "$r_1$" et "$r_2$" dont les tailles en pixels sont variables peuvent faire l'objet d'un second maillage ou ne pas être considérés lors du calcul du dual des différents pixels. Le dernier choix pourrait amener à ne pas considérer une (petite) partie de l'image.

Par exemple pour une image ou $Nl = 640$ et $Nc = 340$ et qu'on prend $k_1$=7, $k_3$=6 a : $k_0 = 91,\ k_2 = 56,\ r_1 = 3,\ r_2 = 4$ c'est à dire $Nl = 91 \times 7 + 3$ et $Nc = 56 \times 6 + 4$. Avec les résidus 3 et 4 on peut effectuer des sous maillages $2 \times 3$ ; ou $2 \times 2$ par exemple.
Le choix entre les algorithmes 6 et 13 est laissé au lecteur en fonction de ses besoins.

## 3.3 Implémentation

Notre algorithme de détection de droite avec la transformée de Hough rectangulaire peut être appliquée à n'importe quelle image numérique mais cela pourrait avoir comme conséquence une lenteur dans l'exécution. Pour palier à cela on effectue des pré-traitements sur l'image considérée.

La détection de formes (droites) en temps réel nécessite une bonne connaissance d'outils informatiques et mathématiques au niveau des pré-traitements ou de la phase de reconnaissance elle même. Nous énumérerons brièvement sans entrer dans le formalisme mathématique [2] des méthodes de pré-traitement d'image dans le but de pouvoir facilement (en temps réduit) extraire des informations, supprimer l'information inutile tel que le bruit et renforcer l'information utile dans l'image en vue d'un traitement ultérieur.

### 3.3.1 Pré-traitement d'image

Le pré-traitement d'images est l'ensemble des opérations effectuées sur une image, soit pour l'améliorer, soit pour en extraire de l'information pour d'autres traitements ou calculs. Généralement, les méthodes de pré-traitements visent à renforcer la ressemblance des pixels d'une même région ou à accentuer les différences de pixels provenant de régions différentes.

**Définition 25.** ***(Bruit)*** *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 peut être dû à plusieurs causes telles que :*

— *l'environnement (physique) lors de la prise de la photo ;*
— *la base qualité du capteur ;*
— *l'échantillonnage.*

2. Le lecteur pourra lire [Ber10] pour un formalisme mathématique.

#### 3.3.1.1 Filtrage

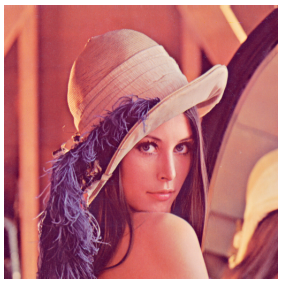
Le filtrage est une des méthodes les plus utilisées en réduction de bruit dans une image. Il vise la réduction de l'amplitude des perturbations liées au bruit, en préservant et en accentuant les zones de transitions ou gradients dans l'image.
Il existe les filtres linéaires (moyenneur, gaussien, ...) et non linéaires (d'ordre, homomorphique,...). L'utilisation de filtre linéaire entraine la naissance de zone flou dans l'image, ce qui n'arrive pas avec les filtres non linéaires.

#### 3.3.1.2 Détection de contours

La détection de contours est une étape préliminaire à de nombreuses applications de l'analyse d'images. Les contours constituent en effet des indices riches, au même titre que les points d'intérêts, pour toute interprétation ultérieure de l'image.
Les variations d'intensités dans une image correspondent à des changements de propriétés physiques ou géométriques des objets de l'image, ou de l'image elle-même. Dans la plupart des cas, ces variations correspondent à des bords ou parties d'objets de l'image. Détecter ces variations est donc une des manières possibles pour extraire les objets de l'image. Les contours correspondent tout simplement aux parties de l'image où le signal présente une forte discontinuité. Les contours extraits et leurs caractéristiques peuvent en effet servir d'entrée à un système de reconnaissance de formes (robotique, analyse d'images médicales, vidéo-surveillance, ...).
Le filtre de Canny et de Sobel sont les méthodes de détection de contours les plus utilisées. le filtre de Canny est optimal pour la détection de contours car il permet :

- Bonne détection : détecter un maximum de contours ;
- bonne localisation : les points détectés doivent être les plus proches possibles du vrai contour ;
- réponse unique : minimiser le nombre de contours détectés plusieurs fois.

Les détecteurs de contours de Canny et de Sobel effectuent implicitement des opérations de filtrage.



(a) Image de départ

(b) Contours avec le filtre de Sobel

(c) Contours avec le filtre de Canny [a]

a. code python à l'annexe A.1

FIGURE 3.3 – Image Lena et détection de ses contours avec les filtres de Sobel et de Canny

### 3.3.2 Exemples d'implémentations

Nous effectuerons à présent des tests des algorithmes présent dans le chapitre 2 sauf pour l'algorithme 9 qui n'est pour le moment pas programmable vue sa spécificité et l'utilisation de l'intersection de demi-plan qu'il induit. Le code python de l'algorithme 12 est à l'annexe A.2, ce code sera utilisé pour tout les tests effectués dans ce chapitre.
Nous recommandons un pré-traitement d'image avec les opérations citées dans l'ordre ci-dessous.

1. Convertir l'image couleur en image en niveaux de gris ;
2. détection de contours avec par exemple le filtre de Canny (code python à l'annexe A.1) ;
3. algorithme de reconnaissance (l'algorithme 12).

Nous ferons varier les paramètres '$\alpha$ et seuil' avec l'image 3.3c en utilisant l'algorithme 12, des valeurs limites aux valeurs moyennes pour nous assurer de la conformité des résultats obtenus. Le paramètre 'n' étant fonction de la couleur de fond de l'image en niveaux de gris, on a n$\in \{0,\ 255\}$.
Dans un premier temps nous fixerons le "seuil" de la matrice d'accumulation à 150 (les pixels de contours sont blancs) et ferons varier $\alpha$ puis ce sera l'inverse.

L'image de la figure 3.3c étant à fond noir, on prend "n=255" pour toute la suite.
Temps de détection [3] est le temps de calcul de l'accumulateur.
Temps final [4] est le temps de tracer des droites trouvées dans l'image (figure 3.4).

#### 3.3.2.1 Variation de $\alpha$

Nous cherchons à déterminer l'influence du paramètre $\alpha$ sur le nombre de droites détectées . Pour cela on fixe le seuil des maxima dans l'accumulateur de données (dans l'algorithme 11) à 150 car l'image a une forte densité de pixels allumés, par contre avec l'image de la figure 3.2b ce seuil ne serait pas judicieux car on a peut de regroupement de pixels allumés dans cette image.

TABLE 3.1 – Résultats sans notre maillage

| seuil | Nombres de droites détectées | Temps de détections | Temps final |
|---|---|---|---|
| 150 | 9 | 204 sec soit 3 min : 24 | 289 sec soit 3 min : |

##### 3.3.2.1.1 Valeurs limites de $\alpha$

Les valeurs limites de $\alpha$ étant 0 ou 1 on étudie le comportement de l'algorithme 12 pour ses valeurs aux bords.
• Si $\alpha = 0$ alors on sélectionne tous les rectangles du maillage rectangulaire et nous aurons que l'effet du maillage sur le résultat.

• Si $\alpha = 1$, dans ce cas on calcul le dual de tous les rectangles entièrement allumés. On détectera pas de droites ou très peu de droites qu'à habituellement.

3. C'est temps sont mis a titre indicatif afin de pouvoir comparer notre algorithme de reconnaissance à celui classique.
4. Quelques secondes entre deux temps n'est formellement pas à prendre en compte.

TABLE 3.2 – Résultats pour $\alpha \leq 0$

| $\alpha$ | seuil | Nombres de droites détectées | Temps de détections | Temps final |
|---|---|---|---|---|
| 0 | 150 | 18 | 236 sec soit 3 min : 56 | 253 sec soit 4 min : 13 |

TABLE 3.3 – Résultats pour $\alpha = 1$ et le seuil fixé à 150

| $\alpha$ | seuil | Nombres de droites détectées | Temps de détections | Temps final |
|---|---|---|---|---|
| 1 | 150 | 0 | 9,14 sec | 9,25 sec |

#### 3.3.2.1.2 $\alpha < 0$ ou $\alpha > 1$

On se met hors du domaine de définition de $\alpha$ dans la pré-condition de l'algorithme 12 pour être rassuré quant à la conformité des résultats.
• Si $\alpha < 0$ on sélectionne tous les rectangles du maillage rectangulaire et nous aurons que l'effet du maillage sur le résultat. On retrouve logiquement les mêmes résultats que pour $\alpha = 0$ car la fonction de comptage calcul le dual des rectangles pour lesquels le taux de pixels allumés est au moins $\alpha$ (table 3.2).
• Si $\alpha > 1$ on ne détecte rien (table 3.4).

TABLE 3.4 – Résultats pour $\alpha > 1$ et le seuil fixé à 150

| $\alpha$ | seuil | Nombres de droites détectées | Temps de détections | Temps final |
|---|---|---|---|---|
| 1,2 | 150 | 0 | 12,17 sec | 12,54 sec |

#### 3.3.2.1.3 $0 < \alpha < 1$

On prend $\alpha$ dans l'intervalle de la pré-condition de l'algorithme 12 afin de mieux comprendre son influence sur la détection de droites.

TABLE 3.5 – Résultats pour $0 < \alpha < 1$ et le seuil fixé à 150

| $\alpha$ | Nombres de droites détectées | Temps de détections | Temps final |
|---|---|---|---|
| 0,1 | 18 | 242 sec soit 4 min : 2 | 261 sec soit 4 min : 21 |
| 0,3 | 17 | 200 sec soit 3 min : 20 | 218 sec soit 3 min : 38 |
| 0,4 | 14 | 118 sec soit 1 min : 58 | 132 sec soit 2 min : 12 |
| 0,5 | 6 | 60,03 sec soit 1 min : 03 | 66 sec soit 1 min : 6 |
| 0,7 | 0 | 13,20 sec | 13,26 sec |

#### 3.3.2.1.4 Interprétation-analyse des résultats

De ces différents résultats on peut à priori dire que plus on est proche de 0 plus on détecte des droites mais la différence du nombre de droites détectées dans les tables 3.2 et 3.1 attire notre attention. Y aurait-il redondance de calculs dans l'implémentation de l'accumulateur ? Le paramètre "seuil" y joue t-il un rôle ? Si oui une étude de la contrôlabilité de la redondance et du seuil par rapport aux droites détectées serait alors envisageable.
Prenons l'exemple d'un conducteur de véhicule munit d'aide au pilotage dont le système d'aide fonctionne à l'aide de détection des lignes sur la route. Comment choisir $\alpha$ pour ne détecter que les droites servant au bon guidage du véhicule ? On veut alors éliminer les petits traits présents sur la route.
Par ailleurs la subdivision de l'image en petites zones, par notre maillage permet la reconnaissance de droites discrètes minces (figure 3.4, image en bas à gauche).

#### 3.3.2.1.5 Exemple de droites détectées

La figure 3.4 présente des exemples de droites détectées avec l'algorithme 12 pour $\alpha = 0{,}3$ et un seuil de 150.

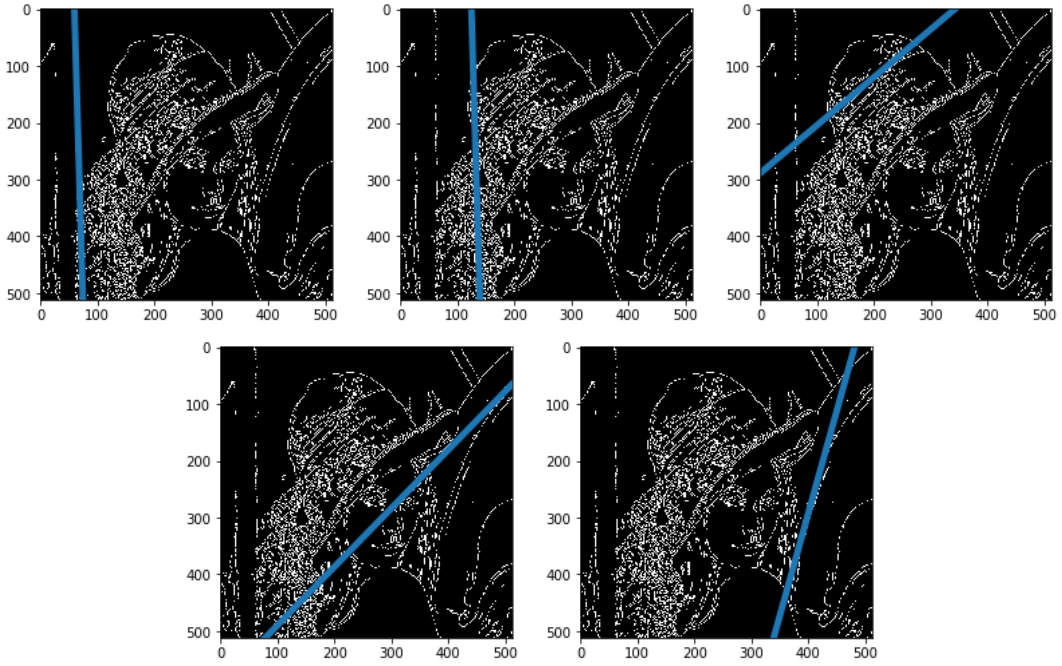


FIGURE 3.4 – Exemples de droites détectées avec l'algorithme 12 $\alpha = 0{,}3$ et un seuil de 150

### 3.3.2.2 Variation du seuil

Nous cherchons à déterminer l'impact du seuil sur le nombre de droites détectées. Pour cela on prendra $\alpha = 0,6$ et conservons n=255 pour compter les pixels blancs, nos résultats seront illustrés dans la table 3.6.

TABLE 3.6 – Résultats pour le seuil variant et $\alpha$=0,6

| seuil | Nombres de droites détectées | Temps de détections | Temps final |
|---|---|---|---|
| 150 | 0 | 22,49 sec | 22,59 sec |
| 100 | 0 | 21,34 sec | 21,46 sec |
| 60 | 9 | 19,28 sec | 29,51 sec |
| 40 | 18 | 19,15 sec soit 2 min : 18 | 37,22 sec |

Du tableau 3.6 on constate que le seuil a une incidence direct sur le nombre de droites détectées et que laisse notre algorithme détecte plus rapidement les droites que celui classique.

## 3.3.3 Un exemple en clair

| $\alpha$ | seuil | Nombres de droites détectées | Temps de détections | Temps final |
|---|---|---|---|---|
| 0 | 50 | 8 | 7,88 sec | 15,39 sec |

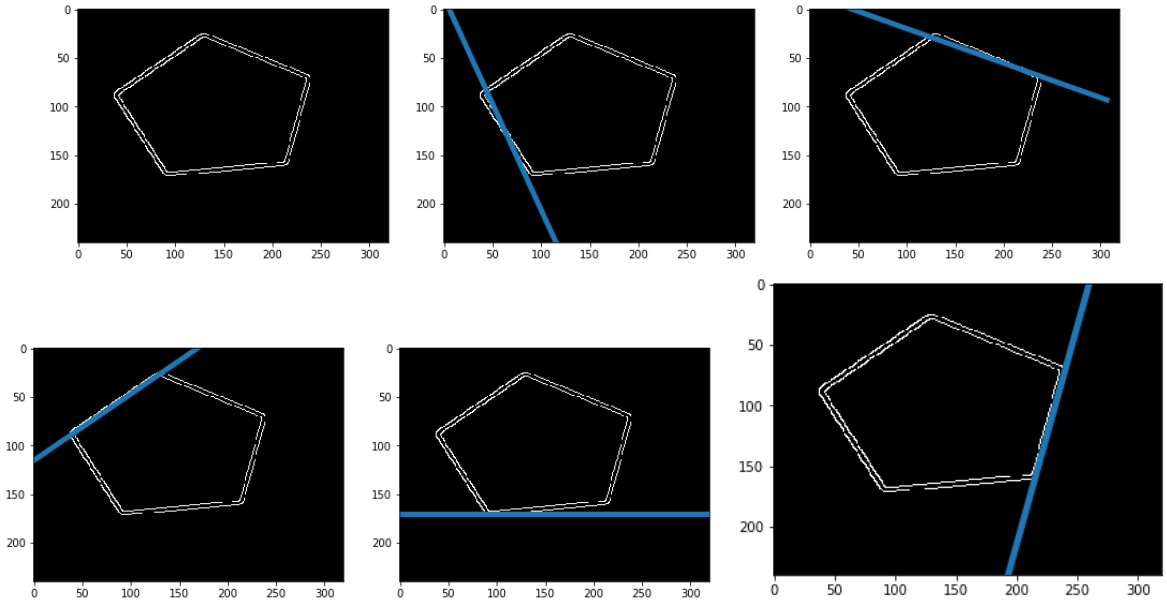


FIGURE 3.5 – Détection de droite dans un pentagone avec l'algorithme 12

# Conclusion

Ce chapitre permet :

— de montrer d'autre facette de nos algorithmes.
A ce niveau Nous avons eu a choisir entre plusieurs algorithmes effectuant une même tâche que nous avons écrit, nous avons opté pour ceux ayant le moins de paramètres possible ou dont on pouvait au mieux contrôler les paramètres, laissant le choix à un lecteur d'un autre avis ou voulant expérimenter autre chose.
— de corroborer les résultats attendus lors des chapitres précédents ;
— de mieux appréhender certaines limites de nos algorithmes de part les simulations ;

L'algorithme 12 permet de faire une détection rapide mais nécessite un bon choix du taux $\alpha$ et du seuil des maxima à prendre dans la matrice d'accumulation. Corriger ces limites peut faire l'objet d'étude(s) approfondie(s) et détaillée(s) mais dans l'ensemble nous pouvons nous réjouïre des résultats obtenus et chercher à pousser la réflexion encore plus loin (un départ pourrait être l'algorithme 13 de ce chapitre).

# Conclusion générale

Les travaux de recherches effectués pour optimiser la détection de droites discrètes avec la Transformée Standard de Hough furent très enrichissant et nous permirent de mieux appréhender la vision par ordinateur, l'intelligence artificielle (le deep-learnig) qui fournissent un cadre technologique d'application des mathématiques. En effet nous acquîmes et exposâmes des connaissances en géométrie et topologie discrètes qui fournissent un cadre abstrait de définition et de conceptualisation de pavages, de droites discrètes, etc. Cela dans le but d'être mieux outillé pour la détermination, l'implémentation de méthodes de maillage, de reconnaissance optimale de droites discrètes avec la transformée de Hough standard et d'un taux de pixels allumés $\alpha$ ainsi que la mise en pratique de ces méthodes de part des simulations. Les simulations avec des images numériques révèlent que le nombre de droites détectées avec l'algorithme 12 (dont le code python est en annexe A.2) est lié aux paramètres $\alpha$ et au seuil de détermination d'extrema dans l'accumulateur de données.

Ces travaux contribuent à la recherche en traitement d'image et en détection de droites dans une image numérique avec les méthodes de maillage rectangulaire (algorithme 6) et de détection de droites analytiques discrètes en 2D (algorithme 8, 9, 12).
Des perspectives :

- D'étude de la contrôlabilité du choix du paramètre $\alpha$ et du seuil, dans le but d'optimiser et de ne détecter que des droites jugées "utiles" ;
- de coupler les algorithmes issus de ses travaux aux deep-learning afin de permettre des reconnaissances automatisées sur de grands volumes d'images et en temps "acceptable" ;
- de conçevoir un maillage triangulaire de l'image et des algorithmes de détections de droites en fonction de ce maillage et dont les temps d'exécutions seront comparés à ceux de ce mémoire. Envisager également une implémentation avec du deep-learning ;

devrait être envisagées avec les connaissances mathématiques et les moyens informatiques actuelles ou à venir.

Annexe A

# Des codes sources python

Cette annexe a pour but de donner les codes python de quelques algorithmes des chapitres précédent sans pour autant les expliquer dans les détails (ce qui sort de l'objectif du travail et allongerai le mémoire). Des abréviations pourront être trouvées dans les codes car ceux-ci sont mis pour uniquement pour indiquer très rapidement ce qu'on fait et pour s'en souvenir.

## A.1 Code source du contour de Canny

La détection de contour est une opération très importante en reconnaissance de forme avec la transformée de Hough. Le filtre de Canny est celui utilisé pour les traitements de ce type dans ce mémoire et dont l'implémentation python à l'aide de opencv est donnée ci dessous. Dans ce programme nous utilisons une interface graphique GUI pour permettre un meilleur paramétrage du niveau du seuil de filtrage et de la taille de la matrice du filtre de Canny.

```
# -*- coding: utf-8 -*-
"""
@author: TRAORE Cheick A. D. Gabrbriel pour une utilisation personnelle
 et non commerciale
"""
import cv2

lowThreshold = 50
highThreshold = 100
maxThreshold = 1000

apertureSizes = [3, 5, 7]
maxapertureIndex = 2
apertureIndex = 0

blurAmount = 0
maxBlurAmount = 20

# Fonctions d'appel des barres
def applyCanny():
```

```
# filtrer l'image avant la détection des contours
  if(blurAmount > 0):
    blurredSrc = cv2.GaussianBlur(src, (2 * blurAmount + 1, 2 * blurAmount + 1), 0);
  else:
    blurredSrc = src.copy()
# dimention de la matrice du noyau du filtre de Canny, il doit être impaire
apertureSize = apertureSizes[apertureIndex];

# application de Canny
edges = cv2.Canny( blurredSrc, lowThreshold, highThreshold, apertureSize )

# affichage et enregistrement de l'image ayant ses contours détectés
cv2.imshow("Edges",edges)
cv2.imwrite('tof.jpg',edges)

# Fonctions de mise à jours
def updateLowThreshold( *args ):
  global lowThreshold
  lowThreshold = args[0]
  applyCanny()
  pass


def updateHighThreshold( *args ):
  global highThreshold
  highThreshold = args[0]
  applyCanny()
  pass


def updateBlurAmount( *args ):
  global blurAmount
  blurAmount = args[0]
  applyCanny()
  pass


def updateApertureIndex( *args ):
  global apertureIndex
  apertureIndex = args[0]
  applyCanny()
  pass

# lecture de  l'image
```

```
src = cv2.imread('20171218_105517.jpg', cv2.IMREAD_GRAYSCALE)

#src=cv2.resize(src, None, 0.6, 0.6, cv2.INTER_LINEAR)

edges = src.copy()
# Display images
cv2.imshow("Edges", src)
cv2.namedWindow("Edges", cv2.WINDOW_AUTOSIZE)

# Les barres de contrôles
cv2.createTrackbar("Faible seuillage", "Edges", lowThreshold,
                           maxThreshold, updateLowThreshold)


cv2.createTrackbar("Taut seuillage","Edges",highThreshold,
                                  maxThreshold,updateHighThreshold)


cv2.createTrackbar("Taille du noyau","Edges",apertureIndex,
                                  maxapertureIndex, updateApertureIndex)


cv2.createTrackbar( "Flou", "Edges", blurAmount,
                                     maxBlurAmount, updateBlurAmount)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## A.2 Code source de l'algorithme 12

Au passage d'un algorithme au programme connait généralement des changements de syntaxes, c'est le cas pour l'algorithme 12 de reconnaissance de droites discrètes qui à été programmé en fonction. Cela permet de prendre facilement les paramètres en entrée.

```
# -*- coding: utf-8 -*-
"""
@author: TRAORE Cheick A. D. Gabriel pour utilisation personnelle
et non commerciale
"""
import cv2
import matplotlib.pyplot as plt
from MAILLAGE import maillage
from DEFIACCU import defAccu
from COMPTAGE import compt
```

```
import math
import time
import scipy.ndimage.filters as filters
import scipy.ndimage as ndimage

img=cv2.imread('Lena-Canny.png') #lecteur de l'image pré-traitée
def select_recogR3(a,n,s,img):
d1=time.time()  # debut du chrono
Nl=img.shape[0]
Nc=img.shape[1]
tab, tabool=maillage(Nl,Nc)
h=tab[0][0]*tab[0][1]
l=tab[1][0]*tab[1][1]
d=tab[0][1]*tab[1][1]
theta_max=math.pi
r_max=(Nl**2+Nc**2)**0.5
acc, r_dim, theta_dim=defAccu(tab,tabool)

for x in range(0,l-tab[0][1],tab[0][1]):
   for y in range(0,h-tab[1][1],tab[1][1]):
   A=(x,y)
   C=(x+tab[0][1],y+tab[1][1])
   if (compt(A,C,n,img)/d)>=a:
      for z in range(A[0],C[0]+1):
            for k in range(A[1],C[1]+1):
               if img[z,k,0] == 0: continue
                  for itheta in range(theta_dim):
                   theta = 1.0 * itheta * theta_max / theta_dim
                   r = z* math.cos(theta) + k * math.sin(theta)
                   ir = r_dim * ( 1.0 * r ) / r_max
                   ir,itheta=round(ir),round(itheta)
                   acc[ir,itheta] = acc[ir,itheta] + 1
#premier Si
if tabool[0]==1 and tabool[1]==0:
 x=h
 for y in range(0,l):
    if img[x,y,0]==n:
       for itheta in range(theta_dim):
          theta = 1.0 * itheta * theta_max / theta_dim
          r = z* math.cos(theta) + k * math.sin(theta)
          ir = r_dim * ( 1.0 * r ) / r_max
             ir,itheta=round(ir),round(itheta)
             acc[ir,itheta] = acc[ir,itheta] + 1
#deuxieme Si
```

```
if tabool[0]==0 and tabool[1]==1:
   y=l
   for x in range(0,h):
      if img[x,y,0]==n:
      for itheta in range(theta_dim):
       theta = 1.0 * itheta * theta_max / theta_dim
       r = z* math.cos(theta) + k * math.sin(theta)
       ir = r_dim * ( 1.0 * r ) / r_max
       ir,itheta=round(ir),round(itheta)
       acc[ir,itheta] = acc[ir,itheta] + 1
#dernier Si
if tabool[0]==1 and tabool[1]==1:
 x=h
 for y in range(0,l):
    if img[x,y,0]==n:
    for itheta in range(theta_dim):
      theta = 1.0 * itheta * theta_max / theta_dim
   r = z* math.cos(theta) + k * math.sin(theta)
      ir = r_dim * ( 1.0 * r ) / r_max
      ir,itheta=round(ir),round(itheta)
      acc[ir,itheta] = acc[ir,itheta] + 1
 y=l
 for x in range(0,h):
    if img[x,y,0]==n:
         for itheta in range(theta_dim):
            theta = 1.0 * itheta * theta_max / theta_dim
            r = z* math.cos(theta) + k * math.sin(theta)
            ir = r_dim * ( 1.0 * r ) / r_max
         ir,itheta=round(ir),round(itheta)
         acc[ir,itheta] = acc[ir,itheta] + 1
#-----------------------------------------------------------------------
f2=time.time()
plt.imshow(acc,origin='lower')
plt.xlim(0,theta_dim)
plt.ylim(0,r_dim)
print('en seconde',f2-d1)
plt.savefig('alpha=0 s=100',bbox_inches='tight')
#-----------------------------------------------étape 3-----------------------
#Recherche des maxima dans la matrice d'accumulation
-----------------------------étape 4-------------------------------------
#Construire les droites détectées
```

Les étapes 3 et 4 sont inspirées du programme de [Dai19]

# Bibliographie


[And00] E. Andres : *Modélisation analytique discrète d'objets géométriques.* Thèse d'habilitation, Université de Poitiers (France), décembre 2000.

[And03] E. Andres : Discrete linear objects in dimension n : the standard model. *Graphical Models*, 65:92–211, 2003.

[APR83] L. M. Adleman, C. Pomerance et R. S. Rumely : On distinguishing prime numbers from composite numbers. *The Annals of Mathematics, Second Series*, 117, No. 1:173–206, 1983.

[Ber04] D. J. Bernstein : On distinguishing prime numbers from composite number : The state of the art in 2004. *Mathematics Subject Classification*, 2004.

[Ber10] M. Bergouioux : *Quelques méthodes mathématiques pour le traitement d'image. DEA. Cours M2-Université d'Orléans.* 2008-2009,2008. pp.110.

[Coe05] D. Coeurjolly : Supercover model and digital straight line recognition on irregular isothetic grids. *In Discrete Geometry for Computer Imagery*, volume 3429:311–322, 2005.

[Dai19] Daidalos : Implementing a simple python code to detect straight lines using Hough transform, février 2019.

[Dex06] M. Dexet : *Architecture d'un modeleur géométrique à base topologique d'objets discrets et méthodes de reconstruction en dimensions 2 et 3.* Thèse en informatique et applications, Université de Poitiers (France), décembre 2006.

[DH72] R.O. Duda et P.E. Hart : Use of the hough transform to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[DS84] L. Dorst et A.W.M. Smeulders : Discrete representation of straight lines. 6(4):450–463, 1984.

[Ezo13] T. Ezome : Test de primalité et de pseudo-primalité. *Publications mathématiques de Besançon*, pages 89–106, 2013.

[Hou62] P.-V.-C. Hough : Method and means for recognizing complex patterns. *In United States Pattent*, 3069654:47–64, 1962.

[Lir11] F. Liret : *Arithmétique, cours et exercices corrigés.* Dunod, 2011.

[Mai85] H. Maitre : un panorama de la transformation de hough. *Traitement du signal*, volume 2(4):305–317, 1985.

[McI84] M. D. McIlroy : A note on discrete representation of lines. 64:481–490, 1984.

[Mon03] J.-M. Monier : *GéométrieMPSI,MP, cours et 350 exercices corrigés.* Dunod, 3ème édition, 2003.

[Pla C] PLATON : *La republique, livre VI, 484a-511b.* 481 av. J.-C.

[Rev89] J.-P. REVEILLES : Structures des droites discrètes. In Journées mathématique et informatique.Marseille-Luminy, 1989.

[Rev91] J.-P. REVEILLES : *Géométrie discrète, calcul en nombres entiers et algorithmique.* Traitement des images, Université Louis Pasteur (France) <tel-01279525>, 1991.

[Rod11] M. RODRIGUEZ : *Redimensionnement adaptatif et reconnaissance de primitives discrètes.* Thèse en informatique & applications, Université de Poitiers (France), décembre 2011.

[Ser13] A. SERE : *Transformations Analytiques appliquées aux Images multi-échelles et bruitées.* Thèse en informatique, Université de Ouagadougou (Burkina Faso), juin 2013.

[SOZ18] A. SERE, F.T. OUEDRAOGO et B. ZERBO : An improvement of the standard hough transform method based on geometric shapes. pages 1–8. Future of Information and Communication Conference (FICC), Singapore, 5-6 April 2018.

[SSA13] A. SERE, O. SIE et E. ANDRES : Extended standard hough transform for analytical line recognition. *(IJACSA) International Journal of Advanced Computer Science and Applications*, 4(3):256–266, 2013.