Introduction au traitement d'images

Détection de contours et segmentation

Résumé :

Ce document est une introduction au traitement d'images s'intéressant notamment à la détection de contours et à la segmentation. Différents concepts y sont abordés : le filtrage et l'opérateur de convolution, la recherche de primitives et détection de contours, les opérateurs morphomathématiques et la détection et fermeture de régions.

Mots clefs :

Traitement d'image, filtrage, détection, contours, régions

Document réalisé par :

Yoann Sculo

Semestre de Printemps 2009

Branche SIT

Sommaire

Introduction

Le traitement d'images est de plus en plus utilisé dans de nombreux domaines pour l'interprétation et le contenu des images. Ce dossier s'inscrit dans le cadre de ma préparation personnelle à mon stage de fin d'études au CNES. Etant amené à travailler sur la segmentation avec récupération des contours, j'ai entrepris un travail de recherche et d'apprentissage sur ce domaine. L'idée étant d'avoir un premier aperçu des techniques de traitement d'images.

La compréhension du traitement d'images commence tout d'abord par la définition d'une image et de son acquisition. J'aborderai ensuite la notion de filtrage et de convolution et leur application à la détection de contours. Je parlerai ainsi des différents filtres utilisés dans ce cadre. Je m'occuperai ensuite des opérateurs morphologiques, puis porterai mon attention à la détection de régions. Je terminerai alors par une ouverture sur les possibilités de mise en œuvre de ces techniques sur ordinateur.

1. Les images

Une image est obtenue par transformation d'une scène réelle par un capteur. La numérisation d'une image consiste à passer de la représentation continue à discrète. Une image peut être définie par une fonction, f(x,y), où x et y sont des coordonnées spatiales planes. L'amplitude de f pour chaque paire de coordonnées (x,y) est appelée intensité de l'image au point donné. Une image peut alors être assimilée à une matrice de points appelés « pixels ». De nos jours les appareils photos numériques effectuent la transformation d'eux-mêmes, fournissant directement une image facilement manipulable par ordinateur. Celle-ci pourra alors être affichée sur un moniteur, imprimée, modifiée, enregistrée etc. Derrière cette transformation se cachent deux opérations successives appelées échantillonnage (discrétisation de l'espace) et quantification (discrétisation des couleurs).

A. Echantillonnage

La phase d'échantillonnage consiste en la transformation de l'image analogique (continu), en image numérique (discret). Les valeurs du signal analogique sont alors capturées à intervalles de temps réguliers. La fréquence de capture est appelée fréquence d'échantillonnage. L'échantillonnage va ainsi permettre de définir la résolution de l'image, c'est-à-dire le nombre de pixels qu'elle contiendra. Plus la résolution est élevée, plus l'image sera détaillée. Sa taille augmentera en conséquence.

B. Quantification

La quantification offre une approximation du signal continu par des valeurs d'un ensemble discret de taille variable. Il est ainsi possible de discrétiser les niveaux de gris ou bien les couleurs de l'image selon le résultat qui nous intéresse.

Le terme « niveau de gris » est utilisé souvent pour se référer à l'intensité d'images monochromes. Les images couleurs, sont formées par une combinaison d'images en 2D. Par exemple, dans le système de couleur RVB, une image en couleur consiste en une

combinaison de 3 composantes (rouge, vert, bleu). Ainsi, le travail de quantification d'une image couleur consistera à décomposer l'image continue en plusieurs composantes, suivant le système de couleurs choisi. Il existe différents types d'encodage couleur, cependant nous nous intéresseront ici au système RVB.

C. FFT et reconstruction d'image

Dans certains cas, il est intéressant de faire la transformation inverse en reconstruisant une image dans le but d'obtenir celle-ci avant quantification. Cette technique permet entre autres d'effectuer un changement d'échelle (agrandissement ou réduction). Il est toutefois important de noter qu'il ne sera pas possible par exemple de zoomer à l'infini au sein d'une image. La limite se base sur le nombre d'informations récupérées lors de l'échantillonnage. Ces informations ne peuvent être augmentées qu'en reprenant une nouvelle photo avec une fréquence d'échantillonnage (résolution) plus élevée.

Cette reconstruction se base ainsi sur la transformée dite de Fourier. La transformée de Fourier 2D d'un signal f est définie par :

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j(ux+vy)}dxdy$$

Celle-ci est réversible, la transformée inverse permettant d'obtenir f en fonction de F s'écrit :

$$f(x,y) = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{j(ux+vy)}dxdy$$

Nous utiliserons par la suite la transformée de Fourier rapide (FFT : Fast Fourier Transform) qui est une implémentation efficace de la transformée.

La transformée de Fourier d'une image donne une image complexe. En général on calcule son module et sa phase mais l'on ne représente généralement qu'uniquement son module (ce qui correspond au spectre de celle-ci, c'est-à-dire l'ensemble des coefficients pondérant les fonctions exponentielles $e^{j(ux+vy)}$).

Figure 1 : A gauche une image naturelle. A droite le module de sa transformée de Fourier

Le repère de l'image obtenue par FFT se base sur une origine (0,0) au centre de l'image. On observe ainsi les basses fréquences au centre de l'image et les hautes fréquences en périphérie. La FFT étant réversible il est alors possible de modifier le module de l'image et de reformuler l'image par une transformée inverse.

Par exemple, il est possible d'appliquer des filtres sur le module afin de modifier l'image initiale. Un filtre passe-bas appliqué sur l'image ira supprimer les hautes fréquences et rendra l'image floue en supprimant de petits détails. Un tel filtre est bien souvent utilisé en préparation d'une détection de contours (voir partie 3). Les filtres passe-haut, quant à eux iront supprimer les basses fréquences, faisant ressortir les contours.

D. Notions d'histogramme et de seuillage

L'histogramme d'une image est une fonction qui donne la fréquence d'apparition de chaque niveau de gris (ou couleurs) dans l'image. Il permet de donner un grand nombre d'informations sur la distribution des niveaux de gris (couleurs) et de voir entre quelles bornes est répartie la majeure partie des niveaux de gris (couleurs). Il ne contient aucune information relative à l'emplacement des pixels ni sur la proximité relative de deux pixels. Cependant, l'information qu'il contient peut concerner notamment la brillance apparente et

le contraste d'une image, et il peut être utilisé notamment afin de manipuler les caractéristiques de l'image.

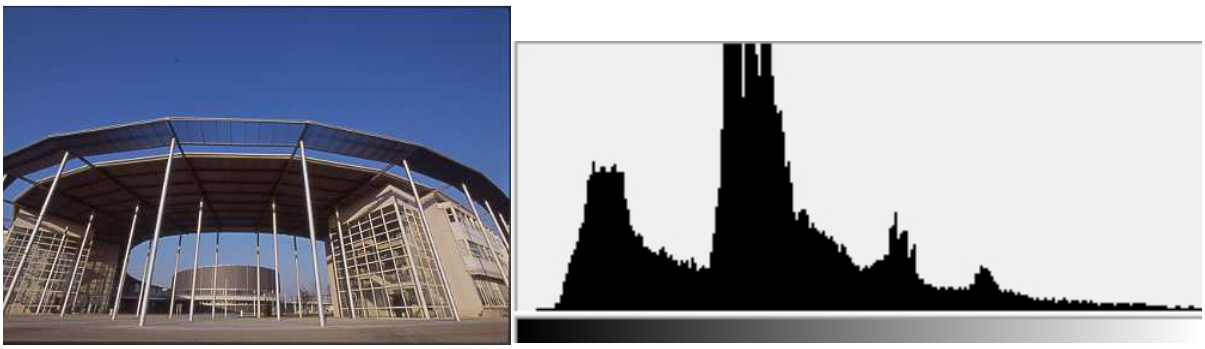


Figure 2 : Une image et son histogramme

L'histogramme est donc un outil très utile pour étudier la répartition des composantes d'une image. De plus, la représentation visuelle de l'histogramme permet de se rendre compte rapidement des défauts de contraste dans l'image. Ainsi en le modifiant, on peut impacter le rendu de l'image. Cette technique est beaucoup utilisée en photographie pour régler le contraste de photos. Cela consiste à appliquer à l'image une fonction de transfert (courbe tonale). Celle-ci indique en abscisse les valeurs initiales et en ordonnées les valeurs modifiées. Il sera alors possible, en fonction de la fonction utilisée de rehausser l'image, d'égaliser l'histogramme, éclaircir ou assombrir l'image. Il est toutefois important de noter que la transformation n'altère pas les informations contenues dans l'image mais les rend plus ou moins visibles.

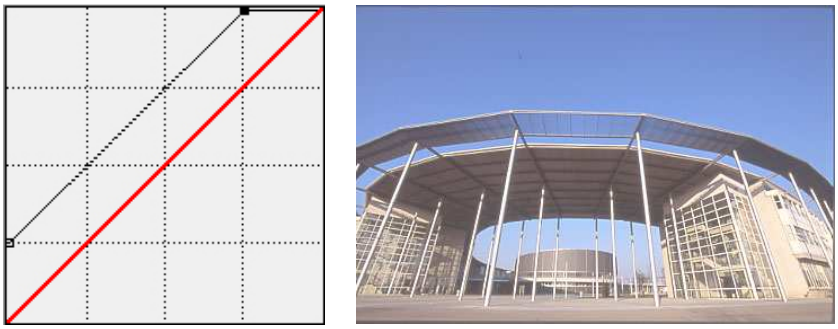


Figure 3 : Exemple d'éclaircissement d'une image par modification de la courbe tonale
En rouge : courbe des valeurs initiales

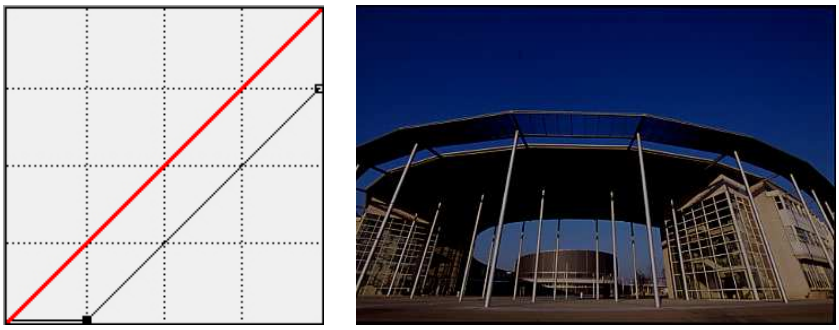


Figure 4 : Exemple d'assombrissement d'une image par modification de la courbe tonale
En rouge : courbe des valeurs initiales

Cela permet alors d'introduire la notion de seuillage. Qui consiste à mettre à zéro tous les pixels ayant un niveau de gris inférieur à une certaine valeur (appelé seuil) et à la valeur maximale des pixels ayant une valeur supérieure. Le résultat du seuillage est une image binaire contenant des pixels noirs et blancs. On parle de « binarisation ».

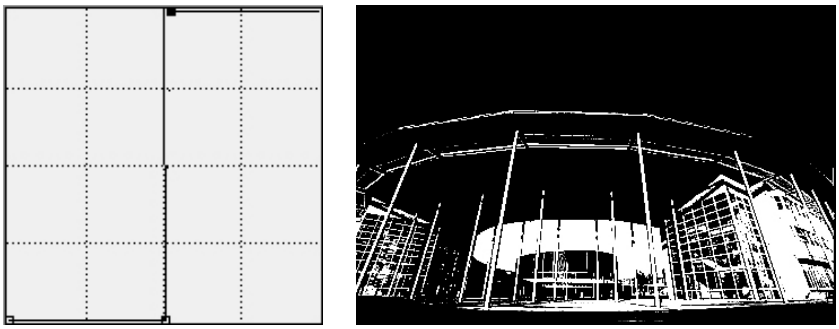


Figure 5 : Courbe tonale de seuillage et le résultat de binarisation

Il est également possible de définir les valeurs de seuil, en utilisant une borne inférieure et supérieure pour faire ressortir les valeurs comprises dans cet intervalle.

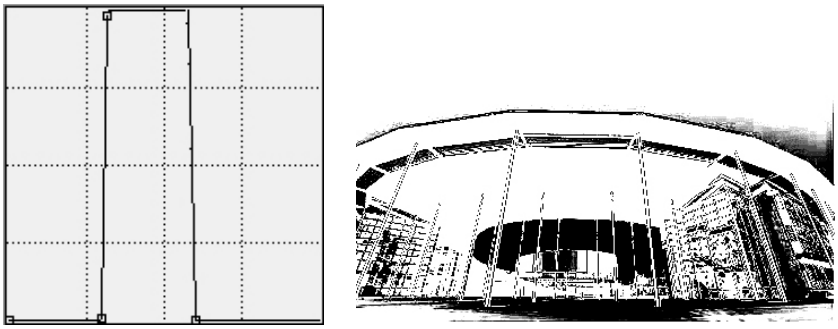


Figure 6 : Courbe tonale de seuillage bornée et le résultat de la binarisation

Le seuillage permet de mettre en évidence des formes ou des objets dans une image. Toutefois la difficulté réside dans le choix du seuil à adopter ou de l'intervalle de seuillage.

Si les pixels dans les objets d'intérêt ont leurs niveaux de gris proches d'une même valeur v, tandis que ceux dans les autres structures ont presque toujours des niveaux de gris éloignés de v, alors l'histogramme de l'image aura un pic autour de v. C'est ce que nous illustrons ci-dessous dans l'histogramme d'une image scanner 3D de l'abdomen d'un sujet. (Figure 7)

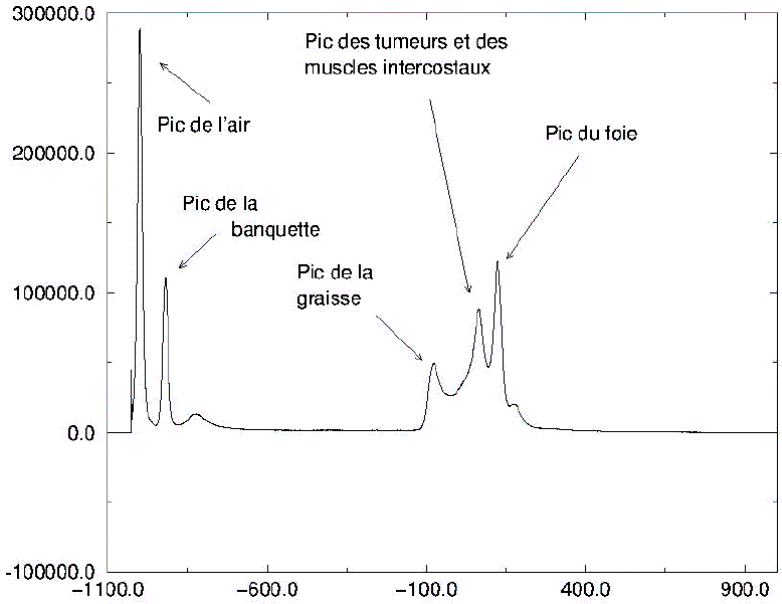


Figure 7 : Histogramme d'une image scanner 3D

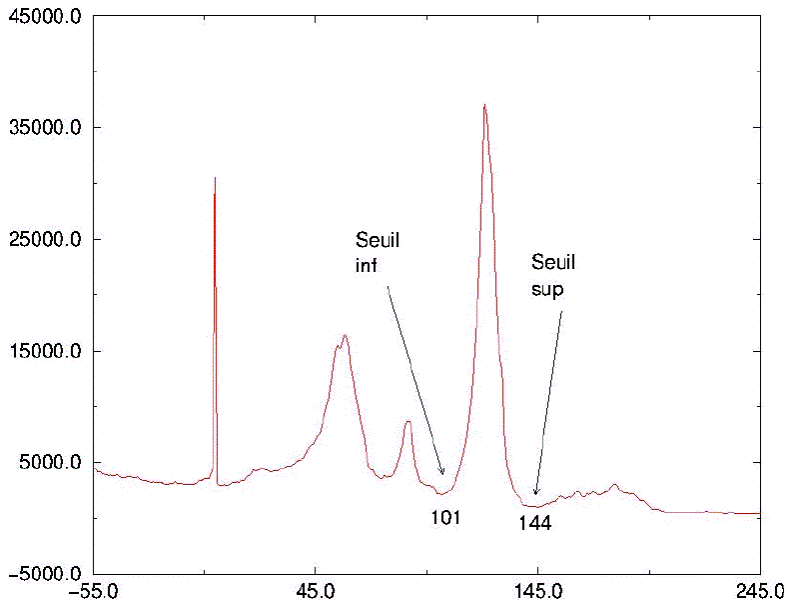


Figure 8 : Histogramme d'une image scanner 3D

En fait, cet histogramme a été lissé, car la plupart des histogrammes ont un aspect bruité, avec des pics crénelés. On pourra donc sélectionner les bornes de l'intervalle de seuillage [a,b], en prenant dans l'histogramme lissé les deux minima de l'histogramme situés de part et d'autre de la valeur v de référence pour le niveau de gris des objets recherchés. C'est ce que nous illustrons ci-dessous pour l'intervalle de seuillage correspondant au foie dans une image scanner abdominale. (Figure 8)

Le seuillage est donc intéressant pour extraire l'information d'une image. Il faut d'abord analyser l'image pour savoir comment bien binariser.

2. Filtrage

i. Notion de voisinage

Dans la suite, nous allons considérer des voisinages d'un pixel dont l'étendue et la forme peuvent varier. Nous allons notamment nous intéresser aux connexités 4 ou 8, c'est-à-dire en prenant en compte 4 ou 8 voisins directs. Nous nous intéresserons à la dimension 3x3.

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Croix (connexité 4)

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Carré (connexité 8)

Toutefois, il existe d'autres formes de voisinage lorsque l'on considère une étendue plus importante : croix, diamant ou carré.

ii. Application de filtres et introduction à la convolution

Le filtrage consiste à balayer l'image par une fenêtre d'analyse de taille finie ou kernel (noyau). Le calcul du nouveau niveau de gris du pixel considéré ne prend en compte que les plus proches voisins de celui-ci.

Bien des traitements d'images sont basés sur ce que l'on appelle l'opérateur de convolution. Ce dernier correspond à une multiplication de matrices. En l'occurrence ici de la matrice image avec un filtre donné. Rappelons tout d'abord la convolution d'une fonction continue f par une fonction continue g :

$$f * g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)g(u-x,v-y)dudv$$

Pour des fonctions discrètes :

$$f * g(x,y) = \sum_{u=-\infty}^{\infty}\sum_{v=-\infty}^{\infty} f(x,y)g(u-x,v-y)$$

Si on applique ceci à une image $I1$ de dimensions finies et à un noyau de convolution K de dimensions 3x3, les pixels de l'image obtenue par convolution de $I1$ par K ont pour valeur :

$$I2(i,j) = \sum_{k=0}^{2}\sum_{l=0}^{2} I1(\mathrm{i}+1-\mathrm{k},\mathrm{j}+1-\mathrm{l})\mathrm{K}(\mathrm{k},\mathrm{l})$$

Si l'on travaille sur des noyaux de convolution de dimension plus générale (2p+1) x (2p+1), l'expression de la convolution de $I1$ par K devient alors :

$$I2(i,j) = \sum_{k=0}^{2p}\sum_{l=0}^{p} I1(\mathrm{i}+\mathrm{p}-\mathrm{k},\mathrm{j}+\mathrm{p}-\mathrm{l})\mathrm{K}(\mathrm{k},\mathrm{l})$$

Ainsi le travail de traitement d'image dans de nombreux cas s'apparente à la suite de convolutions d'une image avec différents filtres suivant le résultat escompté.

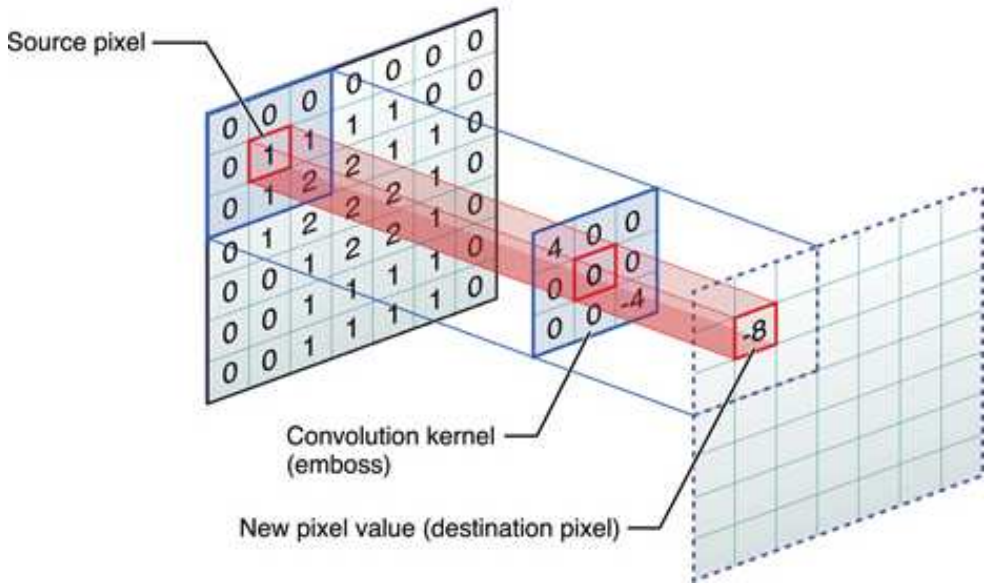


Figure 9 : Schéma explicatif de la convolution

iii. Filtrage et bruit

Les procédés d'acquisition d'images induisent des perturbations qui peuvent être gênantes pour la compréhension et le traitement de la scène. Le but du filtrage est d'éliminer au maximum ces perturbations tout en respectant l'intégrité de la scène originale. L'objectif est donc de réduire les variations d'intensité au sein de chaque région de l'image. Les premières et les plus simples de ces méthodes sont basées sur le filtrage linéaire, mais les limitations de ces techniques ont conduit au développement des filtres non linéaires.

a. Le bruit d'image

A chaque étape de l'acquisition d'une scène, des perturbations vont détériorer la qualité de l'image. Ces perturbations sont regroupées sous le nom de « bruit d'image ». Il existe différentes causes de bruit. On peut notamment citer le contexte d'acquisition (lumière, perturbation de capteurs), les étapes d'échantillonnage et de quantification qui apporte leur propre perturbation, mais aussi l'étape de transmission (perte ou corruption de données). On peut bien remarquer ce bruit notamment sur la transmission analogique de la télévision.

Le bruit est modélisable par :

$$f(a) = Ce^{-K|a|^n}$$

Pour n=1, le bruit est de nature exponentielle ou impulsionnelle.

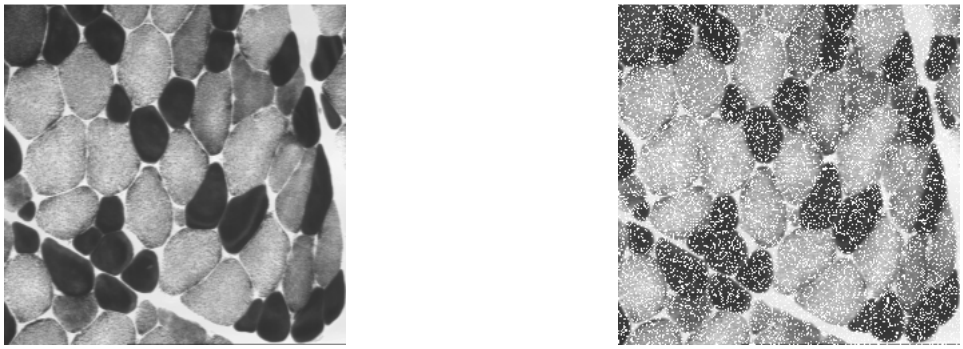


Figure 10 : Image originale à gauche et à droite l'image dégradée par du bruit impulsionnel.

Pour n=2, le bruit est de nature gaussienne.

Figure 11 : Image originale à gauche et à droite l'image dégradée par du bruit gaussien

b. Les filtres linéaires

Filtre moyenneur

Un pixel est remplacé par la moyenne de lui-même et de ses voisins. On peut considérer un voisinage en connexité 4 ou 8, ou même encore plus large. Son kernel en connexité 4 est :

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Figure 12 : A gauche l'image originale, au milieu son image filtrée par un filtre moyenneur (3x3) et à droite un filtre (7x7)

Le filtre moyenneur est un filtre passe-bas permettant ainsi d'éliminer les hautes fréquences, correspondant au bruit. Son inconvénient est qu'il élimine également les hautes fréquences correspondant aux détails de l'image : il rend ainsi l'image moins bruitée mais plus floue.

Filtre smooth

Ce filtre est un exemple de filtre linéaire dont les coefficients, choisis avec plus de soin, permettent un traitement moins grossier de l'image. Les coefficients du noyau pour un filtre 3x3 sont les suivants :

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Figure 13 : A gauche l'image originale, au milieu son image filtrée par un filtre smooth(3x3) et à droite un (7x7)

L'effet du filtre ici aussi augmente avec la taille du noyau. Les détails sont mieux conservés qu'avec le moyenneur.

c. Les filtres non linéaires

Ces opérateurs ont été développés pour pallier aux insuffisances des filtres linéaires : principalement la mauvaise conservation des contours. Ils ont le défaut d'infliger des déformations irréversibles à l'image.

Filtre médian

Le principe consiste à remplacer un pixel par la médiane de ses voisins. Ainsi, même si plusieurs pixels voisins sont bruités, on peut corriger le pixel courant. Ce filtre induit cependant un lissage puisque même des pixels corrects peuvent être modifiés. La netteté est conservée mais le filtre médian élimine les détails fins de manière irrémédiable.

Figure 14 : A gauche l'image originale, au milieu l'image filtrée par un filtre médian (3x3) et à droite un (7x7)

Filtre Nagao

Le principe cette fois-ci consiste à remplacer un pixel par la moyenne des niveaux de gris du voisinage le plus homogène (variance minimale). Les contours sont remarquablement conservés. Les détails fins sont plus apparents qu'avec le médian, cependant les éléments de l'image sont déformés par des effets de blocs. On note également que ce filtre est plus gourmand en ressource que le filtre médian.

Figure 15 : A gauche l'image originale, à droite l'image filtrée par un filtre nagao

d. Débruitage

Les différents filtres que nous avons vu jusque là s'adaptent donc plus ou moins bien aux différents bruits que l'on peut rencontrer sur une image.

Pour un bruit gaussien faible, les filtres linéaires 3x3 conviennent tout à fait. L'utilisation de filtres plus fins ou plus puissants n'entraine aucune amélioration significative. Utiliser un filtre de noyau trop grand entraine une perte dommageable de détail.

Pour un bruit gaussien élevé, les filtres linéaires sont largement aussi efficaces que les autres. Le filtre smooth 7x7 obtient les meilleurs résultats. Le médian 7x7 parvient à bien éliminer le bruit mais au prix d'une perte plus importante de détail.

Pour un bruit impulsionnel de faible intensité, les filtres médian et nagao se trouve relativement efficaces. Par contre, pour un bruit impulsionnel important, les filtres linéaires s'avèrent complètement inefficaces. On privilégie les filtres non linéaires.

Le débruitage est une étape important du traitement d'images. En effet, comme nous pourrons le voir par la suite, les différents filtres de détection de contours sont très sensibles aux bruits. La détection peut alors être ineffective en cas d'image trop bruitée. L'idée est alors d'effectuer un prétraitement en travaillant avec les filtres précédemment cités.

3. Recherche de primitives et détection de contours

La recherche de primitives dans une image est intéressante car elle permet de détecter un ou plusieurs objets dans une scène donnée. Ces primitives peuvent être de type contour, région, point d'intérêt (ou coins), ligne, ou courbe. Il existe différentes méthodes pouvant être utilisées afin de détecter ces primitives. Il est toutefois important de noter qu'il n'existe pas de méthode générale pour la détection d'objet. Il faut s'adapter à chaque cas en fonctions de la situation et des informations que nous avons à notre disposition.

Nous allons dans un premier temps nous intéresser à la détection de contours, à savoir la détection de lieux de sauts d'intensité.

A. Approche basique par un exemple simple

Afin de simplifier au maximum le concept dans un premier temps, nous allons commencer à travailler avec de simples images en noir et blanc. Nous bénéficions alors d'une seule discontinuité de l'intensité et pouvons alors nous intéresser aux limites entre le blanc et le noir. Dans la figure 16, il suffit alors de parcourir l'image de façon horizontale et de s'intéresser à la différence entre un pixel et son voisin. Nous créons ainsi une image basée sur la formule suivante

$$Image_{contour\_horizontal}[i][j] = Image[i][j] - Image[i-1][j]$$

La frontière entre noir / blanc est alors définie par un trait blanc d'un pixel sur un fond noir.

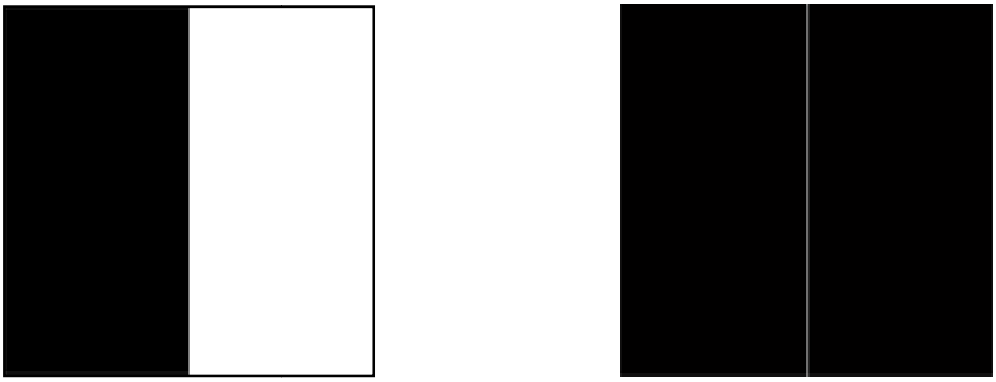


Figure 16 : A gauche une simple image en noir et blanc. A droite le résultat de la détection de contour.

Dans notre cas, la détection de contours ne fonctionne que de manière horizontale, afin de travailler sur la composante verticale nous n'avons qu'à utiliser la formule suivante :

$$Image_{contour\_vertical}[i][j] = Image[i][j] - Image[i][j-1]$$

Afin d'extraire les deux composantes pour former une seule image de contours il suffit de faire la norme du vecteur $[Image_{contour_{horizontal}}, Image_{contour\_vertical}]$.

$$Image_{contour} = \sqrt{Image_{contour\_horizontal}^{2} + Image_{contour\_vertical}^{2}}$$

Il est alors possible de détecter les contours sur des images en noir et blanc

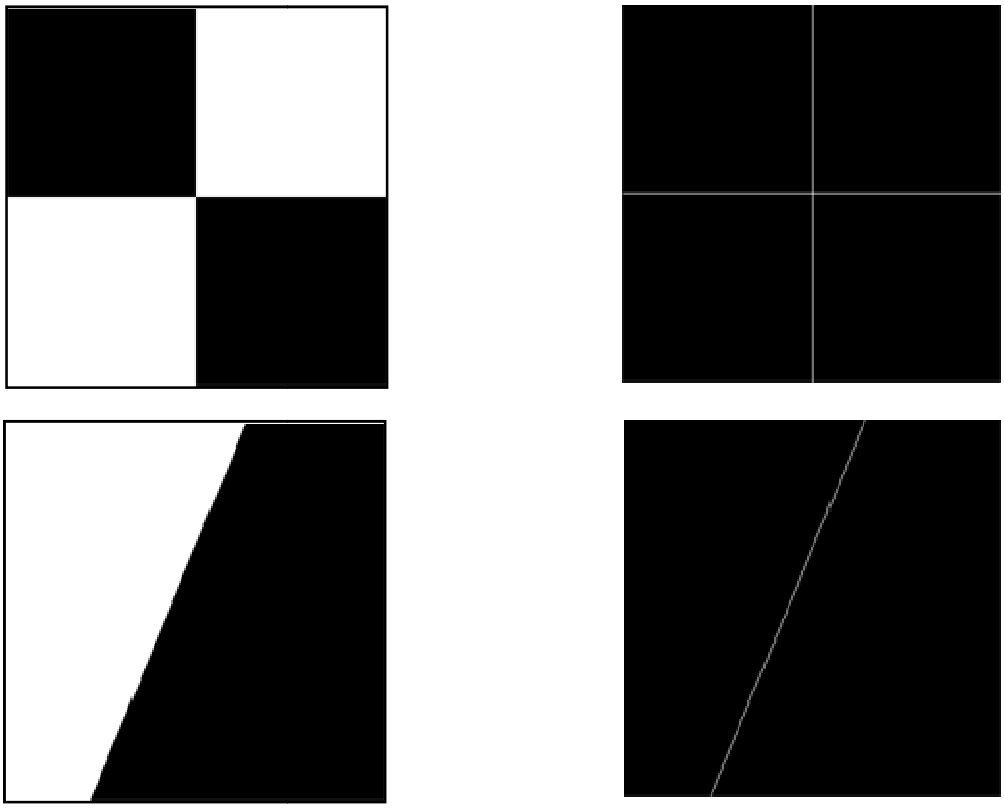


Figure 17 : Résultats de détection des contours verticaux et horizontaux.

Il est alors possible d'appliquer notre détecteur de contour sur une image naturelle en couleur en travaillant sur son seuil de binarisation. Le principe est le même.

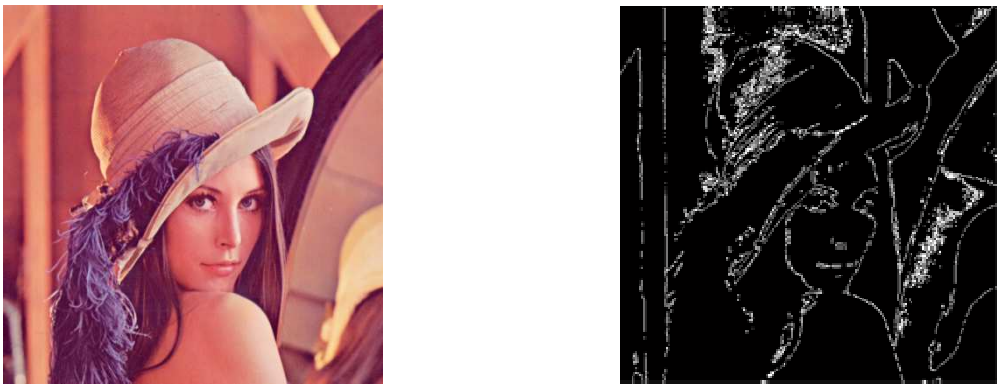


Figure 18 : Détection de contours sur une image naturelle.

B. Approche par convolution

Notre premier détecteur de contour fonctionne mais reste très basique dans son utilisation. De plus il est très sensible au bruit, détecte trop de contours et détecte mieux des contours diagonaux qu'horizontaux ou verticaux. Nous allons donc nous intéresser à d'autres techniques, qui utilisent notamment le concept de convolution.

i. Application de filtres

La technique la plus répandue de recherche de discontinuités se base également sur l'utilisation de filtres sur l'image. Nous utilisons alors l'opérateur de convolution comme précédemment avec les filtres de restauration d'image. Détecter les contours correspond alors à rechercher un filtre qui permet de déterminer les fortes variations de couleurs dans l'image. Pour cela, il y a de nombreux moyens, plus ou moins efficaces selon la qualité ou le type de l'image sur lequel on travaille. Je vais ainsi faire un tour des différents types de filtres utilisables pour la détection de contour.

ii. Détection de points

Tout d'abord, il peut être intéressant de détecter des points isolés dans des zones d'intensité constante (ou s'y rapprochant). L'utilisation du masque suivant (Figure 19) permet alors de détecter les points isolés à l'emplacement sur lequel le masque est centré.

La convolution de l'image et du filtre permet alors, par exemple, de mettre en évidence simplement des points qui, à l'œil nu pourraient paraître invisible. La figure 20 nous montre une image grise foncée contenant des points noirs. La différence est presque invisible à l'œil nu, et la détection de points permet de mettre en évidence ces derniers.

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

Figure 19 : Masque de détection de point

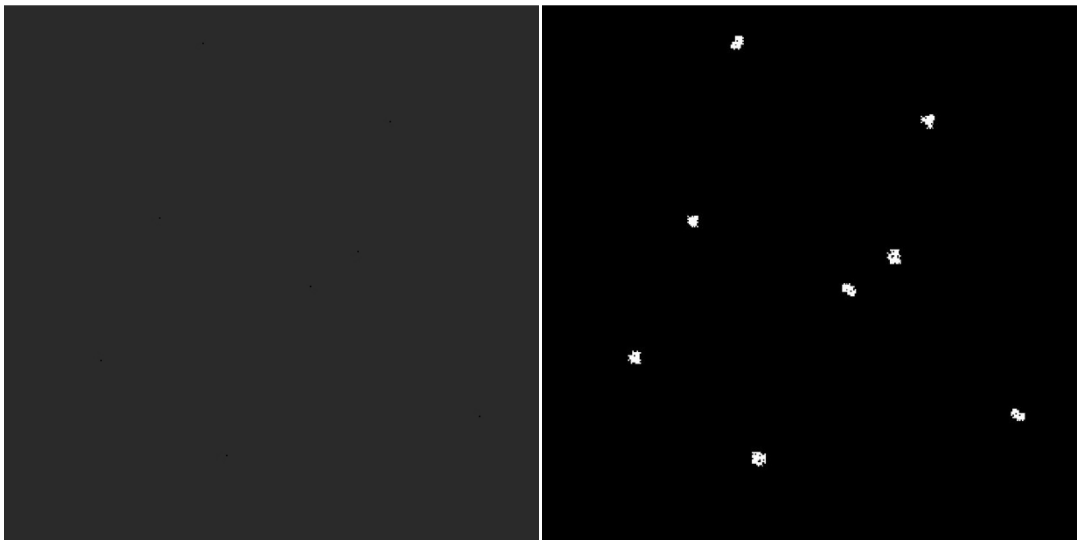


Figure 20 : Détection de points.

iii. Détection de lignes

L'étape suivante est la détection de lignes. Nous pouvons obtenir un résultat similaire à notre premier exemple basique grâce à de simples convolutions. Les filtres présents sur la

figure 21 répondent respectivement aux lignes horizontales, lignes orientées à +45°, lignes verticales et lignes orientées à -45°

$$\begin{pmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{pmatrix} \quad \begin{pmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{pmatrix} \quad \begin{pmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{pmatrix} \quad \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix}$$

Filtre horizontal | Filtre +45° | Filtre vertical | Filtre -45°

Figure 21 : Masques de détection de lignes

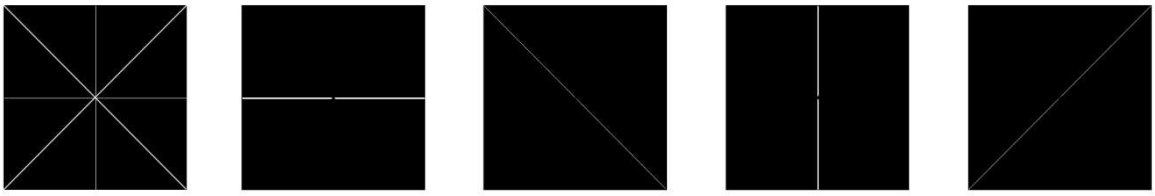


Figure 22 : Résultats de détection, respectifs à la figure XX. L'image tout à gauche étant l'image originale.

iv. Détection de contours

Dérivée première

Nous allons tenter d'améliorer notre premier détecteur en travaillant avec les dérivées. Permettant d'étudier la variation d'un signal, la dérivée est particulièrement adaptée à notre cas et permet de détecter les fortes variations d'intensité de l'image. Les contours correspondent à des maxima locaux de la dérivée première. En 2D, la dérivée première correspond au gradient : $\nabla I = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right]$, vecteur dont les composantes sont la dérivée partielle en x et y de l'image. On admet donc communément que les contours correspondent aux discontinuités d'ordre 0 de l'image. La détection de contours peut s'effectuer de deux manières différentes:

Détecteur de Roberts

Le détecteur de Roberts recherche les dérivées selon les directions diagonales. Il est décomposé en deux masques de convolution pour chacune des composantes du gradient.

$$\frac{\partial I}{\partial x} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \qquad \frac{\partial I}{\partial y} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

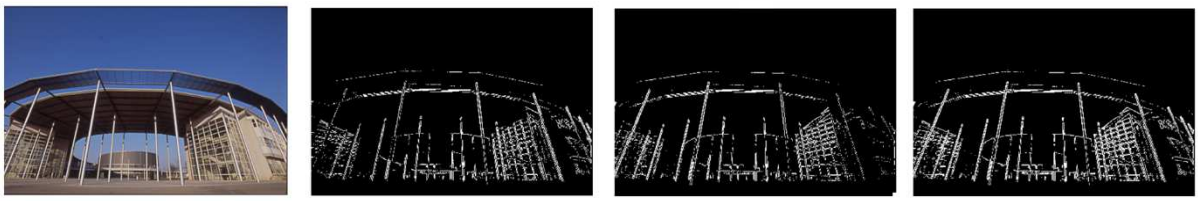
Le calcul du module du gradient permet alors d'obtenir le tracé de contour :

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

Ce qui peut également être réalisé par :

$$\max\left(\left(\frac{\partial I}{\partial x}\right), \left(\frac{\partial I}{\partial y}\right)\right)$$

Le détecteur de Roberts est un des plus vieux détecteurs de contours en traitement d'image numériques, mais il est à la fois le plus simple. Ce détecteur est bien moins utilisé que ceux de Sobel et Prewitt en raison de ses fonctionnalités limitées. En effet il bénéficie des mêmes inconvénients que notre exemple basique. Cependant il est fréquemment utilisé dans des implémentations matérielles où la simplicité et la vitesse sont des facteurs dominants.



Original Gx Gy resultat

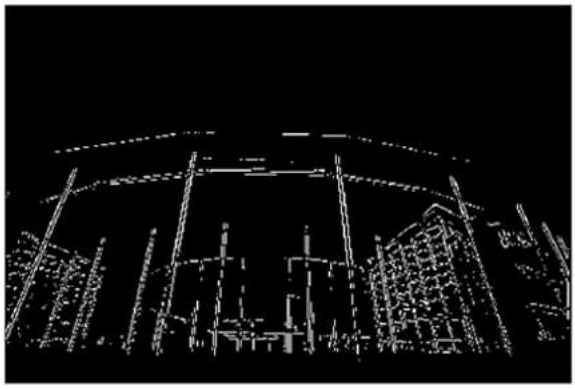
On peut éventuellement remarquer que les images dérivées sont bruitées. En effet, le bruit initial de l'image est amplifié par la dérivée. Cela est d'autant plus visible lorsqu'il s'agit de bruit de haute fréquence. Il convient alors d'effectuer un filtre passe-bas ou un lissage afin d'éliminer ces hautes fréquences.

Détecteur de Sobel

Le détecteur de Sobel utilise le masque suivant pour approximer numériquement les dérivées premières en x et y

$$\frac{\partial I}{\partial x} = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}, \qquad \frac{\partial I}{\partial x} = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 1 & 1 \end{pmatrix}$$

Le filtre de Sobel combine à la fois le lissage par le filtre monodimensionnel [1, 2, 1] et la dérivée selon une direction perpendiculaire au lissage, obtenue par le filtre [1, 0, -1]. On peut voir le résultat comme la moyenne des dérivées des moyennes. Comme pour le détecteur de Roberts, l'image des contours est obtenue en calculant, pour chaque pixel le module du gradient. On peut appliquer ce détecteur sur des images couleurs en appliquant le même algorithme sur les différentes composantes RGB prises séparément.

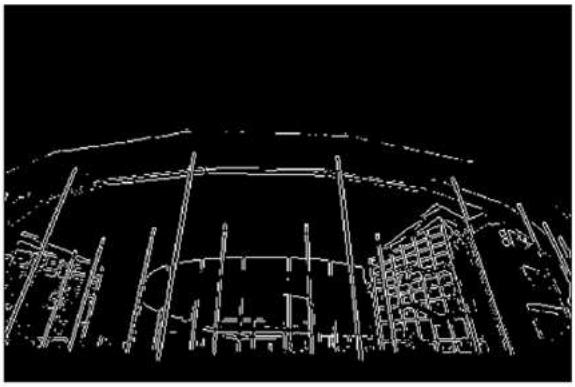



On remarque que certains contours sont bien détectés alors que d'autres posent problème au détecteur. Les portions contenant trop de détails sont notamment trop segmentées.

Détecteur de Prewitt

La détection du contour par la méthode de Prewitt utilise les masques de convolution suivants :

$$\frac{\partial I}{\partial x} = \begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}, \qquad \frac{\partial I}{\partial x} = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 1 & 1 \end{pmatrix}$$

D’autres filtres basés sur les dérivées premières existent, ils s’intéressent généralement à des directions supplémentaires.

Détecteur de Kirsch

Le filtre de kirsch permet de détecter des contours dans plus de 2 directions. Les masques suivants permettent de détecter les contours à 0°, -45°, -90°, et -135°

$$\begin{pmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{pmatrix}, \quad \begin{pmatrix} 3 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & -5 & 3 \end{pmatrix}$$

$$\begin{pmatrix} -5 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & 3 & 3 \end{pmatrix}, \quad \begin{pmatrix} -5 & -5 & 3 \\ -5 & 0 & 3 \\ 3 & 3 & 3 \end{pmatrix}$$

On obtient des résultats similaires aux précédents détecteurs. Le filtre de Kirsch peut être également utilisé de façon plus complexe sur 8 directions afin de gagner en précision sur les contours.

Dérivée seconde

Après nous êtres intéressés aux recherches de maxima des dérivées premières, nous allons nous penser sur les filtres se basant sur les zéros de la dérivée seconde, c'est-à-dire, du Laplacien.

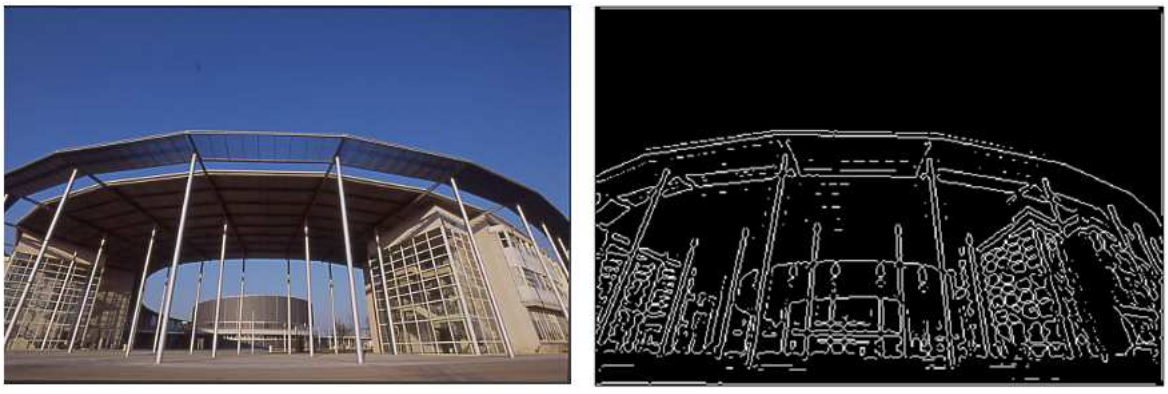
$$\nabla I^2 = \left[\frac{\partial^2 I}{\partial x^2}, \frac{\partial^2 I}{\partial y^2}\right]$$

J'avais indiqué précédemment que le bruit sur l'image initiale pouvait être une source de bruit amplifié sur l'image finale. Les dérivées secondes sont encore plus sujettes à cette manifestation de par la double dérivation. Il est donc courant d'utiliser une combinaison de lissage et Laplacien, ce qui correspond au Laplacien d'une gaussienne.

Les filtres de convolution correspondant au laplacien d'une gaussienne respectivement de connexités 4 et 8 sont les suivants :

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

La figure suivante montre le résultat de la détection. Pour arriver à ce résultat, il faut au préalable lisser l'image en raison de la double dérivation, sans quoi on se retrouve avec une détection très bruitée.



Suivant les types de photographies, on obtient des résultats qui peuvent beaucoup varier. Il peut alors êtres plus intéressant suivant le contexte, de rester sur les dérivées premières.

C. Approche par filtrage optimal

i. Filtre de Canny

Une autre approche de détection de contour dite de Canny consiste à étudier les critères de qualité demandés à un détecteur de contours afin de rechercher un détecteur optimal. L'algorithme de Canny, basé sur le filtre de Sobel, est conçu pour être optimal suivant trois critères :

- **Bonne détection** : Plus le filtre lisse le bruit, plus la détection est bonne. On cherche alors à maximiser le rapport signal sur bruit.
- **Bonne localisation** : Moins le filtre lisse l'image, meilleure est la localisation. On cherche alors à minimiser les distances entre les contours détectés et les contours réels.
- **Non multiplicité des réponses** : On veut une réponse unique par contour et pas de faux positifs.

La mise en œuvre de ce filtre est la suivante :

- **Réduction du bruit** : La première étape consiste à réduire le bruit de l'image avant d'en détecter les contours. On utilise pour cela un filtrage gaussien. Plus le masque utilisé est grand moins le détecteur est sensible au bruit et plus l'erreur de localisation grandit.
- **Gradient d'intensité** : Après le filtrage, l'étape suivante est d'appliquer un gradient ∇I qui retourne l'intensité des contours. L'opérateur utilisé et le suivant :

$$G_x = \frac{\partial I}{\partial x} = (-1 \quad 0 \quad 1) \qquad G_y = \frac{\partial I}{\partial y} = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$$

- **Direction des contours** : Les orientations des contours sont déterminées par la formule :

$$\theta = arctan(\frac{G_x}{G_y})$$

On obtient alors une carte des gradients d'intensité en chaque point de l'image accompagnée des directions des contours.

- **Suppression des non-maxima** : Ca carte des gradients indique une intensité en chaque point de l'image. Une forte intensité indique une forte probabilité de présence d'un contour. Cependant, cette intensité ne suffit pas à décider si un point correspond à un contour ou non. Seuls les points correspondant à des maxima locaux sont considérés comme correspondant à des contours, et sont conservés pour la prochaine étape de la détection. Un maximum local est présent sur les extrema du gradient, c'est à dire là où sa dérivée s'annule.
- **Seuillage des contours** : La différenciation des contours sur la cartée générée se fait par seuillage hystéresis. Cela nécessite deux seuils, un haut et un bas ; qui seront comparés à l'intensité du gradient de chaque point. Si l'intensité du gradient de chaque point est inférieure au seuil bas, le point est rejeté. Si elle est supérieure au seuil haut, le point est accepté comme formant un contour. Si elle est entre le seuil bas et le seuil haut, le point est accepté s'il est connecté à un point déjà accepté.

La taille du filtre de réduction du bruit ainsi que le choix du seuil détermineront l'efficacité de l'algorithme et le temps de calcul nécessaire. Le filtre de Canny s'adapte à de nombreux environnement, cependant dans certains cas le traitement informatique peut être particulièrement lent notamment lorsque la quantité de lissage nécessaire est importante. Le filtre de Deriche est alors souvent préconisé.

Figure 23 : Exemple de filtre de Canny

ii. Filtre de Deriche

Deriche, utilise la même démarche que Canny. L'approche développée par Deriche a consisté en la recherche de l'opérateur optimal sous forme de filtre à une réponse impulsionnelle infinie (RII). L'implémentation récursive du filtre optimal RII permet de meilleurs résultats.

Tout d'abord, on utilise un opérateur de lissage noté SS sur une image x(m,n), correspondant à deux gradients directionnels SSx et SSy :

$$SS(m,n) = k_2(\alpha\,|m| + 1)e^{-\alpha.|m|}k_2(\alpha[n| + 1)e^{-\alpha.|n|}$$

$$SSx(m,n) = k_1 m e^{-\alpha.|m|}.k_2(\alpha\,|n| + 1)e^{-\alpha|n|}$$

$$SSy(m,n) = k_1 n e^{-\alpha.|n|}.k_2(\alpha\,|m| + 1)e^{-\alpha|m|}$$

La première phase consiste alors à parcourir les lignes d'une image x(m,n) par colonne croissante puis colonne décroissante avant de calculer la somme des deux résultats. Le filtrage récursif suivant est alors appliqué :

y1(m,n) = a1x(m,n) + a2x(m,n − 1) + b1y1(m,n − 1) + b2y1(m,n − 2)

pour n=1,...,N et m=1,...,M

y2(m,n) = a3x(m,n + 1) + a4x(m,n + 2) + b1y2(m,n + 1) + b2y2(m,n + 2)

pour n=N,...,1 et pour m=1,...,M

r(m,n) = c1(y1(m,n) + y2(m,n)) pour n=1,...,N et pour m=1,...,M

La seconde phase consiste à traiter par ligne croissante puis décroissante les colonnes du résultat de la phase 1.

y1(m,n) = a5r(m,n) + a6r(m − 1,n) + b1y1(m − 1,n) + b2y1(m − 2,n)

pour m=1,...,M et pour n=1,...,N

y2(m,n) = a7r(m + 1,n) + a8r(m + 2,n) + b1y2(m + 1,n) + b2y2(m + 2,n)

pour m=M,...,1 et pour n=1,...,N

y(m,n) = c2(y1(m,n) + y2(m,n) pour n=1,...,N et pour m=1,...,M

Avant la première étape, on obtient une image plus ou moins lissée. En effet plus α est petit plus le lissage est fort, plus α est grand, plus le lissage est faible. Après passage du filtre récursif, on remarque que les contours apparaissent plus épais avec des valeurs faibles de α. Ils sont alors facilement détectables mais moins bien localisés. Et inversement, pour un α grand, les contours sont plus localisés et plus fins, mais moins facilement détectables.

4. Opérateurs morphomathématiques

Le terme morphologie est associé communément à une branche de la biologie qui traite des formes et des structures des plantes et des animaux. Nous utilisons le même mot dans le contexte mathématique en tant que technique d'extraction d'éléments d'une image tels que des formes et des régions. Les images de contours ne nous permettent pas de segmenter une scène en plusieurs régions. Nous allons ainsi nous intéresser aux opérateurs morphomathématiques qui permettent d'améliorer les images des contours.

A. Notions de base

L'appartenance d'un élément x, à un ensemble A, s'écrit $x \in A$

De manière similaire, si x n'est pas un élément de A, on écrit $x \notin A$

On définit la notion de complément de A tel que $A^c = \{x | x \notin A\}$

L'union de deux éléments est notée $C = A \cup B$

L'intersection de deux éléments est notée $C = A \cap B$

La différence des ensembles A et B est définie par

$$A - B = A/B = A \backslash B = A \cap B^c = A/(A \cap B) = \{x | x \in A, x \notin B\}$$

La réflexion d'un ensemble A est définie par – $A = \{x | x = -a, pour\ b \in A\}$

On peut également rappeler les lois de Morgan

$$A/(B \cup C) = (A/B) \cap (A/C)$$

$$A/(B \cap C) = (A/B) \cup (A/C)$$

Dont les relations suivantes découlent

$$(A \cup B)^c = A^c \cap B^c$$

$$(A \cap B)^c = A^c \cup B^c$$

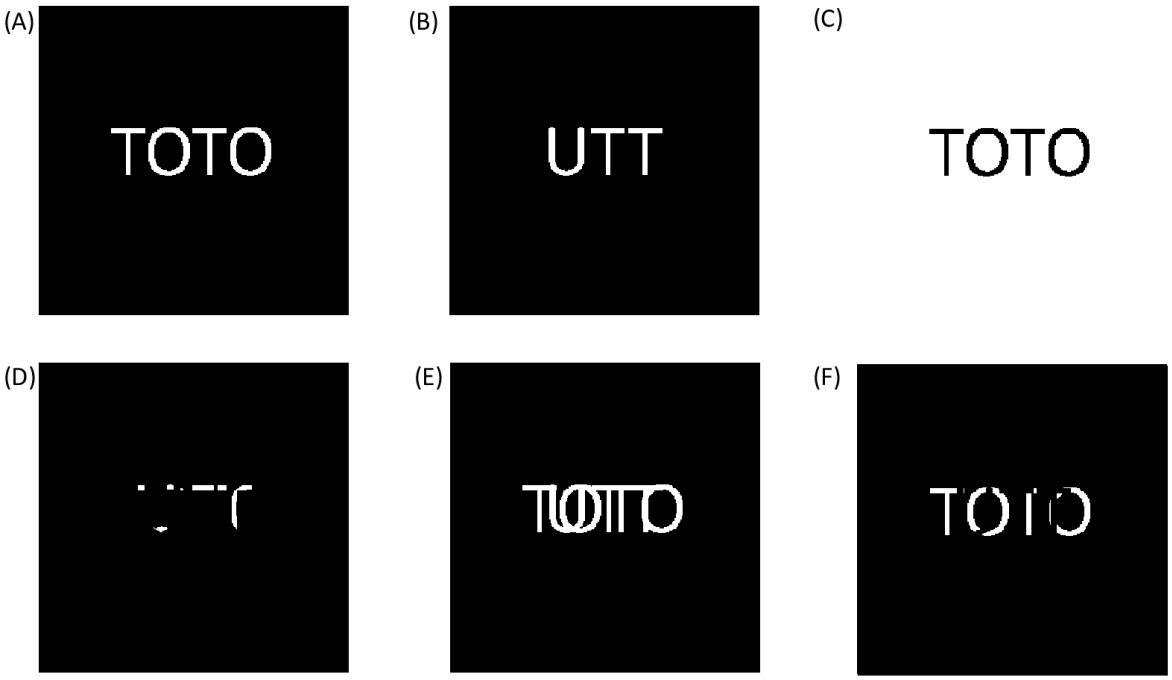


Figure 24 : (A) : Image A ; (B) : Image B ; (C) : -A ; (D) : A∩B ; (E) : AUB ; (F) : A – B

Addition de Minkowski

L'addition d'un élément x à un ensemble A, appelée également translation de l'ensemble A par x, est définie par :

$$B = A + x = \{a + x \; pour \; a \in A\}$$

On en déduit l'addition d'un ensemble B à un ensemble A : c'est le résultat de l'addition de chaque élément de B à A.

$$C = A \oplus B = \bigcup (A + b, b \in B) = \bigcup (B + a, a \in A) = \{x | (-B + x) \cap A \neq \emptyset \}$$

Éléments structurants

Les opérateurs morphomathématiques utilisent de petits ensembles discrets appelés « éléments structurants ». Ces derniers permettent de définir le type de voisinage que l'on souhaite considérer. Suivant le résultat escompté, nous pouvons utiliser différents éléments structurants. Il existe de nombreux éléments structurants de différentes formes (diamant, disque, ligne, octagone, rectangle, etc.) En voici quelques un, simples:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Elément structurant en 4-connexité

(dilatation plus « arrondie »)

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Elément structurant en 8-connexité

(dilatation plus « angulaire »)

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Elément structurant vertical en 2-connexité

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

Elément structurant horizontal

en 8-connexité

On pourra remarquer par la suite que les opérations morphomathématiques reviennent en fait à de simples convolutions de l'image avec l'élément structurant choisi.

B. Dilatation

La dilatation est une opération qui « grandit » ou « épaissit » des objets dans une image binaire. D'un point de vue ensembliste, le résultat de la dilatation de l'ensemble A (Image) par l'ensemble B (élément structurant) est l'addition de Minkowski : c'est l'ensemble des points tels que lorsque B est centré sur un de ces points il y a une intersection non vide entre A et B.

Dilatation d'une image en noir et blanc

Soit K un élément structurant (masque binaire) de taille (2K+1) par (2K+1). L'algorithme de dilatation est le suivant : on parcourt tous les pixels de l'image excepté les bords de l'image d'épaisseur K (on parcourt les lignes de K à h-K et les colonnes de K à w-K). Pour chaque pixel du fond rencontré, s'il possède un voisin, au sens de l'élément structurant, qui appartient à un objet, il prend la couleur de l'objet ; sinon, il n'est pas modifié. Concrètement, pour une image en noir et blanc, on considère que les pixels noirs appartiennent au fond tandis que les pixels blancs appartiennent aux objets. Dans le type d'algorithme, il est important d'écrire le résultat dans une nouvelle image afin de traiter chaque pixel indépendamment du résultat des voisins.

L'exemple suivant permet de montrer un exemple de dilatation.

Figure 25 : Image originale en (A), Images dilatées en 4-connexité (B) et en 8-connexité (C)

Dilatation d'une image en niveaux de gris

Pour travailler sur une image en niveaux de gris, il faut adapter légèrement l'algorithme afin de jouer avec l'intensité pixels voisins. Les pixels de fond sont toujours les pixels noirs. Les autres pixels appartiennent donc aux objets. La condition de modification d'un pixel est identique. Par contre, dans le cas où plusieurs voisins appartiennent aux objets, il faut choisir quelle valeur va être utilisée pour colorier le pixel dilaté : on prend alors la valeur maximale parmi les pixels correspondant au schéma de l'élément structurant.

C. Erosion

La dilatation est une opération qui « rétrécit » ou « amincit » les objets d'une image binaire. La définition mathématique de l'érosion est similaire à celle de la dilatation. Elle est notée :

$$C = A \ominus B = \bigcap(A - b, b \in B) = \bigcap(B - a, a \in A) = \{x | (B + x) \subset A\}$$

avec A pour image et B pour élément structurant

On définit la soustraction de Minkowski par :

$$C = A - (-B) = \bigcap(A + b, b \in B)$$

Le principe de l'érosion est le même que la dilatation, mais de façon inversée. En effet, pour que le pixel prenne la couleur de l'objet, ses pixels connexes doivent tous êtres compris (entièrement inclus) dans l'élément structurant.

Erosion d'une image en noir et blanc

L'algorithme d'érosion est très similaire à celui de dilatation : on parcourt tous les pixels de l'image excepté les bords de l'image d'épaisseur K. Pour chaque pixel rencontré n'appartenant pas au fond, s'il possède un voisin, au sens de l'élément structurant, qui appartient au fond, il prend la couleur du fond ; sinon, il n'est pas modifié.

L'exemple suivant permet de montrer un exemple d'érosion.

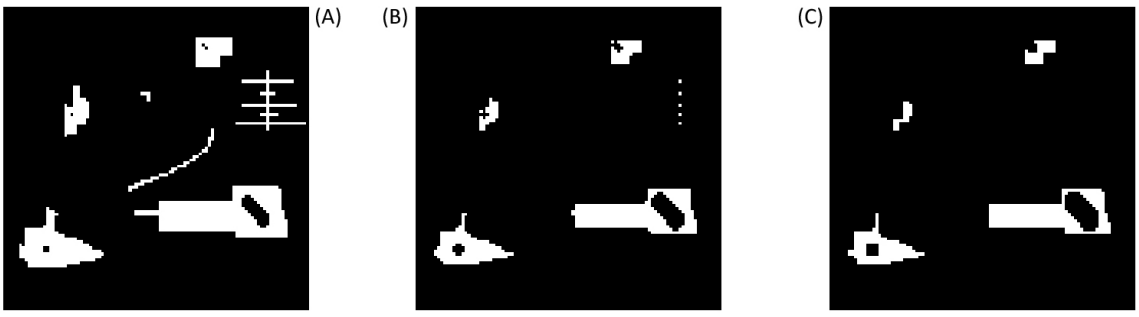


Figure 26 : Image originale en (A), Images érodées en 4-connexité (B) et en 8-connexité (C)

Erosion d'une image en niveaux de gris

Le système est le même que pour la dilatation d'images en niveaux de gris. A la différence, que nous choisirons la valeur minimale pour colorier le pixel courant dans le cas où plusieurs voisins appartiennent au fond.

D. Ouverture et fermeture – Combinaison Dilatation/Erosion

L'ouverture morphologique de A par B, notée $A \circ B$ est simplement l'érosion de A par N, suivie d'une dilatation du résultat par B :

$$A \circ B = (A \ominus B) \oplus B$$

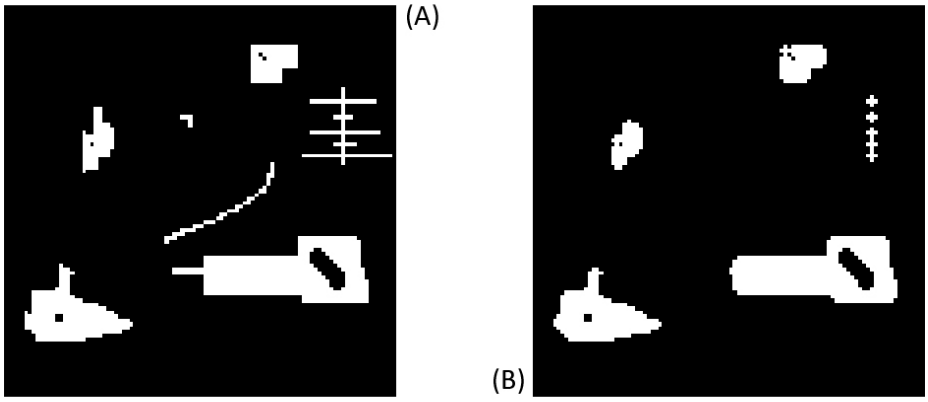


Figure 27 : (A) Image originale ; (B) Résultat d'une ouverture

On remarque que les petites structures ainsi que les détails fins disparaissent de l'image. Cette technique permet ainsi d'éliminer des petits morceaux de contours bruités dans une image de contours.

La fermeture morphologique de A par B, notée $A \cdot B$ correspond à une dilatation suivie d'une érosion :

$$A \cdot B = (A \oplus B) \ominus B$$

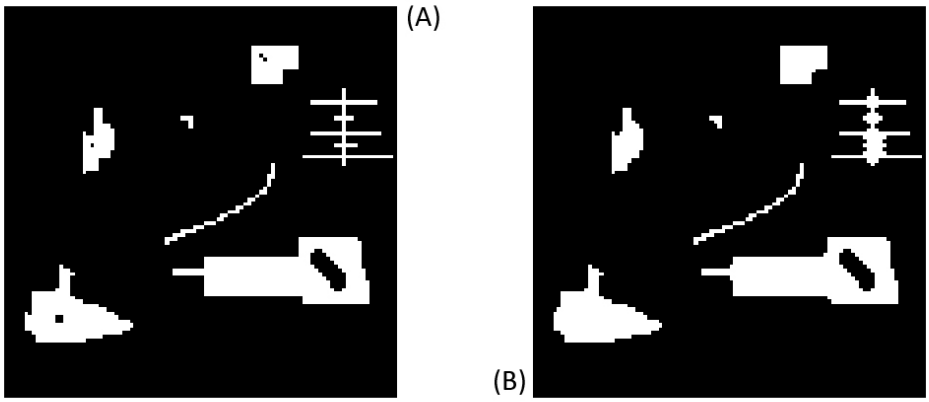


Figure 28 : (A) Image originale ; (B) Résultat d'une fermeture

Comme l'ouverture, la fermeture morphologique tend à lisser les contours des objets. Contrairement à l'ouverture, les structures proches fusionnent généralement. On peut remarquer également que les rainures fines sont remplies, ainsi que les trous plus petits que l'élément structurant.

E. Gradient morphologique

Il est possible d'obtenir les contours des différents objets de l'image en travaillant sur des successions d'opérations sur celle-ci.

En prenant le complémentaire de A par rapport à son érodé : $A/(A \ominus B)$, on obtient le contour intérieur.

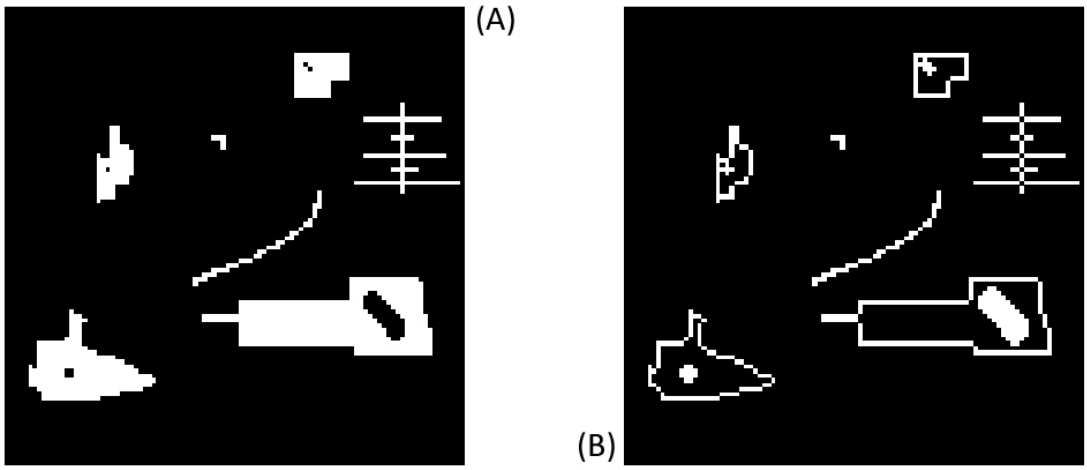


Figure 29 : Original (A) et son contour intérieur (B)

En prenant le complémentaire du dilaté par rapport à A : $(A \oplus B)/A$, on obtient le contour extérieur.

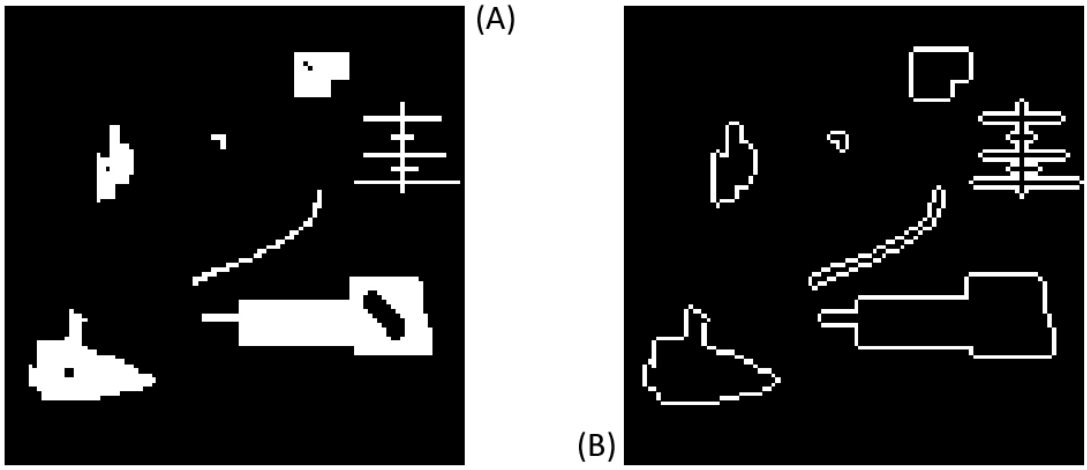


Figure 30 : Original (A) et son contour extérieur (B)

En prenant le complémentaire du dilaté par rapport à l'érodé A : $(A \oplus B)/\ (A \ominus B)$, on obtient le contour moyen, autrement appelé gradient morphologique.

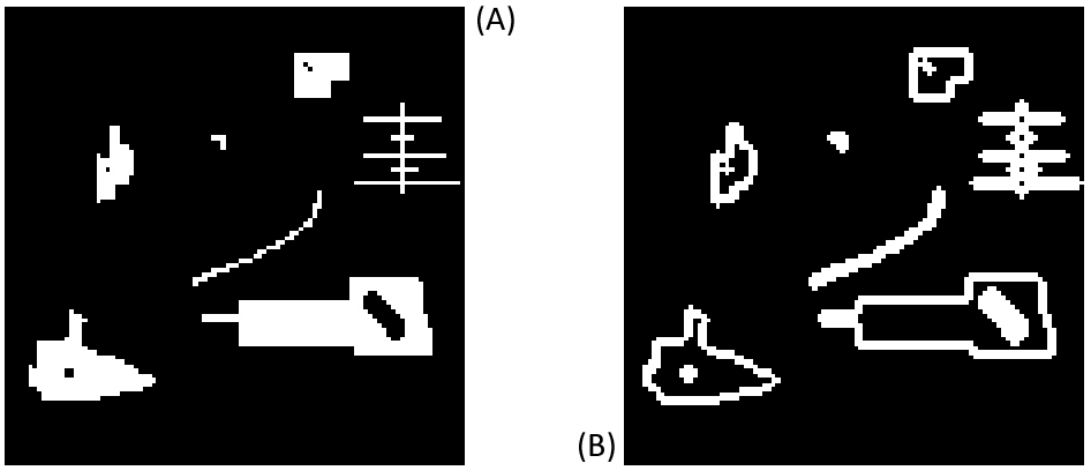


Figure 31 : Original (A) et son contour moyen (B)

F. Amincissement et squelettisation

Le squelette d'une forme correspond à son axe médian, c'est-à-dire la trace des centres des disques maximaux entièrement contenus dans cette forme. La squelettisation permet entre autres la reconnaissance de caractères, la détection de réseaux routiers, la planification de trajectoire, etc. nous allons voir que pour obtenir un squelette, une des méthodes possibles consiste à appliquer un amincissement répété sur l'image jusqu'à convergence.

i. Transformée « tout ou rien » (Hit-or-Miss transformation)

Il est souvent utile de détecter des configurations spatiales données telles que des formes précises, lettres, etc. La transformée Hit-or-miss est utile dans ce cas, et permet en utilisant l'érosion et une paire d'éléments structurants, de détecter la position de telles objets.

Cette transformation est notée

$$A \otimes B = (A \ominus B_1) \cap (A^c \ominus B_2)$$

Où A est l'image originale, B est une paire d'élément structurants, tels que $B = (B_1, B_2)$ avec B_1 et B_2 étant respectivement la forme recherchée et son complémentaire local. La transformation $A \otimes B$ correspond donc à l'ensemble des points qui appartiennent à l'érodé de A par B ainsi qu'à l'érodé du complémentaire de A par le complémentaire local de B. Nous pouvons ainsi par exemple rechercher respectivement les lettres U et T dans une image donnée, la figure suivante nous montre le résultat de la transformée « tout ou rien » vis-à-vis des lettes U et T. Nous obtenons alors les pixels sur lesquels la ou les formes sont centrées.

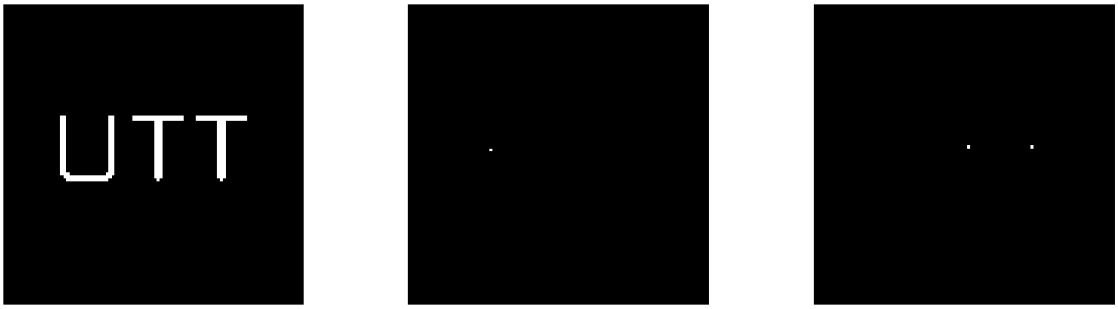


Figure 32 : Détection et localisation des formes U et T

ii. Amincissement

La transformation « tout ou rien » nous permet également de définir l'amincissement noté :

$$A/(A \otimes B)$$

En choisissant B habillement, ceci correspond grossièrement à une érosion qui conserve les structures des formes et les connexions entre formes. Des érosions successives de A par B conduisent à la disparition complète de A, tandis que des amincissements successifs conduisent au squelette (Figure 33).

Le squelette de fond d'une image d'obstacles permet de connaître l'ensemble des lieux les plus loins des obstacles : c'est une information utile pour la navigation d'un robot dans un tel environnement par exemple. (Figure 34)

Dans la reconnaissance d'écriture manuscrite, la squelettisation peut être utilisée en prétraitement afin de faire apparaître que les informations principales. (Figure 33)

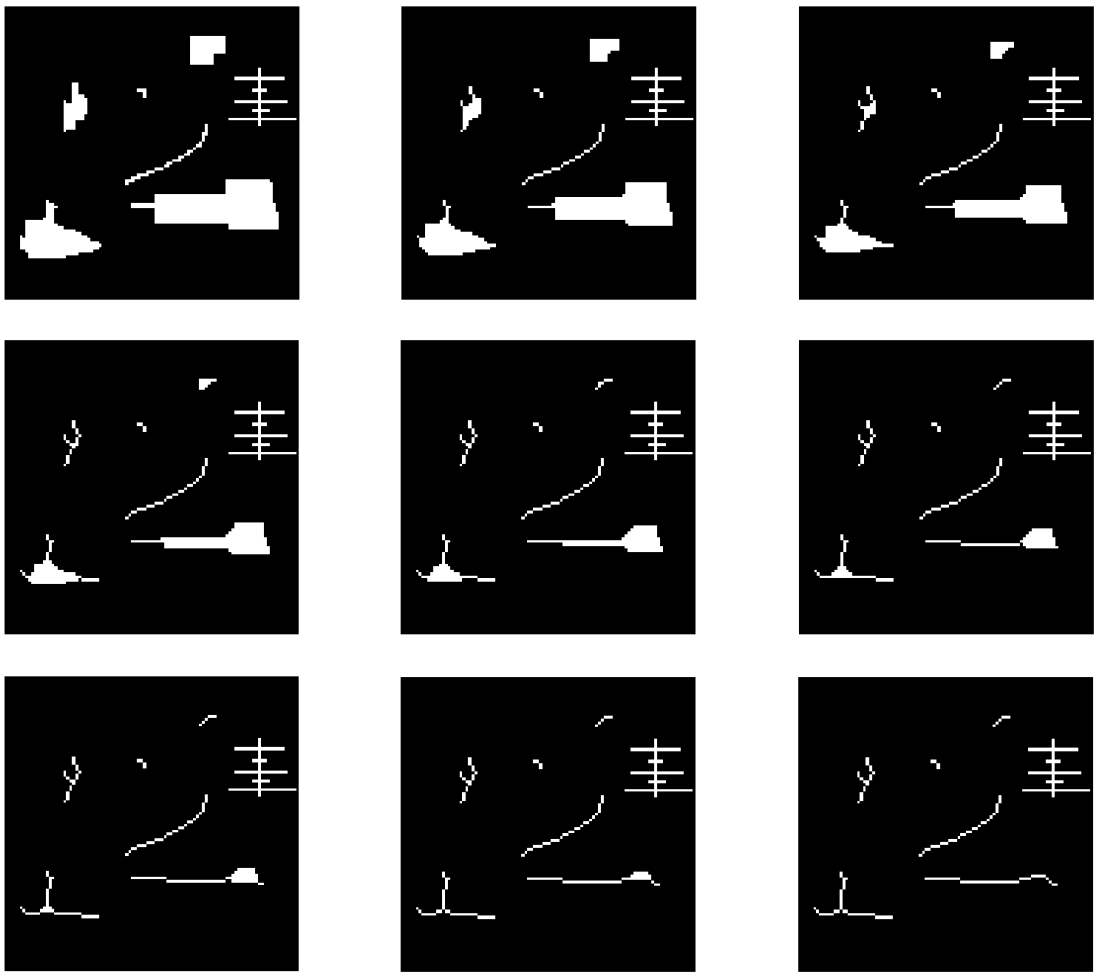


Figure 33 : Squelettisation – Application successive d'amincissements.

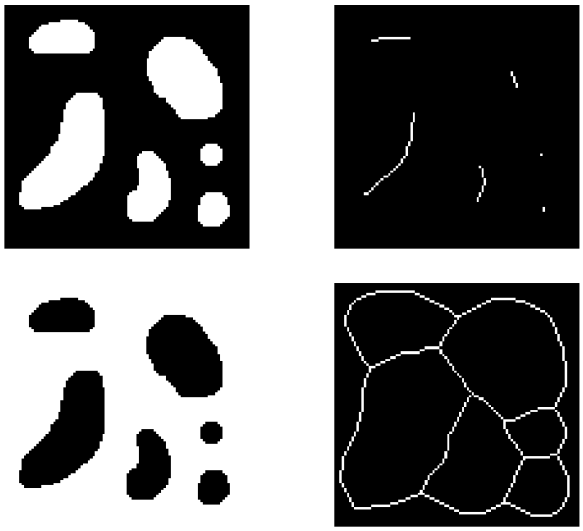


Figure 34 : A gauche les images originales, et à droite, leur squelette.

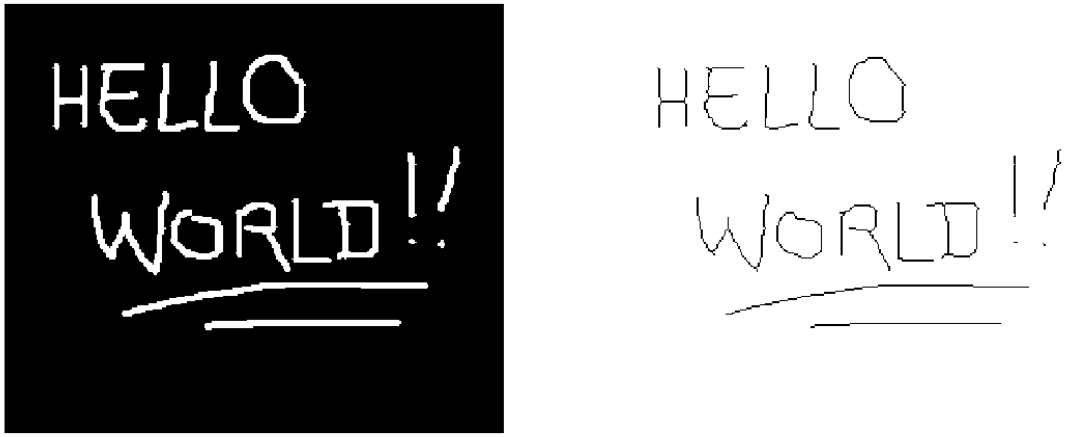


Figure 35 : Squelettisation d'une image manuscrite

5. Détection de régions

A. Détection de régions par seuillage

La façon la plus simple de détecter des régions d'intensité uniforme reste le seuillage. Une binarisation permet, associée à un seuillage adapté, d'isoler des zones sans forcément mettre en œuvre des filtres complexes et couteux en ressources. Cette technique fonctionne bien sur un certain nombre d'images qui ont subît un prétraitement ou du lissage par exemple. Cette méthode très simple peut être judicieusement utilisée dans certains cas. Ce genre de détection peut nécessiter toutefois l'utilisation d'érosions et de dilatations afin de supprimer d'éventuels points erronés. Cependant, tant que l'on peut simplifier le traitement, autant en profiter.

Figure 36 : Exemple de détection de régions par simple seuillage

Il existe une seconde technique de détection de contours, appelée seuillage par hystéresis. Le principe consiste à borner le gradient d'une image entre deux seuils (haut et bas). On obtient alors une image binaire qui ne fait apparaître que les contours détectés. Le problème réside donc dans le choix du seuil : un seuil trop bas conduit à des sur détections (trop de bruit), un seuil trop haut conduit à des contours non fermés.

Figure 37 : Exemple d'image seuillée par hystéresis avec les seuils 44 et 60

B. Détection de contours fermés

La plupart des détecteurs de contours fournissent des contours ouverts, c'est-à-dire qu'ils ne séparent pas les composantes de l'image dans des objets topologiquement distincts. Cela rend souvent plus difficile l'étape de reconnaissance des formes qui doit suivre. Il existe pour cela différentes méthodes qui permettent de récupérer des contours clos.

i. Chaînage de contours

Le chaînage de contours consiste à relier les points des contours connexes obtenus suite à une détection de contours. Il peut s'effectuer par un parcours des points constituant les contours voisins et par la fabrication de listes de points voisins sous la forme de polynômes, voir des arcs ou courbes paramétrées.

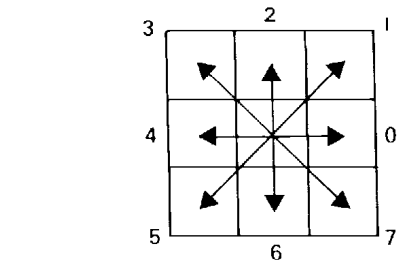
Sur une image sur laquelle on a déjà effectué une détection de contours et un seuillage, on parcourt les pixels ligne par ligne et colonne par colonne. On va alors construire de façon récursive des listes de polygones qui sont alors des listes de points via l'algorithme suivant :

```
Parcours des pixels de l'image
      Si le point courant est un point contour
            Créer un nouveau polygone vide
            Ajouter le point courant à ce polygone
            Marquer le point courant comme déjà pris
            Si le point en colonne suivant est aussi de contour
                  Itérer la construction du polygone courant
                  Ajouter le polygone à la liste des polygones
            Si le point en ligne suivante est aussi de contour
                  Itérer la construction du polygone courant
                  Ajouter le polygone à la liste des polygones
            Si le point en diagonale descendante est aussi de contour
                  Itérer la construction du polygone courant
                  Ajouter le polygone à la liste des polygones
```

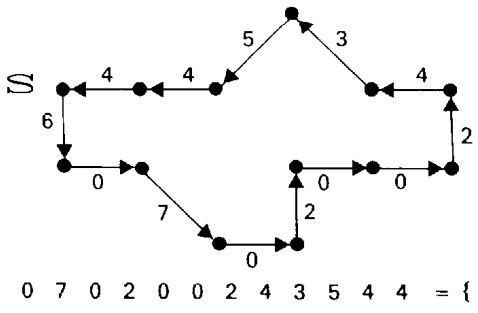
Une fois tous ces polygones récupérés, on cherche à les simplifier et à ne retenir que les sommets pertinents mais aussi à fusionner les polygones qui se suivent. L'idée alors est de représenter ces polygones. On utilise pour cela le codage de Freeman.

ii. Code de Freeman

Le code de Freeman permet de coder un contour chaîné en ne stockant que l'ensemble des déplacements relatifs entre pixels voisins. Le précédent algorithme référençait une liste de points, ce codage permet de représenter le polygone de façon réduite grâce à la notion de direction plutôt que de position.



a.



b.

6 0 7 0 2 0 0 2 4 3 5 4 4 = { Arc $_i$ }

iii. Fermeture de contours

Il existe différentes méthodes de fermeture de contours se basant sur des images de contours.

Une première méthode consiste à utiliser la transformée de Hough (voir partie suivante). Elle fonctionne bien pour des structures polygonales et pas trop nombreuses. Elle peut cependant conduire à des erreurs en fusionnant des segments qui ne devraient pas l'être.

Une seconde approche consiste à utiliser les opérateurs morpho mathématiques si les espaces entre les contours ne sont pas trop gros. On peut également songer à prolonger les contours à leurs extrémités, dans la direction tangente au contour et déterminer si on rencontre ou non d'autres points de contours.

C. Transformée de Hough

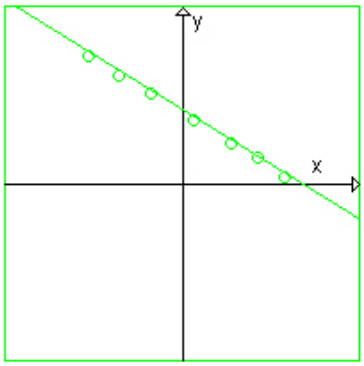
La transformée de Hough fait partie des outils standards dans le domaine de la vision artificielle et permet la détection de droites, mais nous le verrons également plus loin, d'autres formes telles que les courbes, les cercles ellipses voir d'autres d'objets plus complexes. L'approche de cette technique est de chercher à accumuler à l'intérieur d'un espace de paramètres représentatif, des données confirmant la présence de formes particulières.

Le principe de cette transformée est de dire qu'il existe un nombre infini de lignes qui passent par un point, dont la seule différence est l'orientation (l'angle θ). Le but est alors de déterminer quelles sont les lignes qui passent au plus près du schéma attendu.

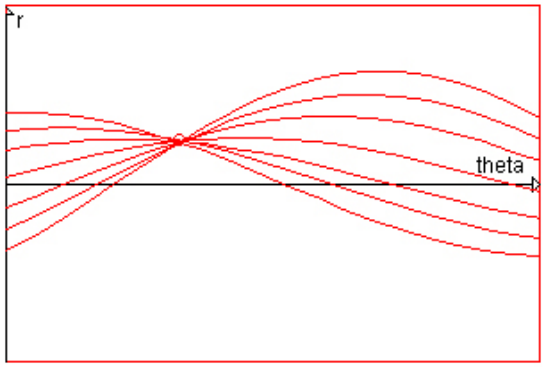
Une droite peut être définie de deux façons.

- Selon ses coordonnées cartésiennes : $y = ax + b$
- Mais aussi à partir de ses coordonnées polaires : $r = xcos(\theta) + ysin(\theta)$

On obtient une sinusoïde appelée « espace de Hough » lorsque l'on transforme toutes les lignes possibles qui relient un point à un autre. C'est-à-dire en calculant la valeur de r pour chaque angle. Si les courbes relatives à deux points de coupent, alors l'endroit d'intersection dans l'espace de Hough correspond aux paramètres d'une droite qui relie ces deux points. Il est alors simple de détecter des droites dans l'image en recherchant l'intersection dans l'espace de paramètres.



Espace de l'image en (x,y)



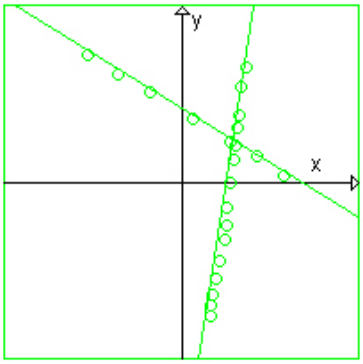
Espace de Hough = Espace de paramètres (r, θ)

i. Algorithme

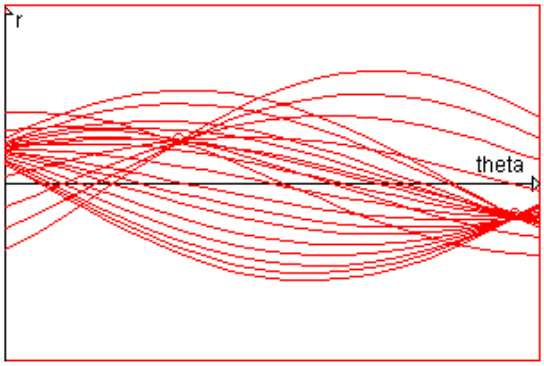
L'algorithme de Hough consiste alors pour chaque point de l'image et pour chaque valeur de θ à calculer sa norme de vecteur r et à pondérer les couples (r, θ) en fonction du nombre de fois que le couple est trouvé dans l'image. En d'autres termes, à chaque fois que l'on rencontre deux couples (r, θ) identiques dans l'image, on incrémente le poids du couple. Le couple bénéficiant du poids le plus fort correspondent alors à une droite détectée sur l'image.

ii. Détection de plusieurs droites

L'intérêt la plus part du temps résidant dans la recherche de plusieurs droites au sein d'une image donnée, l'algorithme de Hough s'applique à la recherche de plusieurs droites avec quelques modifications. L'idée est alors de ne plus considérer le maximum dans l'espace de Hough, mais d'analyser les différents maxima que l'on peut trouver. La difficulté réside alors dans la détection de ces points sans pour autant détecter deux fois la même droite. La solution est alors de parcourir l'espace des paramètres en regardant zone par zone selon une fenêtre d'exclusion.



Espace de l'image en (x,y)



Espace de Hough = Espace de paramètres (r, θ)

Il est alors simple de prendre une image à laquelle on va appliquer une détection de contour, comme par exemple un filtre de Canny. On peut alors ensuite rechercher sur celle-ci les droites à l'aide de la transformée de Hough.

iii. Transformée généralisée de Hough

La transformée généralisée de Hough fonctionne sur le même principe que la transformée de Hough. On recherche la présence d'une courbe, caractérisée par un certain nombre de paramètres. Chaque point de l'image analysée pondère l'ensemble des blocs de paramètres générant des courbes auxquelles il appartient. Et de la même façon que précédemment, les blocs de paramètres ayant le poids le plus fort correspondent aux courbes détectées. On peut alors appliquer cette transformée à de la recherche de cercles par exemple.

6. Ouverture sur la programmation

Il existe de nombreux outils de traitement d'images. Tout d'abord, on trouve des outils classiques de visualisation et de traitement d'images tels que Photoshop, The Gimp ou ACDSee. Ils permettent de manière simple et visuelle de manipuler et éditer des images. Leur utilisation est essentiellement à but esthétique. Toutefois, ces logiciels utilisent en partie les fonctions, filtres et outils que nous avons vus précédemment. Ils rendent alors le traitement d'images plus abordable pour le grand public grâce à de nombreux outils visuels.

Il existe plusieurs boîtes à outils adaptées au traitement d'images scientifique, tels que Matlab, Scilab et Khoros. Ces derniers permettent de manipuler et d'éditer des images de A à Z offrant aux utilisateurs la possibilité de programmer eux-mêmes leurs filtres et différents traitement. Les possibilités sont infinies et sont adaptées aux besoins les plus poussés en termes de traitement d'image. Il est à noter que Matlab a été utilisé pour la réalisation des exemples de ce document.

Les performances de ces boîtes à outils restent inférieures à celles obtenues par un programme Java, C ou C++. En effet, il est également possible de créer sa propre bibliothèque de traitement d'images. Etant relativement proche de la machine et bas niveau, les performances de traitement d'images sont alors accrues. Pour les personnes ne

souhaitant pas réinventer la roue, il existe des bibliothèques de traitement d'images telles que l'API Java2D pour Java ou CImg et OpenCV pour le C/C++.

Remarque sur mon travail avec Matlab

J'ai utilisé Matlab tout le long de ce travail de recherche sur le traitement d'images. Ayant commencé avec Scilab je n'ai pas réussi à mettre en œuvre la boîte d'outils de traitement d'images sur celui-ci. Par conséquent, je suis passé à Matlab. Il m'a permis de tester tous les algorithmes et filtres de détection de contours que j'ai évoqués jusqu'à présent. Cela m'a pris beaucoup de temps de comprendre et maitriser le logiciel et ses outils, mais j'ai pu bien pu comprendre leur fonctionnement. Toutefois, à certains moments, il m'a été impossible d'utiliser pleinement les outils de traitement d'images de Matlab. Certains scripts, bien que venant du site www.mathworks.com , ne daignaient pas fonctionner sur mon ordinateur. J'ai par conséquent, faute de temps, été obligé d'utiliser des images trouvées sur Internet pour illustrer mes propos. Je tenais juste à préciser ce point pour indiquer qu'il ne s'agit en aucun cas d'un raccourci de ma part, mais une conséquence de soucis techniques sur mon ordinateur. Avec un peu plus de temps, j'aurais sans doute pu réparer la boîte d'outils de Matlab afin d'utiliser mes propres images en illustrations.

Conclusion

Ce travail de recherche m'a permis de vraiment découvrir le domaine du traitement d'images et tout ce qui peut graviter autour. Je n'avais au départ aucunes connaissances relatives à ce milieu, et je suis très satisfait d'avoir pu apprendre les bases ainsi que faire un tour des techniques de base. J'ai pu notamment bien prendre connaissance de ce qui m'intéressait pour mon stage, à savoir la détection de contours. Toutefois, il ne s'agit là que d'une partie infime de ce que l'on peut faire en traitement d'images. En effet les possibilités offertes sont vraiment très nombreuses, et ce domaine est au centre de toutes les attentions notamment en recherche. Je suis donc très loin d'avoir vu la totalité de ce qui peut se faire, cependant cela me donne un aperçu intéressant et n'a fait qu'attiser encore plus ma curiosité et mon intérêt pour ce domaine.

Bibliographie

[1] Diane Lingrand. *Introduction au traitement d'images*, 2e édition. Vuibert, Paris, 2008

[2] E. Angelini, I. Bloch, Y. Gousseau, S. Ladjal, H. Maître, D. Matignon, B. Pesquet-Popescu, F. Schmitt, M. Sigelle, F. Tupin. *Analyse des images - Polycopié du cours SI241 - Analyse des Images*. Ecole Nationale Supérieure des Télécommunications, 2008

[3] Humbert Florent. *Introduction au traitement numérique des images*. 27 mai 2007

[4] Humbert Florent. *La transformée de Fourier en traitement d'images*. 1er avril 2007

[5] E. Angelini, I. Bloch, Y. Gousseau, S. Ladjal, B. Pesquet-Popescu, M. Sigelle, F. Tupin, *Méthodes Avancées de Traitement d'Images - Polycopié du cours SI343 – MATIM*. Ecole Nationale Supérieure des Télécommunications. 22 février 2007

[6] Isabelle BLOCH, Henri MAÎTRE, Florence TUPIN. UE SI344 - *Vision artificielle et raisonnement dans les images - Tome 1 : Vision artificielle*. Ecole Nationale Supérieure des Télécommunications. Décembre 2006

[7] Radu HORAUD et Olivier MONGA. *Vision par ordinateur : outils fondamentaux*. Deuxième édition. Editions Hermès

[8] *Cours de traitement d'images - Corrigé Labo 1: Familiarisation avec Matlab, Manipulations de base sur les niveaux de gris, Images calculées, Introduction a la Transformée de Fourier*

[9] *Introduction à Matlab*. Groupe Vision, CUI, Université de Genève. Février 2002

[10] Gonzalez Woods & Eddins. *Digital Image Processing Using Matlab*

[11] *Illustrations des cours de traitement des images (MTI, MBAI, ANIM) et de morphologie mathématique (I3M)* - http://www.tsi.enst.fr/tsi/enseignement/ressources/mti

[12] Rachid DERICHE. *Techniques d'extraction de contours*. INRIA Sophia-Antipolis

[13] http://wikipedia.org

[14] http://www.tsi.enst.fr

[15] http://www.irit.fr/ACTIVITES/EQ_TCI/EQUIPE/dalle/IRR/

[16] http://www.tsi.enst.fr/tsi/enseignement/ressources/mti/ellipses/appletHough.htmlb

[17] http://merciber.free.fr/these/these1.htm

[18] http://www.tsi.enst.fr/tsi/enseignement/ressources/mti/Ohlander/images2.html

[19] http://www.commentcamarche.net/contents/video/traitimg.php3

[20] http://gr6.u-strasbg.fr/~ronse/TIDOC/FEAT/seuil.html

[21] http://www.tsi.enst.fr/tsi/enseignement/ressources/mti/hyste-dyn/node3.html