الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
Université Akli Mohand Oulhadj - Bouira -
ⵜⴰⵙⴷⴰⵡⵉⵜ ⴰⴽⵍⵉ ⵎⵓⵃⴻⵏⴷ ⵓⵍⵃⴰⵊ - ⵜⵓⴱⵉⵔⴻⵜ -

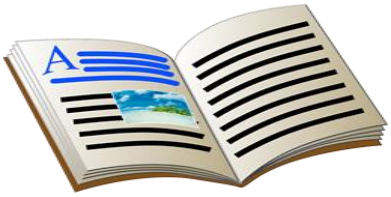
وزارة التعليم العالي والبحث العلمي
جامعة أكلي محند أولحاج
- البويرة -

Faculté des Sciences et des Sciences Appliquées

كلية العلوم والعلوم التطبيقية

Département de Génie Électrique

Polycopié de cours/TP



En : Télécommunication

Spécialité : Systèmes des Télécommunications

Niveau : Master

TRAITEMENT D'IMAGES

R. Kasmi

2017

Sommaire

Liste des figures

Chapitre 1 : Perception de la couleur

1.1 Colorimétrie

La colorimétrie est la discipline qui permet de déterminer les propriétés d'un objet par rapport à sa couleur. Plus particulièrement, c'est la science qui permet de quantifier ou de mesurer la couleur d'un objet.

1.2 La perception de la lumière et couleur

La lumière atteint les cellules de la rétine (bâtonnets et les cônes), un flux nerveux est transmis au cerveau via les nerfs optiques. Le cerveau à son tour interprète le flux nerveux à une image.

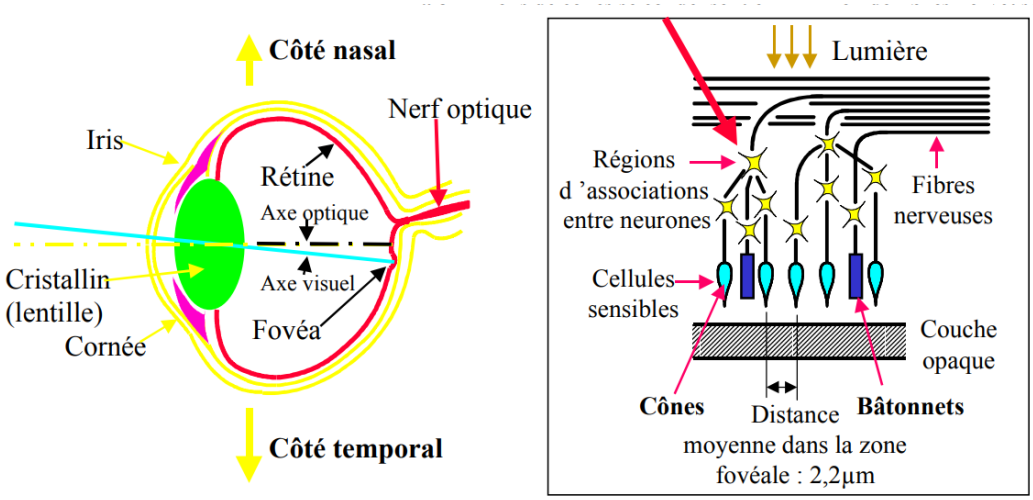


Figure 1.1 coupe horizontale de la rétine. (b): schéma simplifié

La lumière blanche est un mélange de couleurs. Chaque couleur correspond une grandeur physique appelée longueur d'onde. L'œil humain n'est sensible qu'aux radiations dont les longueurs d'onde sont comprises entre 400 nm et 800 nm.

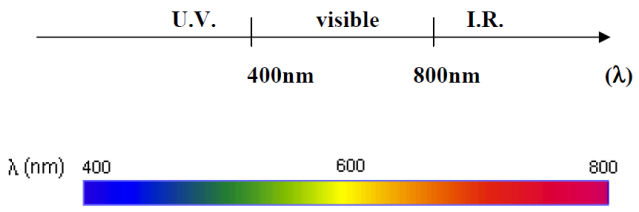


Figure 1.2: Spectre de la couleur visible

1.3 Représentation d'une image 2D

Une image est une représentation graphique d'un ou de plusieurs objets. Une image numérique est une matrice bidimensionnelle constituée d'un nombre fini d'éléments appelés pixels. Chaque pixel est caractérisé par un niveau de gris.

1.3.1 Différents types d'images

Selon le codage des pixels, trois types d'images peuvent être obtenues, image binaire, aux niveaux de gris et couleur, codés respectivement sur 1 bit, n bits (n >1) et 3X n bit (n>1).

a) Image binaire

Pour ce type d'image, chaque pixel est codé sur 1 bit. Ce qui fait que les pixels sont représentés par deux états logiques 0 (noir) et 1 (blanc)

$N = 2^1$ $(N: nombre\ de\ niveaux)$

pixel =0 ou pixel =1

L'image apparaît noir et blanc (1 bit par pixel)

Figure 1.3: Image binaire.

b) Image aux niveaux de gris

Chaque pixel est codé sur n bits, ce qui confère aux pixels des valeurs entières comprises entre 0 et 2^n. Une image dont les pixels sont codés sur 8 bit à 256 niveaux de gris

Exemple:

- n=2 bits par pixel==> nombre de niveaux N = $2^2 = 4$

- n=8 bits par pixel ==> nombre de niveaux est $N = 2^8 = 255$

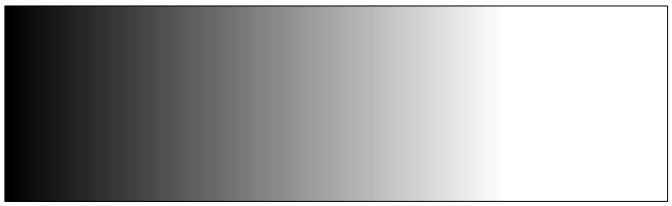


Figure 1.4: Image à 256 niveaux de gris.

- 24 bits par pixel ==> nombre de niveaux est $N = 2^{24} = 16.7$ millions

Remarque: L'œil humain ne peut discerner 300 000 couleurs différentes et environ une trentaine de niveaux de gris.

c) Image couleur

Une image en couleur est constituée de trois plans, dans le cas du système couleur RVB, : un plan rouge (R), plan vert (V), plan bleu (B).

Les trois couleurs R, V, B sont appelées ***couleurs primaires***. Les couleurs que nous percevons sont une combinaison des trois couleurs primaires, appelé ***synthèse additive***.

Une couleur **C** peut être obtenue par combinaison linéaire suivante:

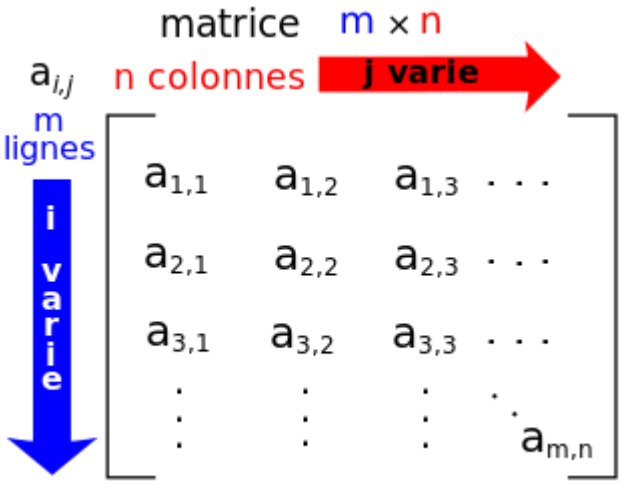
$$C ==> rR+vV+bB \qquad (1.1)$$

r, v et b : constantes qui contrôlent les intensités des couleurs R, V et B respectivement.

Le codage de la couleur d'une image est réalisé sur trois octets, chaque octet représente la valeur d'une composante couleur par un entier compris entre 0 et 255 (codage: 8 bits).

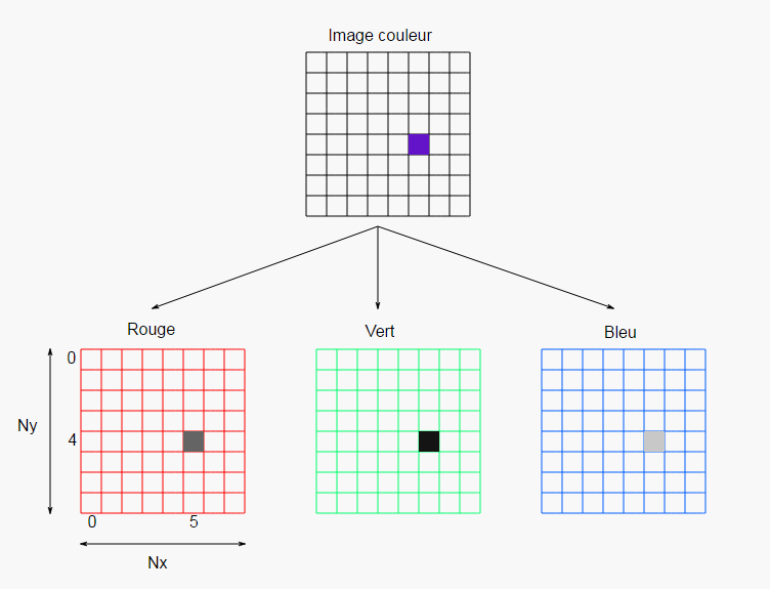
R	V	B	Couleur
0	0	0	noir
255	0	0	rouge
0	255	0	vert
0	0	255	bleu
128	128	128	gris
255	255	255	blanc

Par convention, on notera en premier l'indice qui repère les lignes et le deuxième les colonnes, conformément au repérage des coordonnées d'un point sur un plan (x, y). On remarque néanmoins que l'origine se trouve sur le coin supérieur gauche de l'image.



Une image couleur de Ny lignes et Nx colonnes est représenté comme suit:

Exemple:



On voit sur la figure le pixel (5,4) dont les niveaux de gris des trois couches sont (100,20,200), ce qui donne une couleur violette.

1.4 Systèmes de représentation de la couleur

Une couleur peut être représenté sur plusieurs **espaces couleurs** ou systèmes de couleurs tel que RVB, XYZ, YUV, HSV, YIQ, l*a*,b.

chaque modèle est adapté à une situation:

- affichage sur un écran
- impression
- algorithmes de traitement d'images
- …

On peut convertir une image couleur d'un espace de couleur à un autre espace couleur.
Exemple: la conversion RVB à HSV (Teinte Saturation Valeur) se fait comme suit

$$H = \cos^{-1} \frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}} \quad (1.2)$$

$$S = 1-\frac{3}{(R+G+B)}[\ MIN(R,B,G)\ ] \quad (1.3)$$

$$V = \frac{1}{3}(R+G+B) \quad (1.4)$$

Chapitre 2: Capteurs d'images et dispositifs d'acquisition numériques

2.1. Introduction

Le traitement d'images est un domaine très vaste qui a connu, et qui connaît encore, un développement important depuis quelques dizaines d'années. On désigne par traitement d'images numériques l'ensemble des techniques permettant de modifier une image numérique afin d'améliorer ou d'en extraire des informations.

2.2. Schéma de principe d'une chaîne de traitement d'images.

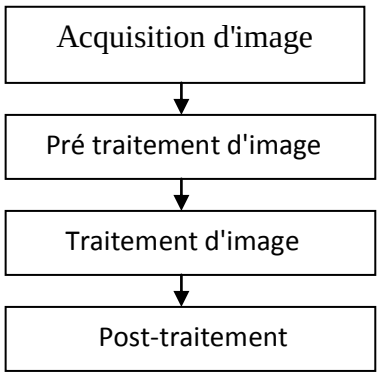


Figure 2.1 Schéma général du traitement d'image.

- **Acquisition d'image** : Deux catégories , caméras numériques (CCD, CMOS) et scanners
- **Prétraitement** : ensemble des techniques qui améliorent l'image ou met en valeur un objet à détecter.
- **Traitement d'image** : ensemble des techniques qui permettent d'extraire des informations de l'image.
- **Poste traitement** : ensemble des techniques qui permettent d'améliorer le résultat de l'étape précédente (traitement d'image)

2.3. Acquisition d'images (capteurs CCD et CMOS)

Les capteurs d'images permettent de transformer les photons en signal électrique.
Dans le domaine de la photographie numérique, les deux grandes technologies les plus utilisées sont: **CCD** (Charge Coupled Devices) et **CMOS (**Complementary Metal Oxide Semiconductor).

Les deux capteurs sont constitués de cellules photosensibles capables de transformer la lumière en un courant d'intensité variable (effet photoélectrique). Les charges (électrons) collectées sont stockées dans un puits de potentiel créé par une électrode. La charge est proportionnelle à l'intensité de lumière reçue.

La différence principale entre ces deux technologies est le processus de transfert de l'information (charge) des cellules de capteurs vers la processus de l'appareil.

- CMOS: identifie la valeur de chaque cellule séparément,
- CCD : travaille sur l'intensité d'une ligne/colonne de cellules directement.

a) Principe de fonctionnement d'un capteur CCD

Chaque cellule du capteur correspond à un Photosite (figure 2.2)

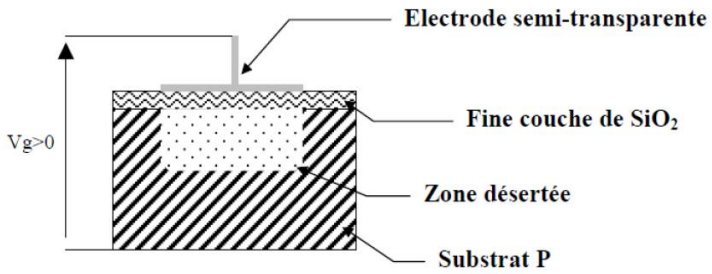


Figure 2.2: Photosite

Une fois les charges collectées dans les zones désertées des Photosites, les charges sont transférées de Photosite en Photosite par le jeu de variation de potentiel cyclique contrôlé par une horloge externe. Puis, les charges sont envoyées dans un registre de sortie. Enfin, les charges sont transformées en tension. Ce signal sera, à l'extérieur du CCD, amplifié et numérisé.

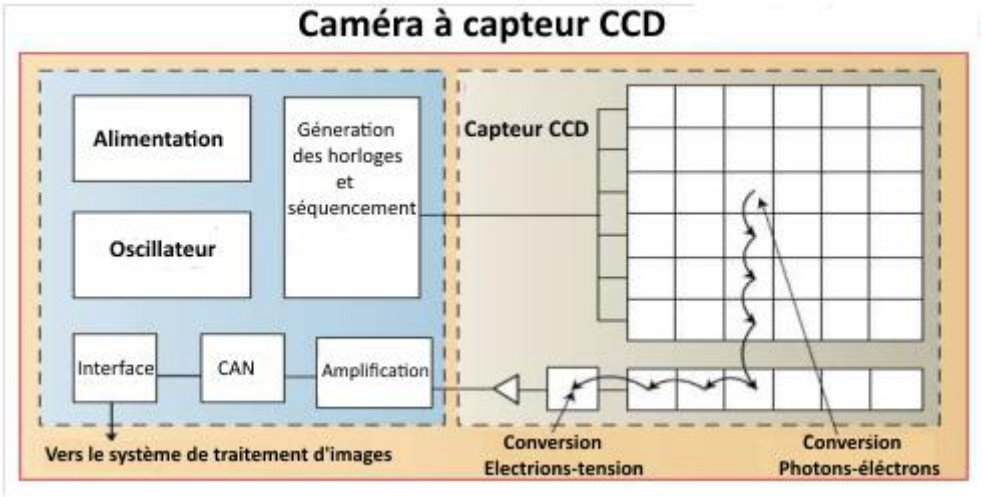


Figure 2.3: Le fonctionnement général d'une caméra à capteur CCD.

Il existe 3 types de capteurs CCD

- ***Le CCD "plein cadre " (full frame)***: l'ensemble de la surface contribue à la détection

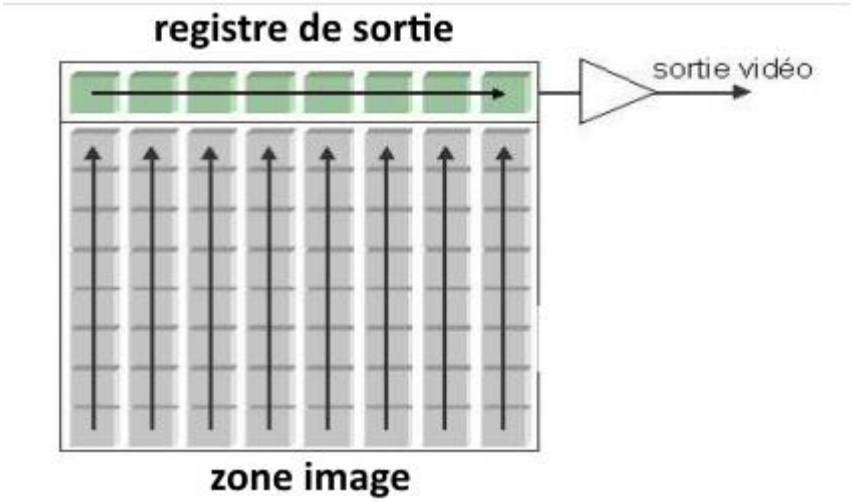


Figure 2.4: Schéma d'un capteur CCD plein cadre

Transport des charges avec un capteur à **transfert parallèle-série**

1- Après le temps d'intégration, l'obturateur est fermé et les charges sont transférées dans le registre horizontal.

2-Les charges sont évacuées en série

- **Le CCD « à transfert de trame » (*full-frame transfer*)** : il associe deux matrices CCD de même dimension, l'une exposée à la lumière, l'autre masquée.

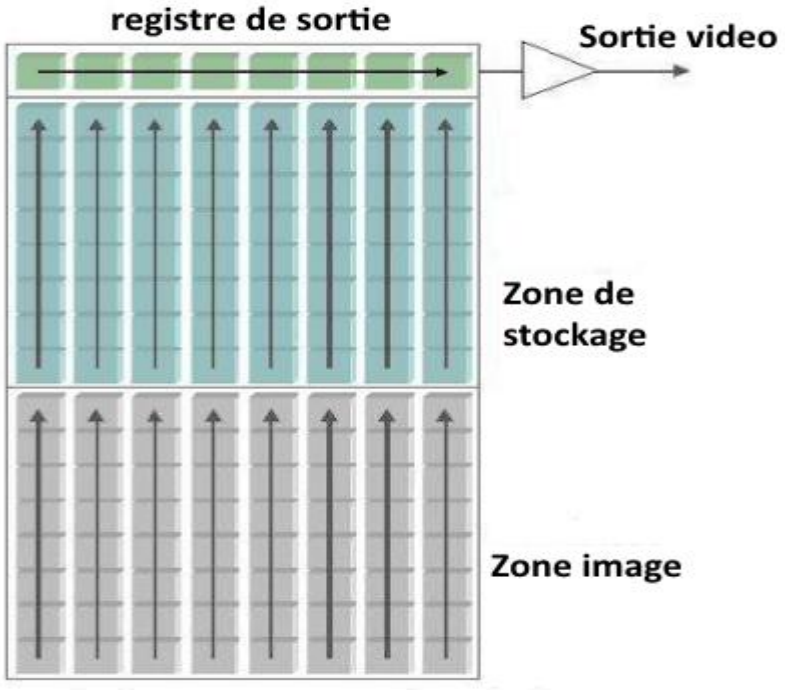


Figure 2.3: Schéma d'un capteur CCD à transfert de trame.

Transport des charges avec un capteur à **transfert de trame**

1- les charges sont transférées dans la surface de stockage

2- puis transférées vers le registre de sortie

3- Enfin, évacuées en série

- ***Le CCD « interligne »*** : plus complexe ; il associe un photosite à chaque cellule CCD.

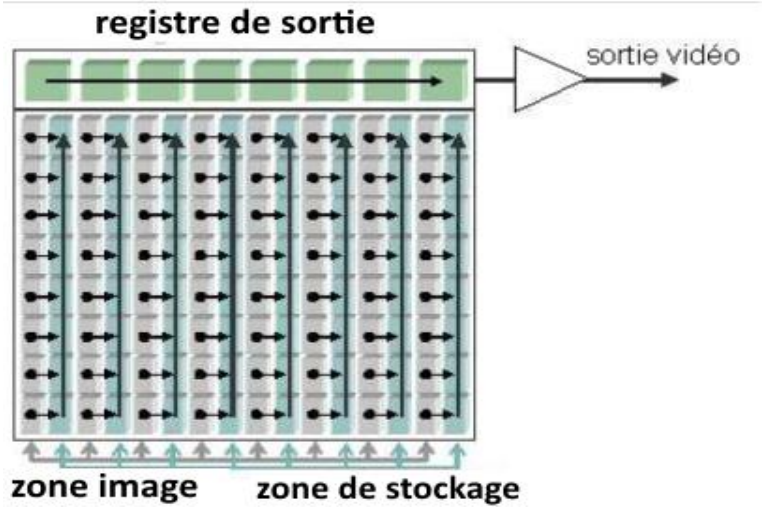


Figure 2.4: Schéma d'un capteur CCD interligne.

Transport des charges avec un capteur à **transfert interligne**

1- les charges sont transférées dans les colonnes de stockage
2- puis transférées vers le registre de sortie
3- Enfin, évacuée en série

Points positifs:

- Qualité d'images élevée: qualité d'image élevée,
- Bruit très faible : Le bruit est faible avec un capteur CCD (moins de composants électroniques dans capteur)
- Haute sensibilité : cela permet l'emploi de ces capteurs où il y a peu de lumière.

Points négatifs:

- Saturation du capteur aux fortes luminosités: Cela peut créer des taches circulaires blanches, appelées blooming..
- Nécessité d'horloges multiples pour piloter les transferts de charges

b) Principe de fonctionnement d'un capteur CMOS

Les capteurs CMOS (Complementary Metal Oxide Semiconductor) ou en français (Semi-conducteur à Oxyde de Métal Complémentaire) fonctionnent différemment, même si le principe de détection de charge reste le même

La différence avec le CCD, est que à l'intérieur de chaque photosite, la charge générée est convertie directement en tension utilisable. Suivant le schéma ci-après:

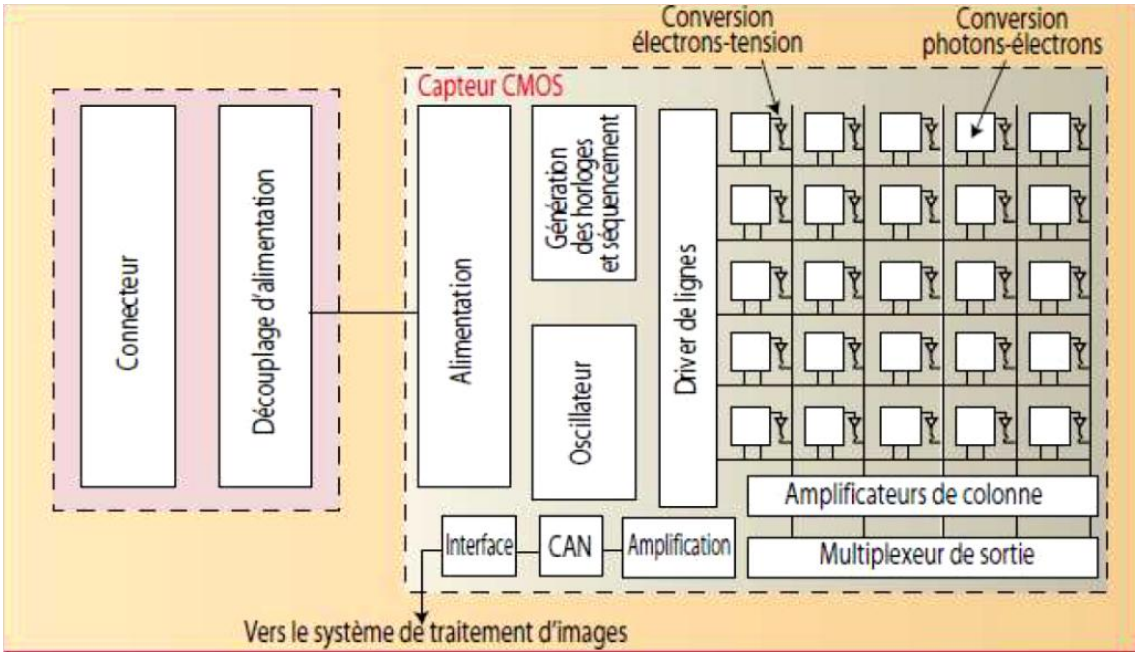


Figure 2.5: Principe de fonctionnement d'un capteur CMOS.

Points positifs:

- Fenêtrage: possibilité d'adresser individuellement les pixels et de ne lire que certaines zones de la matrice.
- Compacité: L'électronique et le capteur sont présents sur le même composant.
- Faible consommation.
- Grandes vitesses d'acquisition.

Points négatifs:

- Sensible aux bruits
- Facteur de remplissage modéré

2.4. La résolution

C'est le nombre de points contenu dans une longueur donnée (en pouce). Elle est exprimée en points par pouce (PPP, en anglais: DPI pour Dots Per Inch). Un pouce mesure 2.54 cm.

La résolution permet ainsi d'établir le rapport entre la définition en pixels d'une image et la dimension réelle de sa représentation sur un support physique (affichage écran, impression papier...)

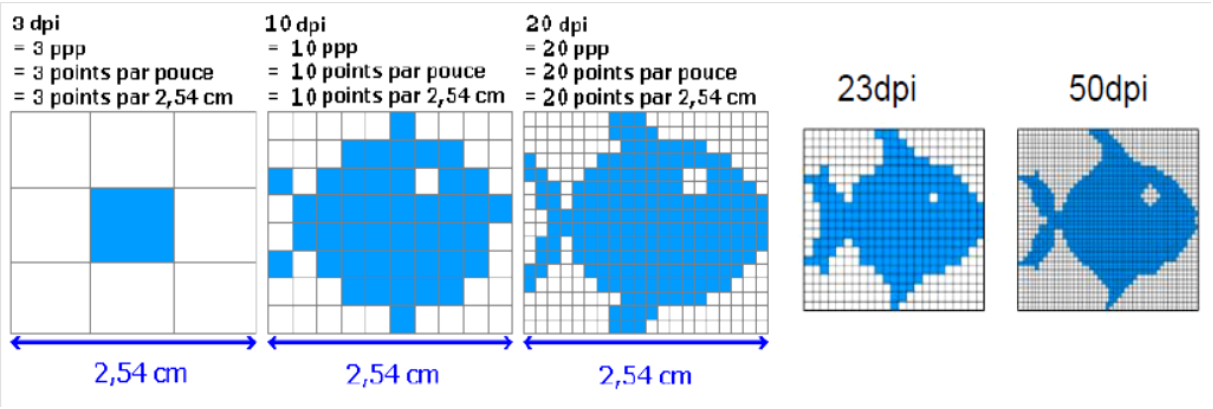


Figure 2.6: Résolution d'image.

Résolution = définition (nombre de pixels) / dimension (largeur)

Ex: la résolution d'une image de 300 pixels de large mesurant 2 pouces de coté :
Résolution = 300 / 2 = 150dpi

Formules :
Résolution = définition / dimension

Exercice:
Quelle serait la définition en pixel d'une feuille de 8,5 pouces de largeur et 11 pouces en hauteurs cannée à 300dpi?

Réponse:
300 X 8,5 = 2550 pixels
300 X 11 = 3300 pixels
La définition de l'image serait donc de 2550 X 3300 pixels

Formule: Calculer le poids d'une image en octet
Nombre de pixel total X codage couleurs (octet) = Poids (octet)

Exemple: quel est le poids d'une image d'une définition de 640 x 480 codée sur 1 bit (noir et blanc)?

(640x480) x 1bit
307200 x (1/8) = 38400 octets
38400 / 1024 = 37,5 ko

2.5. Les formats de fichier bitmap

a) Principaux formats de fichiers non compressés

Ce sont les formats de fichiers dit « non destructifs ». Ils enregistrent chaque pixel d'une image comme nous l'avons vu précédemment (CCD, CMOS), les images bitmap occupent beaucoup de mémoire. De part leur poids élevé, ils ne sont pas adaptés pour le web mais doivent être utilisés lorsqu'on à besoin de préserver la totalité des informations d'une image pour retravailler dessus par exemple.

.PSD : Format natif de Photoshop, c'est un méta-fichier qui peut contenir du bitmap et du vectoriel (primitives géométriques). La couleur peut être codée sur 8, 16, 24 ou 32 bits, en Noir et Blanc, RVB et CMJN. Il gère la transparence, les couches alpha et peut prendre énormément de poids suivant le nombre de calques utilisés (chaque calque ajouté pèse !)

.BMP : Format natif de Windows, il permet d'enregistrer des images bitmap en 1, 4, 8 ou 24 bit en mode RVB. Il gère également les palettes pour les couleurs en mode indexées.

.TIFF : il permet de stocker des images de haute qualité en noir et blanc, couleurs RVB, CMJN jusqu'à 32 bits par pixels. Il supporte aussi les images indexées faisant usage d'une palette de couleurs, les calques et les couches alpha (transparence).

.RAW : C'est un format brut qui « code» les images avec un maximum d'information suivant le capteur de l'appareil qui l'a crée. Il permet ensuite de développer numériquement ses photos en les enregistrant en **.tiff** avec les réglages souhaitées (températures de couleurs, contrastes...).

b) Principaux formats de fichier compressés

Ce sont les formats de fichiers dit « destructifs ». Ils permettent, selon un algorithme particulier, de gagner plus ou moins de mémoire en supprimant certaines informations peu ou non perceptible par l'oeil humain. Ils sont particulièrement adaptés à internet, mais ne doivent pas être utilisés lors d'un travail de création sous Photoshop car chaque nouvel enregistrement

détériore un peu plus le fichier. On les utilisera donc pour exporter des images destinées à la visualisation sur internet ou l'archivage.

.JPG : Norme de compression pour les images fixes ; Elle donne la possibilité de sélectionner le taux de compression en fonction du niveau de restitution recherché (qualité réglables sur une échelle de 0 à 12). Elle supprime les informations redondante et les détails fins. Fonctionne en 8 bit/pixel en RVB ou CMJN.

.GIF : C'est un format léger qui peut également contenir des animations. Une image GIF ne peut contenir que 2, 4, 8, 16, 32, 64, 128 ou 256 couleurs parmi 16.8 millions dans sa palette en mode RVB. Elle supporte également une couleur de transparence.

.PNG : il permet de stocker des images en noir et blanc (jusqu'à 16 bits par pixels), en couleurs réelles (True color, jusqu'à 48 bits par pixels) ainsi que des images indexées, faisant usage d'une palette de 256 couleurs. Il offre enfin une couche alpha de 256 niveaux pour la transparence.

CHAPITRE 3: Traitements de bases sur l'image

3.1 Introduction:

Le traitement d'images numériques est un ensemble d'outils mathématiques opérant sur des images, dans le but d'améliorer leurs aspects visuels, d'extraire des caractéristiques ou de détecter des objets dans l'image.

3.2 Contraste:

Le contraste peut être défini de plusieurs façons :

a) Variance des niveaux de gris (N nombre de pixels dans l'image)

$$C = \frac{1}{N}\sum_{n=1}^{N}(Img(i,j) - Moy)^2$$

Img: Image aux niveaux de gris

Moy: moyenne des pixels de Img

b) Variation entre niveaux de gris max et min

$$C = \frac{\max(Img) - \min(Img)}{\max(Img) + \min(Img)}$$

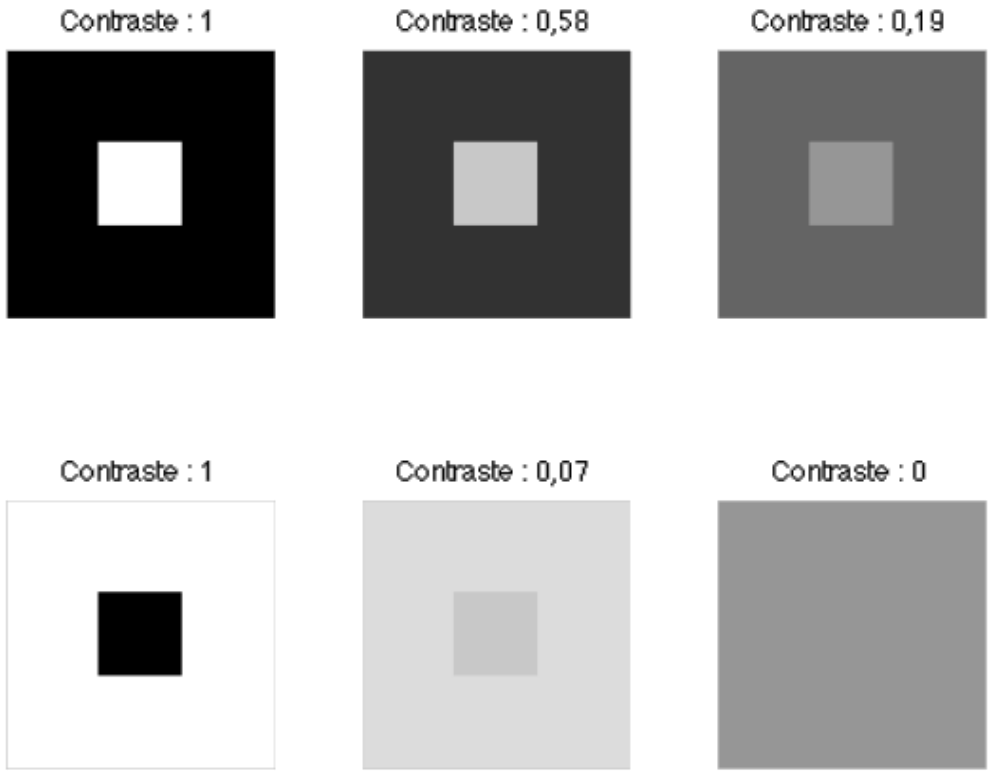


Figure 3.1: mesure de contraste

3.3 Luminance

La luminance (ou brillance) est un terme relatif. C'est une grandeur qui quantifie la sensation visuelle de la luminosité. Définie comme la moyenne des pixels de l'image.

$$L = \frac{1}{M \times N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} I(i,j)$$

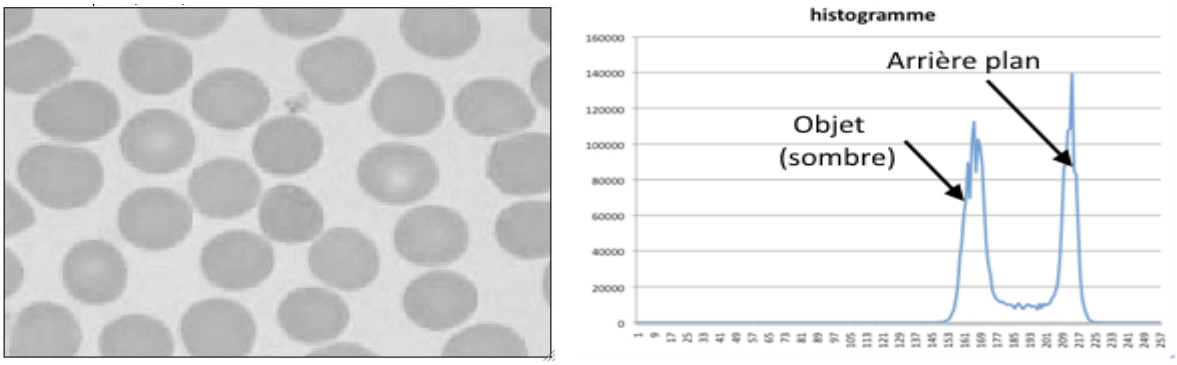
On peut transformer une image couleur a une image aux niveaux de gris:

$$IG = 0.2989\ R + 0.5870\ G + 0.1140\ B$$

IG: Image aux niveaux de gris (luminance).

3.4 Histogramme

Histogramme d'une image est défini comme une fonction discrète ***h(n)*** qui mesure le nombre de pixels pour chaque niveau de gris ***n.***



(a) **(b)**

Figure 3.2: (a) Image. (b) son Histogramme.

a) Algorithme générateur de l'histogramme

```
Pour chaque pixel  i=0 à N-1
   V=Image(i)
   Hist(V)= Hist(V)+1;
fin
```

b) Exemple de calcul d'histogramme:

4	3	2	3	3
2	2	3	1	4
1	4	4	1	2
1	2	2	2	2
2	2	3	3	2

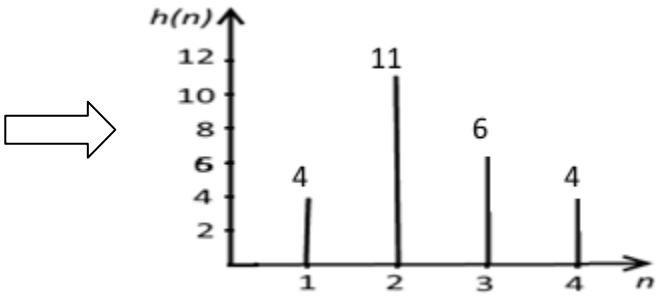
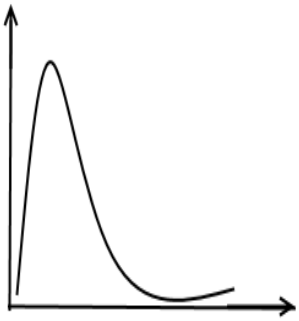
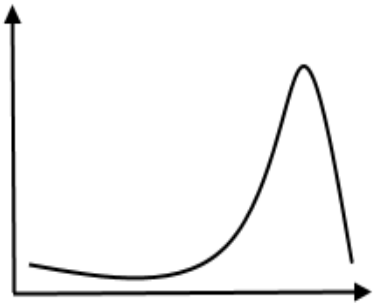
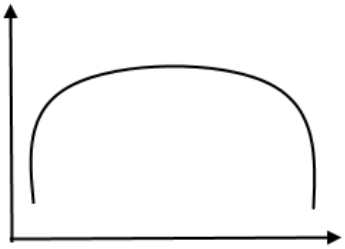


Image Histogramme

Figure 3.3: Calcul de l'histogramme.

c) Analyse de l'histogramme

		
La majorité des pixels ont de faible valeurs ==>Image sombre	La majorité des pixels ont des valeurs élevées ==>Image claire	*L'histogramme est constant ==> image à une luminosité homogène*

d) Histogramme cumulé

"Histogramme cumulé" désigne un graphique qui représente le nombre de pixels ayant une valeur en dessous d'un niveau de gris donné.

$$C(k) = \sum_{i=1}^{k} (h(i))$$

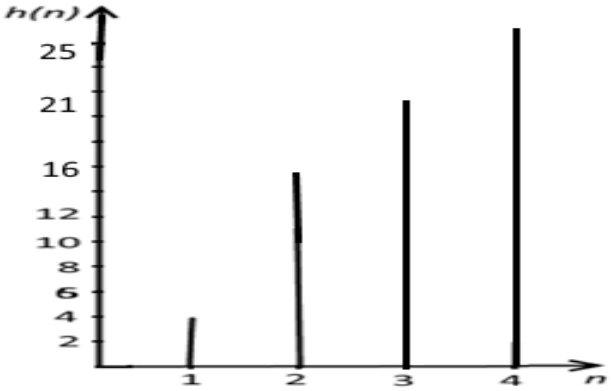


Figure 3.4: Histogramme cumulé.

Les niveaux 1, 2, 3 et 4 représentent respectivement 4, (4+11=15), (4+11+6=21), (4+11+6+4=25)

3.5 Correction de la dynamique de l'image par les transformations affines sur l'histogramme

a) Égalisation d'histogramme

On cherche à aplanir l'histogramme en modifiant les valeurs des pixels dans l'image de façon à obtenir une image avec une luminance uniforme.

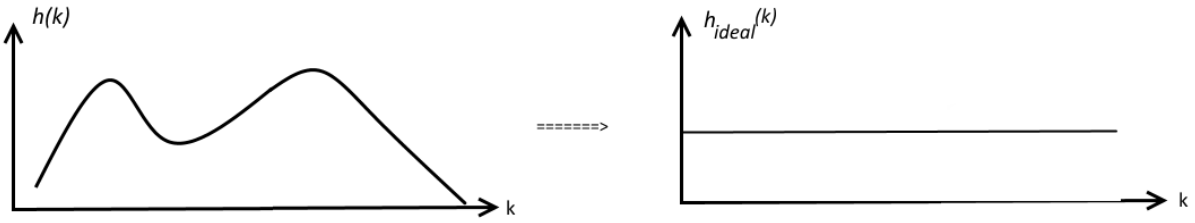


Figure 3.5: égalisation de l'histogramme.

1- calcule de l'histogramme h(k) avec $k \in [1\ 255]$

2-histogramme cumulé $C(k) = \sum_{i=1}^{k}(h(i))$

3- transformation des niveaux de gris de l'image

$$I'(x,y) = \frac{C(I(x,y))\times 255}{N}$$

b) Ajustement de l'histogramme

L'ajustement de l'histogramme a pour objet de modifier la dynamique de l'histogramme. C'est-à-dire une répartition des niveaux de gris sur un intervalle [a b] sera modifiée pour qu'elle se répartisse sur un autre intervalle [a2 b2].

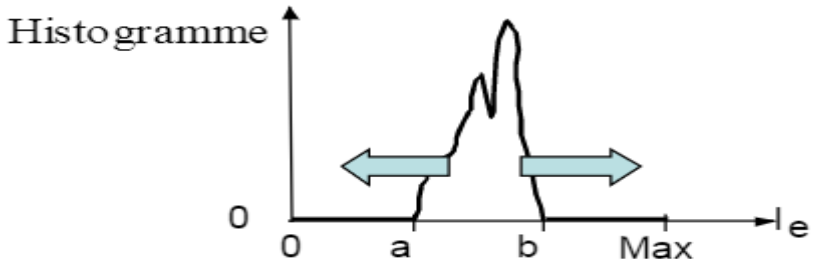
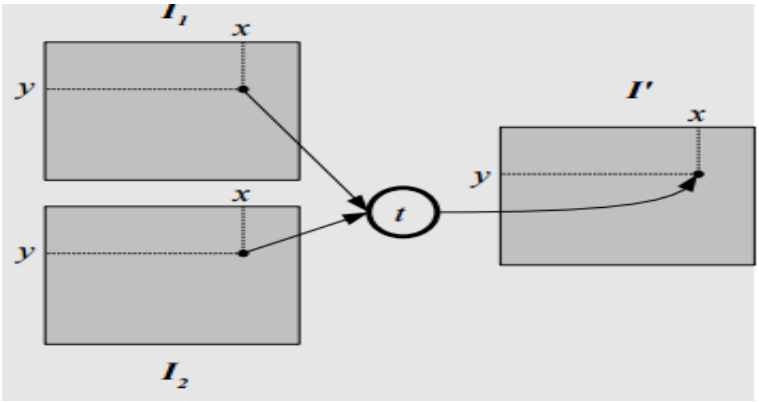


Figure 3.6: Ajustement de l'histogramme.

3.6 Opérations logiques et arithmétiques sur les images

Le principe est d'appliquer, pixel par pixel, des opérations logiques et arithmétiques classiques tel que addition, soustraction, ET logique, OU logique, à deux (ou plusieurs) images de même taille.



c) Addition

$$Image3(x,y) = Image1(x,y) + Image2(x,y)$$

Si un pixel résultant (Image3) dépasse 255 (on suppose que les images sont codées sur 8 bit), le phénomène de saturation apparait.

Saturation: si un pixel dépasse 255, il prend la valeur 255.

Utilisations:

- augmentation de la luminance d'une image (par addition d'une constante ou d'une image avec elle-même)
- diminution du bruit dans une série d'images

$$Image3(x,y) = MiINIMUM\ ((Image1(x,y) + Image2(x,y)),255)$$

d) Soustraction

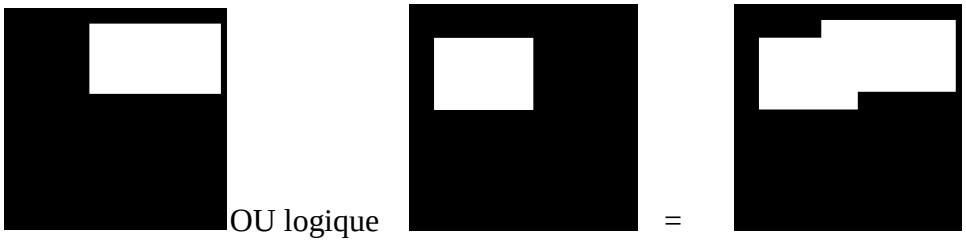
Utilisations:

- diminution de la luminance d'une image
- détection de changements entre images défauts (par comparaison avec une image de référence)

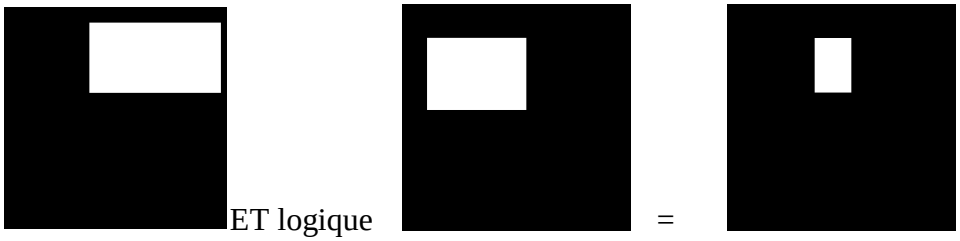
- mouvements (par comparaison avec une autre image de la séquence)

e) Opérations logiques

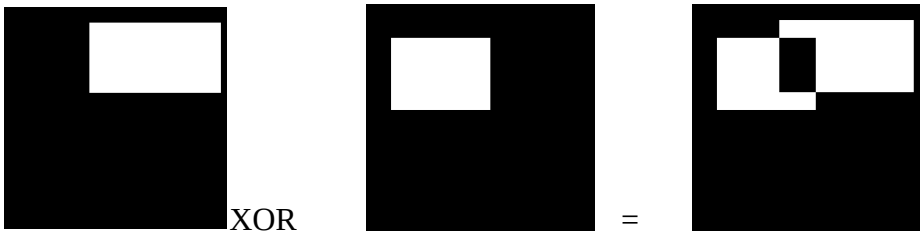
1. OU logique



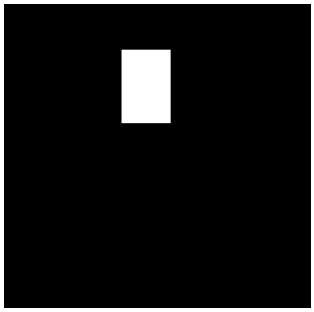
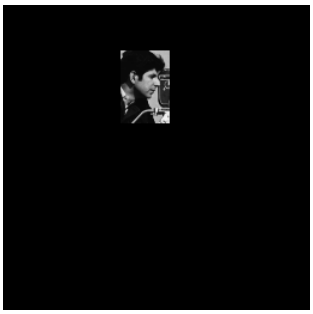
2. ET logique



3. XOR: exclusion logique



Avec les opérateurs logiques on peut créer des masques, qui peuvent être utilisés pour sélectionner ou supprimer une partie d'une image donnée.

 **ET Logique**  **=** 

CHAPITRE 4: Filtrage numérique des images

Dans ce chapitre, on étudiera les différentes techniques de filtrage d'images 2D. Le filtrage a pour objet soit d'améliorer la qualité du rendu visuel en atténuant ou en supprimant certaines dégradations (suppression de pixels bruit), soit de supprimer des structures ou des objets qui gênent une analyse donnée.

4.1 Définition du bruit

Le bruit est un signal parasite aléatoire qui peut suivre une distribution de probabilité connue ou non connue. Les sources du bruit sont diverses: capteur, acquisition, lumière, amplification, ...

Modèles simplifiés : 3 types de dégradations

- Bruit additif : $u = Img + b$
- Bruit multiplicatif : $u = Img \times b$
- Bruit convolutif : $u = Img * h$

Img: image original.

b : bruit.

a) Bruit impulsionnel (poivre et sel)

Il est obtenu en insérant n pixels blancs et n pixels noirs aléatoirement dans une image.

Figure 4.1: Ajout d'un bruit impulsionnel.

b) Bruit additif gaussien (bruit blanc gaussien)

Obtenu en ajoutant à une image des valeurs aléatoires indépendantes qui suivent une loi gaussienne :

$$G_{\sigma,\mu}(x) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{\frac{(x-\mu)^2}{2\sigma^2}}$$

L'analyse fréquentielle d'un bruit gaussien est une constante d'où le nom bruit blanc.

Figure 4.2: Ajout d'un bruit gaussien.

c) Bruit convolutif ou Modèle de flou

L'image observée est dégradée par un filtre convolutif (en général passe-bas) :

$$ImFloué = ImOriginal * h$$

$*$:convolution

Figure 4.3: Ajout d'un bruit convolutif.

4.2 Filtrage spatial linéaire

a) Produit de convolution

Le filtrage est réalisé par une convolution entre l'image ***f*** bruitée et un masque ***h.***

Le produit de convolution est définie comme:

$$f'(i,j) = (f * h) = \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} \sum_{m=\frac{d-1}{2}}^{\frac{d-1}{2}} f(i,j)h(n-i, m-j))$$

h : masque carré de taille ***d*** impair:

f': image filtrée.

Le principe de calcul de convolution

1- Faire une rotation du noyau (marque h) de Pi par rapport a son centre

2-Centrer le masque sur un pixel donné de l'image

3-Sommer les produits entre les pixels d'image et éléments du masque.

4-Remplacer la valeur du pixel centré par la valeur trouvée par la convolution.

Exemple d=3

$$h = \begin{pmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{pmatrix}$$

La convolution au pixel (i, j) de f par le noyau h est calculée :

$$f'(i,j) = w_1 f(i-1, j-1) + w_2 f(i-1, j) + w_3 f(i-1, j+1) \quad +$$
$$w_4 f(i, j-1) + w_5 f(i, j) + w_6 f(i, j+1) + \quad w_8 f(i+1, j+1) +$$
$$w_8 f(i+1, j) + w_9 f(i+1, j+1) \qquad (4.1)$$

b) Filtre (Lissage) moyenneur

La valeur d'un pixel est relativement similaire à celle de ses voisins, de ce fait, une image bruitée peut être filtrée en atténuant le bruit par un ***moyennage local***, cette opération est appelée lissage (smoothing)

le lissage est fait comme:

$$f'(i,j) = \frac{1}{d^2} \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} \sum_{m=\frac{d-1}{2}}^{\frac{d-1}{2}} f(i+n, j+m)$$

Si un filtre (masque ***h***) de taille ***d***, dont tous les éléments ont comme valeurs $w_i = \frac{1}{d^2}$, le résultat de la convolution est un **lissage moyenneur.**

Exemple du litre moyenner **d=3**

$$h = \frac{1}{3^2} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

(a) (b) (c)

Figure 4.4: Filtrage moyenneur: (a) Image original.(b) Masque 5X5 (b) Masque 11X11.

On remarque que plus ***d*** est grand, plus le lissage est important, et plus l'image filtrée perd les détails de l'image originale.

Les effets du filtre moyenner:

==> Permet de lisser l'image *(smoothing)*

==> Remplace chaque pixel par la valeur moyenne de ses voisins

==> Réduit le bruit

==> Réduit les détails non-désirés

==> Brouille ou rend flou les contours de l'image *(blur edges)*

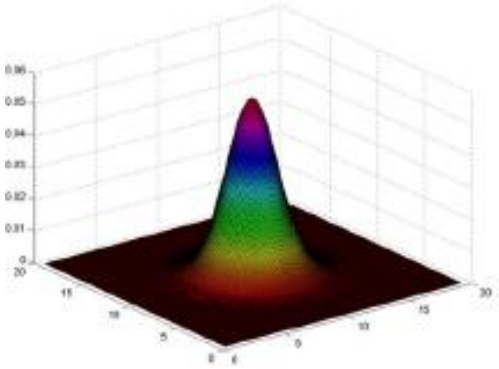
c) Lissage gaussien

Le masque suit une loi de distribution gaussienne, noyau centré et d'écart-type σ

$$G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{i^2+j^2}{2\sigma^2}}$$

Exemple:

$$\frac{1}{98}\times\begin{bmatrix}1 & 2 & 3 & 2 & 1\\ 2 & 6 & 8 & 6 & 2\\ 3 & 8 & 10 & 8 & 3\\ 2 & 6 & 8 & 6 & 2\\ 1 & 2 & 3 & 2 & 1\end{bmatrix}$$



- Si σ est plus petit qu'un pixel==> le lissage n'a presque pas d'effet
- Si σ est grand==> plus on réduit le bruit, mais plus l'image filtrée est flouée
- Si σ est choisi trop grand ==> tous les détails de l'image seront perdus

Les paramètres d'un masque doivent être choisis de sorte à avoir un compromis entre le taux de bruit à enlever et la qualité de l'image en sortie.

(a) (b) (c)

Figure 4.5: Filtrage gaussien: (a) Image original.(b) Masque 5X5 (b). Masque 11X11.

4.3 Filtre médian (non linéaire)

*** La médian:** Soit une séquence discrète a1; a2; : : : ; aN (N impair). a_i est la valeur médiane de la séquence si :

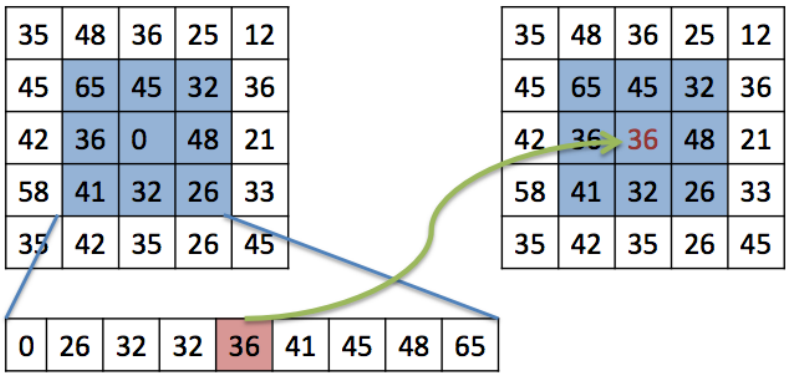
Il existe (N-1)/2 éléments de valeur inférieure

Il existe (N-1)/2 éléments de valeur supérieure

***Filtre médian**

1- Déplacer une fenêtre de taille impaire sur l' image

2- Remplacer le pixel central (sur lequel est positionnée la fenêtre) par la valeur médiane des pixels inclus dans la fenêtre



Les effets du filtre moyenneur:

==> Très adapté au bruit type "poivre et sel" (faux "blanc" et "noir" dans l'image)

==> Préserve les contours

==> Réduit le bruit additif uniforme ou gaussien (lissage de l'image)

==> Si le bruit est supérieur à la moitié de la taille du filtre, alors le filtre est inefficace

(a) (b)

Figure 4.6: Filtrage médian: (a) bruit impulsionnel (b) résultat du filtrage.

4.4 Filtrage fréquentiel

a) Notion de fréquence

Dans un signal 1D la fréquence est calculé : F=1/T (T: période). Un signal de haut et basse fréquence sont comme le montre la figure suivante:

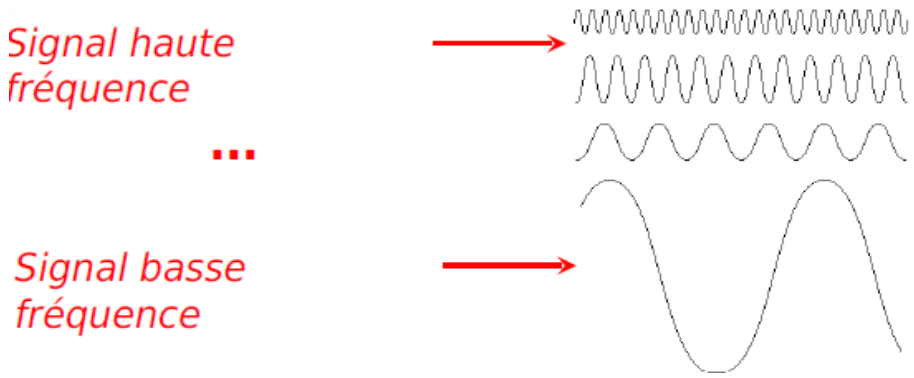
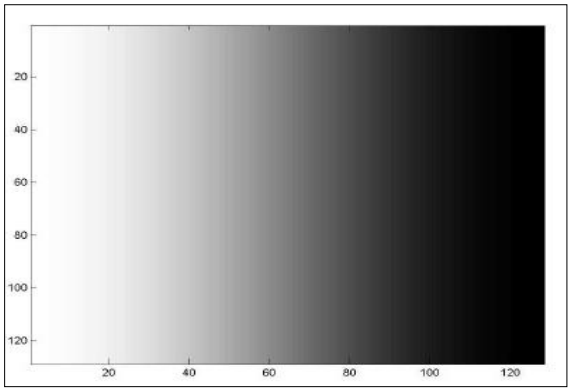


Image numérique

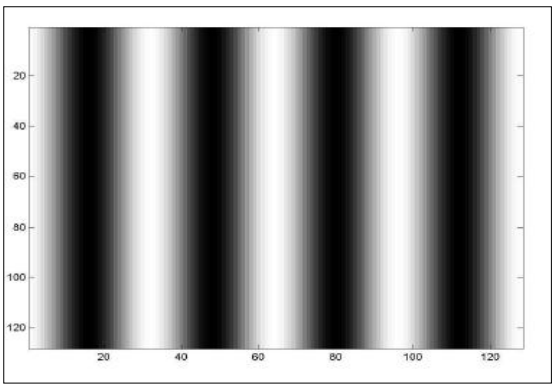
Une image est un signal bidimensionnel. Elle contient un nombre fini de points appelés pixels. Une image est représentée par une fonction mathématique g(m,n) avec m et n les coordonnées horizontale et verticale respectivement d'un pixel donné. Chaque pixel est associé à une valeur appelée niveau de gris

Dans l'image, la notion de fréquence correspond aux changements de l'intensité. On parle de ***fréquence spatiale*** (image) au lieu de fréquence temporelle.

- **Basse fréquence** : région homogène, flou
- **Haute fréquence** : contour, changement brusque d'intensité, bruit



Basse fréquence



Haute fréquence

Figure 4.7: Fréquence spatial.

Transformation de Fourier d'une image

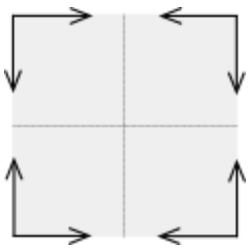
$$G\ (k,l) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} g(m,n) e^{-2j\pi(\frac{km}{M}+\frac{ln}{N})}$$

$$(k,l) \in \left[-\frac{M}{2}+1, \frac{M}{2}\right] \times \left[-\frac{N}{2}+1, \frac{N}{2}\right]$$

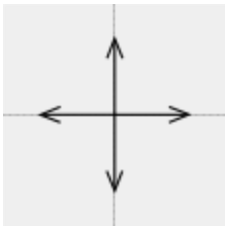
Ajustement

Les fréquences sont disposées de manière peu naturelle sur l'image obtenue

Par convention, l'image est réajustée en inter-changeant les quatre quadrants afin d'avoir les basses fréquences au centre de l'image et les hautes fréquences en périphérie, comme sur le schéma ci-après:



Résultat de la transformée de Fourier



Après inversion des quadrants

b) Analyse de la TF de l'image

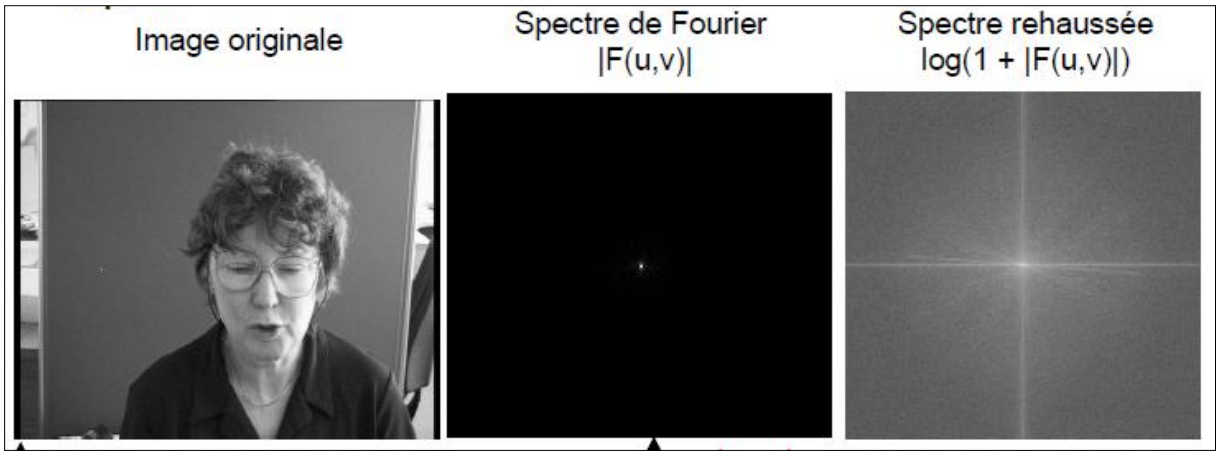


Figure 4.8: Transformée de fourrier d'une image.

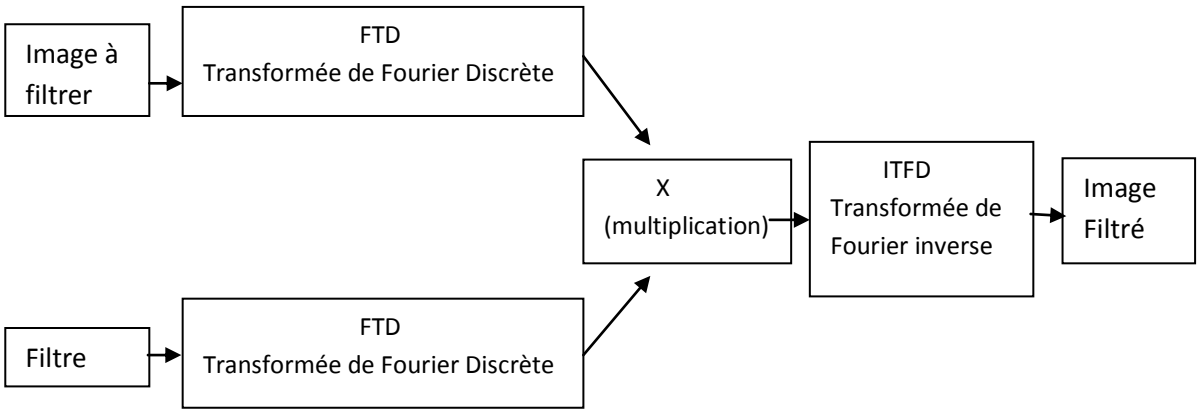
4.5 Principe général du filtrage fréquentiel

1- Calculer la transformée de Fourier de l'image IM de l'image im à filtrer

2- Calculer la transformée de Fourier H du filtre h

3- Calculer $IM_{filtré} = IM \times H$

4- Calculer $im_{filtré}$: la transformée de Fourier inverse de l'image obtenue $IM_{filtré}$



- **Hautes fréquences** : loin du centre de la TF
- **Basses fréquences** : proche du centre de la TF
- **Composante continue (DC)** : centre de l'image
- **Fréquence zéro** (moyenne de l'image): Centre le l'image

a) Un filtre passe-haut

Un filtre passe-haut est un système linéaire ne modifiant pas ou peu les hautes fréquences de l'image d'entrée.

- Basses fréquences et fréquence fondamentale éliminées
- L'information d'intensité est enlevée lors de la reconstruction de l'image (IDFT)
- Hautes fréquences préservées
- Les changements brusques d'intensité (bruit, frontières, ...) sont mis en évidence.

Pour filtrer les bases fréquence, il suffit de multiplier TFD de l'image par un masque binaire (filtre) dont les pixels centrés sont à 0.

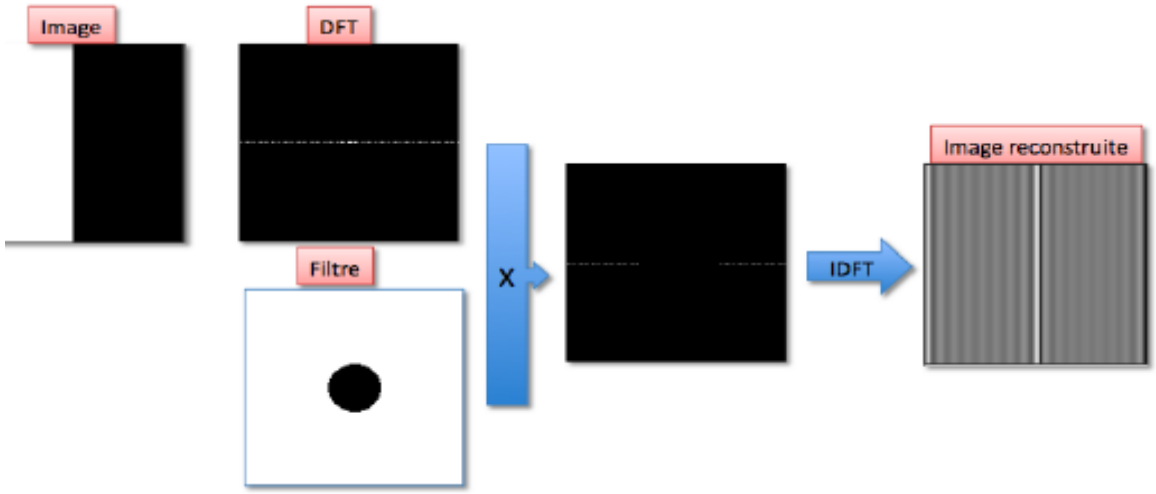


Figure 4.9: filtrage fréquentiel passe haut.

L'image reconstruite a les contours nets (peut être utilisé pour la détection de contours.

Exemple: filtre passe-haut 2D de ***Butterworth*** d'ordre n

$$H(U,V) = \frac{1}{1 + \left(\frac{D_0}{\sqrt{U^2 + V^2}}\right)^{2n}}$$

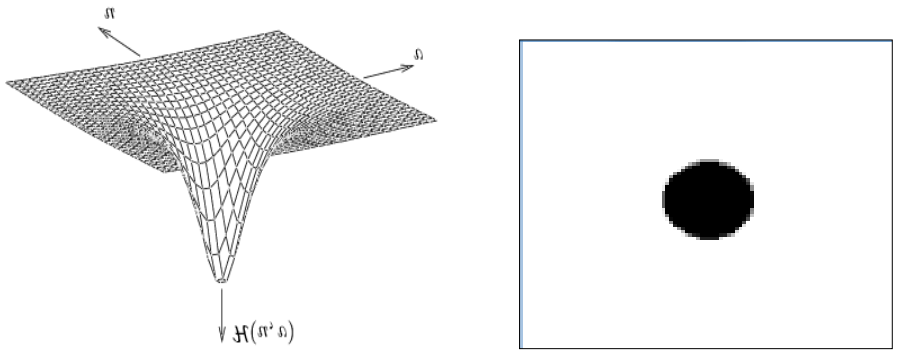


Figure 4.10: Filtre passe-haut de Butterworth d'ordre n=1.

b) Un filtre passe-bas

- Basses fréquences et fréquence fondamentale sont conservées

==> L'information d'intensité est restituée lors de la reconstruction de l'image (IDFT)

- Hautes fréquences éliminées : les changements brusques d'intensité

(bruits, frontières, ...) sont atténués voire éliminés

==> Étalement des frontières

Le même procédé suivi dans le filtrage passe haut, sauf que le filtre passe-bas (masque) utilisé dans la multiplication avec la TFD de l'image est l'inverse du filtre passe-haut.

Exemple: Filtre passe-bas 2D de ***Butterworth*** d'ordre n

$$H(U,V) = \frac{1}{1 + \left(\frac{\sqrt{U^2 + V^2}}{D_0}\right)^{2n}}$$

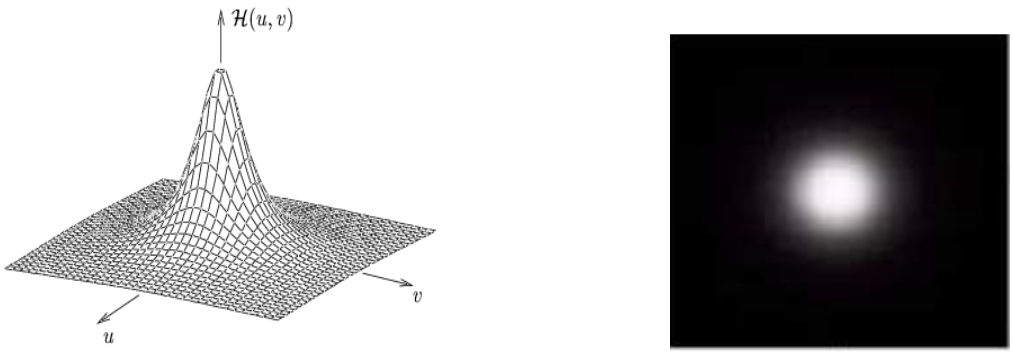


Figure 4.11: Filtre passe-bas de Butterworth d'ordre n=1.

c) Un filtre passe-bande

Le même procédé suivi dans le filtrage passe haut et basse fréquence, sauf que le filtre passe-bande (masque) est comme suit:

- Un filtre passe-bande est complémentaire d'un filtre passe-bas et d'un filtre passe-haut
- Un filtre passe-bande est un système linéaire qui préserve une plage de fréquences.
- L'image reconstruite est une combinaison d'un nombre réduit d'images de base (sinusoïdes)

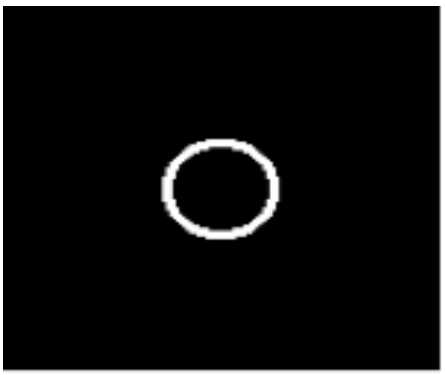


Figure 4.12: Filtre passe bande.

CHAPITRE 5: Détection de contours

5.1. Définition

Les contours sont les changements d'intensité dans l'image. Ou les frontières qui séparent deux ou plusieurs objets dans l'image.

5.2. Types de contours

On peut distinguer trois types de contours, selon la 'vitesse' de changement d'intensité entre deux régions.

1- Marche d'escalier: changement abrupt (**exp**: passage de niveau de gris de 0 à 200)
2- Rampe (**exp**: [1 2 3 5 7 9 11]
3- Toit (**exp**: [0 0 200 0 0]

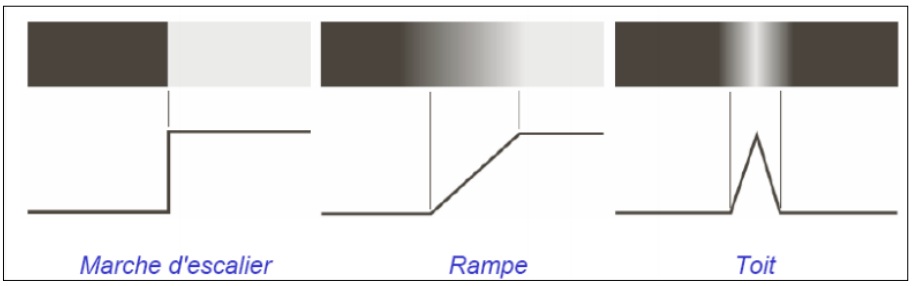


Figure 5.1: Types de contours.

5.3. La première dérivée d'une image

a) Opérateur de gradient

En considérant l'image dans un repère orthogonal (Oxy) tel que (Ox) désigne l'axe horizontal et (Oy) l'axe vertical, le **Gradient de l'image** (ou plutôt de la **luminance *f***) en tout pixel de coordonnées (x, y) est désigné par :

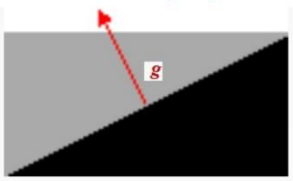
$$\vec{Grad}\, f = \vec{\nabla} f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$$

Le **module du gradient** quantifie l'importance du contour, c'est-à-dire l'amplitude du saut d'intensité relevé dans l'image :

$$\|\vec{\nabla} f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

La **direction du gradient** permet de déterminer l'arête présente dans l'image. En effet, la direction du gradient est orthogonale à celle du contour :

$$\alpha_0 = \arctan\left(\frac{\partial f / \partial y}{\partial f / \partial x}\right)$$



Le principe de la détection de contours par l'utilisation du gradient consiste à:

1- Calculer le gradient de l'image dans deux directions orthogonales
2- Calcule de module du gradient.
3- Sélection des contours les plus marqués, c'est-à-dire les pixels qui dépassent un seuil donné.
4- Les directions des contours étant orthogonales à la direction α_0 déterminée en tout pixel de l'image.

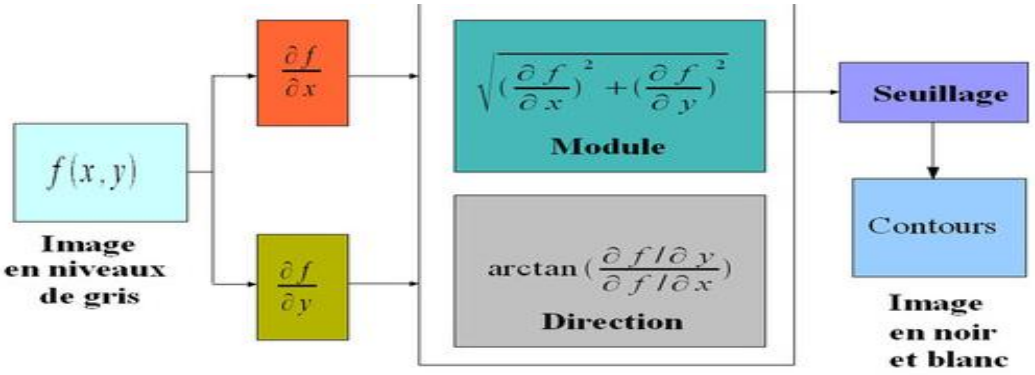


Figure 5.2: Schéma de détection de contours par le Gradient.

On approxime les dérivées par les "différences finies".

$$Gx = I(x+1,y) - I(x,y),$$

$$Gy = I(x,y+1) - I(x,y).$$

Cela revient a convoluer l'image avec les deux filtres Rx = [-1 1] et Ry = transpose([-1 1]).

b) Masque de Roberts,

Il fournit une première approximation de la première dérivée d'une image discrète. Roberts a proposé 2 masques dérivatifs diagonal.

0	1
-1	0

1	0
0	-1

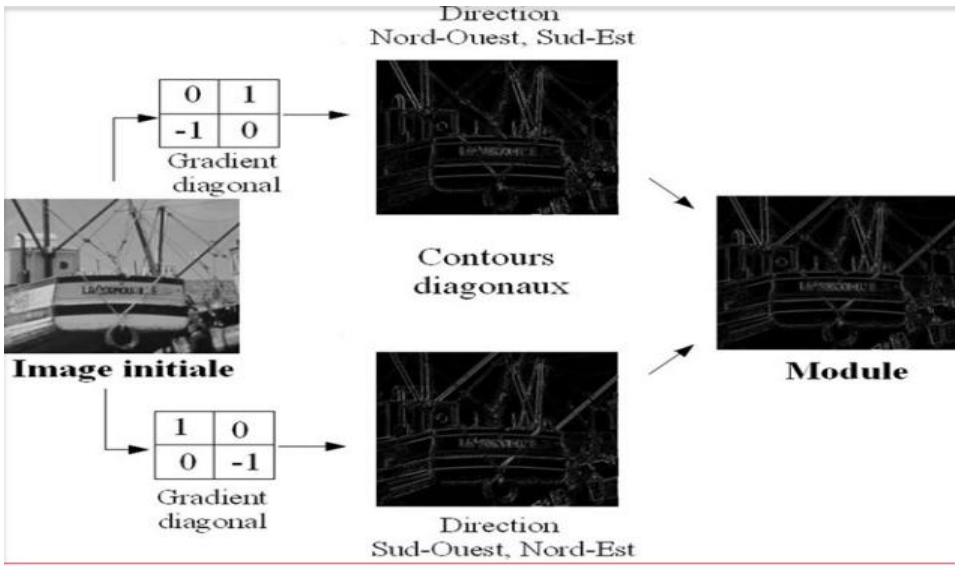


Figure 5.3.Schéma de détection de contours par les masques de Roberts.

c) Masque de Prewitt,

La convolution ce fait avec les masques de Prewitt suivants

Opérateur de Prewitt :

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

d) Masque de Sobel

La convolution ce fait avec les masques de Sobel suivants

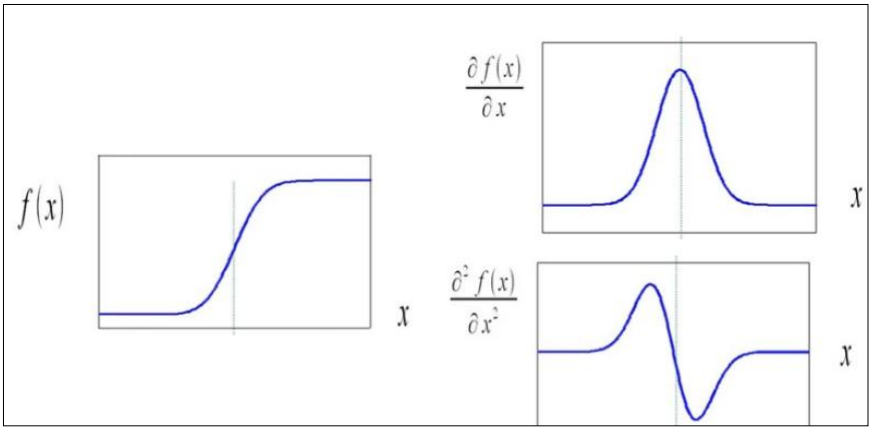
Opérateur de Sobel :

$$h1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad h2 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Remarque: ces méthodes sont sensibles aux bruits. Pour limiter les effets du bruit, un lissage est compris dans le calcul (filtre moyenner pour Prewitt, filtre gaussien pour Sobel).

5.4. La deuxième dérivée d'une image

Les opérateurs de gradient vus précédemment exploitent le fait qu'un contour dans une image correspond au maximum du gradient dans la direction orthogonale au contour.



a) Opérateur Laplacien

Le passage par zéro de la dérivée seconde d'une rupture d'intensité permet également de mettre en évidence le contour. La dérivée seconde est donc déterminée par le calcul du Laplacien :

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial y}\right)$$

$$\nabla^2 f = \nabla_x\left(f(x+1,y) - f(x,y)\right) + \nabla_y\left(f(x,y+1) - f(x,y)\right)$$

$$\nabla_x\left(f(x+1,y) - f(x,y)\right) = f(x+1,y) - f(x,y) - \left(f(x,y) - f(x-1,y)\right)$$
$$\nabla_x\left(f(x+1,y) - f(x,y)\right) = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

$$\nabla_y\left(f(x,y+1) - f(x,y)\right) = f(x,y+1) - f(x,y) - \left(f(x,y) - f(x,y-1)\right)$$
$$\nabla_y\left(f(x,y+1) - f(x,y)\right) = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

$$\nabla^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \qquad (5.1)$$

Cette opération de calcul de Laplacien peut alors être appliquée à une image par l'intermédiaire d'un filtrage avec le masque 3*3 suivant :

0	1	0
1	-4	1
0	1	0

D'autres masques peuvent être utilisés

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	-8	-1
-1	-1	-1

1	-2	1
-2	4	-2
1	-2	1

Après le filtrage de l'image au moyen d'un de ces masques, les contours sont détectés comme les passages par zéro.

Sensibilité aux bruit:

De même que lors de l'utilisation de l'opérateur gradient, l'idée consiste d'abord à filtrer l'image par un filtre passe-bas avant d'appliquer l'opérateur Laplacien.

Mathématiquement cela revient à convoluer l'image initiale $f(x,y)$ avec la dérivée seconde de la réponse impulsionnelle $h(x,y)$ du filtre passe-bas comme le montre l'écriture suivante dans le cas monodimensionnel :

$$\frac{\partial^2}{\partial x^2}\left(f(x,y)*h(x,y)\right)=\frac{\partial^2 f(x,y)}{\partial x^2}*h(x,y)=f(x,y) \qquad (5.2)$$

b) Filtre de Marr-Hildreth

Pour remédier à la sensibilité aux bruits Marr et Hildreth ont proposé de replacer le filtre passe-bas par un filtre gaussien.

$$H_{\sigma,\mu}(x,y)=\frac{1}{2\pi\sigma^2}\mathrm{e}^{\frac{(x^2-y^2)}{2\sigma^2}} \qquad (5.3)$$

La dérivée de la fonction gaussienne bidimensionnelle s'écrit alors :

$$H'_{\sigma,\mu}(x) = \frac{-(x+y)}{2\pi\sigma^4}\ \mathrm{e}^{\frac{(x^2-y^2)}{2\sigma^2}} \qquad (5.4)$$

Et la dérivée seconde s'exprime :

$$H''_{\sigma,\mu}(x,y) = -\frac{1}{\pi\sigma^4}\left(1-\frac{(x+y)}{\pi\sigma^2}\right)\mathrm{e}^{\frac{-(x^2-y^2)}{2\sigma^2}}$$

La réponse impulsionnelle du filtre **Laplacien de Gaussienne** (LOG), appelée également « *chapeau mexicain* ».

La détection de contours avec le filtre LOG :

1. Convolution de l'image avec le filtre LOG (filtre 2D) ;
2. Détection des passages par zéro de l'image résultante ;
3. Seuillage de l'image afin de ne considérer que les passages par zéro d'amplitude suffisante.

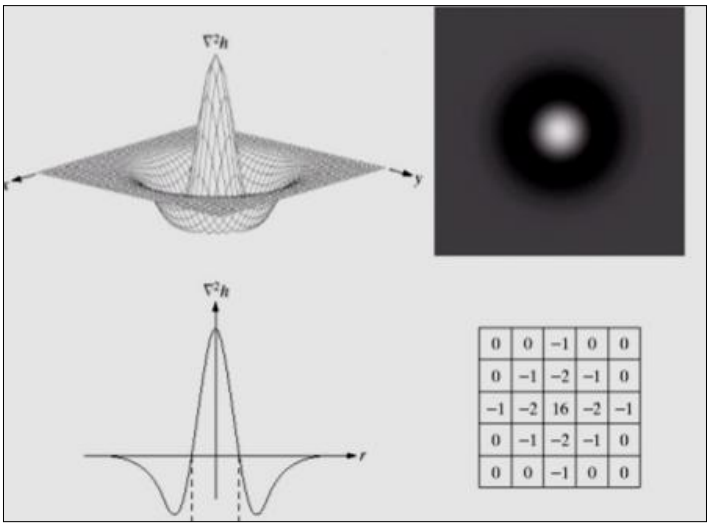


Figure 5.4: Laplacien du gaussien(LoG).

5.5. Comparaison des méthodes basées sur la 1ier dérivée (gradient) et celles basées sur 2eme dérivés (Laplacien)

	GRADIENT	LAPLACIEN
Avantages	• Fournit l'orientation du contour • Bonne localisation malgré le lissage • La suppression des non-maxima locaux fournit des contour fins	• Proche du système visuel humain • La détection par zéro fournit des contour fermés
Inconvénients	• Sensible aux bruits • Contours non fermés • Le choix du seuillage	• Grande sensibilité aux bruits • Pas d'information sur l'orientation du contour • Le choix du seuillage

CHAPITRE 6 : Segmentation d'images

6.1. Définitions

La segmentation consiste à séparer les zones homogènes dans l'image selon un critère ou plusieurs critères (niveau de gris, couleur, texture, forme...).

La segmentation est appliquée dans plusieurs domaines:

- Imagerie médicale: tumeur du cerveau, échographie, segmenter des lésions de la peau..
- Images satellitaires: cartographie, aménagement des sols...
- Indexation : rechercher dans une base d'images, les images
- Reconnaissance d'objets

Il existe plusieurs méthodes de segmentation:

1- Méthodes statistiques: Isodata, Kmeans, fcm (fuzzy c means)
2- Approches basées région: croissance de régions, division et fusion..
3- Approches basées contours: dérivative, contour actif (Snake, GVF, GAC)..

6.2. Méthodes statistiques

a) Basé sur l'histogramme

Les segmentations basées sur l'histogramme cherchent un seuil ou plusieurs seuils optimaux calculés à partir de l'histogramme de l'image.

Exemple:

L'histogramme suivant, d'une image donnée, est en forme de deux gaussiennes qui correspondent à deux zones dans l'image (objet à détecter et l'arrière-plan). Le seuil est la valeur qui sépare les deux pics (gaussiennes).

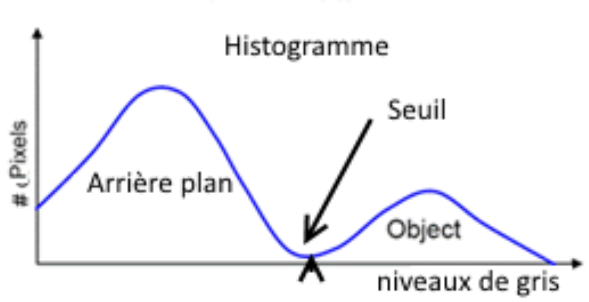


Figure 6.1: Histogramme d'une image à deux zones.

b) Méthode d'Otsu

La méthode d'Otsu cherche à minimiser la variance intra-classe d'une part et à maximiser la variance inter classe.

***Variance intra-classe* :**

$$\sigma_w^2 = \omega_1(T) \times \sigma_1^2(T) + \omega_2(T) \times \sigma_2^2(T)$$

w_1 : représente la probabilité qu'un pixel soit dans la classe 1

w_2 : représente la probabilité qu'un pixel soit dans la classe 2

σ_1 : représente la variance de la classe 1

σ_2 : représente la variance de la classe 2

Otsu montre que minimiser la variance intra-classe revient à maximiser la variance inter-classe

Variance inter-classe :

$$\sigma_b^2(t) = \sigma^2 - \sigma_w^2 = w_1(t)w_2(t)[\mu_1(t) - \mu_2(t)]^2$$

σ : représente la variance de l'image

σ_w : représente la variance intra-classe

Algorithme:

1. Calculer l'histogramme et les probabilités de chaque niveau d'intensité
2. Définir les $w_1(0)$, $w_2(0)$ et initiaux

a. Parcourir tous les seuils possibles T=1, 2 ...255

b. Mettre à jour w_i et μ_i

c. Calculer $\sigma_b^2(t)$

3. Le seuil désiré correspond au $\sigma_b^2(t)$ maximum.

c) Algorithme des K-means

Permet de répartir un ensemble d'échantillons (pixels) en sous-ensembles (K classes) homogènes selon un critère de distance.

K et les paramètres (moyenne u_k^0 , et écart type σ_k^0) : doivent être initialisés à priori.

Algorithme de Kmeans

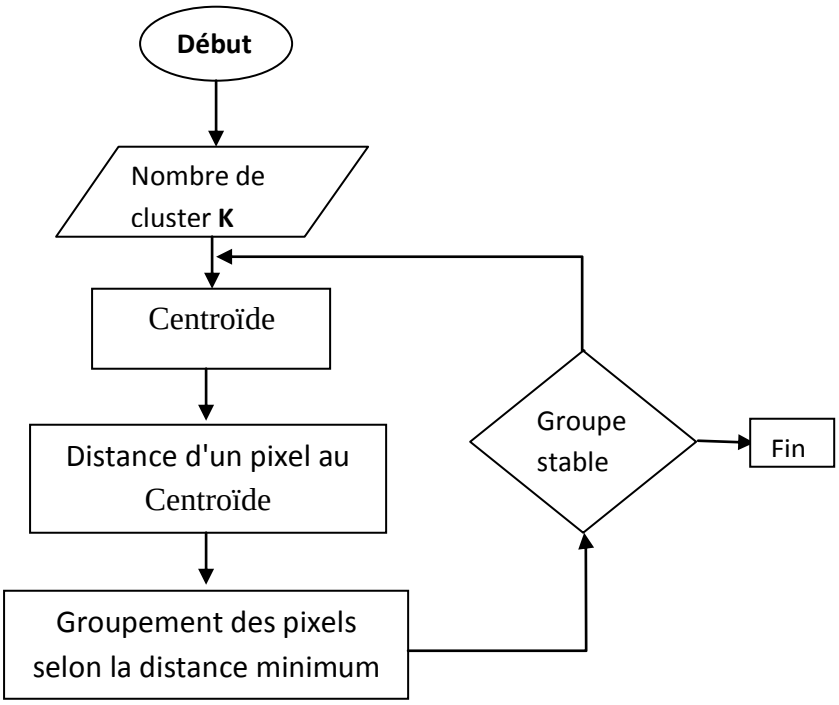
1- Définir aléatoirement les centres des ***k*** classes

2- Balayer l'image, en calculant la distance entre chaque pixel et les centres de classes.

3- Attribuer les pixels aux classes les plus proches

4- Recalculer les centres de classes en considérant les pixels attribués pour chaque classe. (centre de classe est la moyenne des pixels de la même classe)

5- Répéter les étapes 2,3 et 4 jusqu'à ce que les centres de lasses ne changent pas.



6.3. Méthodes Géométriques

a) Méthode de croissance de région

L'algorithme procède comme:

1 - Attribuer une étiquette à un pixel de départ (seed).

2 - Tous les pixels voisins et similaires (seuil à déterminer) seront étiquetés par la même étiquette (créant ainsi une région)

3- La croissance de la région se poursuit en ajoutant les pixels voisins, similaires, jusqu'à ce que le critère de similarité n'est plus vérifié.

b) Méthode de division et fusion (Split and merge)

Étape 1: division

L'idée est de diviser l'image en de petites régions (split) de tailles identiques, homogènes selon un critère donné tel que: Région est divisé si sont écart type est <seuil.

Étape 2: fusion

Rassemble toutes les régions adjacentes qui satisfont le critère de similarité.

Exemple: abs(μ_{R1}-μ_{R2})<seuil. μ moyenne d'une région.

6.4. Modèle de contour actif.

l'idée est d'initialiser un contour autour de l'objet à détecter. Le contour initial évolue jusqu'aux contours de l'objet.

a) Snake

Le modèle Snake utilise comme contour initial des points dynamiques (courbe bidimensionnelle), entourant l'objet à détecter. L'évolution du contour initial vers les limites de l'objet est guidée par deux forces biaisées (énergies). Kass et al ont introduit la théorie du Snake. L'évolution de la courbe initial est basée sur la minimisation de l'énergie le long de la courbe. L'énergie comprend des énergies internes et externes.

Soit C le contour initial et (x, y) ses coordonnées.

$C(s) = (x(s), y(s))$; s = [0, 1].

$$E_{snake} = \int_0^1 E_{snake}\big(C(s)\big)ds \qquad (6.1)$$

$$E_{snake} = \int_0^1 E_{int}\big(C(s)\big) + E_{ext}\big(C(s)\big)ds \quad (6.2)$$

L'énergie interne E_{int} préserve le lissage et la fermeté de la forme initiale, elle est représentée par deux termes liés à la dérivée de premier et second ordre de C.

$$E_{int} = \alpha \left|\frac{\partial C(s)}{\partial s}\right|^2 + \beta \left|\frac{\partial}{\partial s}\left(\frac{\partial C(s)}{\partial s}\right)\right|^2 \quad (6.3)$$

$$E_{snake} = \int_0^1 \alpha \left|\frac{\partial C(s)}{\partial s}\right|^2 + \beta \left|\frac{\partial}{\partial s}\left(\frac{\partial C(s)}{\partial s}\right)\right|^2 + \int_0^1 E_{ext}\big(C(s)\big)\, ds \quad (6.4)$$

La première dérivée rend le contour flexible et maintient la distance entre deux points. Il représente la quantité d'étirement du contour, et il est contrôlé par α.

La dérivée de deuxième ordre préserve la rigidité du contour et empêche l'apparition des coins, il représente la courbure du contour initial, et il est contrôlé par β.

L'énergie externe est considérée comme une fonction de traction qui attire le contour initial aux points des arêtes des objets.

$$E^1{}_{ext} = -|\nabla \mathrm{Image}|^2 \quad (6.5)$$

$$E^2{}_{ext} = -|\nabla(G_\sigma(\mathrm{x},\mathrm{y}) * \mathrm{IG}(\mathrm{x},\mathrm{y}))|^2 \quad (6.6)$$

G_σ : est la fonction gaussienne bidimensionnelle avec écart type (σ)

∇ : opérateur gradient

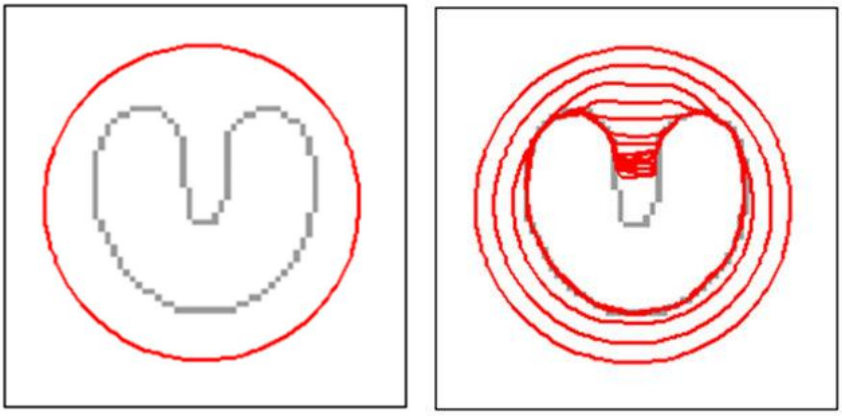
$*$: opérateur de convolution

L'équation (6.5) améliore les contours et l'équation (6.6) lisse et met en évidence les contours, pour éviter tout problème de blocage des points du contour dans l'énergie minimale.

Il existe trois inconvénients majeurs liées à l'approche Snake. Premièrement, le contour initial devrait être proche du contour de l'objet à segmenter, plus le contour initial est large (loin de l'objet) plus le contour à plus de chance de s'accrocher au minimum d'énergie local (en raison de la petite plage de capture du gradient).

Deuxièmement, l'évolution de la courbe dépend des paramètres (α, β), les valeurs optimales sont difficiles à trouver.

Enfin, le Snake est incapable d'évoluer vers des concavités, ce qui augmente les erreurs de segmentations, comme le montre la figure 6.2



(a) (b)

Figure 6.2: Snake n'atteigne pas à la cavité (a) contour initial, (b) évolution du contour.

b) GVF (Gradient Vector Flow)

Pour surmonter la contrainte d'initialisation du contour requise dans le Snake classique (le contour initial devrait être proche de l'objet), Xu et al ont introduit le GVF. L'idée est de remplacer l'énergie externe utilisée par la méthode Snake par un champ de forces externes (GVF) qui a une plus grande plage de capture. Cela permet d'attirer le contour initial vers l'objet même s'il est loin. De plus, le GVF peut se déplacer dans des concavités et des coins qui sont hors de portée du Snake.

L'équation (6.2) est réécrite comme

$$E_{snake} = \int_0^1 \alpha \left|\frac{\partial C(s)}{\partial s}\right|^2 + \beta \left|\frac{\partial}{\partial s}\left(\frac{\partial C(s)}{\partial s}\right)\right|^2 + GVF \qquad (6.7)$$

L'approche étend la propriété attrayante du gradient près des bords à l'image entière à l'aide d'un processus de calcul de diffusion.

GVF est définie comme la solution à l'équilibre de l'équation de diffusion vectorielle suivante (6.8).

$$\begin{cases} v_t = \mu\nabla^2 \mathrm{v} - (\mathrm{v} - \nabla f)|\nabla f|^2 & (2.8a) \\ \mathrm{v}_0 = \nabla f & (2.8b) \end{cases} \qquad (6.8)$$

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{6.9}$$

f : plan des contours: dérivée d'image (nivaux de gris).

v_t : dérivée partielle de v par rapport à t

μ : paramètre utilisé pour mettre en évidence les contours.

Algorithme:

$GVF = [u, v]$

initialization de u, v

$v = f_x; \quad u = f_y$

$[f_x, f_y] = \text{gradient}(f)$

u, v sont calculés après un certain nombre d'itérations comme

$$v = v + \mu\nabla^2 u - \left(f_x^2 + f_y^2\right)\left(u - f_x\right) \tag{6.10}$$

$$u = u + \mu\nabla^2 v - \left(f_x^2 + f_y^2\right)(v - f_x) \tag{6.11}$$

La carte contours de GVF comporte des vecteurs qui pointent vers les bords (normaux aux bords) comme le montre la figure suivante.

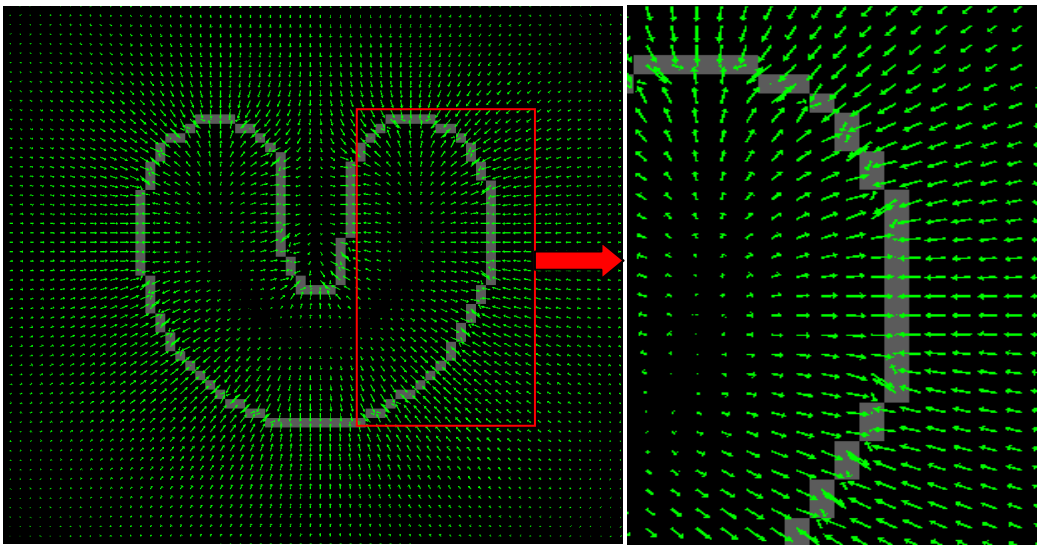


Figure 6.3: U-forme:Le GVF pousse le contour initial vers les cavités.

TRAVAUX PRATIQUES

TP 1: Représentation et manipulation des images sous Matlab

TP 2: Traitements de base sur les images, histogramme et opérations arithmétiques

TP 3: Filtrage d'images

TP 4: Détection de contours

TP 5: Segmentation d'images

TP 01

Représentation et manipulation des images sous Matlab

But du TP

1 - Coder une matrice et l'afficher sous forme d'une image

2 - Lecture et affichage d'une image réelle

3 - Conversion d'un espace de couleur vers un autre (RGB, HSV, l*a*b*..)

4 - Manipulation d'image (extraire un pixel, un carré...)

1- Coder une matrice est l'afficher sous forme d'une image

1.1 Niveau de gris

a) Générer une matrice ***Img*** de 100*100 en utilisant la commande '***ones***'

b) Multiplier la matrice ***Img*** par ***x*** (x: réel positif)

c) Coder les éléments de la matrice ***Img*** sur 8 bits par la commande (***uint8***)

d) Afficher la matrice codée, pour différente valeur de ***x*** (0, 50, 100, 200, 250) (commande d'affichage **imshow)**

1.2 Colorimétrie

a) Générer une image couleur (système RGB)

- Afficher les plans R, V et B séparément.

```
R=uint8(x*ones(100)); subplot(311), imshow(R); title('R');
V=uint8(y*ones(100)); subplot(312), imshow(V); title('V');
B=uint8(z*ones(100)); subplot(313), imshow(B); title('B');
```

- Afficher l'image couleur constituée par des plans R, V et B précédents

```
Img(:,:,1)=R;  Img(:,:,2)=V;  Img(:,:,3)=B;
figure,imshow(Img)
```

2 - Lecture et affichage d'une image réelle

2.1 Image couleur

```
im = imread(filename,fmt)
```

- **filename** : est le nom de l'image.
- **fmt** : est l'extension du format de stockage de l'image (jpg, tif, png,...).

exemple:

```
Img = imread('cameraman', 'tif')
figure,imshow(Img)
```

- si l'image à lire est dans un dossier le dossier MATLAB dont la racine est

```
filename  ='C:\Users\nom-PC\Desktop\
Img = imread(['C:\Users\nom-PC\Desktop\' 'NOM-IMAGE.jpg'])
figure,imshow(Img)
```

2.2.Luminance de l'image

La luminance d'une image correspond à l'intensité des pixels de l'image.

Pour une image en couleur **RGB**, la luminance est
I=0.2989*R+0.5870*G+0.1140*B

Sous Matlab, la conversion de l'image couleur en niveaux de gris (luminance) se fait par la commande **rgb2gray**

```
Img_gray = rgb2gray(Img)
figure,subplot(121),imshow(Img)
subplot(122),imshow(Img_gray)
```

3 - Conversion d'un espace de couleur vers un autre (RVB, HSV, l*a*b*..)

RGB à HSV

```
HSV=rgb2hsv(RVB) ;
figure,subplot(121),imshow(RVB)
subplot(122),imshow(HSV)
```

RGB à L*a*b*

```
RGB=uint8(RGB);
colorTransform = makecform('srgb2lab');
lab = applycform(RGB, colorTransform);

figure,subplot(121),imshow(RVB)
subplot(122),imshow(lab)
```

4 - Manipulation de l'image (extraire un pixel, une fenêtre ...)

- Chaque pixel est localisé par les coordonnées ligne et colonne

Pxl=Img(Ligne, colonne, plan),
plan= 1, pour le plan rouge (R)
plan=2 , pour le plan vert (V)
plan:=3, pour le plan bleu (B)

- L'affichage d'une partie ou une fenêtre de l'image se fait en précisant les dimensions des lignes et de colonnes de la fenêtre à extraire.

Fenêtre =Img(L1: L2 , C1:C2)

example:

```
Img=imread('cameraman.tif')
```

```
Ftr =Img(Img(35:100,80:150)
figure, imshow(Ftr)
```

Commands Matlab

ones: créer une matrive ou un vecteur dont les éléments égales à 1.
uint8: entiers non signés sur 8 bits
imshow : affiche une figure
subplot : crée des sous figure dans une figure
imread : lit une image
rgb2gray: transforme une image couleur au niveau de gris
makecform : créer une structure de transformation de couleur

TP 2

Traitements de bases sur l'image

Histogramme et opération

But du TP

1- Générer l'histogramme d'une image

2- Égalisation de l'histogramme

3- Ajustement de l'histogramme

4- Opérations logiques et arithmétiques sur les images

1-Générer l'histogramme d'une image

a) Lire l'image '*cameraman.tif*' sous Matlab

```
Img=imread('cameraman.tif')
```

b) Générer son histogramme

```
[H,n]=imhist(image)
```

d) Afficher l'image et son histogramme sur la même figure.

```
figure,
subplot(211),imshow(Img), title('image aux niveaux de gris')
subplot(212),plot(H), title('L'histogramme')
```

e) Localiser les parties de l'image qui correspondent aux lobes qui apparaissent dans l'histogramme.

f) Afficher l'image inversée (255-Img)

g) Afficher sur la même figure l'histogramme de l'image original et l'histogramme de l'image inversée.

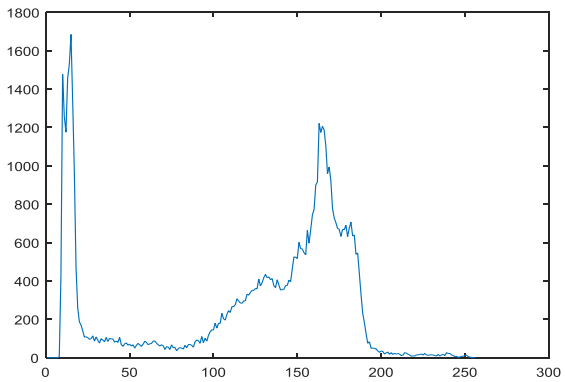
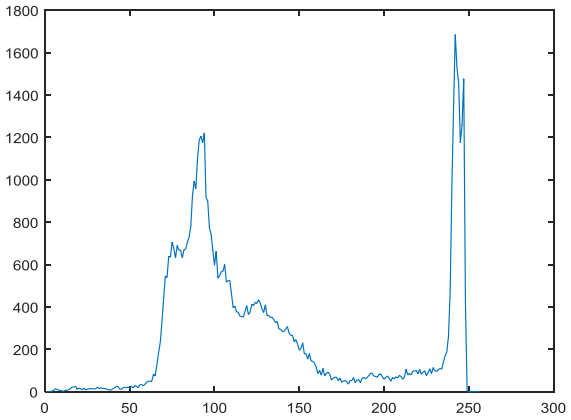


Image est globalement sombre: apparition de deux lobes sur l'histogramme, le lobe à gauche (objet **A**) dont les valeurs des pixels sont autour de 20 indique que l'objet **A** est sombre.

Le deuxième lobe (objet **B**) dont les valeurs des pixels sont autour de 175 correspond a la partie claire (objet B) de l'image.

Exemple 2

Si on inverse l'image précédente (255-Image), on obtient l'histogramme suivant:



Le lobe à gauche dont les valeurs des pixels sont autour de 90: objet B devient plus sombre.
Le lobe à droite dont les valeurs des pixels sont autour de 230: objet A devient clair.

2- Égalisation de l'histogramme

a) Cette partie consiste à afficher une image de Matlab (`'tire.tif'`)
b) Rehausser l'image en utilisant la méthode d'**égalisation de l'histogramme (histeq)**
c) Afficher sur la même figure
- Image originale
- Histogramme Image originale
- Image rehaussée
- Histogramme de l'image rehaussée

```
I = imread('tire.tif');
figure,
subplot(221),imshow(I),title('Image original')
[H2,n]=imhist(I)
subplot(222),plot(H2),title('Histogramme Image
original')

% Rehaussement de l'image par l'égalisation de l'histogramme
J2 = histeq(I);
subplot(223),imshow(J2),title('Image rehaussée')
[H2,n]=imhist(J2)
subplot(224),plot(H2),title('Histogramme de l''image
rehaussée')
```

3- Ajustement de l'histogramme

Cette partie consiste à

a) Afficher une image de Matlab (`pout.tif`)
b) Rehausser l'image en utilisant la méthode **d'ajustement de l'histogramme (imadjust)**

c) Afficher sur la même figure

- Image originale
- Histogramme Image originale
- Image rehaussée
- Histogramme de l'image rehaussée

```
I = imread('pout.tif);
figure,
subplot(221),imshow(I),title('Image original')
[H2,n]=imhist(I)
subplot(222),plot(H2),title('Histogramme Image
original')

% Rehaussement de l'image par ajustement de l'histogramme
J2 = imadjust(I);
subplot(223),imshow(J2),title('Image rehaussé')
[H2,n]=imhist(J2)
subplot(224),plot(H2),title('Histogramme de l''image
rehaussé')
```

Remarque: *pour plus d'information et d'options des commandes* ***histeq et imadjust*** *taper*

help imadjust

help histeq

4- Opérations arithmétiques et logiques sur les images

a) Addition (Augmentation de la luminance)

```
Img=imread('cameraman.tif');
figure,
subplot(311),imshow(Img)
subplot(312),imshow(Img+60)
subplot(313),imshow(Img+120)
```

b) Soustraction

```
Img=imread('cameraman.tif');
figure,
subplot(311),imshow(Img)
subplot(312),imshow(Img-60)
subplot(313),imshow(Img-120)
```

d) Opérations logiques

```
[a,b]=size(Img)

Ib1=zeros(a,b);
Ib1(20:100,100:250)=1;
figure,imshow(Ib1)


Ib2=zeros(a,b);
Ib2(40:120,30:140)=1;
figure,imshow(Ib2)

% ET logique
I_ET=Ib2&Ib1;
figure,imshow(I_ET)

% OU logique
I_OU=Ib2 | Ib1;
figure,imshow(I_OU)

% XOR operateur
I_xor=xor(Ib2,Ib1);
figure,imshow(I_xor)

%%========
% Affiché une partie de l''image utilisant le masque I_ET
figure,imshow(uint8(double(Img).*(I_ET)))

% Caché une partie de l''image utilisant l'inverse du masque I_ET
figure,imshow(uint8(double(Img).*(1-I_ET)))
```

Commandes Matlab

imread : lecture d'une image
figure : génère une figure vide
subplot : permet l'affichage plusieurs images sur la même figure.
imshow : affichage d'une image
imhist : génère l'histogramme de l'image ***im*** (niveau de gris)
histeq : égalisation de l'histogramme
imadjust : ajustement de l'histogramme

TP 3

Filtrage d'image

Médian, Moyenneur et Filtrage Fréquentiel

But du TP

1 - Filtre moyenneur (linéaire)
2 - Filtre médian (non linéaire)
3 - Filtrage Fréquentielle

1 - Filtre moyenner (linéaire)

Si un filtre (masque ***h***) de taille ***d***, dont tout les éléments ont comme valeur $w_i = \frac{1}{d^2}$ la convolution permet d'effectuer un **lissage par moyennes.**
Exemple **d=3**

$$h = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

- Sous Matlab, on peut générer un filtre spatial en choisissant le type de filtrage.

La fonction ***fspecial*** crée un filtre bidimensionnel ***h*** de type spécifié (moyen, gaussien, laplacian...). La syntaxe*:* ***h = fspecial(type,dimention).***

- La fonction ***imnoise*** permet d'ajouter un bruit dont on peut spécifier le type (gaussien, poisson, salt & pepper,...)

  la syntaxe*:* ***h = fspecial(type,dimention)***

Ajout de brouit

```
Im_org=imread('cameraman.tif');
figure,imshow(Im_org)

%% Filtrage moyenneur
 %1- Ajout de bruit à une image
     %Im_bruite = imnoise(Im_org,'salt & pepper',0.01);
      Im_bruite = imnoise(Im_org,'gaussian',0.01,0.01);
 figure,
 subplot(1,2,1),imshow(Im_org), title('Original')
 subplot(1,2,2),imshow(Im_bruite),title('Brouité')
```

Image originale

Ajout bruit gaussien

- Modifier les paramètres du bruit et gaussien et voir l'effet sur l'image.
- Modifier le type de bruit à ajouter ('salt & pepper', 'poisson').

Filtrage

```
%2-Création d'un filtre H (Masque )
    H=fspecial('average',3);
        %équivalent à H=(1/9)*[1 1 1;1 1 1;1 1 1];
    %équivalent à H=(1/9)*ones(3)

% 2- filtrer l'image 'cameraman' par différente
%    dimensions du filtre et voir l'effet
    ImF1=imfilter(Im_bruite,H);

    H2=fspecial('average',7);
    ImF2=imfilter(Im_bruite,H2);

    H3=fspecial('average',11);
    ImF3=imfilter(Im_bruite,H3);
    figure,
   subplot(1,4,1),imshow(Im_bruite),title('Bruit: Gaussian')
  subplot(1,4,2),imshow(ImF1), title('3x3')
  subplot(1,4,3),imshow(ImF2), title('7x7')
  subplot(1,4,4),imshow(ImF3), title('11x11')
```

L'effet du filtre moyenner sur une image dont on a ajouté un bruit de type 'Bruit:Gaussian'

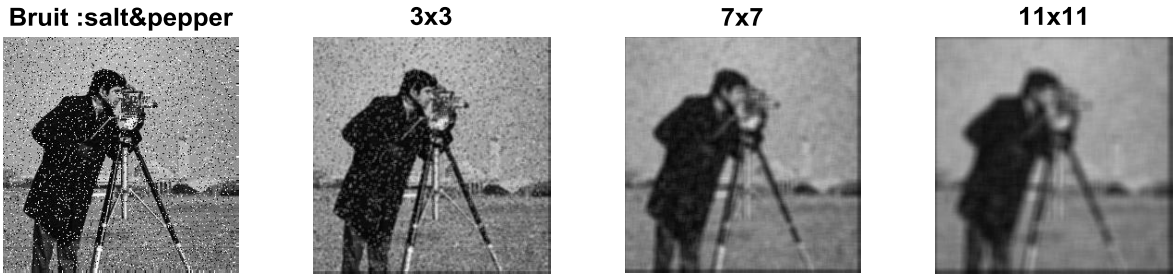
Bruit:Gaussian

3x3

7x7

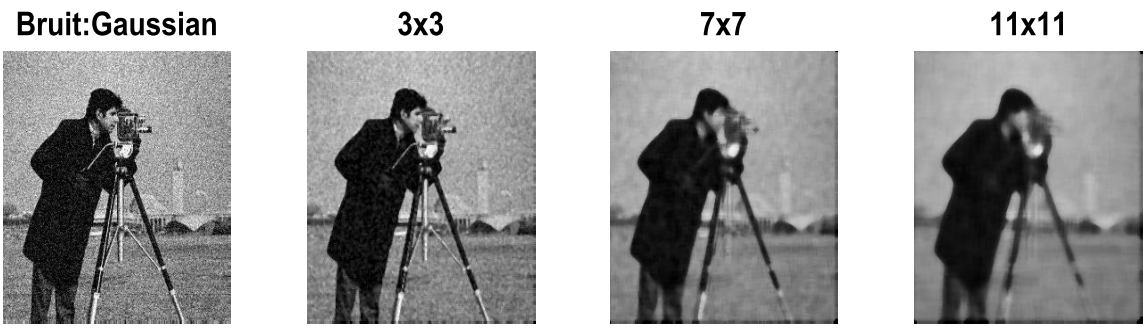
11x11

L'effet du filtre moyenner sur une image dont on a ajouté un bruit de type 'salt & pepper'

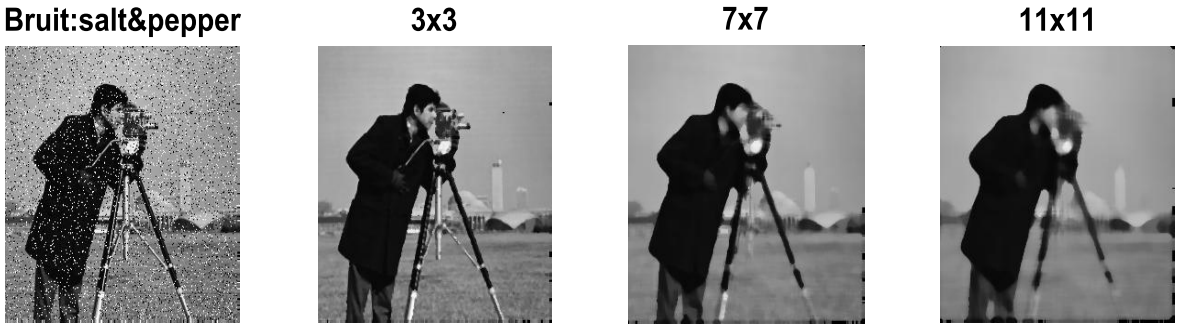


2 - Filtre médian (non linéaire)

```
%%  2 - Filtre médian (non linéaire)


%1- Ajout de bruit à une image

    %Im_bruite = imnoise(Im_org,'salt & pepper',0.1);
     Im_bruite = imnoise(Im_org,'gaussian',0.01,0.01);

%2-Filtre median
ImF1 = medfilt2(Im_bruite, [3 3]);
ImF2 = medfilt2(Im_bruite, [7 7]);
ImF3 = medfilt2(Im_bruite, [11 11]);

figure,
subplot(1,4,1),imshow(Im_bruite), title('Bruit: Gaussian')
subplot(1,4,2),imshow(ImF1), title('3x3')
subplot(1,4,3),imshow(ImF2), title('7x7')
subplot(1,4,4),imshow(ImF3), title('11x11')
```



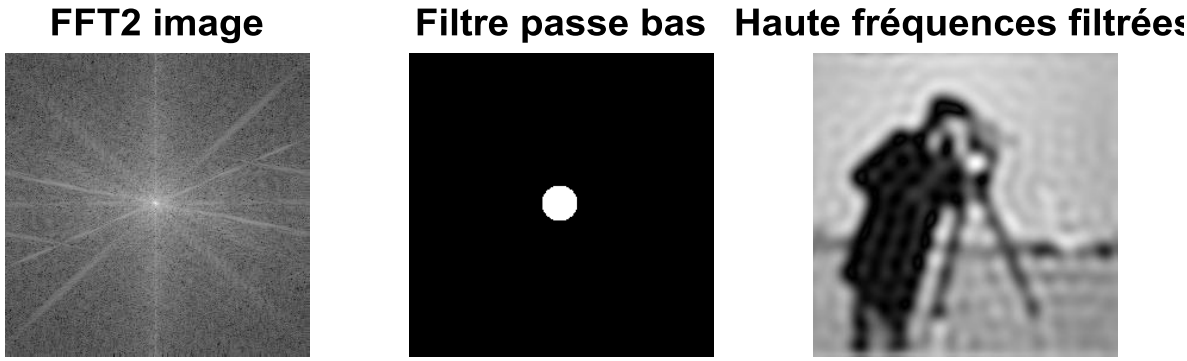
```
Im_bruite = imnoise(Im_org,'salt & pepper',0.1);
```



3. Filtrage fréquentiel

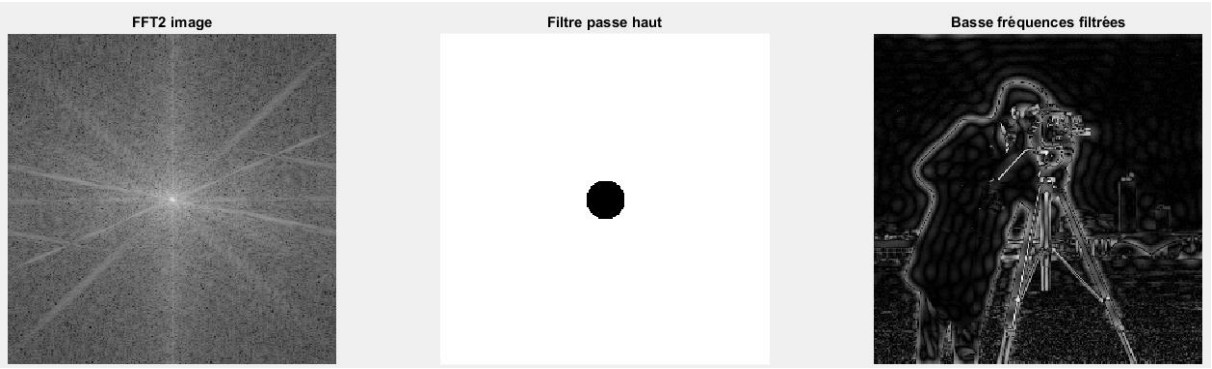
Filtre passe-bas: laisse passer les basses fréquences et qui atténue les hautes fréquences.

```
imfft=fft2(Im_org);
imfft2=log(abs(fftshift(imfft)) + 1);
figure,imshow(imfft2, [])
%=================
[x,y]=meshgrid(-128:127,-128:127);
z=sqrt(x.^2+y.^2);
H=z<15;
H1=H;
 %===========
af1=af.*H1;
af1Sh=log(1+abs(af1));
figure,imshow(af1)
inv_af1=ifft2(af1);
%===========
% FFT inverse
imfft_inv=ifft2(((imfft)).*(H));
figure,imshow((abs((imfft_inv))),[])
%=========
figure,
subplot(1,3,1),imshow(log(abs(fftshift(imfft))),[]),title('FFT2 image')
subplot(1,3,2),imshow(H1),title('Filtre passe bas')
subplot(1,3,3),imshow(log(1+abs(inv_af1./max(inv_af1(:)))),[]),title('Haut
e fréquences filtrées')
```



Filtre passe-haut: laisse passer les hautes fréquences et qui atténue les basses fréquences.

```
H2=1-H;
 af1=af.*H2;
af1Sh=log(1+abs(af1));
%figure,imshow(af1)
inv_af1=ifft2(af1);
%============
%figure,
subplot(1,3,4),imshow(log(abs(fftshift(imfft))),[]),title('FFT2 image')
subplot(1,3,5),imshow(H2),title('Filtre passe haut')
subplot(1,3,6),imshow(log(1+abs(inv_af1./max(inv_af1(:)))),[]),title('Bass
e fréquences filtrées')
```



commandes Matlab

imnoise : ajout de bruit à une image
imfilter : permet de filtré l'image
medfilt2 : filtre median
fft2 : TF 2D
ifft2 : TF 2D inverse
fftshift : Réarrange la TF

TP 4

Détection de contour

Détection de contours et segmentation

But du TP

1 - Détection de contours
1.1. Méthodes dérivatives d'ordre 1
1.2. Méthodes dérivatives d'ordre 2
2 - Filtre optimal (critères d'optimalité, Canny et Derriche)

1 - Détection de contour

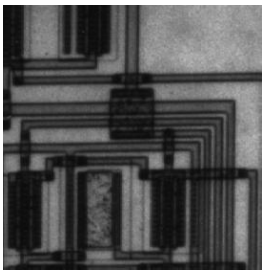
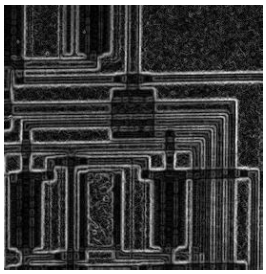
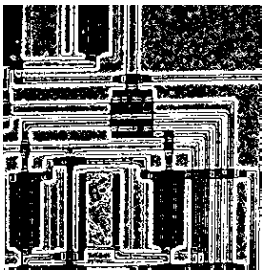
1.1. Méthodes dérivatives 1ier ordre

a) Gradient

```
clc
close all
clear all

I = imread('circuit.tif');
figure, imshow(I)
```

Détection de contours en utilisant le gradient

```
%% === Gradient =======

[Gx,Gy]=gradient(double(I));
% module
Gxy=sqrt(Gx.^2+Gy.^2);
Gxy=Gxy./max(Gxy(:));
s=0.2;
Gxy_s=Gxy>s;
figure,
subplot(131),imshow(I),title('original')
subplot(132),imshow(Gxy),title('Module du Gradient')
subplot(133),imshow(Gxy_s),title('Contours, module seuillé')
```

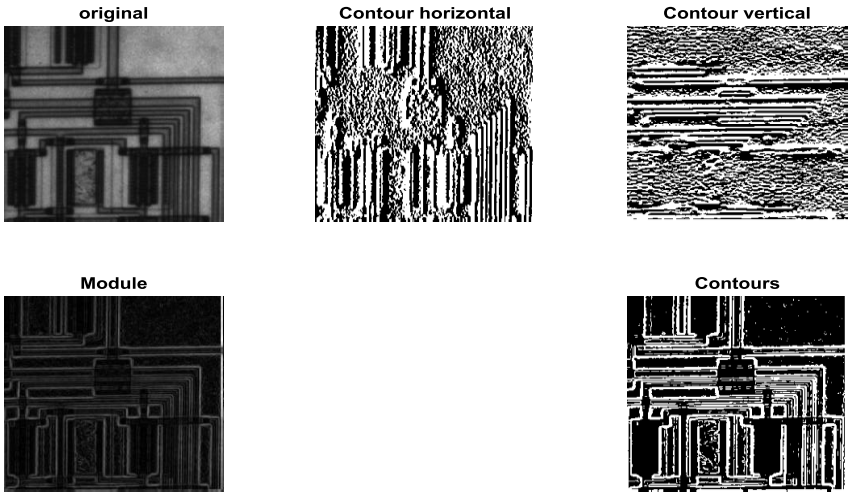
(a) (b) (c)

(a): image original, (b): Module du gradient, (c): contours détectés (module seuillé)

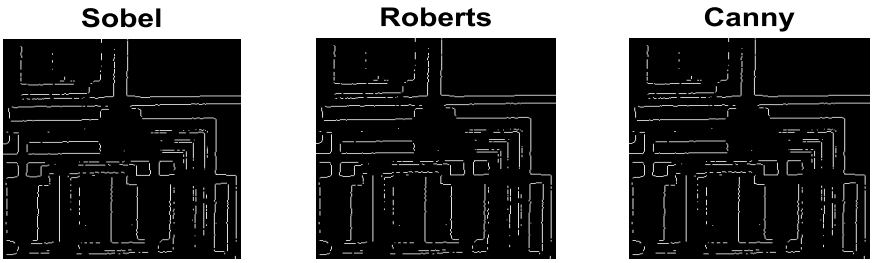
Changer le seuil **s** et voir les changement.

b) Masques (prewitt, sobel,..)

```
%% contours par masque de prewitt
H1=[- 1 0 1 ;-1 0 1;-1  0 1 ];
H2=[- 1 -1 -1;0 0 0; 1  1  1];
Cx=conv2(I,H1);
Cy=conv2(I,H2);
figure,
subplot(231),imshow(I),title('original')
subplot(232),imshow(Cx),title('Contour
horizontal')
subplot(233),imshow(Cy),title('Contour vertical')
% calcule du module
Cxy=sqrt(Cx.^2+Cy.^2);
Cxy=Cxy./max(Cxy(:)); % normalisé [0 1]
subplot(234),imshow(Cxy,[]),title('Module')
% seuillage
s=0.12;
Cxy_s=Cxy>s;
subplot(236),imshow(Cxy_s),title('Contours')
% prewitt: fonction matlab "Edge"
P_cont = edge(I,'Prewitt');
figure,imshow(P_cont),title('Prewitt')
%====================================
```



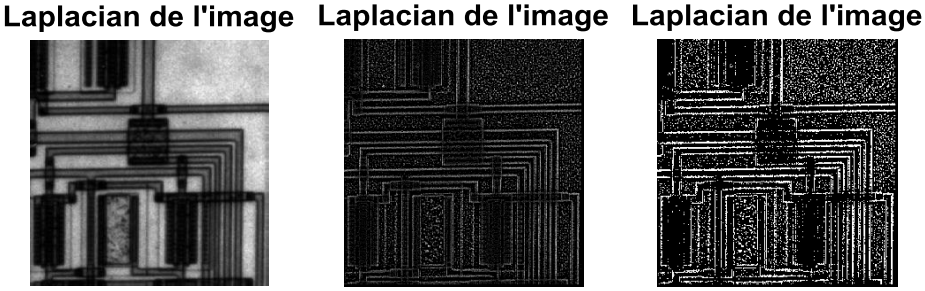
```
%==================================
%% contour par masque de Sobel
S_cont = edge(I,'Sobel');
figure,imshow(P_cont),title('Sobel')

%% contour par masque de Roberts
R_cont = edge(I,'Roberts');
figure,imshow(P_cont),title('Roberts')

%% contour par masque de Canny
C_cont = edge(I,'Canny');
figure,imshow(P_cont),title('Canny')
```



1.2. Méthodes dérivatives 2ier ordre

a) Laplacien

```
H = fspecial('laplacian');
I_lp = imfilter(I,H,'same');

 figure,
subplot(1,3,1),imshow(I,[]),title('Lap
lacian de l''image');
subplot(1,3,2),imshow(I_lp,[]),title('
Laplacian de l''image');
s=10
I_lps=I_lp>s;
```



TP 5
Segmentation d'images

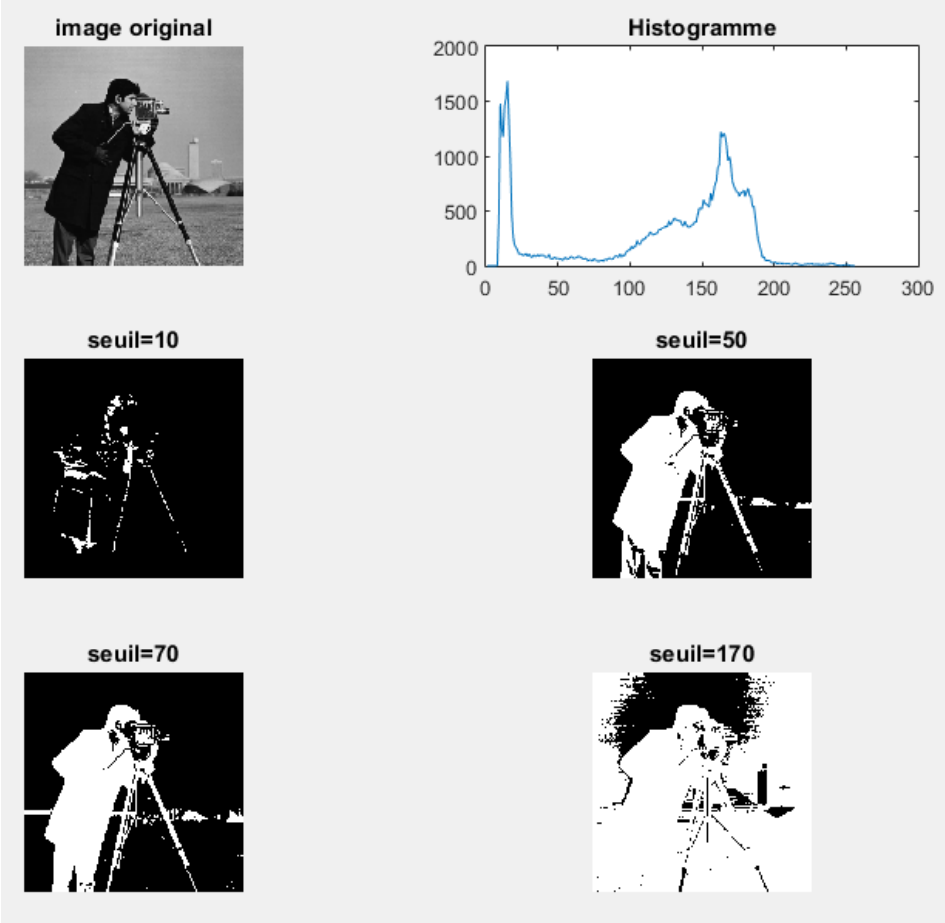
But du TP

1 - Segmentation basée histogramme
2- Segmentation par Otsu
3 - Segmentation k-means.

1- Segmentation basée histogramme

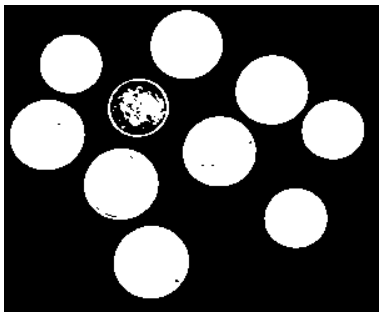
```
clc
close all
clear all

I = imread('cameraman.tif');
figure, imshow(I)
```

```
%% segmentation par histogramme
[h, n,]=imhist(I);

Ib1=I<10;
Ib2=I<50;
Ib3=I<70;
Ib4=I<170;

figure,subplot(3,2,1),imshow(I),title('image original')
subplot(3,2,2),plot(h), title('Histogramme')
subplot(3,2,3),imshow(Ib1),title('seuil=10')
subplot(3,2,4),imshow(Ib2),title('seuil=50')
subplot(3,2,5),imshow(Ib3),title('seuil=70')
subplot(3,2,6),imshow(Ib4),title('seuil=170')
```



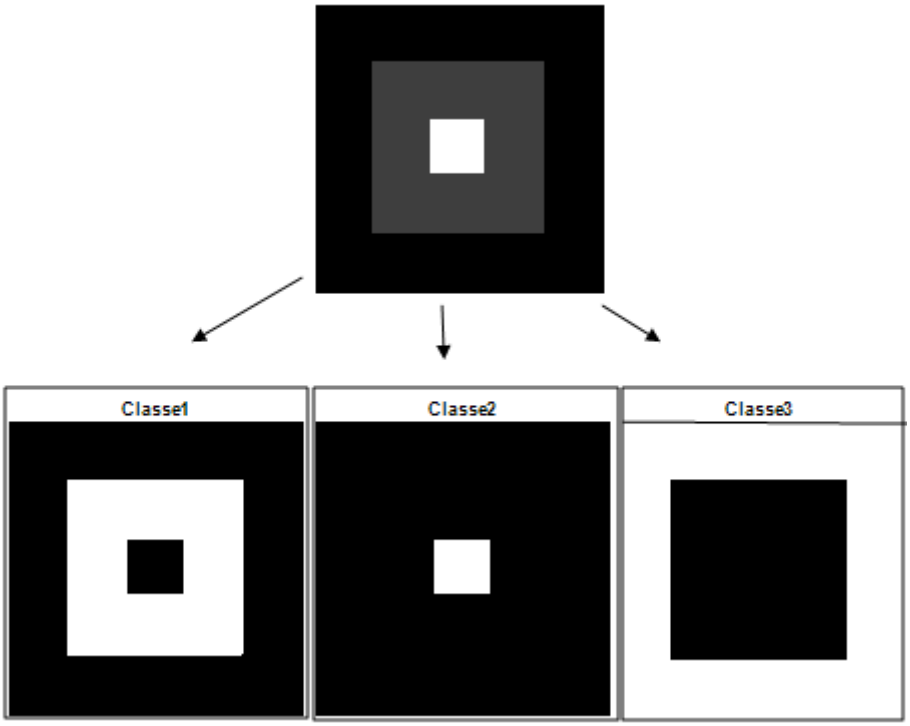
2- Segmentation par Otsu

```
I = imread('coins.png');
level = graythresh(I);
BW = im2bw(I,level);
figure,subplot(121),imshow(I);
subplot(122),imshow(BW)
```



3 - Segmentation k-means.

```
clc
close all
clear all
Img=255*ones(250,250);
Img(1:50,:)=50;  Img(:,1:50)=50; Img(end-50:end,:)=50;
Img(:,end-50:end)=50;
figure,imshow(Img)
Img(51:101,51:end-51)=100;  Img(51:end-51,51:101)=100; Img(end-
101:end-50,51:end-51)=100; Img(51:end-51,end-101:end-50)=100;
figure,imshow(Img,[])
nrows = size(Img,1);
ncols = size(Img,2);
% image==> verteur
ab=Img;
ab = reshape(ab,nrows*ncols,1);
% K: Nombre de classes
K = 3;
[cluster_idx
cluster_center]=kmeans(ab,K);%,'distance','sqEuclidean','Replica
tes',10);

% Les classe
pixel_labels = reshape(cluster_idx,nrows,ncols);
for i=1:K
figure, imshow(pixel_labels==i),
title(['Classe',num2str(i)]);
end
```



Références

1. Stéphane Bres, Jean-Michel Jolion, Frank Lebourgeois, ‘’Traitement et analyse des images numériques’’.Hermès- Lavoisier. 2003.

2. Richard Berry, James Burnell **,** ‘’The Handbook of astronomical Image processing’’. 2nd Edition. 2006.

3. Rafael C. Gonzalez & Richard E Woods, "Digital Image Processing", Prentice Hall, 2008.

4. Radu Horaud et Olivier ,‘’Vision par ordinateur’’. Editions Hermès, 1995 – 2ème édition.

5. J.P. Cocquerez et Sylvie Philipp**,** ‘’Analyse d’images : Filtrage et segmentation’’. Elsevier-Masson.

6. Diane Lingrand , ‘’Introduction au traitement d'images’’. Vuibert 2008.

7. Gilles Burel, ‘’Introduction au traitement d'images. Simulation sous Matlab’’. Hermès - Lavoisier. 2001.

8. M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models, "International journal of computer vision”, vol. 1, pp. 321-331, 1988.

9. C. Xu and J. L. Prince, "Gradient vector flow deformable models," Handbook of Medical Imaging, pp. 159-169, 2000