Université des Antilles et de la Guyane **janvier 2001**
Pointe à Pitre

INFORMATIQUE

ANALYSE d'IMAGES

notes de cours - version 1.3

Jacky DESACHY

TABLE DES MATIERES

1. HISTORIQUE ET GENERALITES

1.1 INTRODUCTION

A titre indicatif notons que prés de 90% de l'information reçue par l'homme est visuelle. La production d'images de qualité, de même que leur traitement numérique (et si possible) automatique a donc une importance considérable.

La plupart des appareils scientifiques fournissent des images (microscopes, téléscopes, radiographies, Résonance magnéto-nucléaire, ...) et de nombreux domaines d'applications utilisent l'image comme source d'information et/ou de visualisation.

Le traitement numérique des images va mettre en oeuvre deux types d'approches principales:

- amélioration d'images pour visualisation et éventuellement interprétation « manuelle » par un expert humain.

- La vision par ordinateur qui consiste à réaliser des opérations de perception (d'interprétation) automatique (par ordinateur) de façon analogue au système de perception visuelle humain.

1.1.1 Vision humaine

Quelques indications rapides:

Une personne regardant autour d'elle peut décrire pratiquement immédiatement ce qu'elle voit.

La scène est perçue à l'aide des cellules de la rétine qui correspondent grosso modo à 1.000.000 pixels (1 pixel = 1 point image), chaque pixel étant quantifié en couleur (rouge, vert, bleu) et en intensité.

fonctionnement de la perception :

SCENE → *Connaissances enregistrées*

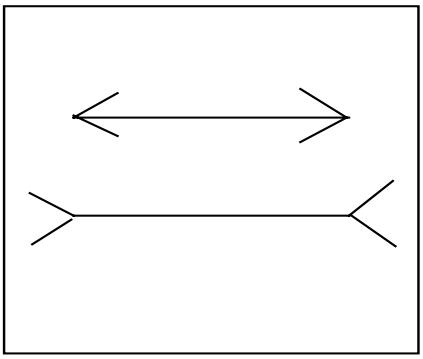
dans le cerveau

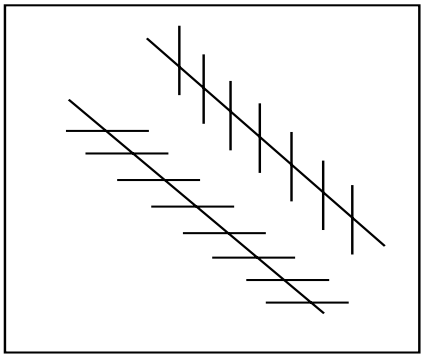
INFORMATION RETINIENNE → RECONNAISSANCE → INTERPRETATION

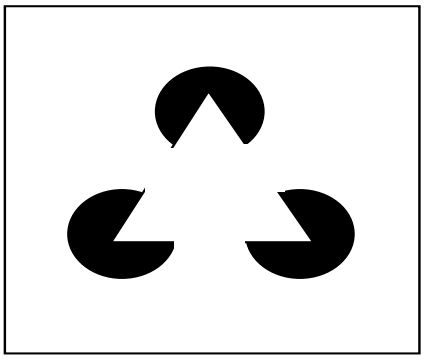
performances moyennes de l'homme:

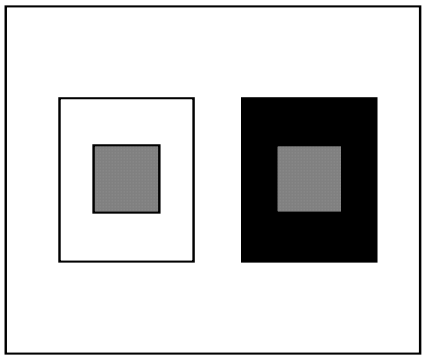
- temps d'identification d'une image : ~100 ms
- Nombre de scènes mémorisées : ~100 000

Des imperfections cependant.....

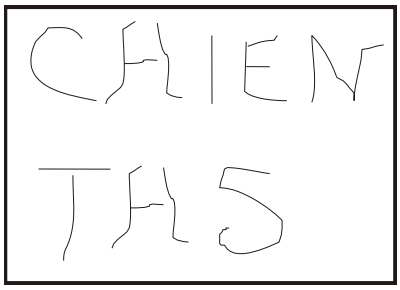








une capacité de s'adapter au contexte



1.1.2 Naissance de l'image numérique

Les premières applications des images numériques remontent aux années 1920 et ne concernent que les premiers pas en transfert d'images et en correction-amélioration d'images.

-> Transmission d'images de journaux par câble sous marin entre Londres et New York en 1921 (une image transmise en trois heures) par codage au départ et décodage à l'arrivée par une imprimante spécialisée.

Chaque niveau de gris étant codé par un caractère particulier (déclenchant une impression plus ou moins dense en surface encrée)

1921 : 5 niveaux de gris
1929 : 15 niveaux de gris

--> en 1929 premier brevet déposé sur la reconnaissance optique de caractères

---> les potentialités du traitement numérique des images digitales pour le transfert et l'amélioration des images sont apparues avec le développement des grands ordinateurs et surtout avec les nécessités des programmes de recherche spatiale:

Jet Propulsion Laboratory en 1964:
images de la lune obtenues par RANGER7 avec la correction numérique des distorsions de la caméra par ordinateur. Ces méthodes furent utilisées pour les missions suivantes.

----> Puis l'explosion des applications dans tous les domaines....

mais toujours pas de méthodologie universelle pour automatiser par ordinateur la vision humaine….d'où de nombreuses recherches dans ce domaine en pleine expansion.

1.2 VISION PAR ORDINATEUR :

Domaine d'investigation et d'applications nées à la fin des années 50

Définition:

Extraction d'informations à partir d'images représentant un monde tridimensionnel (et de séquences d'images représentant un monde à quatre dimensions (3D + temps))

Modélisation et descriptions d'objets réels à partir d'images

En final c'est l'ordinateur qui réalise le travail de perception du système de vision humaine de façon automatique.

Les grands domaines de la vision par ordinateur :

Traitement d'image consécutif à l'obtention de l'image numérique

Reconnaissance de formes et interprétation du monde 3D au travers d'images 2D (planes)

Vision tridimensionnelle (ex images stéréoscopiques) et modélisation de l'espace 3D et des objets

Infographie, synthèse d'images et réalité virtuelle

- **Texte**
    - Reconnaissance de texte
    - Lecture en ligne
- **Domaine spatial :**
    - images de télédétection satellitaires (mono, stéréo, multibandes..) et tout le problème de leur interprétation.
    - cartographie
    - environnement
    - agriculture
    - aménagement du territoire
    - détection de pollution
    - analyse de planètes.....
- **Recherche scientifique**
    - astronomie
    - océanographie
    - spectroscopie
    - analyse de trajectoires de particules
- **Domaine médical:**
    - aide au diagnostic
    - Scanner, radiographies, RMN, ... souvent des images 3D
    - Reconstruction 3D d'organes, tumeurs…

- **Automatisation des procédés industriels**

    - Soudage, découpage
    - Montage de cartes électroniques
    - Tri
    - Cueillette de fruits

- **Vision:**

    - contrôle qualité
    - inspection de soudures…
    - décision
    - aide à la navigation, déplacement (robots mobiles)
    - images stéréographiques

- **Télécommunications:**

    - compression
    - qualité image

- **applications militaires**

    - navigation automatisée
    - détection de cibles, guidage d'engins

- Amélioration, restauration d'images:

- distorsions géométriques et radiométriques
- correction des défocalisations , bougés, bruit, ...

1.3 DEFINITIONS

1.3.1 Introduction

Une image est la représentation d'une scène acquise à l'aide de systèmes de production d'images (appareils photographique, caméra, radiographes, scanner, sonar,.....).

Sa forme peut être analogique (ex: négatif, photographie, vidéo..) ou numérique (images numérisées suivant divers formats (images compressées ou non…) ou obtenues par des capteurs fournissant des images numérisées) et dans ce cas un traitement par ordinateur est possible.

SCENE	***IMAGE***
Béatrice Dalle live	*ex: photographie de Béatrice Dalle*

1.3.2 Définition d'une image

Une image peut être considérée comme une fonction I(X) définie sur un espace multidimensionnel.

- X est un vecteur de coordonnées définissant une position dans un espace multidimensionnel (ex. X = (x1, x2) dans le 2D)

- I(X) est une valeur scalaire

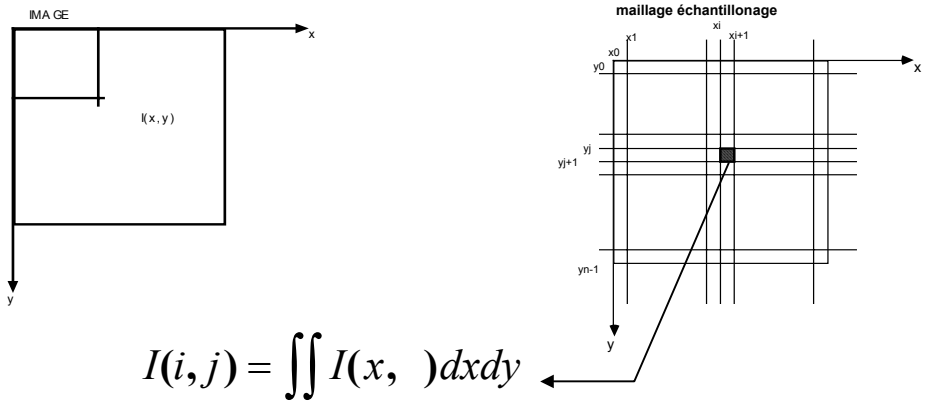
Les images les plus courantes sont définies dans un espace de dimension 2 (images 2D) : photographies N/B ou couleur, radiographies,... ou de dimension 3 (images 3D) : images médicales tomographiques par exemple,...

1.3.3 Définition d'une image numérique

Les fonctions image sont échantillonnées pour former des ensembles de points (**pixels**, PICTure Elements)

en général le maillage d'échantillonnage est rectangulaire (mais il peut être aussi triangulaire ou plus complexe).

- exemple image 2D:



$$I(i,j) = \iint I(x,\ )dxdy$$

d'où une matrice de dimension n x n :

$$\begin{matrix} I(0,0) & I(0,1) & \dots & I(0,n-1) \\ I(1,0) & I(1,1) & \dots & I(1,n-1) \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ I(n-1,0) & \dots & \dots & I(n-1,n-1) \end{matrix}$$

Cette image peut être stockée ligne par ligne par exemple, ce qui signifie que les traitements sur cette image seront réalisés si nécessaire plutôt ligne par ligne !

1.3.4 Echantillonnage et résolution

L'échantillonnage de l'image peut être plus ou moins fin (64 X 64 ou 128 X 128 ou 4092 X 4092 , ... ou 523 X 765) mais chaque pixel va représenter une certaine partie de la scène réelle, plus ou moins grande mesurée par ce qu'on appelle la résolution (spatiale) de l'image.

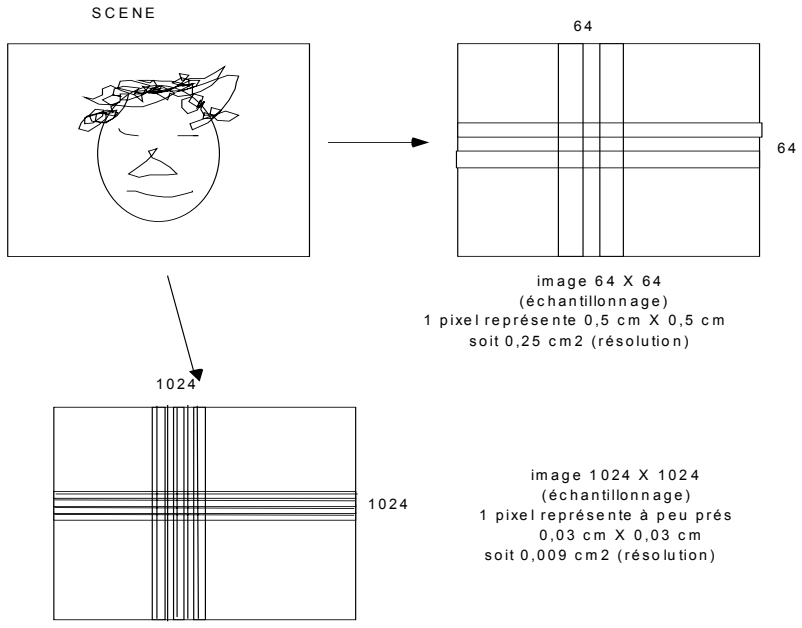


image 225 x 409 d'une image prise au microscope électronique d'un alliage métallique

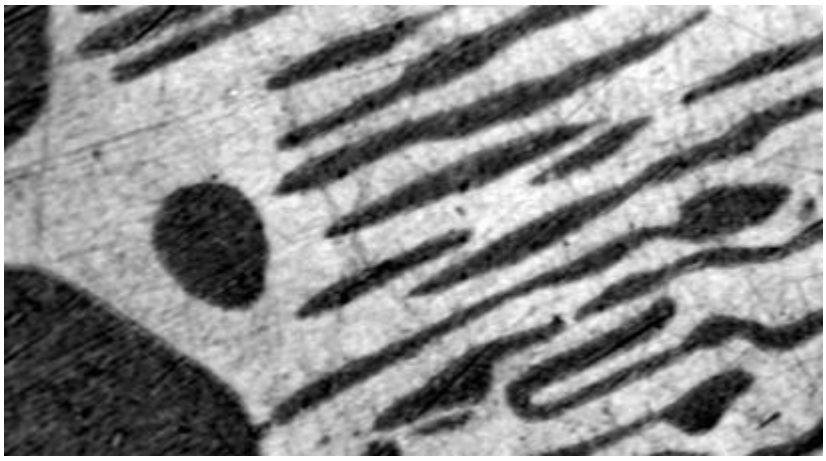


image 45 x 82 de la même scène (avec zoom X5)

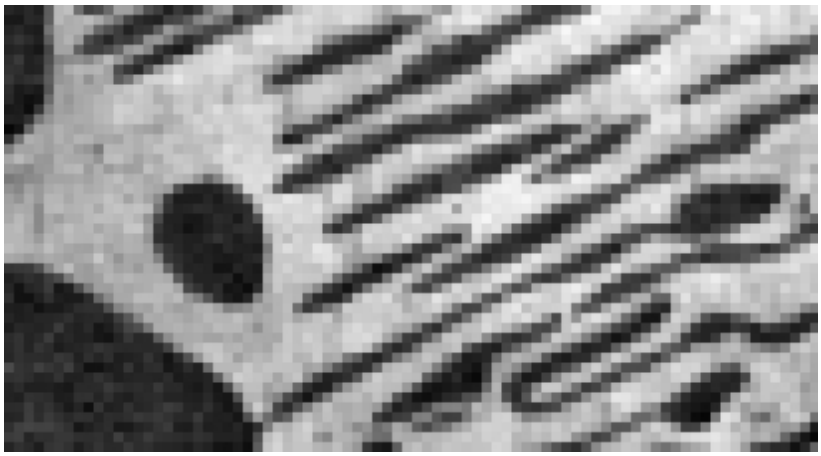
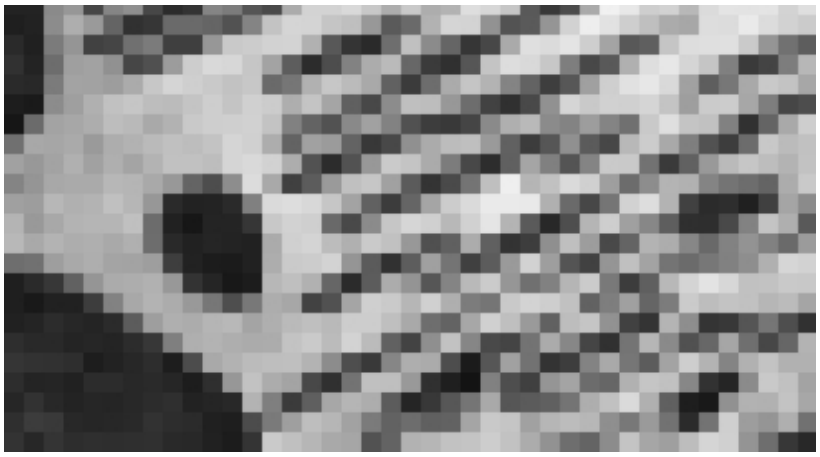


image 22 x 41 de la même scène (avec zoom X10)



1.3.5 Quantification

La fonction image est elle même discrétisée

- exemple :

pixel I(i,j)		
------> 1 bit	0,1	image binaire
------> 8 bits	0,1,2, ...255	image niveaux de gris classique
------> 16 bits	0,1,2,…65535	images d'astronomie

2 ACQUISITION, RESTITUTION ET CODAGE

2.1 GENERALITES

Deux éléments sont nécessaires pour l'acquisition d'images numériques:

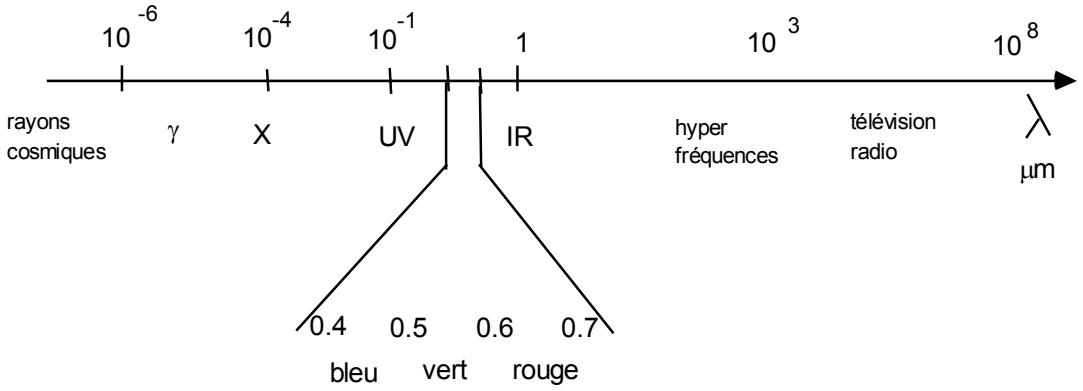
- un dispositif photosensible à une bande spectrale du spectre électromagnétique

produisant un signal.

- un système de digitalisation convertissant le signal en une valeur numérique.

Notons que ces deux éléments peuvent être regroupés en un seul.

Spectre électromagnétique en fonction de la longueur d'onde :

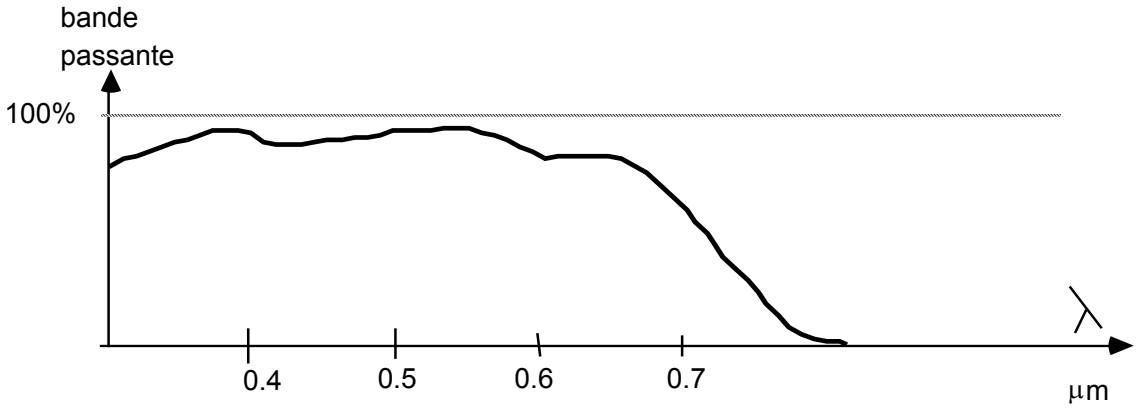


exemple 1 :

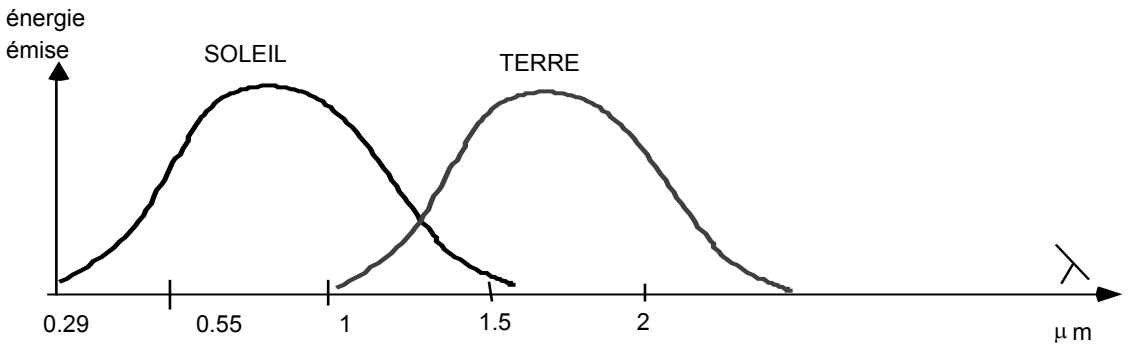
image infrarouge thermique entre 2 et 9 μmètres

exemple 2 :

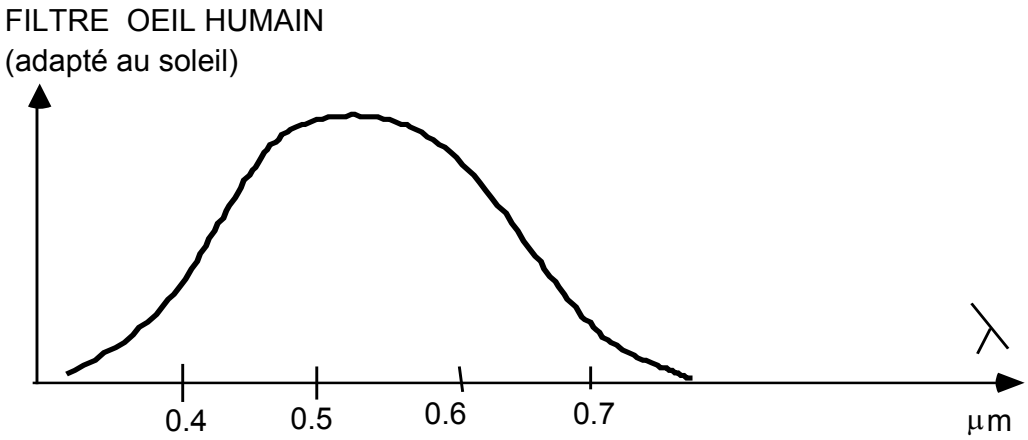
appareil photographique avec une pellicule N/B panchromatique



exemple 3:



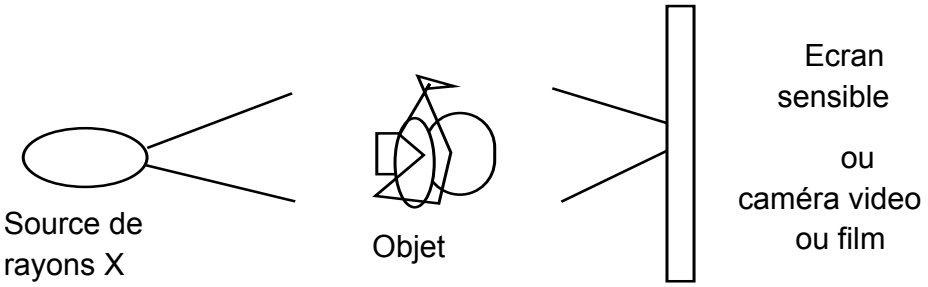
exemple 4 :



2.2 ACQUISITION, CODAGE

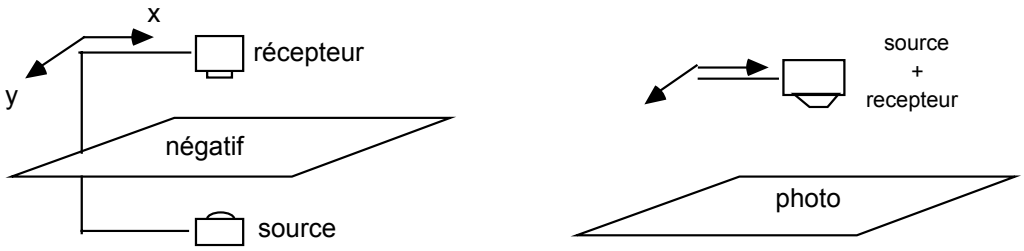
Voici quelques exemples d'acquisition et de codage d'images numériques

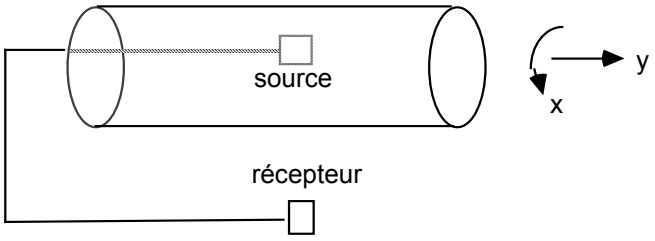
Rayons X :



Visible, Infrarouge :

Scène --> appareil photo --> film --> scanner, microdensitomètre





trés lents mais possibilité de trés haute résolution

SCENE --> CAMERA VIDEO --> convertisseur A/D --> image numérique

La caméra de télévision classique où l'image de la scène est focalisée sur une cible photosensible. Cette cible est « scannée » ligne par ligne par un faisceau d'électrons . Une

intensité électrique proportionnelle à l'intensité lumineuse en chaque point est ensuite traitée pour fournir un signal vidéo

(inconvénients : nombre limité de lignes (625) , distorsions , sortie non linéaire par rapport aux intensités lumineuses réelles, effet de rémanence entre trames)

SCENE --> CAMERA CCD --> image numérique

Caméra CCD (**C**harge **C**oupled **D**evices) :

- Caméra CCD ligne

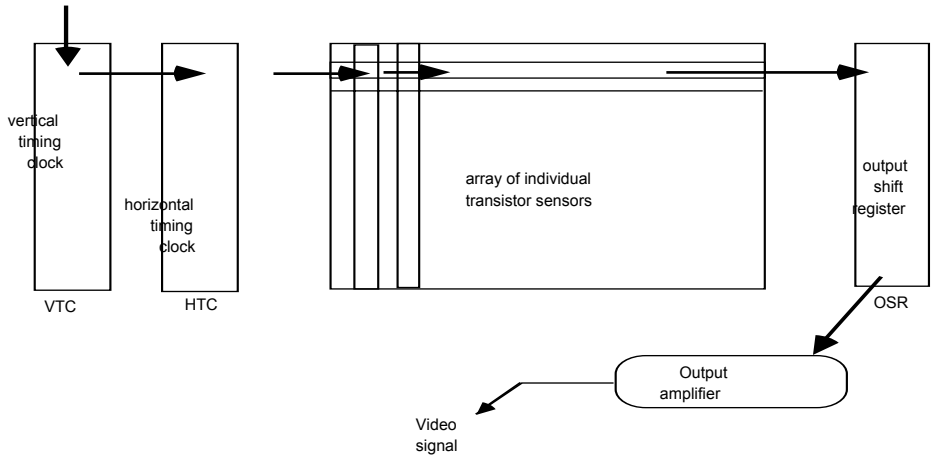
  Une ligne de capteurs photosensibles et l'image produite est un ensemble de lignes successives
  (une ligne = de 256 à 4096 capteurs)

- Caméra CCD matricielle

  Matrice de capteurs de 32 X 32 à 2048 X 2048
  (~ 40 ms par image)

(intérêt: distorsions géométriques plus faible que les caméras vidéo et sortie vidéo quasi linéaire par rapport aux intensités lumineuses sur la scène)

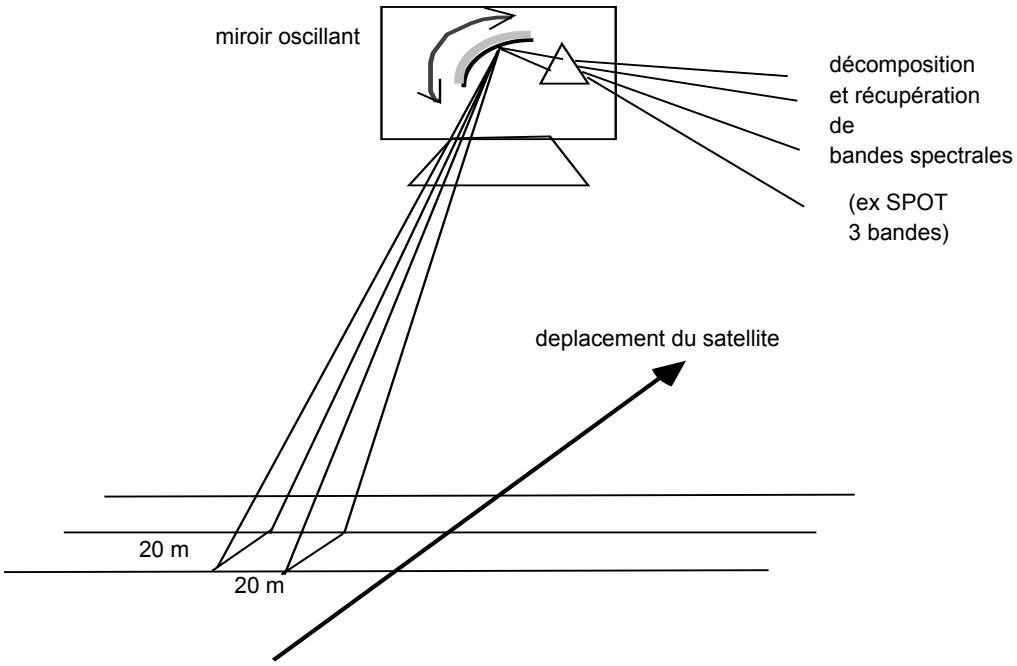
fonctionnement d'une caméra CCD:



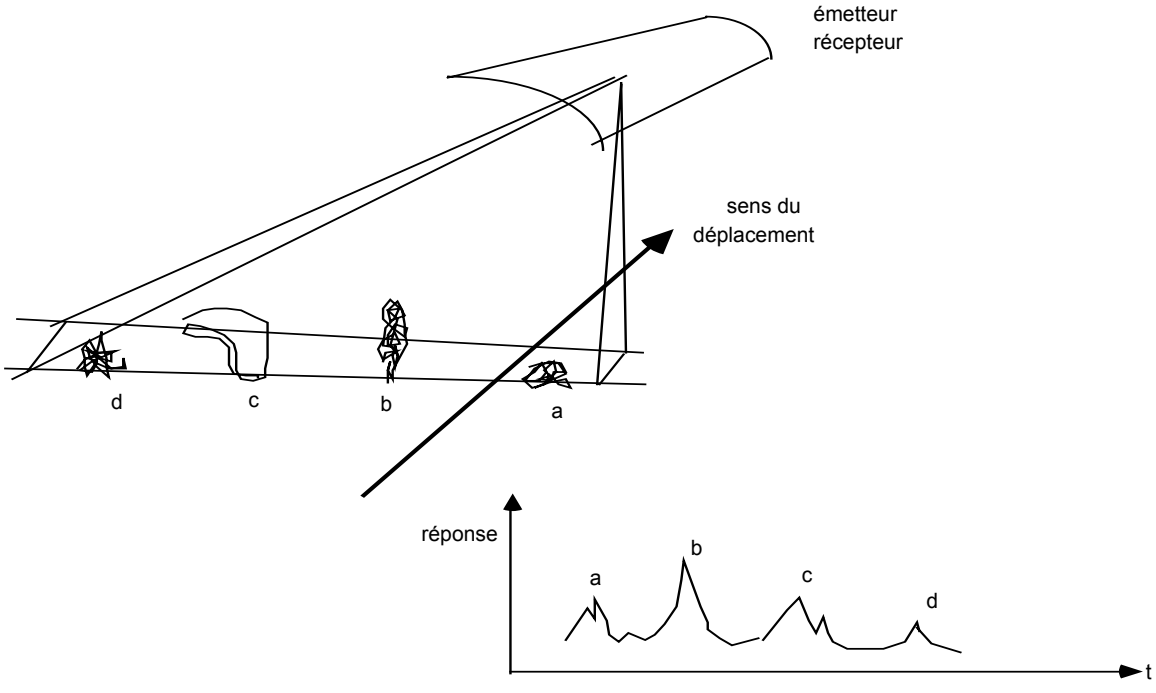
Le VTC choisit chaque ligne de détecteurs consécutivement, puis HTC décale le contenu de la ligne de détecteurs dans l'OSR qui sera amplifié en tension comme une fonction du temps.

Caméra multispectrale à balayage:

Par exemple celles des satellites de télédétection (Landsat, SPOT, ..)



Images hyperfréquences (radar):



2.3 DIFFERENTS TYPES D'IMAGES

2.3.1 Images Noir et blanc (monochromes)

Ces images sont dites à niveaux de gris , car on ne prend pas en compte ici la couleur mais seulement l'intensité lumineuse (l'exemple classique correspond aux photographies noir et blanc).

Parmi ces images on peut trouver:

images binaires

où chaque pixel est représenté par un bit (0/1)
avec en général (0 ---> noir , intensité nulle et 1 ---> blanc , intensité maximale).

Notons que la plupart des systèmes de traitement d'images placeront chaque pixel dans un octet (code 0 ou 255 (pour coder le 1 de l'image binaire)) pour des facilités d'accès et d'écriture des algorithmes.

images en niveaux de gris

Dans ce cas on dispose d'une échelle de teintes de gris , et la plupart du temps on dispose de 256 niveaux de gris avec:

0 ----> noir , ...127 ---> gris moyen , ..., 255 ---> blanc

ceci est commode car l 'unité d'information est l'octet (unsigned char en langage C).

Certaines images peuvent être codées sur deux octets ou plus (certaines images médicales , des images astronomiques,...) ce qui peut poser des problèmes dans la mesure où les systèmes de traitement d'images courants supposent utiliser des pixels d'un octet !

Notons au passage que l'humain standard ne reconnaît au plus que 64 niveaux de gris

2.3.2 images couleur

Ces images sont en général codées en utilisant le codage des trois couleurs fondamentales (rouge , vert , bleu) , on parle alors d'images RVB.

(cela correspond au codage des téléviseurs couleur Français)
Chaque couleur est codée sous forme d'un octet, d'où

⟶ image RVB
composante Rouge , intensité de 0 à 255
+ composante Verte , intensité de 0 à 255
+ composante Bleue , intensité de 0 à 255

on code ainsi 2^{24}=16 777 216 couleurs différentes (Notons que l'homo standardus ne reconnaît pas plus de

350 000 couleurs)

La couleur peut aussi être codée sur un octet (256 couleurs) et l'affichage étant réalisé après passage dans une table de couleurs (transcodage).

Le codage peut ainsi être réalisé en affectant 3 bits au rouge et au vert et 2 bits au bleu (pour tenir compte par exemple de la plus faible sensibilité de la vision humaine au bleu).

Il existe d'autres techniques de représentation de la couleur pour les images (on passe d'un espace 3D , l'espace RVB , à un autre espace 3D défini par une autre base)

Ainsi le standard T.V. US **NTSC** utilise un autre codage appelé YIQ.

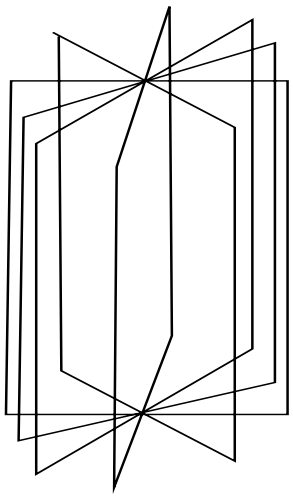
2.3.3 Images 3D

Des images 3D sont des images qui représentent une scène en trois dimensions . Le « pixel » est alors appelé un **voxel** , et représente un volume élémentaire.

Ces images peuvent évidemment être d'un des deux types définis précédemment (N/B ou couleur).

Des exemples d'images de ce type se rencontrent dans les images médicales.

Les images tomographiques axiales sont ainsi des images construites à partir de plusieurs radiographies faites sous des angles de vue différents



autres exemples : les images scanner , les images de résonance magnétique...

Toutes ces images pouvant être éventuellement « fusionnées » pour former des images plus complexes (les rayons X fournissant l'ossature, le scanner les tissus, et la RMN décrivant les fonctions physiologiques par exemple).

Cas particuliers

Les images obtenues par un laser tournant pour obtenir une partie de l'information de forme des objets d'une scène.

Images stéréographiques:

On dispose alors d'une paire d'images (N/B ou couleur) prises sous des points de vue différents. A partir de telles images il est possible d'obtenir de l'information sur la scène 3D.

2.4 MEMORISATION DES IMAGES NUMERIQUES

- Image 1024 X 1024 codée sur un octet

---> 1 Moctets

- Image SPOT XS (multispectrale)

3 bandes 5000 X 5000

---> 75 Moctets

Trois méthodes de mémorisation :

--> Mémoire utilisée pour les traitements numériques:

* Mémoire centrale
* Mémoires d'images

En général elles servent de mémoires de rafraîchissement d'écran vidéo (accès à 30 millisecondes au moins)
Certaines des cartes contenant ces mémoires d'image permettent certains traitements (zooming, scrolling (déplacements verticaux), panning (déplacements horizontaux) tailles courantes 1024 X 1024 X 8 X 4 bits

--> Mémoires en ligne:

disques magnétiques, magnéto-optiques, ...

--> Mémoires d'archivage:

bandes magnétiques, CDrom, ...

3 CONNEXITE DANS LES IMAGES

Le problème que nous voulons aborder ici concerne les relations entre pixels dans les images numériques et ceci pour préparer l'étude des divers algorithmes de traitement d'images impliquant des opérations entre pixels voisins.

Notamment la notion de connexité entre pixels est une notion particulièrement importante pour la détection des frontières d'objets dans une image et des pixels composant un objet :

Deux pixels seront considérés comme connexes (appartenant au même objet donc) s'ils satisfont deux critères:

- d'une part un critère de similarité (par exemple même niveau de gris)

- s'ils sont adjacents (voisins)

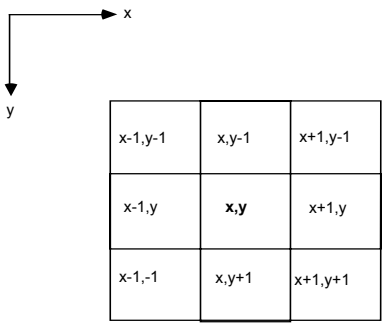
3.1 VOISINAGE D'UN PIXEL

Un pixel I(x,y) possède quatre voisins horizontaux et verticaux qui forment ce qu'on appelle le 4-voisinage de I(x,y) . Si on considère un pixel comme un carré élémentaire , les pixels présentant un coté commun avec I(x,y) sont appelés les 4-voisins de (x,y).

le pixel I(x,y) possède aussi quatre voisins diagonaux, ce sont les pixels qui ont un sommet commun avec (x,y).

L'ensemble des huit voisins du pixel (x,y) représentent ce qu'on appelle les 8-voisins (8-voisinage). Ces huit voisins forment la fenêtre 3x3 du pixel I(x,y) .

On dit aussi que ces pixels sont 8-adjacents de (x,y)



3.2 CONNEXITE

La connexité est une propriété de liaison entre deux pixels qui fait qu'on les considère comme faisant partie de la même région dans une image.

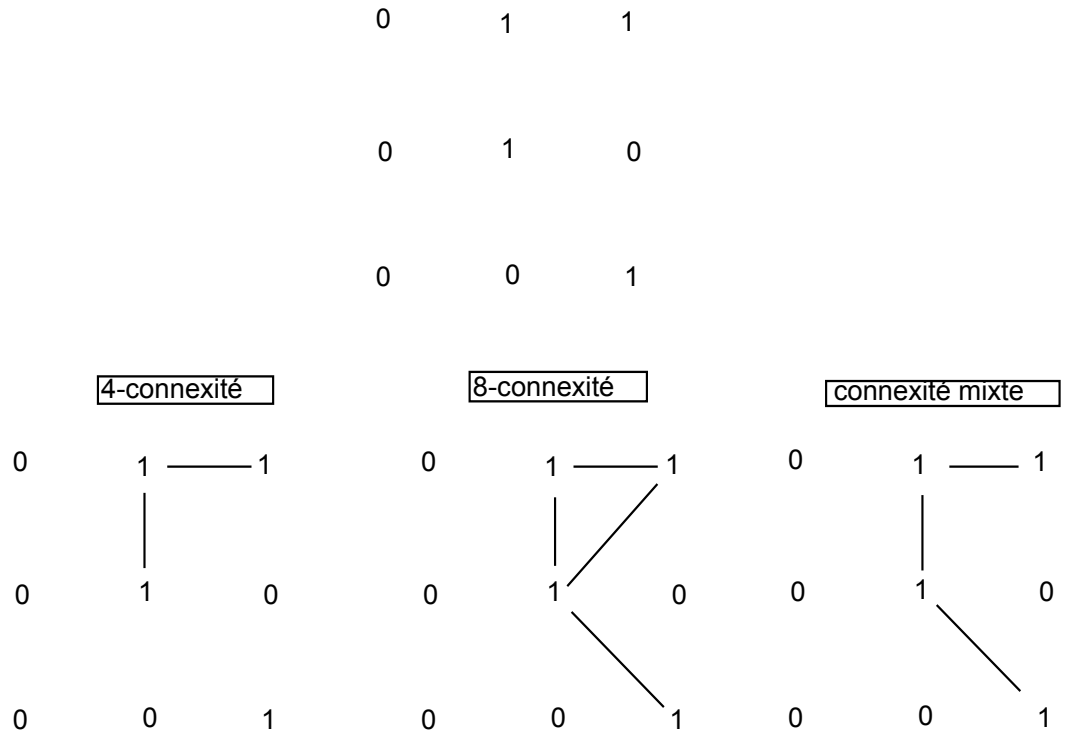
En supposant que deux pixels P et Q vérifient déjà un certain critère de similarité, on peut définir différent types de connexités:

4-connexité : Les deux pixels sont tels que Q est un des 4-voisins de P

8-connexité : Les deux pixels sont tels que Q est un des 8-voisins de P

connexité mixte : ou bien P et Q sont 4-voisins
ou bien P et Q sont voisins diagonaux et aucun des 4-voisins communs à P et Q ne sont 4-connexes

exemple:



On remarque que la « méthode » de connexité mixte consiste à appliquer en priorité la 4-connexité , et la 8-connexité là où la 4-connexité ne “marche“ pas.

3.3 CHAINAGE DE PIXELS

La notion de connexité permet de relier des pixels adjacents (vérifiant le même critère de similitude). Il existe un chaînage de longueur n entre deux pixels P et Q si on peut trouver une suite de (n+1) pixels P0=P, P1, P2, ... , Pn=Q telle que deux pixels consécutifs sont toujours connexes.

On parlera de 4-chainage ou de 8-chainage suivant le type de connexité choisie.

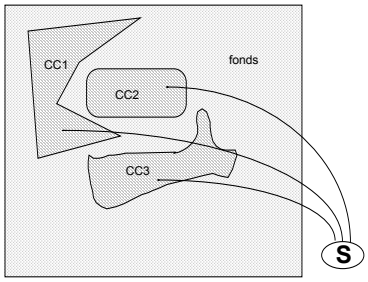
Les deux pixels P et Q sont alors dits connexes et la relation de connexité qui les lient est une relation d’équivalence qui a la propriété d’ être réflexive, symétrique et transitive (notons que la connexité mixte n’est pas transitive).

3.4 REGION CONNEXE, COMPOSANTE CONNEXE

Un sous ensemble R de pixels (satisfaisant le même critère de similarité) d’une image est une région connexe si quelque que soit un couple de pixels P et Q de ce sous ensemble , il existe au moins un chaînage entre ces pixels formé de pixels de R .

Soit S un sous ensemble de pixels d’une image (ces pixels satisfaisant le même critère de similarité) , alors quelque soit un pixel P de S , l’ensemble des pixels de S connectés par chaînage à P forment une composante connexe (région connexe) de S.

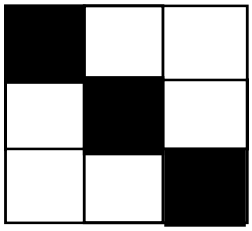
On en déduit que deux composantes connexes distinctes sont disjointes.



sous ensemble S , formé de trois composantes connexes

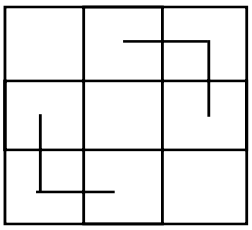
3.5 FONDS, FORME ET CONNEXITE

Lorsque qu'on examine une image considérée comme un ensemble de composantes connexes (la **forme** : rassemblant tous les pixels satisfaisant le même critère de similarité) sur un **fonds** (rassemblant les autres pixels) comme dans la figure précédente on imagine aisément que les composantes connexes considérés seront différentes suivant que l'on utilise la 4-connexité , la 8-connexité où la connexité mixte.
Exemple: soit la portion d'image ou le fonds est en noir (0) et la forme en blanc (1)

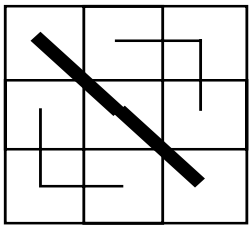


0 1 1
1 0 1
1 1 0

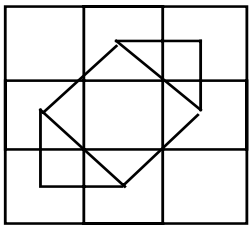
4-connexité pour le fonds et la forme



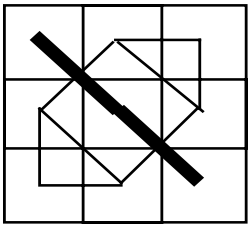
4-connexité pour la forme
8-connexité pour le fonds



8-connexité pour la forme
4-connexité pour le fonds



8-connexité pour le forme
8-connexité pour le fonds



Ceci explique que la recherche des régions connexes est souvent réalisée en prenant les conventions suivantes:

--> 4-connexité pour la forme et 8-connexité pour le fonds

ou bien --> 8-connexité pour la forme et 4-connexité pour le fonds.

3.6 ETIQUETAGE DE COMPOSANTES CONNEXES

C'est l'opération qui consiste dans une image binaire (forme/fonds) à trouver toutes les classes d'équivalence de la forme , c'est à dire les différentes composantes connexes et à les étiqueter.

L'étiquetage consiste à affecter une étiquette identique (par exemple un numéro) à tous les pixels d'une même composante connexe.

exemple d'algorithme général :

1) scanner les pixels de l'image ligne par ligne et ne considérer que ceux de la forme (critère de similarité, pixel à 1,...).

Si le pixel n'a pas de voisin connexe déjà étiqueté, créer une nouvelle étiquette et l'affecter à ce pixel.

Si le pixel a exactement un seul pixel connexe étiqueté, lui affecter cette étiquette.

Si le pixel a plus d'un pixel connexe (avec des étiquettes différentes) lui affecter une de ces étiquettes et mémoriser que toutes ces étiquettes sont équivalentes.

2) Refaire une passe sur l'image en groupant les étiquettes équivalentes sur une seule étiquette

(Voir Gonzalez et Woods pp 42 et 43 pour le détail de la méthode en 4-connexité et 8-connexité)

3.7 DISTANCES ENTRE PIXELS

Il existe différentes mesures de distances entre deux pixels (x1,y1) et (x2,y2)

Distance Euclidienne : $\sqrt{\left((x1-x2)^2+(y1-y2)^2\right)}$

Distance « City block » : $|x1 - x2| + |y1 - y2|$

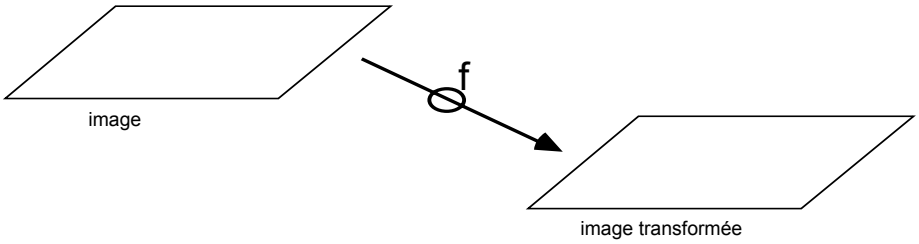
Distance « chessboard » : $\max(|x1 - x2|, |y1 - y2|)$

Les deux premiéres distances indiquées sont respectivement l'expression des normes vectorielles $L1, L2, L\infty$. Soit un vecteur (v1,v2, ... ,vn) alors

$$L_p(v) = \|v\|_p = \left(\sum_{i=1}^{n} |vi|^p \right)^{1/p}$$

4 TRANSFORMATIONS PONCTUELLES D'IMAGES

Nous allons décrire ici des méthodes de traitement d'images qui vont consister à transformer ces images de façon à améliorer leur visualisation pour une interprétation manuelle , à mettre en évidence des détails , des structures ou bien à réaliser un prétraitement pour faciliter l'application d'un algorithme de traitement ultérieur. Le résultat est une nouvelle image.



4.1 DIFFERENTS TYPES DE TRANSFORMATIONS D'IMAGES

Il existe différents types de transformations d'images :

- transformations ponctuelles

En chaque point on applique une formule de modification du niveau de gris (ou de la couleur) indépendamment de la localisation géométrique de ce pixel

J=f(i) i ancien niveau de gris, j nouveau niveau de gris

on a donc une table de transcodage appelée LUT (**L**ook **U**p **T**able)

- transformations locales

le nouveau niveau de gris en un point est fonction de l'ancien niveau de gris et de la localisation du pixel

j=f(i,x,y)

j(x,y) peut être plus simplement une fonction des niveaux de gris des pixels voisins de (x,y)

- transformations globales

On considère alors l'image comme un nuage de points dans un espace donné et on réalise un changement de base (ex transformée de Fourier...)

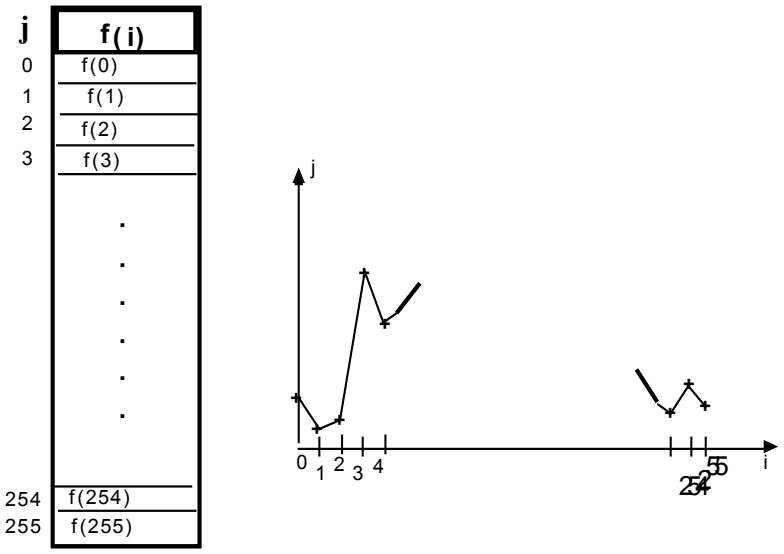
4.2 TRANSFORMATIONS PONCTUELLES

Ces transformations s'appliquent pixel par pixel et ne font intervenir que la valeur du niveau de gris

La transformation est donc définie par une fonction f telle que pour tout niveau de gris i on obtient un nouveau niveau de gris j=f(i).

Cette fonction est donc définie sur l'ensemble des valeurs de niveaux de gris possibles... On peut donc représenter cette fonction par une tabulation.

On appelle cette table une LUT (Look Up Table).



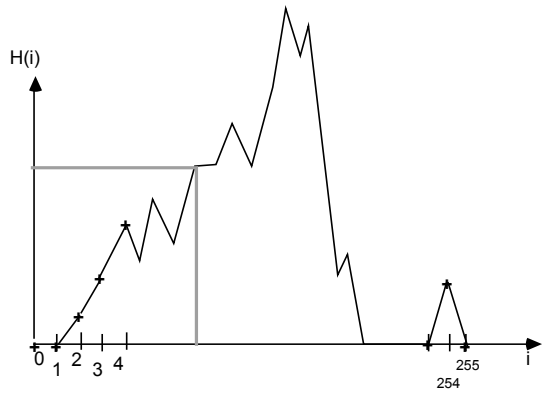
Cette LUT va définir une modification sur la répartition des niveaux de gris dans la nouvelle image. De ce fait on débute généralement l'étude d'une image par le calcul de la répartition statistique des niveaux de gris de l'image initiale, c'est le calcul de l'histogramme des niveaux de gris.

4.2.1 Histogramme d'image

Pour obtenir un histogramme il faut calculer le nombre de chacun des niveaux de gris présents dans l'image. Si on travaille sur une image en niveaux de gris avec 256 niveaux alors on obtient 256 nombres entiers

i ------> H(i) i=0....255

H(i) : Nombre d'occurrences du niveau de gris i dans l'image



L'étude d'une image numérique débute le plus souvent par le calcul et l'analyse de son histogramme. On obtient ainsi des informations sur :

- la dynamique réelle de l'image (nombre de niveaux de gris réellement utilisés) et donc le contraste disponible....

- Le caractère aléatoire ou non de la distribution des niveaux de gris et la similitude de l'histogramme (ou de morceaux) avec une distribution statistique connue (par exemple une gaussienne).

- la présence de pics significatifs

- .....

Exemple:

Inverser une image à 256 niveaux (prendre le négatif)

alors f(i)=255-i

o ----- > 255
1 -----> 254
2 -----> 253
.....
254 ----> 1
255 ----> 0

image initiale						image finale					
0	5	120	121	3	0	255	250	135	134	252	255
0	0	100	99	88	2	255	255	155	156	167	253
0	1	87	90	30	0	255	254	168	165	225	255
0	0	50	73	80	2	255	255	205	182	175	253

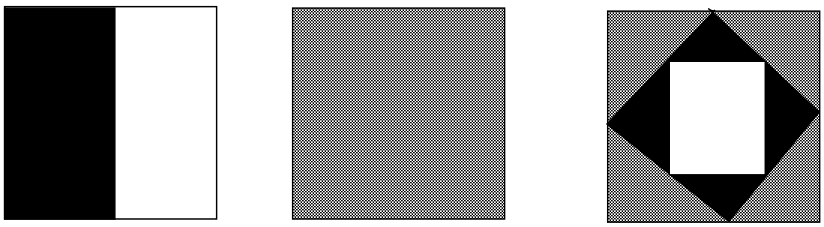
0 0 7 32 80 2 255 255 248 223 252 255

Tous les paramètres statistiques sur les niveaux de gris peuvent être calculés à partir de l'histogramme

$$\text{moyenne :} \quad \mu = \frac{1}{256}\sum_{i=0}^{255} H(i)$$

$$\text{variance :} \quad \sigma^2 = \frac{1}{256}\sum_{i=0}^{255} H(i)\times(i-\mu)^2$$

Notons que les trois images ci dessous ont même histogramme:



Ceci provient du fait que l'histogramme ne donne aucune indication sur la répartition dans l'espace des niveaux de gris (ce que nous appellerons les **fréquences spatiales** dans l'image)

4.2.2 Histogramme cumulé

Pour certaine transformations il est intéressant de calculer ce qu'on appelle l'histogramme cumulé. C'est une fonction du niveau de gris i telle que :

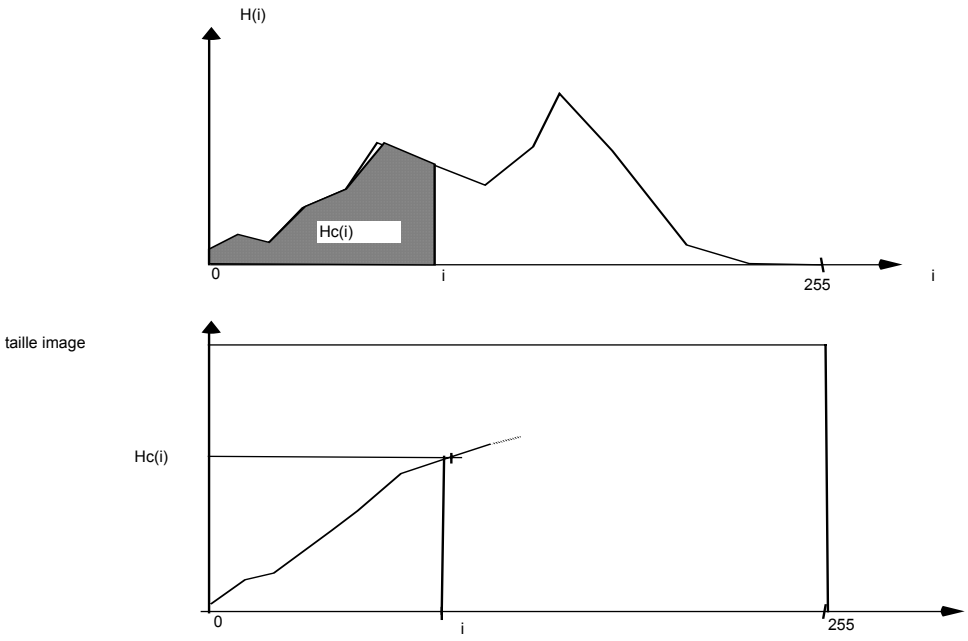
i ---> $Hc(i) = \sum_{k=0}^{i} H(k)$ avec H(k) valeur de l'histogramme

Hc(0)=H(0) et Hc(i)=Hc(i-1)+H(i) i>0

Hc(255)= NP nombre total de pixels de l'image

Réciproquement on peut passer de Hc(i) à H(i) par la formule suivante:

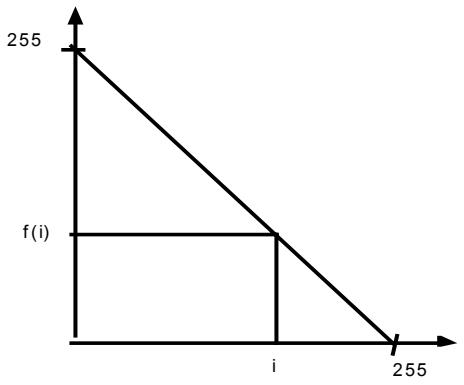
H(0)=Hc(0) ..... H(i)=Hc(i)-Hc(i-1)



4.2.3 Quelques exemples de transformations ponctuelles

LUTs i -----> f(i)

4.2.3.1 Inversion d'image



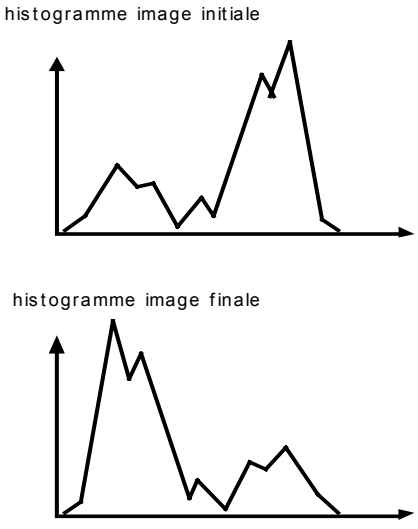


image initiale

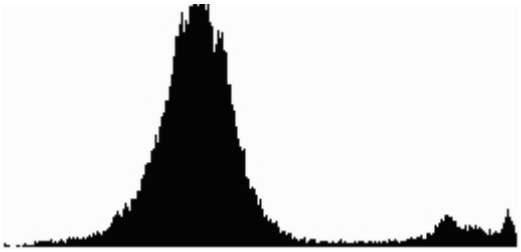
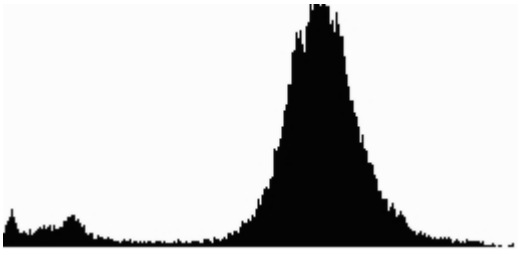
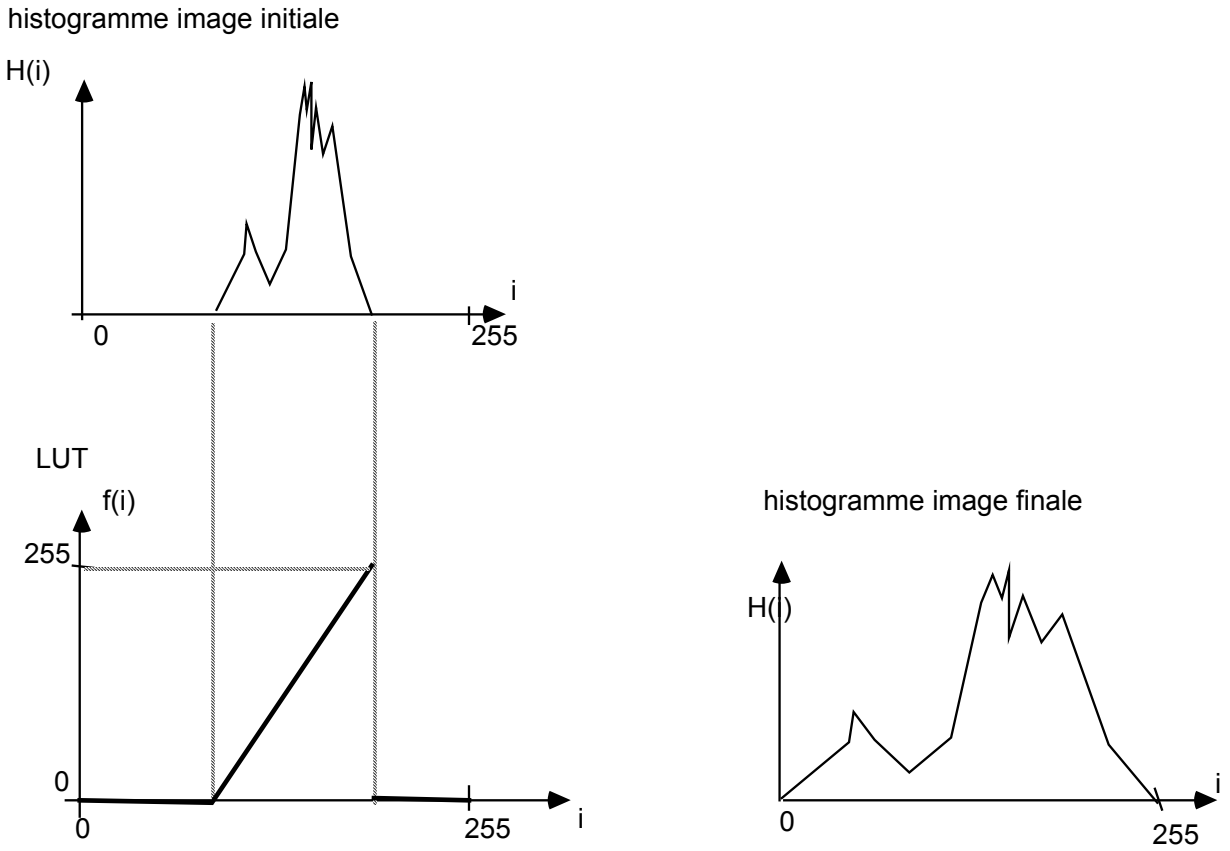


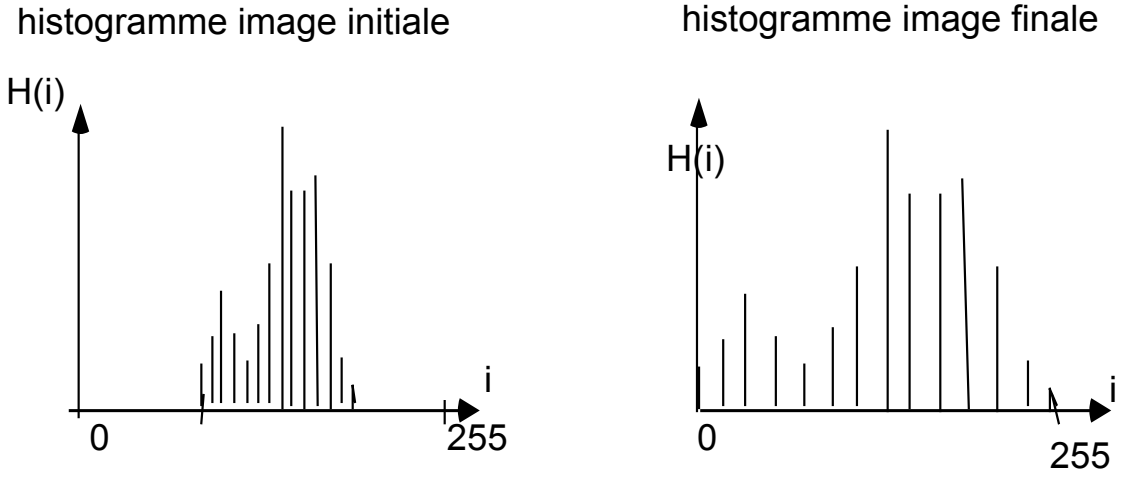
image inversée



4.2.3.2 Etalement de la dynamique des niveaux de gris



L'image obtenue sera alors beaucoup plus contrastée.....en fait on obtiendra:



4.2.3.3 Recadrage de dynamique

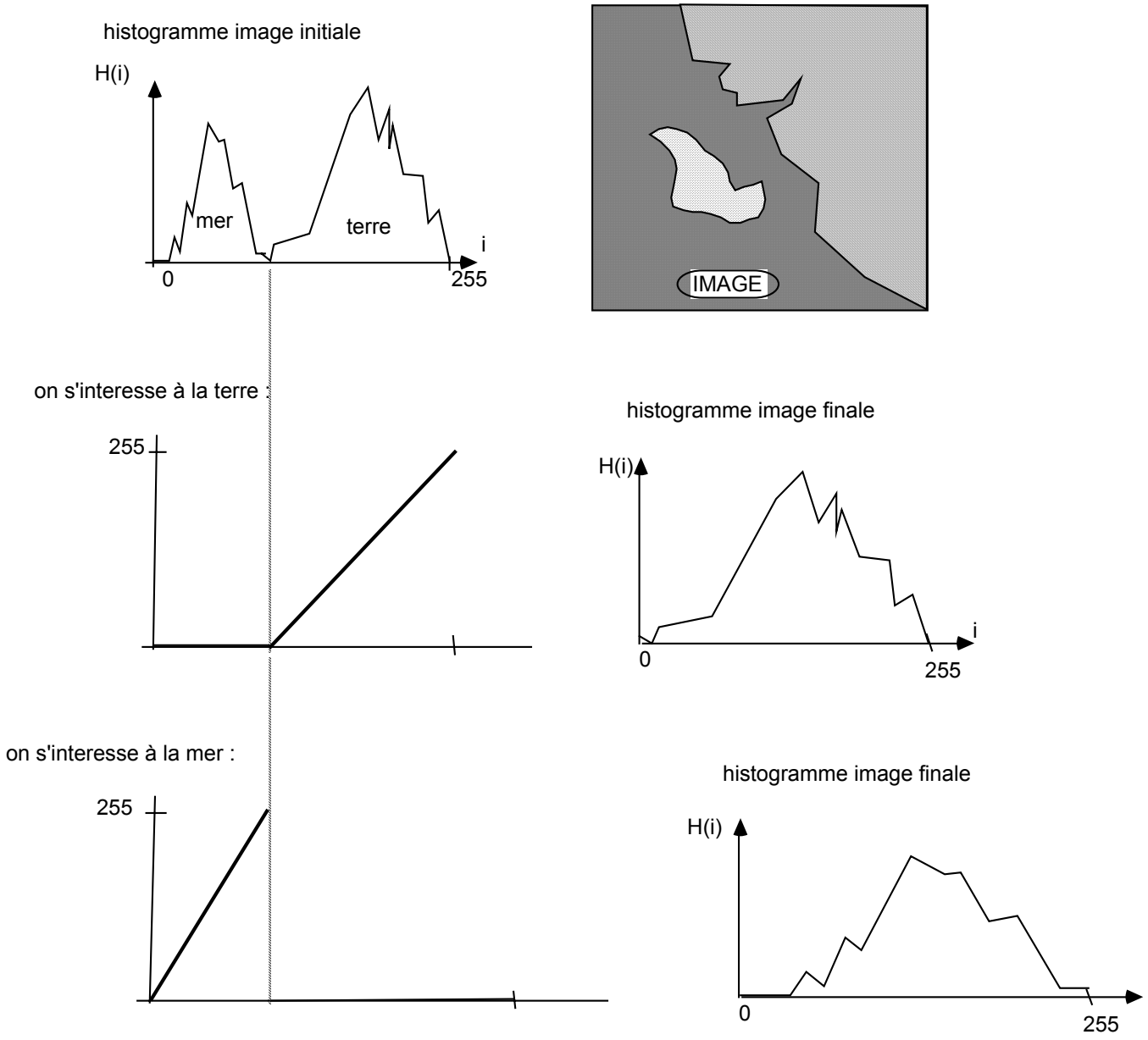


image initiale

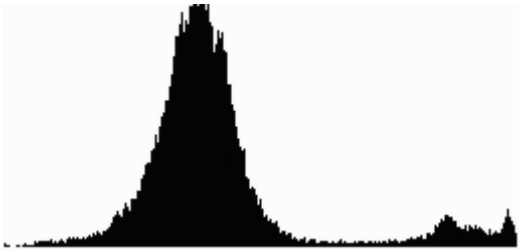
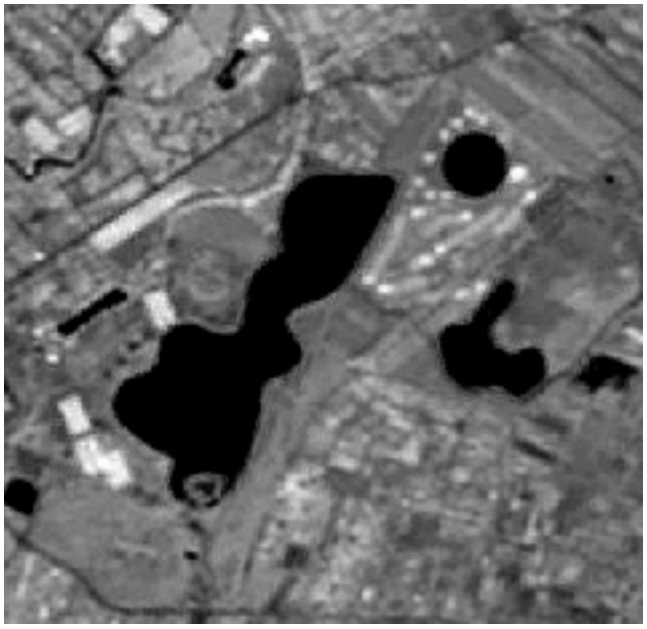


image de la "terre"



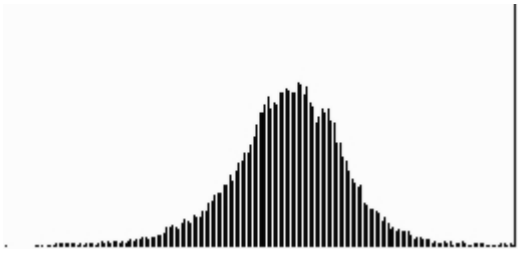
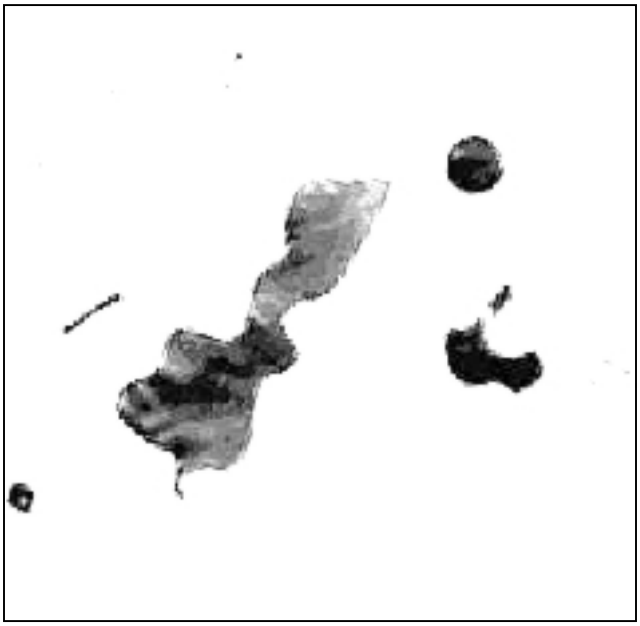
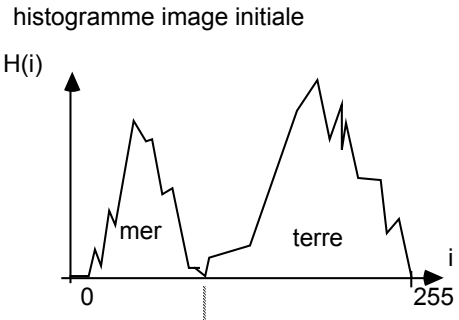
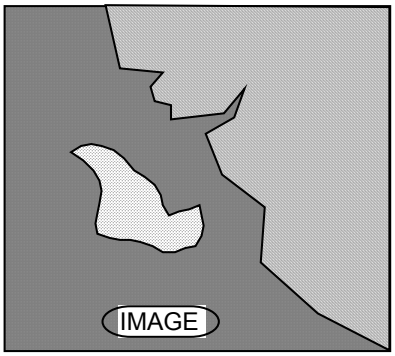
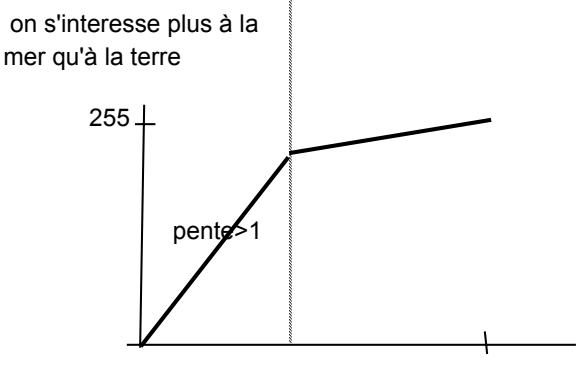


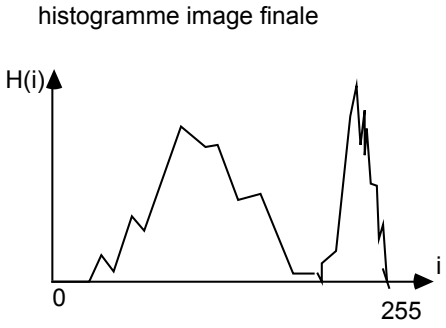
image de l'eau"





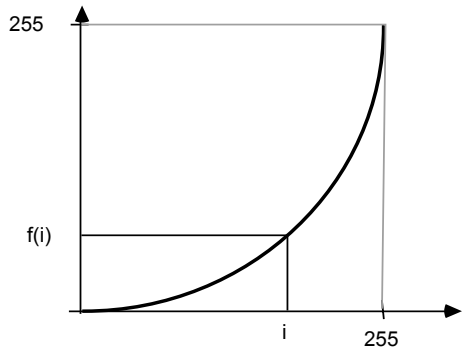






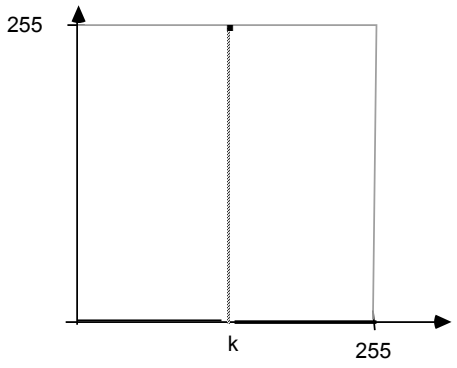
4.2.3.4 Amélioration visuelle

l'oeil répond linéairement aux intensités lumineuses et supposons disposer d'une image numérique en densité, alors comme Densité=log (intensité) on peut appliquer un LUT exponentielle.

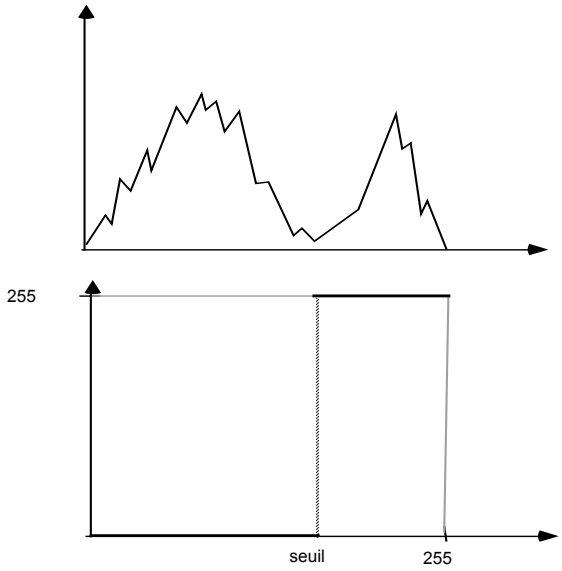


4.2.3.5 Mise en évidence des structures

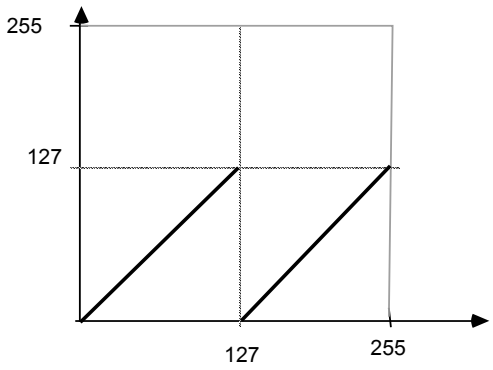
- isodensité



- binarisation



- Suppression du bit de fort poids



4.2.3.6 Egalisation d'histogramme

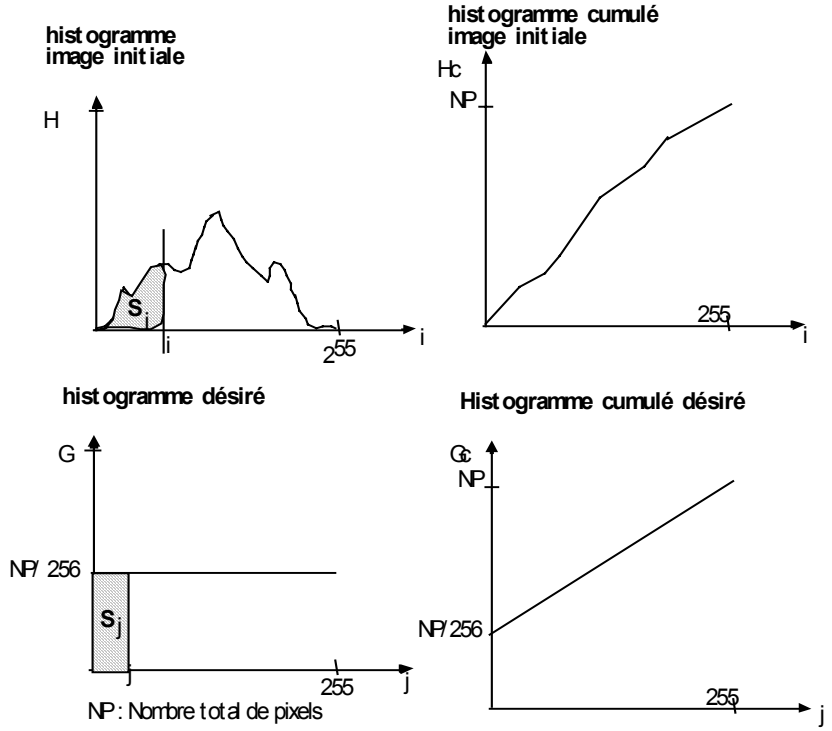
La transformation ponctuelle va consister à obtenir une image où les niveaux de gris sont répartis de façon la plus égalitaire possible entre tous les niveaux possibles. Il est clair qu'on ne pourra pas obtenir un histogramme totalement égalisé car le principe de la transformation reste du type j=f(i).

Notons que dans cette transformation certains niveaux de gris peuvent être regroupés (soit il existe i1 et i2 tels que f(i1)=f(i2))

Pour fixer les idées on supposera que le nombre de niveaux de gris utilisés est de 256 et que la taille de l'image traitée est de NP.

Les raisonnements seront faits en supposant que les histogrammes sont des fonctions continues.

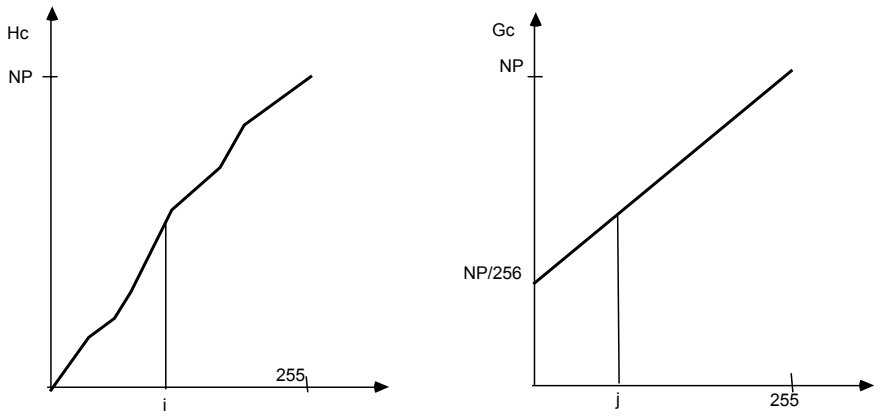
On a donc la situation suivante:



On cherche donc en fait une fonction de transformation (« LUT »)
qui à un niveau i fait correspondre un niveau j : j=F(i) et qui produise une image dont l'histogramme cumulé soit celui défini par Gc (en fait tel que $S_j=S_i$).

soit en fait Gc(j)=(NP/256)x(j+1)

Pour une valeur de i donnée on cherche le j tel que Gc(j)=Hc(i)



Soit donc

Gc(j)= Hc(i) soit (NP/256) x (j+1) = Hc(i)

Cette égalité n'étant définie que pour des valeurs de i telles que Hc(i) >= (NP/256) et ceci dans la mesure où j doit être une valeur comprise entre 0 et 255.

Par exemple j=O correspond en fait à Hc(i1)=NP/256 et donc pour toutes les valeurs de i entre o et i1 on prend j=F(i)=0

d'où,

la fonction F cherchée vérifie :

(NP/256) x (F(i)+1) = Hc(i) avec l'hypothèse précédente

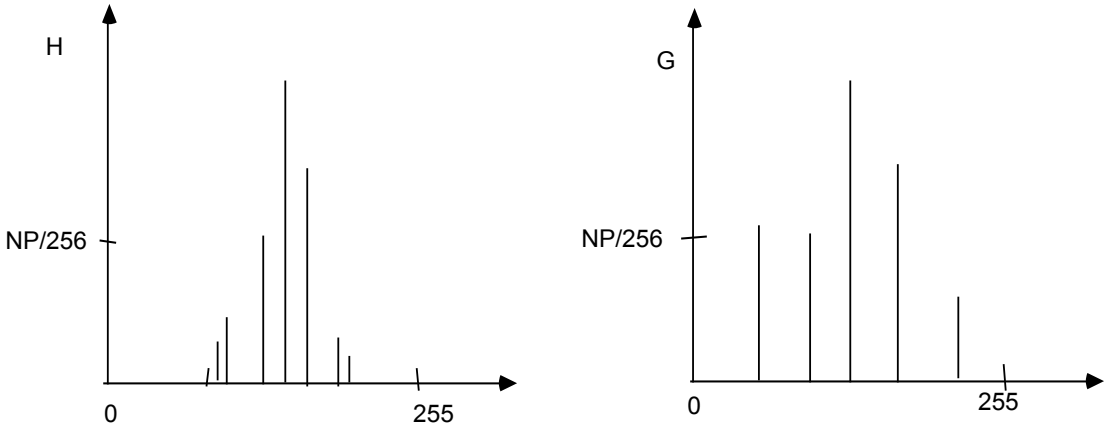
Soit

F(i) = (256/NP) X Hc(i) - 1 avec i > i1

pour le cas discret on obtient

F(i) = **E**[(256/NP) x Hc(i) - 1] (avec **E** : partie entière de...)

Notons que l'égalisation d'histogramme consiste à regrouper des niveaux de gris de valeur voisine pour former un nouveau niveau de gris en quantité voisine de NP/256 et d'autre part ces nouveaux niveaux de gris sont « harmonieusement » répartis sur 0...255



Notons aussi que l'image est améliorée pour la visualisation mais en réalité on a en général perdu de l'information (car on ne peut définir de transformation inverse puisque des niveaux de gris peuvent avoir été regroupés sur un seul).

L’image finale obtenue est en apparence (visualisation) plus informative peut-être pour l’expert qui regarde l’image alors que l’on a objectivement perdu de l’information.

Image initiale

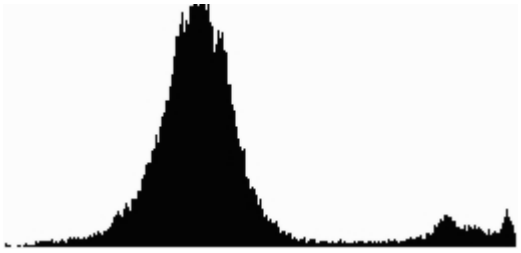
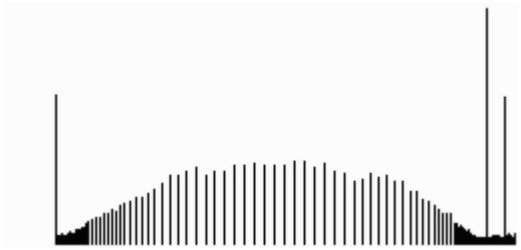


image égalisée



4.2.3.7 Obtention d'un histogramme à forme prédéfinie

De façon analogue à l'égalisation d'histogramme il est possible de trouver une transformation ponctuelle permettant d'approcher un histogramme de forme prédéfinie.

Le raisonnement est tout à fait identique à celui de l'égalisation d'histogramme.

5 DETECTION DE CONTOURS

Un contour peut être considéré comme une frontière entre deux régions différentes (suivant un critère de similarité donné , par exemple un niveau de gris identique à un epsilon prés ou une couleur identique...).

La détection d'un contour est généralement basée sur la détection du changement à la frontière de deux régions. De ce fait on peut pressentir que le bruit dans l'image (anomalies sur les niveaux de gris par exemple...) va compliquer ce travail de détection.

L'obtention des contours est une étape importante dans le processus d'interprétation automatique d'une image puisque cela permet de matérialiser les contours des objets recherchés.

Les méthodes de détection des contours comportent en général deux phases :

- D'abord trouver les pixels censés appartenir à un contour (éventuellement avec une mesure de certitude) en s'appuyant sur une propriété particulière (par exemple un changement local de niveau de gris). On appellera ces points : des **points-candidats** ou des **points contour.**

- Puis relier ces points contour de façon à obtenir de véritables contours (lignes, courbes,...)

La première phase va faire appel à des techniques permettant de « supprimer » le bruit (opération de lissage) et/ou de mettre en évidence et détecter les points frontière.

Les méthodes correspondantes s'appuient sur des « filtrages » de l'image.

Ce filtrage peut être fréquentiel (passage par la transformée de Fourier par exemple) ou spatial (convolution de l'image avec un filtre).

5.1 FILTRAGE SPATIAL

Pour la détection des contours nous allons avoir à faire des lissages (suppression du bruit) ou à repérer les variations sur un paramètre (par exemple le niveau de gris) en utilisant des notions de dérivées (gradients, laplaciens).

Ce travail peut être réalisé avec l'aide de filtres spatiaux que nous allons décrire maintenant

Définition:

Le filtrage spatial d'une image I consiste à **convoluer** cette image considérée comme une fonction I(x,y) avec une fonction f(x,y).

f(x,y) s'appelle la **"réponse impulsionnelle du filtre"**

Le résultat est une nouvelle image dite « filtrée » définie par la formule suivante:

En considérant l'image comme une fonction continue sur R^2

$$I_f(x,y) = (f * I)(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x',y') \times I(x-x',y-y') \times dx' \times dy'$$

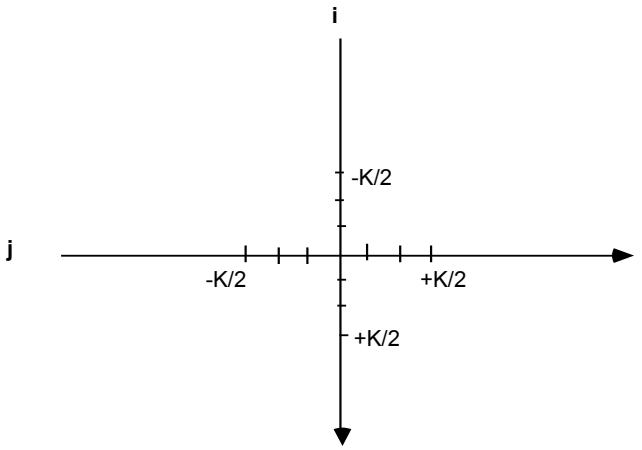
qui s'écrit aussi

$$I_f(x,y) = (f * I)(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x-x',y-y') \times I(x',y') \times dx' \times dy'$$

l'opérateur ***** s'appelle une **convolution**.

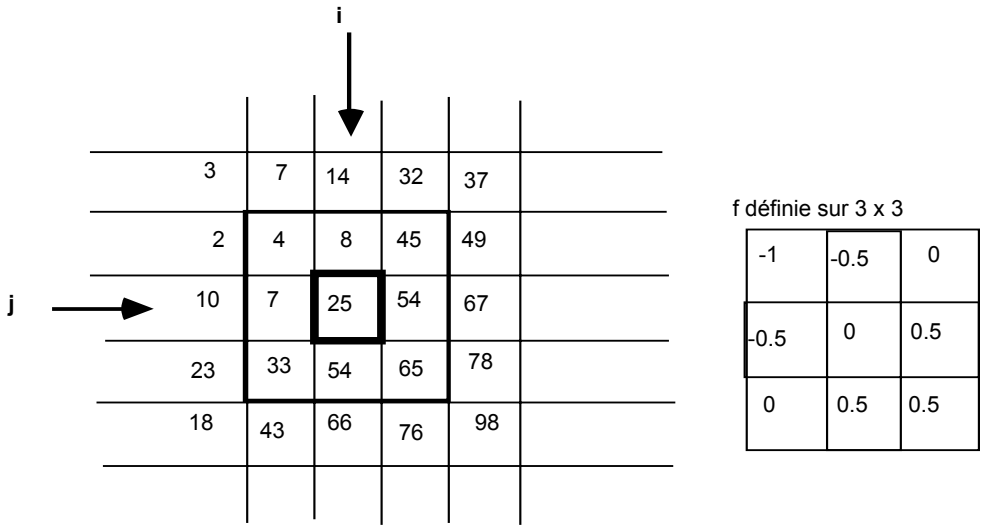
Dans le domaine discret , ce qui est le cas des images numériques,
on obtient la formulation suivante :

- l'image I est supposée de taille 2N+1 x 2N+1
- la fonction f est définie sur une fenêtre 2K+1 x 2K+1



et $I_f(i,j) = \sum_{i'=-K/2}^{+K/2} \sum_{j'=-K/2}^{+K/2} f(i-i', j-j') \times I(i', j')$

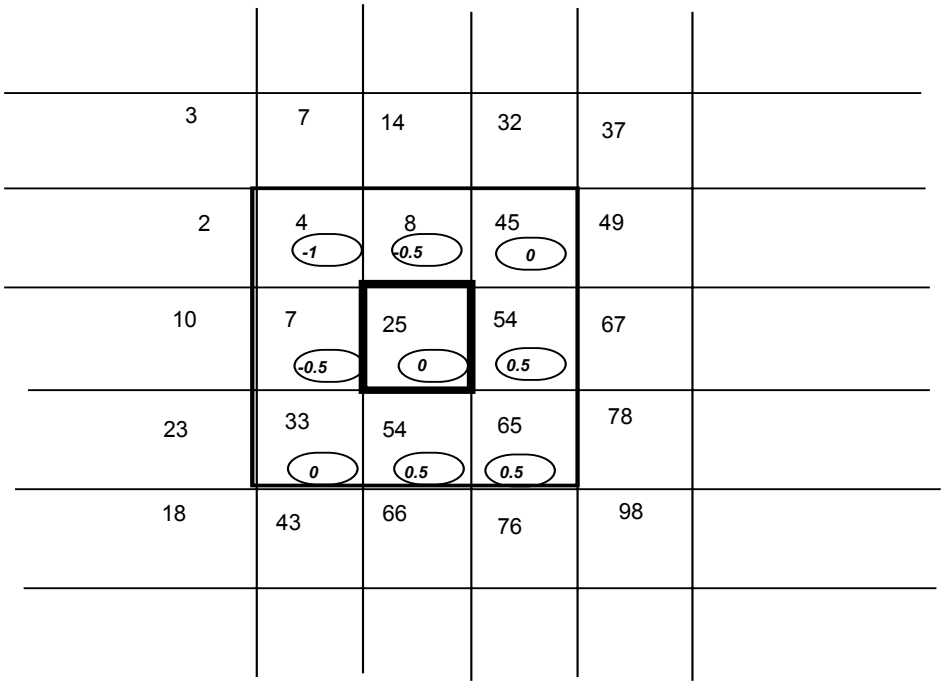
exemple :



f définie sur 3 x 3

-1	-0.5	0
-0.5	0	0.5
0	0.5	0.5

CONVOLUTION en (i,j) par la fonction f (le masque)



I(i,j) =25

I_f(i,j)= (-1)x4 + (-0.5)x8 + (0)x45 + (-0.5)x7
+ (0)x25 + (0.5)x54 + (0)x33 + (0.5)x54 + (0.5)x 65 = 75

Notons que l'opération de convolution discrète réalisée met en évidence un point qui semble appartenir à une frontière entre régions de niveaux de gris assez éloignés.

Cette opération de convolution discrète appliquée à toute l'image fonctionne par décalages et multiplications puisqu'on va déplacer le filtre sur toute l'image et réaliser la convolution sur tous les pixels image. Les pixels de bord d'image ne peuvent être convolués que si le filtre reste intérieur à l'image. Le filtrage linéaire réalisé consiste donc à remplacer chaque pixel de l'image par une combinaison linéaire des niveaux de gris sur un voisinage donné de ce pixel (fenêtre 3x3, 5x5, 7x7)

Exemple d'application de ce filtre:
Image initiale

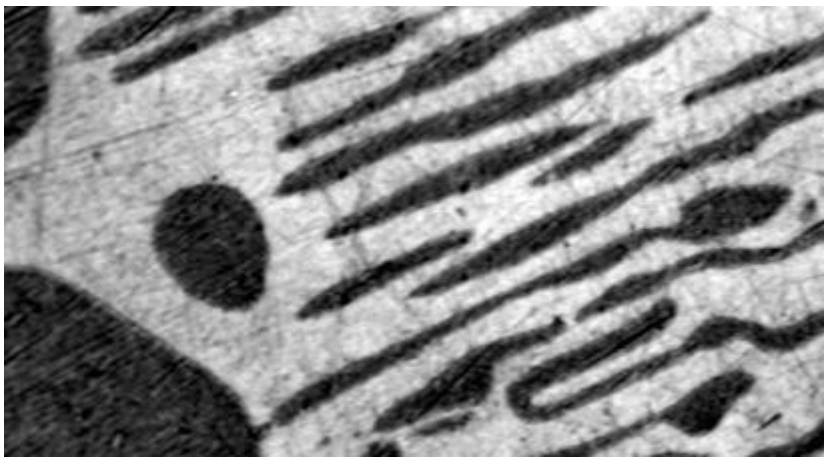


Image filtrée

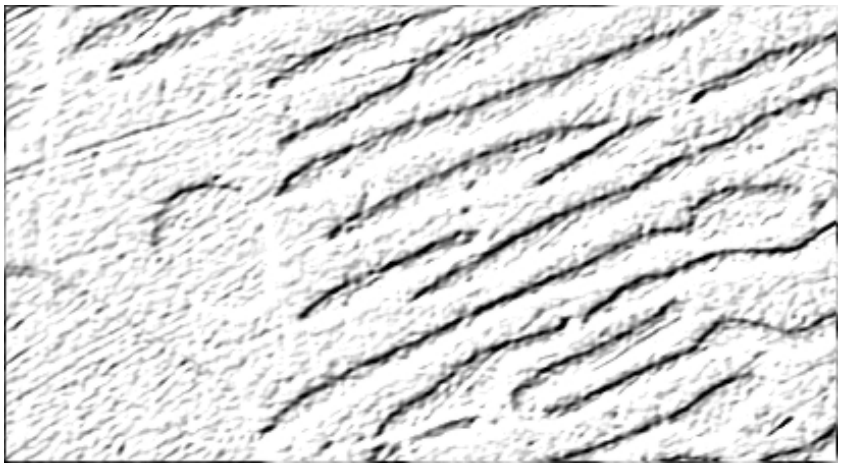
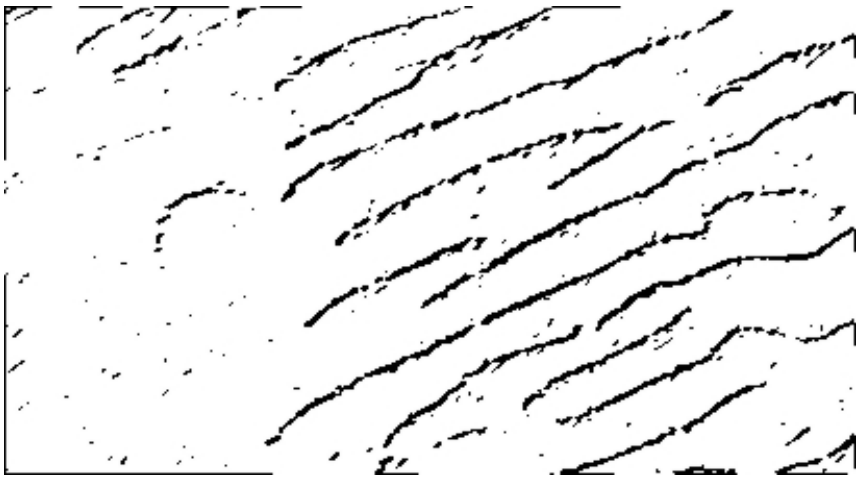


image filtrée seuillée



Remarque :

On dit que la fonction f(x,y) s'appelle **« réponse impulsionnelle du filtre »** pour la raison suivante:

Dans le cas continu , soit une image « fonction de Dirac »

et telle que $I(x,y) = 0 \quad pour \quad x \quad ou \quad y \quad \neq \quad 0$

$et \quad \int I(x,y)dxdy = 1$

alors $\forall \quad f(x,y) \quad on \quad a \quad I_f(x,y) = f(x,y)$

Dans le cas discret on obtient le même résultat (image telle que i(0,0)=1 et I(x,y) =0 pour x ou y ≠0

5.1.1 Divers filtres spatiaux linéaires

Il existe différents types de filtres et la taille et le contenu du filtre vont correspondre à divers types d'opérations:

- lissage d'images (« éliminer les pixels isolés » considérés comme du bruit)

- Calcul de gradient (« dérivée première »),
  de laplacien (« dérivée seconde ») , ...

5.1.1.1 Lissage

Nous allons présenter différents types de lissage. Certains d'entre eux sont des filtres linéaires , mais nous en présenterons certains qui ne le sont pas.

5.1.1.2 Filtrage par la moyenne

Cette méthode permet de « lisser » les images , c'est à dire de diminuer les différences de niveaux de gris entre pixels voisins. Cette méthode très simple est censée supprimer le bruit.

Le filtrage par la moyenne consiste à remplacer chaque pixel par la valeur moyenne de ses voisins (le pixel lui même y compris). Cette méthode a pour effet de modifier les niveaux de gris trop différents de leurs voisins (en ce sens on peut penser « supprimer » le bruit, c'est à dire des niveaux de gris « anormaux ».

Suivant la "violence" du lissage que l'on veut réaliser on choisira une taille de filtre plus ou moins grande (3x3, 5x5,..) mais on doit comprendre que les contours de l'image de départ deviendront alors plus « flous ».

filtre 3x3 :

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

filtre 5x5 :

1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25

image initiale

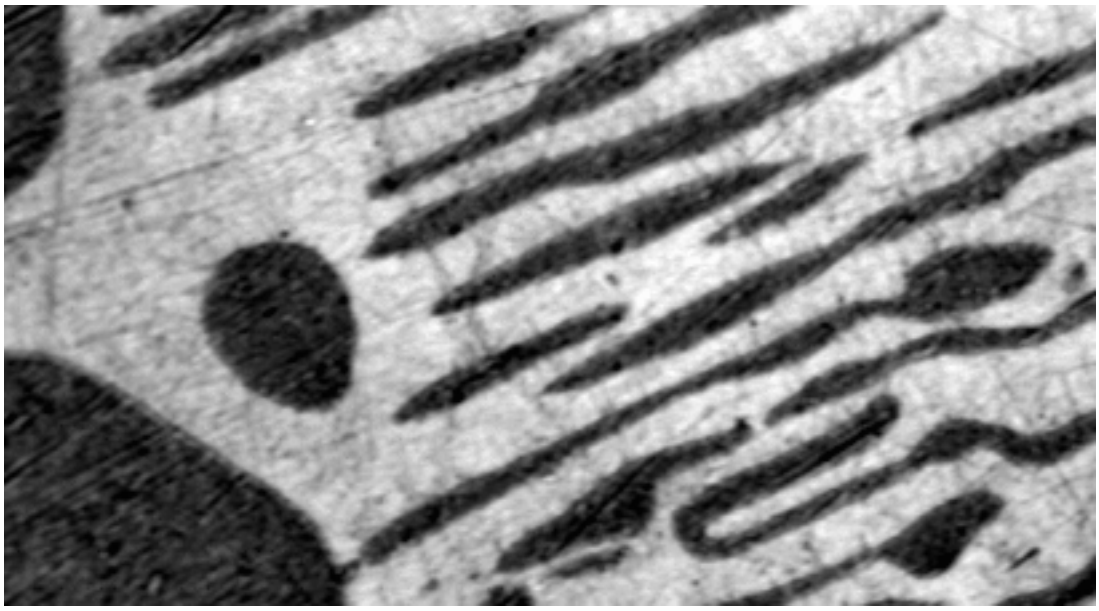
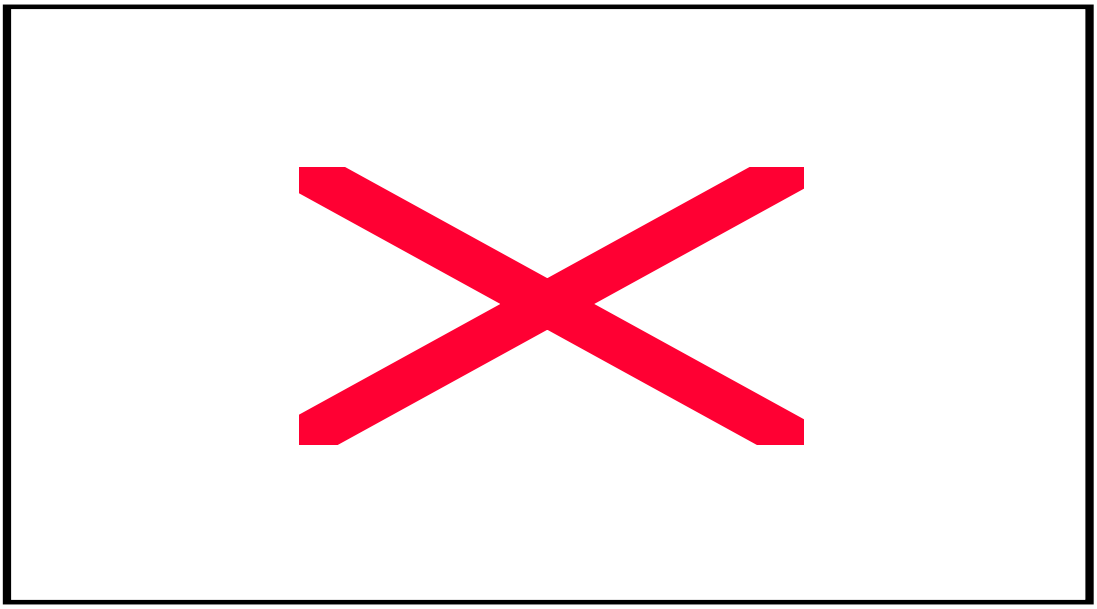


image filtrée (moyenne 3X3)

image filtrée (moyenne 5x5)

image filtrée (moyenne 7x7)

image filtrée (moyenne 63x63)
les pixels situés trop prés des bords ne sont pas traités!



Les inconvénients évidents de ce filtre de moyenne sont les suivants:

- Un pixel isolé avec un niveau de gris “anormal“ pour son voisinage va perturber les valeurs moyennes des pixels de son voisinage.

- Sur une frontière de régions le filtre va estomper le contour et le rendre flou, ce qui est gênant en visualisation bien sûr mais éventuellement aussi pour un traitement ultérieur qui nécessiterait des frontières nettes.

Il est possible de moduler ces effets néfastes en réalisant en chaque pixel une convolution “conditionnelle“

Par exemple en un pixel de niveau de gris NG1 on applique le filtre.
Supposons obtenir une valeur NG2, alors on décidera d’appliquer le filtre que si

$$|NG1 - NG2| \geq Seuil$$

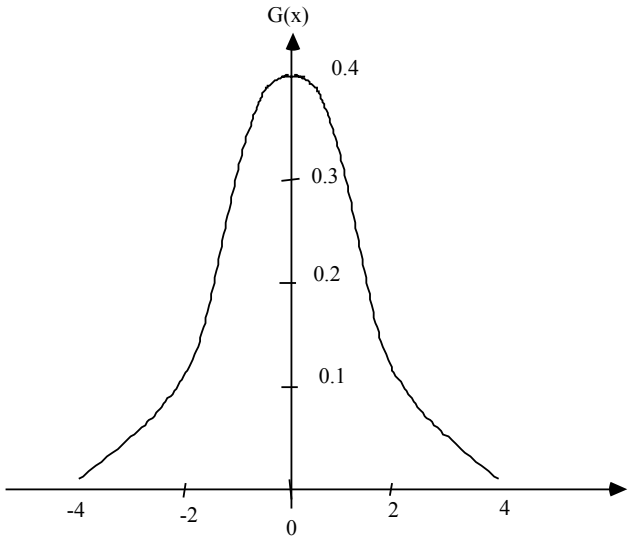
5.1.1.3 Filtrage Gaussien

Le filtre gaussien est un opérateur de lissage utilisé pour estomper les « détails » et le bruit. Ce filtre a une logique analogue au filtre moyenne.

Distribution Gaussienne en 1D **:** $$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

avec μ : moyenne et σ écart-type

Exemple avec $\mu = 0$ et $\sigma = 1$:



En 2D et en supposant que la distribution est circulaire symétrique la distribution gaussienne a pour expression :

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x-\mu 1)^2+(y-\mu 2)^2}{2\sigma^2}}$$

Le filtrage Gaussien va utiliser cette distribution pour définir un filtre de convolution. Comme on travaille sur des images discrètes on utilise une approximation discrète de la distribution gaussienne dans un filtre fini de convolution.

Voici par exemple un filtre de taille 5 x 5 représentant une gaussienne de moyenne nulle et d'écart type égal à 1.4:

1/115 *

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Application:

Image initiale

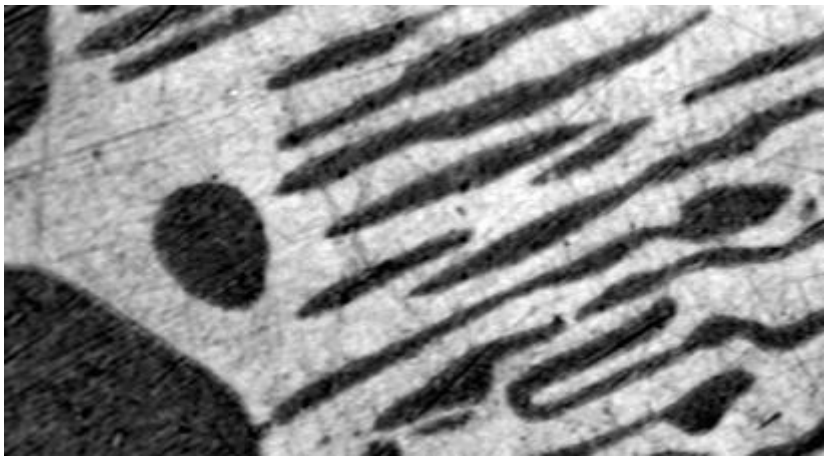


image filtrée

L'intérêt de ce filtre est que le filtrage peut être réalisé en deux passages en traitant d'abord ligne par ligne avec un filtre à une dimension puis on traite l'image obtenue colonne par colonne avec le même filtre 1D. Ceci est rendu possible du fait du caractère symétrique circulaire de ce filtre. Bien entendu le calcul est largement accéléré.

Le filtre 1D utilisé pour réaliser le filtre 5 x 5 vu précédemment est approximativement le suivant:

1/10.7 * | 1.3 | 3.2 | 3.8 | 3.2 | 1.3 |

Notons aussi qu'un filtre 2D gaussien avec un écart-type élevé peut être réalisé par application successive de filtres gaussiens d'écart-type plus faible.

propriétés:

le degré de lissage est déterminé par l'écart-type associé au filtre gaussien (mais alors plus l'écart-type est élevé plus il faut prendre un filtre grand (5 x5,7 x 7).

.

Dans la mesure où les « poids » sont plus élevés au centre du filtre qu'à l'extérieur (à la différence du filtre moyenne) les contours sont mieux conservés qu'avec le filtre moyenne.

5.1.2 Lissage par filtres non linéaires

Il est possible de réaliser des filtres non linéaires (un filtre est non linéaire si ayant deux images I1 et I2 et un filtre F alors $I1_F(x,y)+I2_F(x,y) \neq (I1+I2)_F(x,y)$ pour tout x et y), c'est à dire ici non réalisables par une convolution . Leur résultat est souvent une limitation intéressante du bruit tout en préservant les contours « nets ».

Nous décrirons ici le filtre médian et le filtre par le max. (« conservative smoothing »)

5.1.2.1 Filtrage médian

Le filtre médian réalise un lissage de l'image un peu plus performant que le filtre moyenne en ce qui concerne les détails dans l'image.

Méthode:

Chaque pixel est traité en considérant ses voisins sur un voisinage donné.

Le pixel lui même et ses voisins forment alors un ensemble dont on calcule la « médiane ». Le pixel sera alors remplacé par cette valeur médiane.

exemple:

exemple : voisinage 3 x 3

12	14	5	4	2
4	99	23	56	4
5	126	121	120	5
4	97	12	9	4
2	4	5	4	2

tri 9 12 23 56 97 99 120 121 126

valeur médiane 97

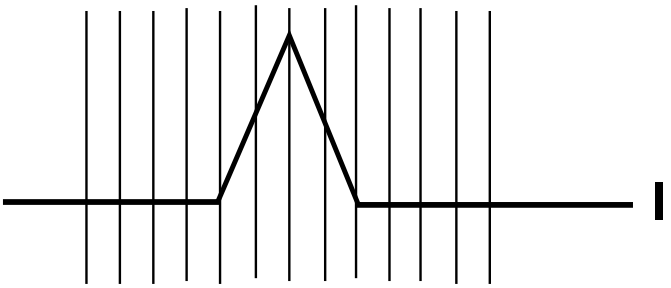
Intérêt du filtre médian:

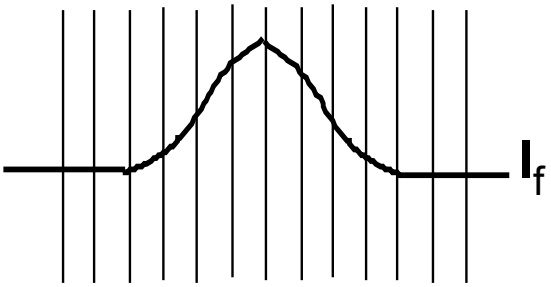
- un pixel non représentatif dans le voisinage affectera peu la valeur médiane.

- La valeur médiane choisie étant le niveau de gris d'un des pixels considérés , on ne crée pas alors de nouveaux niveaux de gris dans l'image. Ainsi lorsque le filtre passe sur un contour très marqué il le préservera mieux .

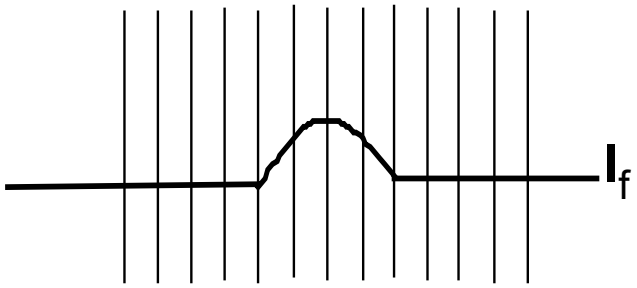
exemple:

Comparaison filtre moyenne et filtre médiane
pour un bruit de type impulsionnel

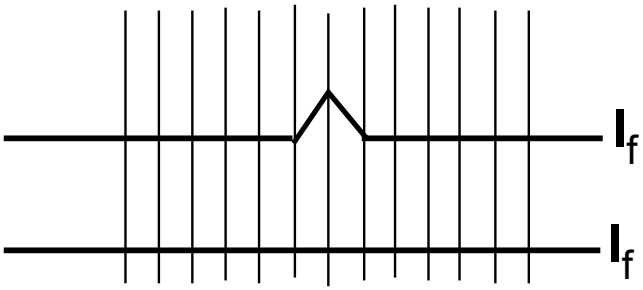




filtre moyenne
sur "fenêtre"
de taille 3



filtre médiane
sur "fenêtre"
de taille 3



filtres médiane
sur fenêtre de
taille croissante
supérieure à 3

5.1.2.2 Filtrage par le max. (« conservative smoothing »)

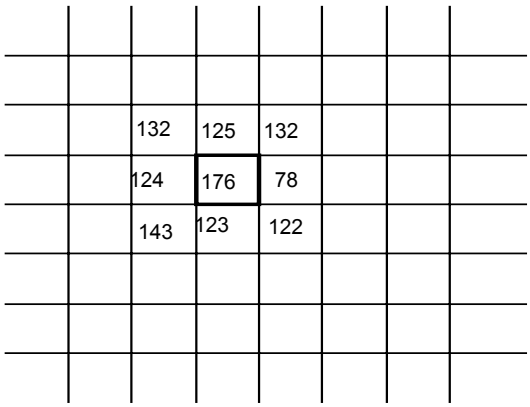
Ce filtre de lissage supprime bien le bruit de type “poivre et sel“ c’est à dire qu’il “adoucit“ les pixels isolés ayant un niveau de gris très différent des niveaux de gris de leur voisinage et il a la particularité de bien préserver les contours très marqués.

Ce filtre s’assure en fait que tout pixel a son niveau de gris placé dans la gamme de ses voisins.

méthode:

On considère le niveau de gris du pixel à traiter, et d’autre part tous ses voisins (à l’exception de lui même) . Sur les voisins on calcule le niveau min et le niveau max., si le niveau de gris du pixel à traiter est compris entre le min et le max. alors on le laisse inchangé sinon on le remplace par le max.

exemple :



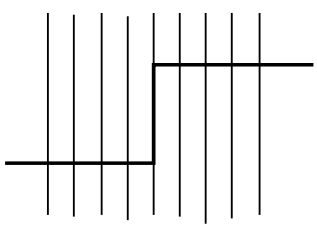
Voisins : 78 122 124 125 132 132 143
min : 78 max : 143

Nouvelle valeur : 143

5.1.3 Détection des points-contour

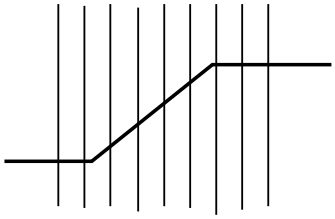
Il existe des filtrages permettant de mettre en évidence les points contours. Les points contours sont en fait des points où l'on trouve des discontinuités sur un paramètre (la plupart du temps le niveau de gris)

Notons qu'on peut trouver divers types de contours que nous représentons ci-dessous par un profil sur l'image:



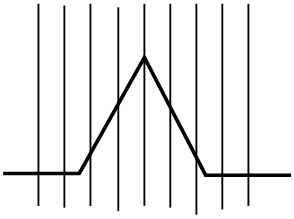
"marche"

profil: 7 7 9 6 123 125 121 120 126



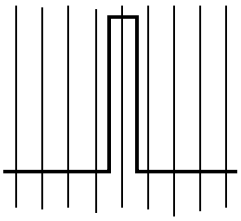
"rampe"

profil: 7 8 24 43 56 98 120 119 125



"toit"

profil: 7 7 34 56 118 54 29 8 7



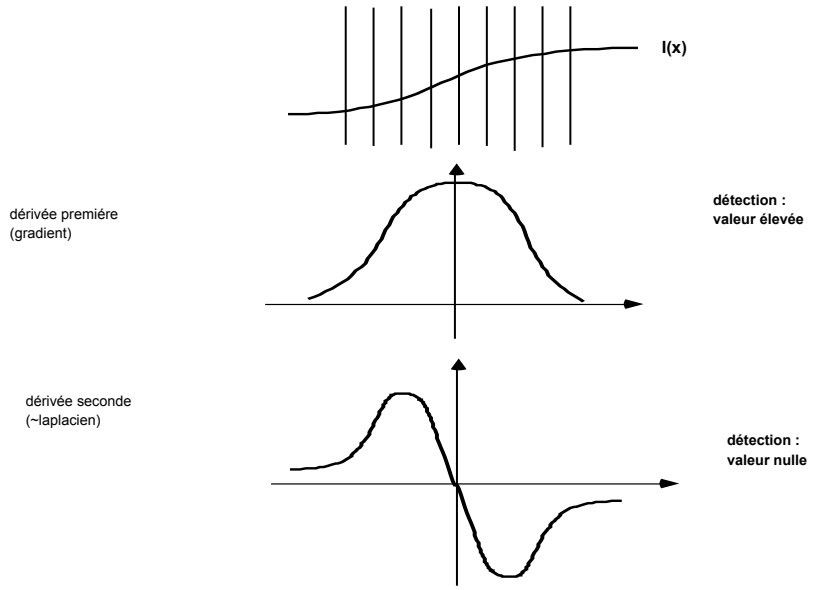
"pic"

profil : 6 9 7 7 128 6 8 8 9

Une méthode de mise en évidence des points contours peut donc consister à singulariser les pixels où on trouve une forte discontinuité.

Par exemple par un calcul de gradient (dérivée d'ordre 1) ou de Laplacien (lié aux dérivées secondes).

Explication:



Une méthode de mise en évidence des points contour peut donc prendre la forme suivante:

Lissage ⟶ Dérivation par filtrage⟶ sélection des points sur critère

Les méthodes classiques:

1) Calcul du gradient de l'image en chaque pixel et extraction des maxima locaux de la norme G du gradient.

$$\vec{G}(x,y) = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix}$$

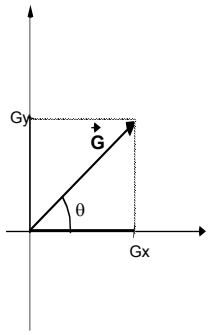
2) Calcul du Laplacien et extraction des zéros (voisins de 0)

$$\nabla^2 I(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Calcul de gradient sur une image:

Le gradient est un vecteur $\vec{G}$ à deux composantes pour une image 2D qui sont:

$$G_x(x,y) = \frac{\partial I(x,y)}{\partial x} \qquad G_y(x,y) = \frac{\partial I(x,y)}{\partial y} \qquad G = \left\| \vec{G} \right\|$$



Vecteur gradient défini par

module du gradient $\quad G = \left(G_x^2 + G_y^2\right)^{1/2} \; ou \quad \mathbf{max}\left(\left|G_x\right|, \left|G_y\right|\right)$

direction du gradient $\quad \theta = arctg\left(\frac{G_y}{G_x}\right)$

Notons que si on calcule les dérivées 1D sur tous les profils passant au point (x,y), ces dérivées ont un module maximum dans les directions Θ+kπ

Relation entre contour et gradient :

Prenons l'exemple à 1 dimension suivant:

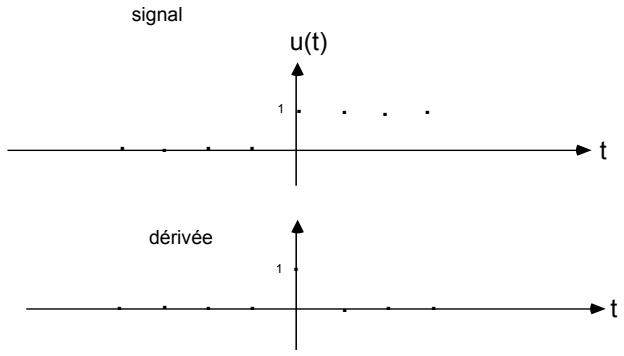
fonction marche u(t) = 1 si t >= 0
= 0 si t < 0

et soit la fonction de Dirac δ(t) $\quad \left[\begin{array}{l} = 1 \text{ si } t = 0 \\ = 0 \text{ si } t \neq 0 \text{ et } \int_{-\infty}^{+\infty} \delta(t)dt = 1 \end{array} \right.$

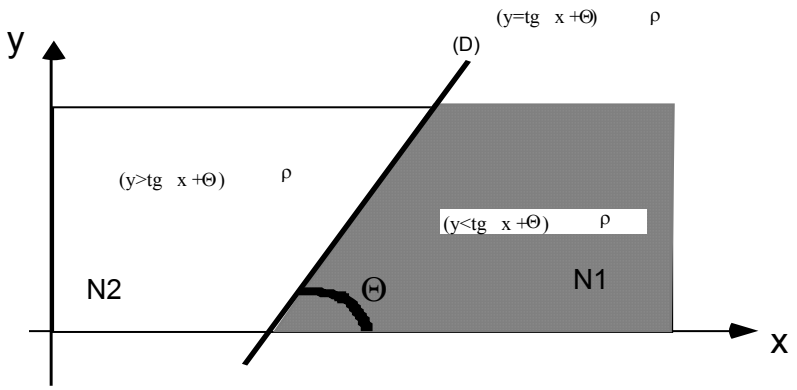
alors $u(z) = \int_{-\infty}^{z} \delta(t)dt$ et donc u'(t)=δ(t)

si on travaille en discret (valeurs de t entières signées)

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



reprenons cet exemple en 2D et soit l'image suivante:



Deux régions (niveaux de gris N1 et N2)
séparées par une frontiére de type "marche"

Alors le contour a pour équation dans le repère indiqué:

droite (D) - x sinΘ + y cosΘ-ρcosΘ=0 (car y=tgΘ x +ρ)

La fonction image vaut donc:

$I(x,y)=N1 + (N2-N1) * u(-x \sin\Theta + y \cos\Theta - \rho_1)$ si on note $\rho_1=\rho\cos\Theta$

les composantes du vecteur gradient en un pixel (x,y) sont :

$Gx = (N2-N1) * \delta(-x \sin\Theta + y \cos\Theta - \rho_1) * (-\sin\Theta)$

$Gy = (N2-N1) * \delta(-x \sin\Theta + y \cos\Theta - \rho_1) * (\cos\Theta)$

Sur le contour (D) on a donc

$$G^2 = G_x^2 + G_y^2 = (N2 - N1)^2 \; donc \quad G = |N2 - N1|$$

Et le vecteur gradient $\vec{G}$ est orthogonal à la direction du contour (D)

Il a donc une direction de $\Theta+\Pi/2$

5.1.4 Quelques propriétés des filtres linéaires

Soit I un signal (image par exemple !) et un filtre spatial linéaire de réponse impulsionnelle f

alors (I*f)' = I*f' (I*f)"=I*f" ....

Démonstration :

Transformée de Fourier:

$$F\{f\} = \int_{-\infty}^{+\infty} f(t) \times e^{-j\omega t} dt = F(j\omega) \qquad signal \qquad 1D$$

$$alors \qquad \boxed{F\left\{\frac{df}{dt}\right\} = j\omega \times F(j\omega)} \quad (1)$$

$$car \qquad F\left\{\frac{df}{dt}\right\} = \int_{-\infty}^{+\infty} \frac{df(t)}{dt} \times e^{-j\omega t} dt$$

$$u' \qquad v \qquad et \qquad u'v = (uv)' - uv'$$

$$donc \quad \frac{df}{dt} \times e^{j\omega t} = (f \times e^{-j\omega t})' - f \times (-j\omega) \times e^{-j\omega t}$$

$$F\left\{\frac{df}{dt}\right\} = \left[f \times e^{-j\omega t}\right]_{-\infty}^{+\infty} - \int_{-\infty}^{+\infty} f \times (-j\omega) \times e^{-j\omega t} dt$$

$$F\left\{\frac{df}{dt}\right\} = 0 - 0 + j\omega \times \int_{-\infty}^{+\infty} f(t) \times e^{-j\omega t} dt$$

$$F(j\omega)$$

$$f(t) * g(t) = \int_{-\infty}^{+\infty} f(t-\tau) \times g(\tau) \times d\tau \quad et \qquad g(t) * f(t) = \int_{-\infty}^{+\infty} g(t-\tau) \times f(\tau) \times d\tau$$

alors $\boxed{F\{f(t) * g(t)\} = F\{f(t)\} \times F\{g(t)\}}$ (2)

$$F(j\omega) \times G(j\omega)$$

car $$F\{f(t)*g(t)\} = \int_{-\infty}^{+\infty}\left[\int_{-\infty}^{+\infty} f(\tau)\times g(t-\tau)\times d\tau\right]\times e^{-j\omega t}\times dt$$

$$= \int_{-\infty-\infty}^{+\infty+\infty} f(\tau)\times g(t-\tau)\times e^{-j\omega t}\,d\tau dt$$

$$= \int_{-\infty-\infty}^{+\infty+\infty} g(t-\tau)\times e^{-j\omega(t-\tau)}dt \times e^{j\omega\tau} f(\tau)\times d\tau$$

$$= \int_{-\infty-\infty}^{+\infty+\infty} f(\tau)\times e^{-j\omega\tau}\times d\tau \times g(t-\tau)\times e^{-j\omega(t-\tau)}dt$$

$$= \int_{-\infty}^{+\infty} f(\tau)\times e^{-j\omega\tau}d\tau \times \int_{-\infty}^{+\infty} g(t-\tau)\times e^{-j\omega(t-\tau)}dt$$

$$= \int_{-\infty}^{+\infty} f(\tau)\times e^{-j\omega\tau}d\tau \times \int_{-\infty}^{+\infty} g(x)\times e^{-j\omega(x)}dx$$

$F(j\omega)$ $G(j\omega)$

$F\{f(t)\}$ $F\{g(t)\}$

et

$$\boxed{(f*g)' = f*g' = g*f'} \qquad (3)$$

car

$F\{f(t)\} = F(j\omega)$ *et* $F\{g(t)\} = G(j\omega)$

$$F\left\{\frac{d(f*g)}{dt}\right\} = j\omega\times F\{f(t)*g(t)\} = j\omega\times F\{f(t)\}\times F\{g(t)\}$$

$$= j\omega\times F(j\omega)\times G(j\omega)$$

d' aprés (2)

$$\frac{d(f*g)}{dt} = F^{-1}\{j\omega \times F(j\omega) \times G(j\omega)\}$$
$$= F^{-1}\{j\omega \times F(j\omega)\} * F^{-1}\{G(j\omega)\}$$
$$= \frac{df}{dt} * g(t)$$
$$= F^{-1}\{F(j\omega)\} * F^{-1}\{j\omega \times G(j\omega)\}$$
$$= f * \frac{dg}{dt}$$

A la recherche des filtres spatiaux linéaires réalisant gradient ou laplacien

Les propriétés des convolutions vues précédemment vont nous permettre de définir les filtres spatiaux linéaires permettant de calculer gradient et laplacien sur une image.

Ainsi pour calculer le vecteur gradient nous avons vu qu'il va falloir calculer des dérivées partielles de l'image suivant les lignes et les colonnes.

Or l'image sur laquelle on veut faire ce traitement est en général lissée (par convolution avec un filtre de lissage).

par exemple pour le gradient , on devra calculer

$$Gx(x,y) = \frac{\partial(I*f)}{\partial x} \quad et \quad Gy(x,y) = \frac{\partial(I*f)}{\partial y}$$

Or nous avons vu que

$$\frac{\partial(I*f)}{\partial x} = I*\frac{\partial f}{\partial x} \qquad \frac{\partial(I*f)}{\partial y} = I*\frac{\partial f}{\partial x}$$

On en déduit que le calcul du gradient d'une image (supposée lissée par convolution) peut être calculée directement par convolution de l'image initiale avec des filtres linéaires (dérivées d'un filtre de lissage).

Nous allons voir des exemples de ceci par la suite.

Le gradient peut donc être calculé par deux convolutions de l'image brute avec des filtres

$$\frac{\partial f}{\partial x} \qquad \frac{\partial f}{\partial y}$$

De la même façon le laplacien d'une image lissée peut être calculé directement par une convolution sur l'image brute:

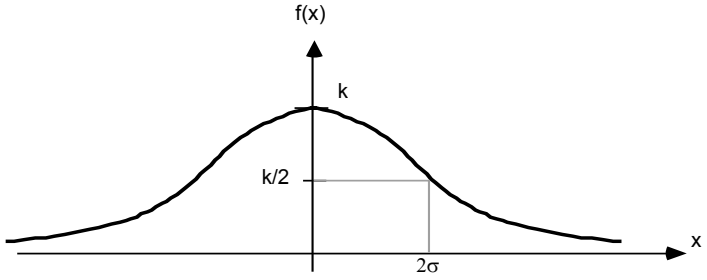
$$\nabla^2 (I * f) = I * \nabla^2 f$$

5.1.4.1 Exemples de définition de filtres de calcul de gradient

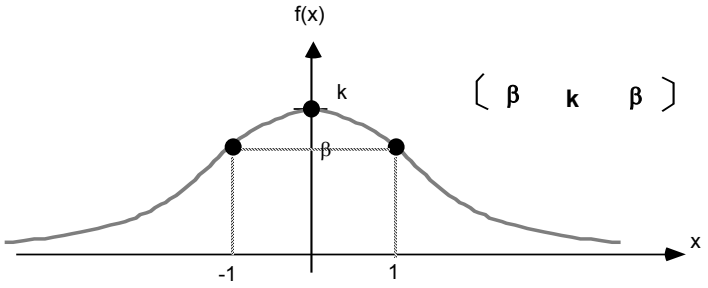
Nous allons prendre d'abord le cas d'un signal à une dimension et nous supposons rechercher les contours du type échelon (⎯⎯⎯⎯) .

Nous allons supposer raisonner dans le cadre continu, et nous appliquerons au cas discret. Puis nous étendrons au 2D (image).

soit donc une fonction de filtrage du bruit: $f(x) = k \times e^{-\frac{x^2}{2\pi\sigma^2}}$



en discret on prendrait comme filtre de taille 3 :



$$\frac{df}{dx} = f'(x) = -k \times \left(-\frac{x}{\pi\sigma^2} \right) \times e^{-\frac{x^2}{2\pi\sigma 2}}$$

$$f'(1) = \alpha = \frac{k}{\pi\sigma^2} \times e^{-\frac{1}{2\pi\sigma^2}} \quad f'(0) = 0 \quad f'(-1) = -\alpha$$

Exemple de filtre (pardon : « réponse impulsionnelle du filtre » !!)

permettant d'obtenir $\frac{d(I * f)}{dx}$: **[-α 0 + α]**

En deux dimensions :

filtre gaussien

$$f((x,y) = k \times e^{-\frac{(x^2+y^2)}{2\pi\sigma^2}}$$

$$\frac{\partial f}{\partial x} = k \times \left(-\frac{x}{\pi\sigma^2}\right) e^{-\frac{(x^2+y^2)}{2\pi\sigma^2}}$$

$$\frac{\partial f}{\partial y} = k \times \left(-\frac{y}{\pi\sigma^2}\right) e^{-\frac{(x^2+y^2)}{2\pi\sigma^2}}$$

Filtre 3 X 3 associé pour la convolution

$$\frac{\partial(I * f)}{\partial x} = I * \frac{\partial f}{\partial x} \qquad \frac{\partial(I * f)}{\partial y} = I * \frac{\partial f}{\partial y}$$

Soit en dérivant sur l'horizontale:

$$\left[\frac{\partial f}{\partial x}(x,-1) \qquad 0 \qquad \frac{\partial f}{\partial x}(x,+1)\right]$$

soit

$$\left[-\frac{k}{\pi\sigma^2}e^{-\left(\frac{x^2+1}{2\pi\sigma^2}\right)} \qquad 0 \qquad +\frac{k}{\pi\sigma^2}e^{-\left(\frac{x^2+1}{2\pi\sigma^2}\right)}\right]$$

Ce qui donne le filtre de convolution 3 X 3 suivant pour calculer la composante horizontale
du gradient:

$$\begin{pmatrix} -\alpha = -\frac{k}{\pi\sigma^2}e^{-\left(\frac{2}{2\pi\sigma^2}\right)} & 0 & +\alpha = +\frac{k}{\pi\sigma^2}e^{-\left(\frac{2}{2\pi\sigma^2}\right)} \\ -\beta = -\frac{k}{\pi\sigma^2}e^{-\left(\frac{1}{2\pi\sigma^2}\right)} & 0 & +\beta = +\frac{k}{\pi\sigma^2}e^{-\left(\frac{1}{2\pi\sigma^2}\right)} \\ -\alpha = -\frac{k}{\pi\sigma^2}e^{-\left(\frac{2}{2\pi\sigma^2}\right)} & 0 & +\alpha = +\frac{k}{\pi\sigma^2}e^{-\left(\frac{2}{2\pi\sigma^2}\right)} \end{pmatrix}$$

Suivant cette logique on obtient des filtres du type suivant pour calculer le gradient:

Pour le calcul de la composante horizontale du gradient:

$$\begin{vmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{vmatrix} \quad \text{ou bien} \quad \begin{vmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{vmatrix}$$

et pour le calcul de la composante verticale

$$\begin{vmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{vmatrix} \quad \text{ou bien} \quad \begin{vmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{vmatrix}$$

On obtient ainsi en chaque point de l'image après les deux convolutions (horizontale et verticale) le vecteur gradient définies par ses deux composantes :

(Gx , Gy)

On peut alors calculer en chaque point

- le module du gradient : $G = \sqrt{\left(G_x^2 + G_y^2\right)}\ ou\ bien |Gx| + |Gy|$

- la direction du gradient : $\theta = arctg(Gy / Gx)$

ces filtres portent le nom de leur inventeur:

filtre de Sobel:

-1 0 +1	-1 -2 -1
-2 0 +2	0 0 0
-1 0 +1	+1 +2 +1

Application:

Image initiale

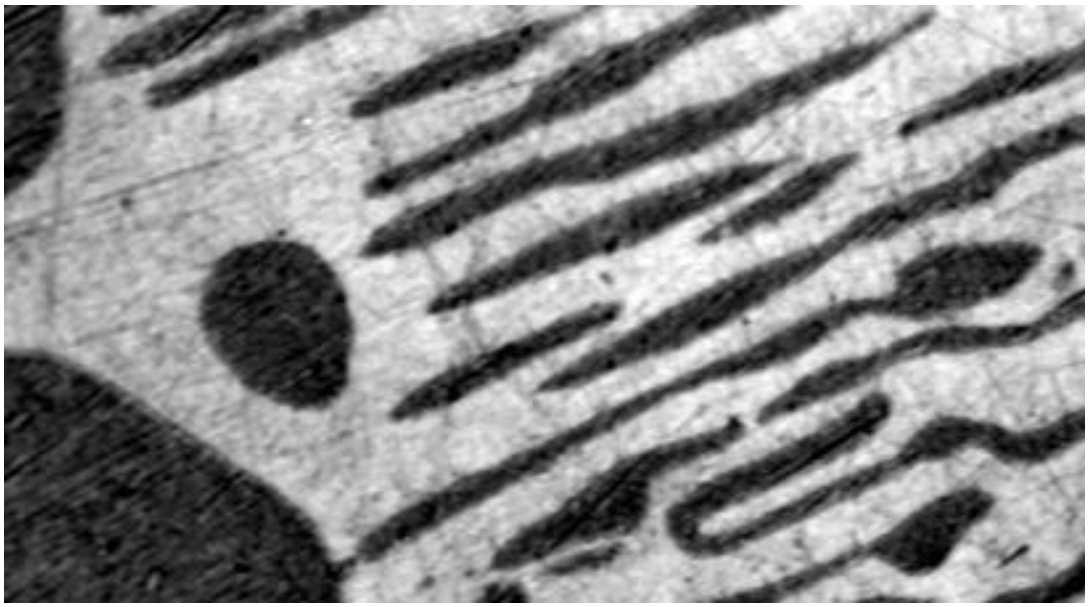


image filtrée (Sobel horizontal)

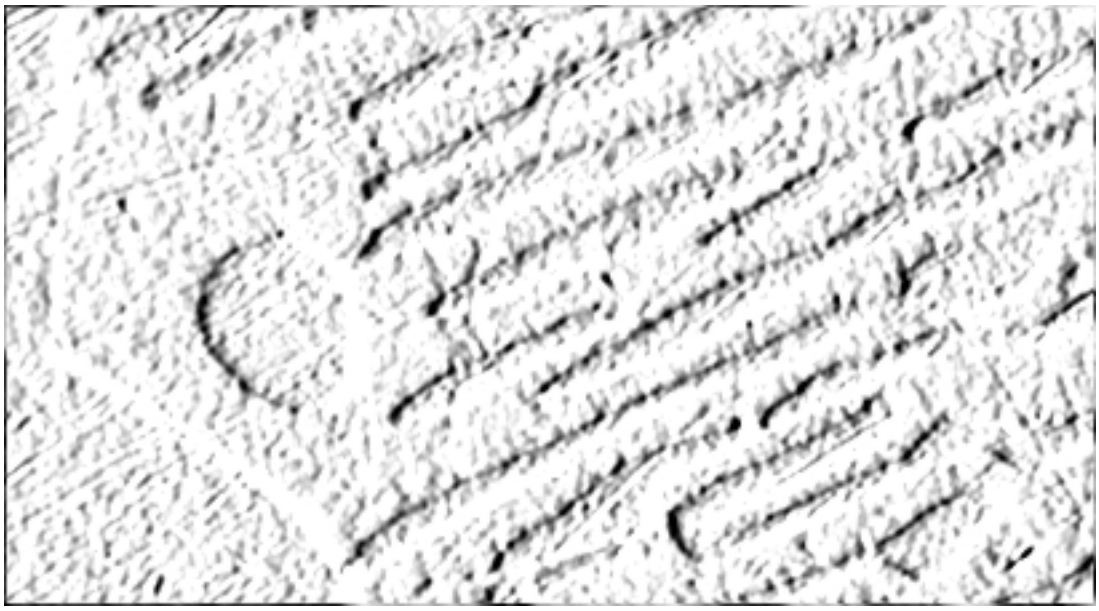
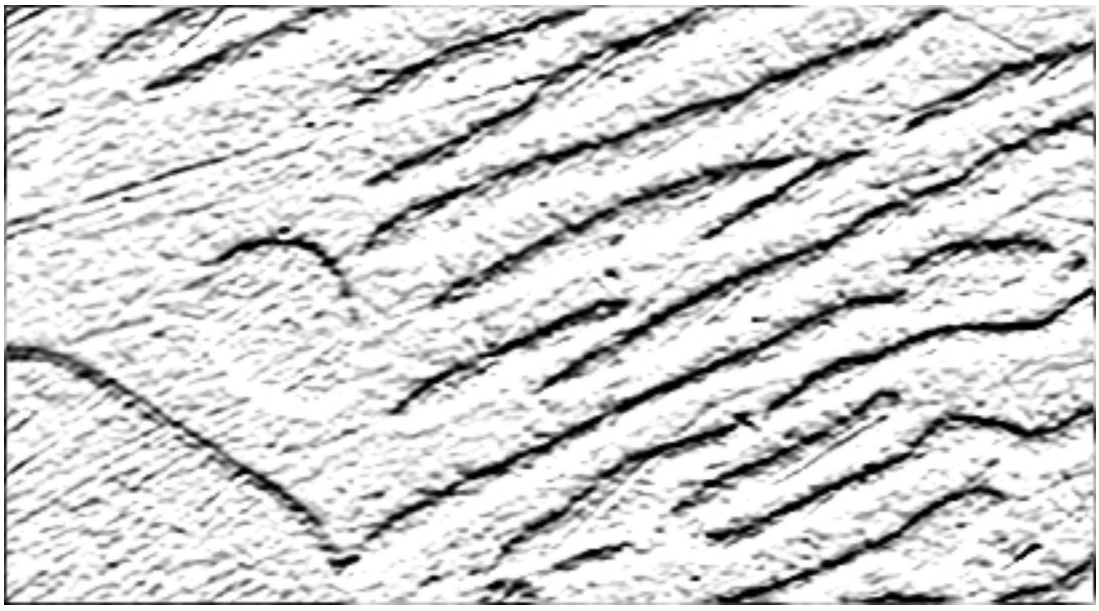


image filtrée (Sobel vertical)



filtres de Prewitt :

-1 0 +1	-1 -1 -1
-1 0 +1	0 0 0
-1 0 +1	+1 +1 +1

Application:

Image initiale

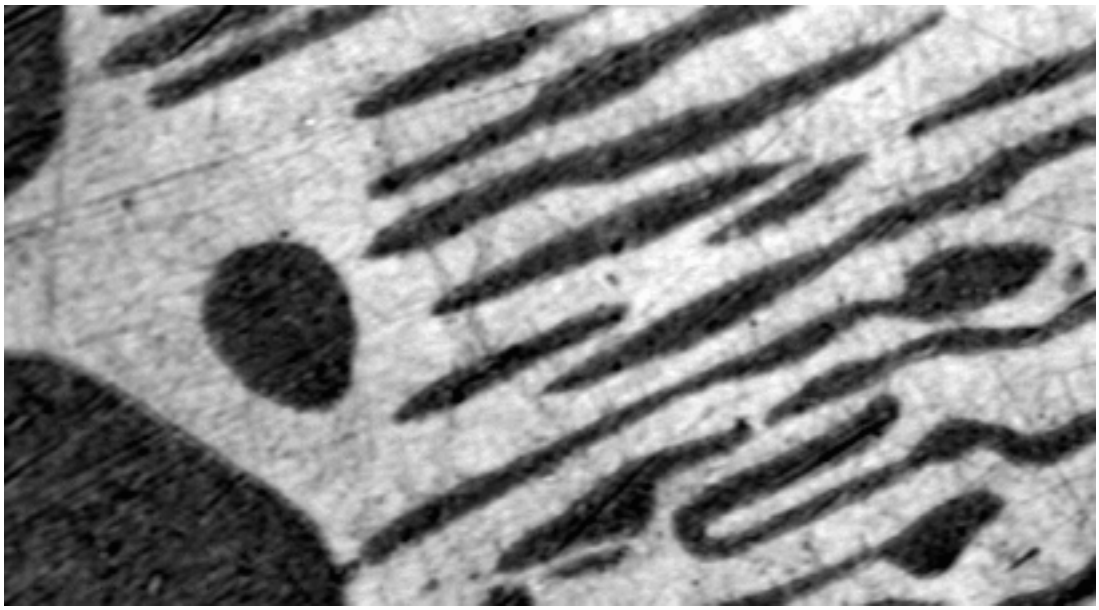


image filtrée (Prewitt horizontal)

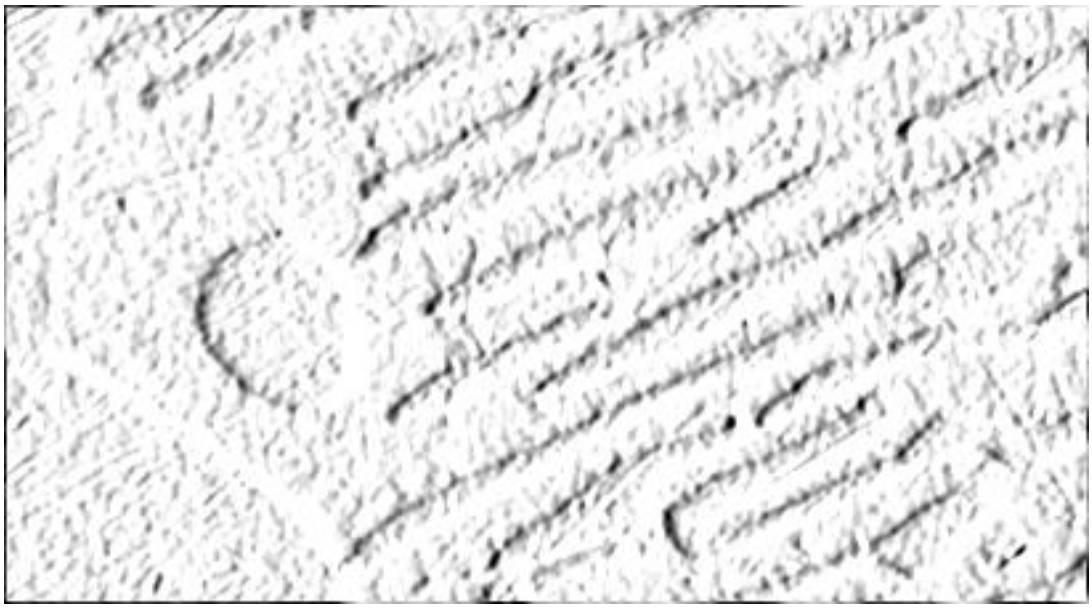
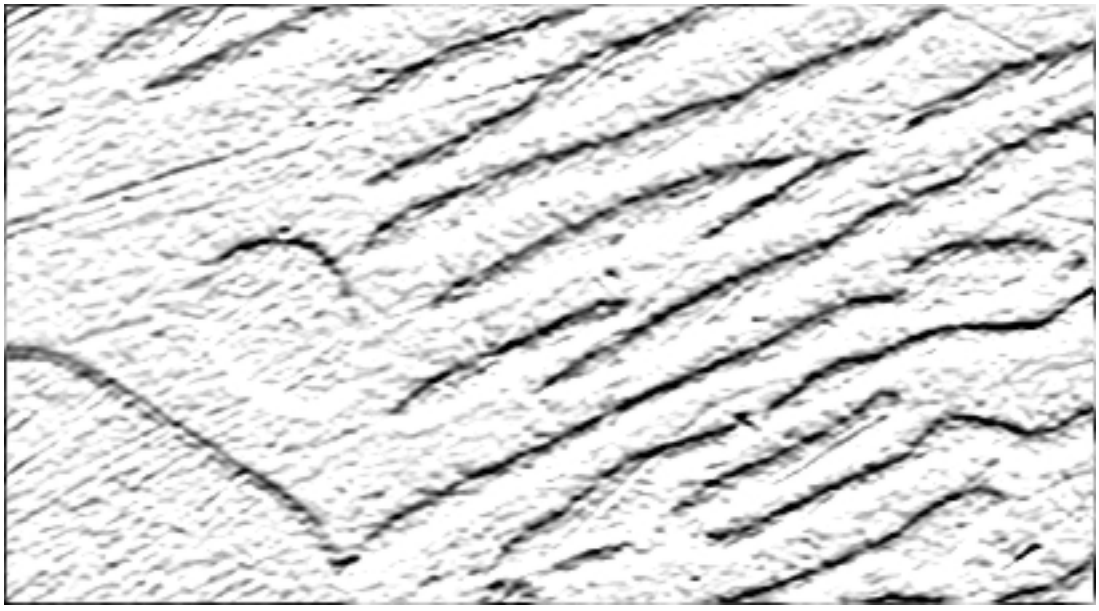


image filtrée (Prewitt vertical)



Détecteur de contour de ROBERT :

C’est un filtre de convolution travaillant sur un voisinage 2 x 2 très rapide à calculer

Il permet de calculer le gradient en un point en traitant successivement chacune des composantes du gradient calculées suivant les directions 45° et 135° (au lieu de 0° et 90° pour les méthodes classiques).

Voici les filtres utilisés par cette méthode:

+1	0
0	−1

0	+1
−1	0

En un point de coordonnées x et y on appliquera la convolution en considérant les points suivants:

(x,y) , (x+1,y) , (x,y+1) , (x+1,y+1)

Le premier opérateur calcule la composante G1 du gradient (calcul de la composante dans la direction 135° , le deuxième la composante G2 du gradient (calcul de la composante dans la direction 45°)

Application:

Image initiale

image filtrée (Robert 135°)

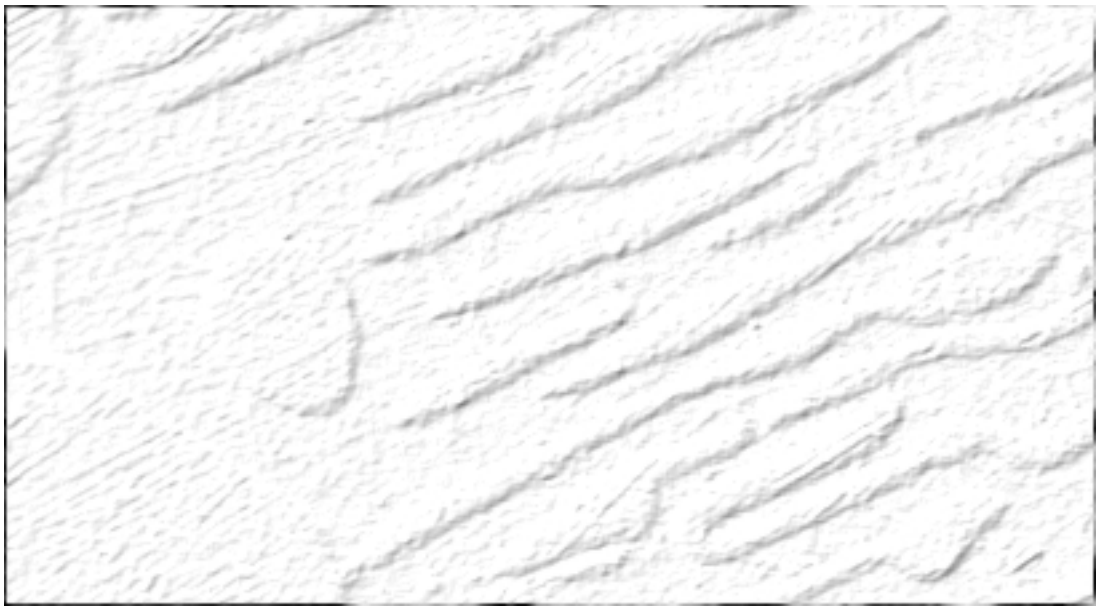


image filtrée (Robert 45°)

On peut alors calculer

- l'amplitude du gradient $G = \sqrt{\left(G_1^2 + G_2^2\right)} \quad ou \quad bien \quad |G_1| + |G_2|$

- L'orientation du gradient $\theta = arctg\left(\frac{G_1}{G_2}\right) + \frac{\pi}{4}$

L'avantage évident de cet opérateur est sa rapidité (le calcul de l'amplitude consiste à faire la somme des valeurs absolues de la différence de deux pixels).

(pix1 pix2)
(pix3 pix4)

$$|G| = |pix1 - pix4| + |pix2 - pix3|$$

Mais l'inconvénient majeur de cet opérateur est son extrême sensibilité au bruit du fait de sa « petite taille ». Le détecteur de Sobel donnera évidemment de meilleurs résultats mais avec une complexité de calcul nettement plus grande.

D'autre part il ne détecte réellement que les contours très aigus.

Notons ici que l'utilisateur n'affiche la plupart du temps que l'image de l'amplitude du gradient (les points très intenses apparaissent alors à priori comme ayant de grande chance d'appartenir à des contours de régions).

Détecteur de gradient de Kirsch :

La méthode proposée consiste à filtrer l'image avec 8 masques directionnels.

L'orientation du contour (orientation du filtre + 45°) est dans ce cas déduite par le filtre donnant le résultat le plus élevé (qui est considéré comme représentant l'intensité du gradient).

Exemple de tels filtres:

$$\begin{vmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{vmatrix} \qquad \begin{vmatrix} 3 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & -5 & 3 \end{vmatrix}$$

Application:

Image initiale

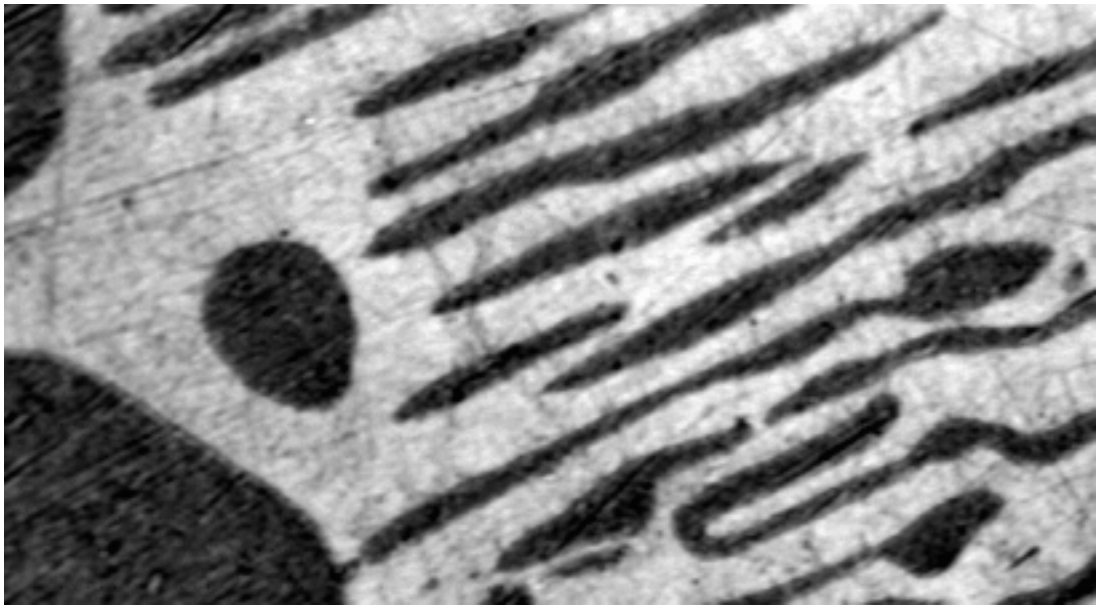


image filtrée (premier filtre de Kirsch)

image filtrée (deuxième filtre de Kirsch)

Les autres masques se déduisent par permutations circulaires.

Notons qu'il existe d'autres masques de ce type suivant la même logique.

5.1.4.2 Exemple de définition de filtres de calcul de laplacien

De même que l'on a pour le calcul de dérivée première:

$$\frac{\partial(I * f)}{\partial x} = I * \frac{\partial f}{\partial x} \qquad \frac{\partial(I * f)}{\partial y} = I * \frac{\partial f}{\partial x}$$

Le gradient peut être calculé par deux convolutions de l'image brute avec des filtres:

$$\frac{\partial f}{\partial x} \qquad \frac{\partial f}{\partial y}$$

De la même façon, le laplacien d'une image lissée peut être calculé directement par une convolution sur l'image brute:

$$\nabla^2 (I * f) = I * \nabla^2 f$$

Expression du laplacien de f(x,y):

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

De façon analogue à ce que nous avons vu pour le calcul de gradient d'une image, on voit que la fonction f est une fonction de lissage de l'image pour laquelle nous allons chercher un filtre de convolution qui approximera le calcul du laplacien sur l'image.

- Cas d'un signal monodimensionnel:

alors le laplacien se réduit à $\nabla^2 f(x) = \frac{\partial^2 f(x)}{\partial x^2}$

Prenons une fonction de lissage de type gaussienne (on suppose raisonner en continu)

$$f(x) = k \times e^{-\frac{x^2}{2\pi\sigma 2}}$$

alors

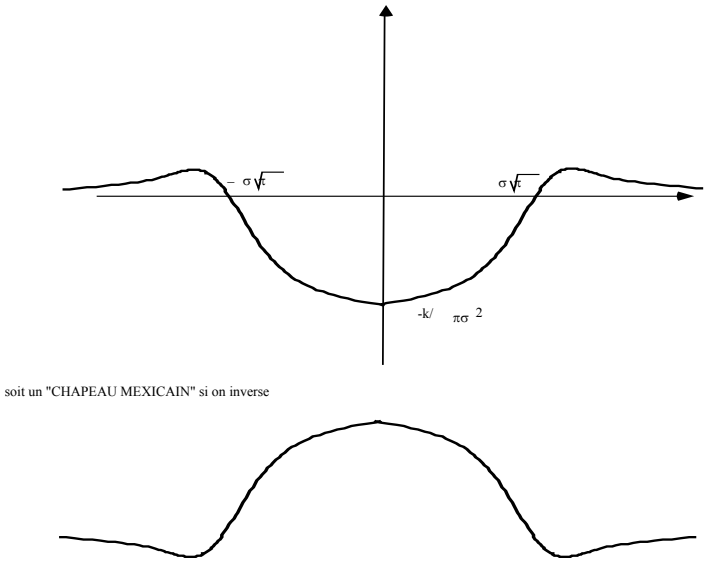
$$\frac{\partial f}{\partial x} = k \times \left(-\frac{2x}{2\pi\sigma^2} \right) \times e^{-\frac{x^2}{2\pi\sigma 2}}$$

$$\frac{\partial^2 f}{\partial x^2} = k \times \left(-\frac{1}{\pi\sigma^2} \right) \times e^{-\frac{x^2}{2\pi\sigma 2}} + k \times \left(-\frac{x}{\pi\sigma^2} \right) \times \left(-\frac{x}{\pi\sigma^2} \right) \times e^{-\frac{x^2}{2\pi\sigma 2}}$$

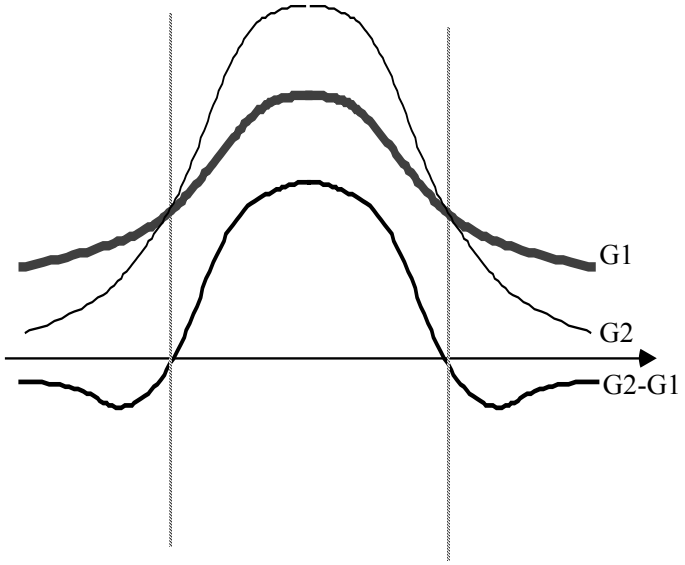
soit

$$\frac{\partial^2 f}{\partial x^2} = \left(\frac{k}{\pi\sigma^2} \right) \times \left(\frac{x}{\pi\sigma^2} - 1 \right) \times e^{-\frac{x^2}{2\pi\sigma 2}}$$

Ce qui donne la fonction suivante :



La forme de ce « chapeau Mexicain » fait qu'on le remplace souvent dans l'approximation du laplacien (en fait l'opposé du laplacien, ...mais comme on ne s'intéresse qu 'au passage par 0 du Laplacien !...) par la différence de deux fonctions gaussiennes de paramètres μ , σ différents.



- Cas d'un signal image :

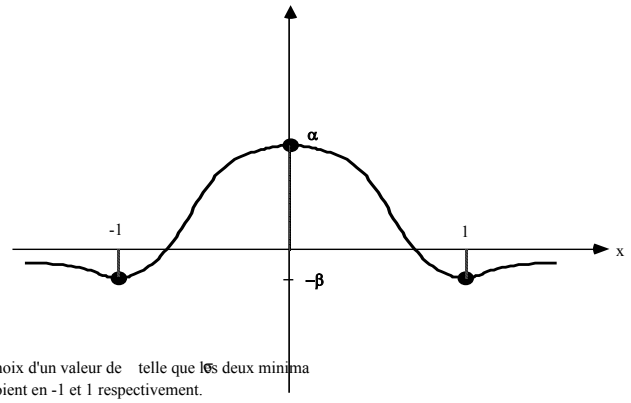
$$f(x,y) = k \times e^{-\left(\frac{x^2+y^{\dot{E}}}{2\pi\sigma^2}\right)}$$

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = \frac{k}{\pi\sigma^2}\left(\frac{\left(x^2+y^2\right)}{\pi\sigma^2} - 2\right) \times e^{-\left(\frac{x^2+y^2}{2\pi\sigma^2}\right)}$$

$$= \frac{k}{\pi\sigma^2}\left(\frac{\left(\rho^2\right)}{\pi\sigma^2} - 2\right) \times e^{-\left(\frac{\rho^2}{2\pi\sigma^2}\right)} \quad \text{en coordonnées polaires}$$

D'où un vrai chapeau Mexicain en 2D

D'où les filtres utilisés pour la convolution de l'image afin d'obtenir une image de laplacien.



Choix d'un valeur de telle que les deux minima soient en -1 et 1 respectivement.
les valeurs de et étant modulées par les valeurs de k et σ

D'où les filtres possibles

- en 1D :

$$\begin{vmatrix} -1 & 2 & -1 \end{vmatrix}$$

- en 2D :

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

Application:

Image initiale

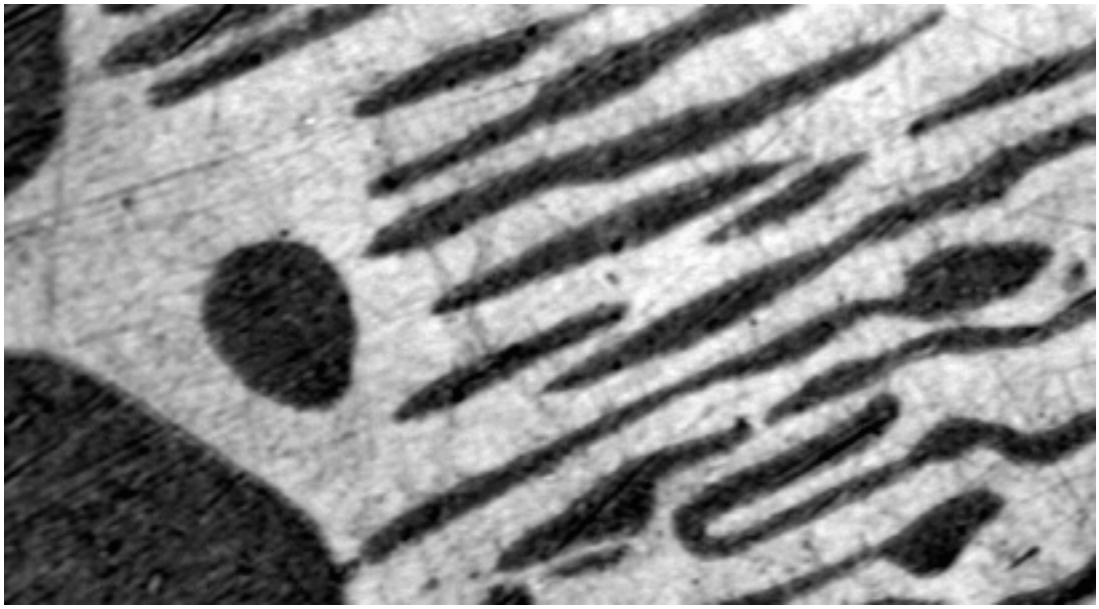
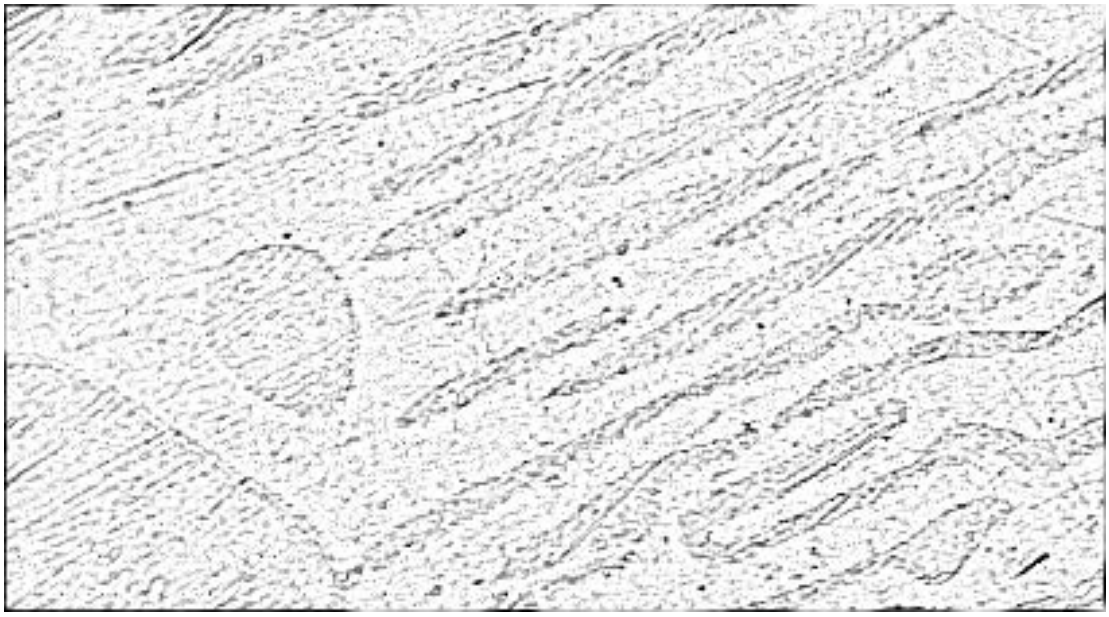


image filtrée



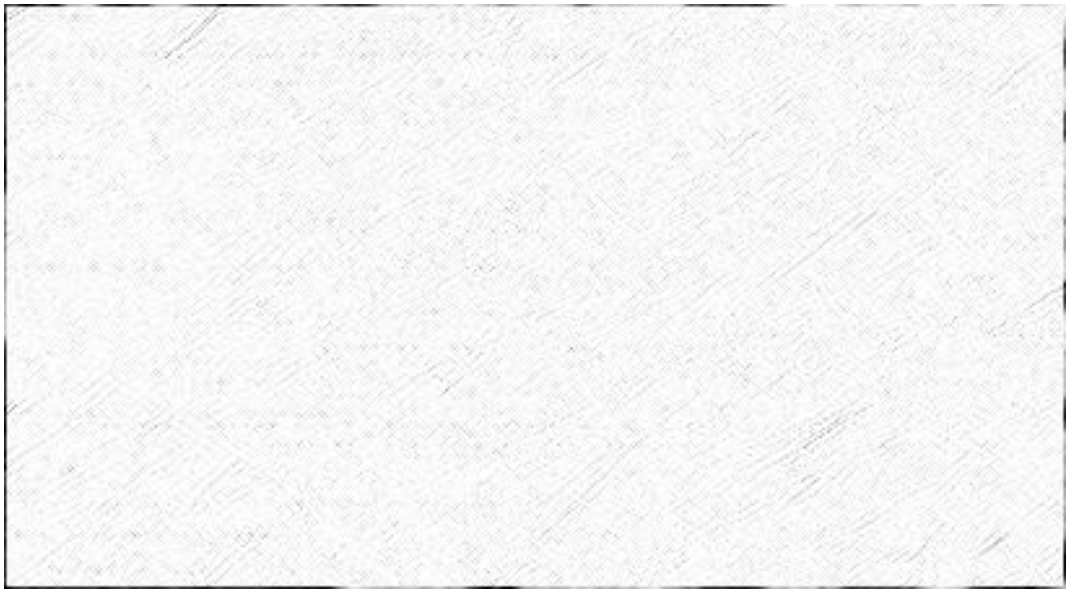
- et les variantes :

$$\begin{vmatrix} 0 & -1 & 0 \end{vmatrix} \qquad \begin{vmatrix} 1 & -2 & 1 \end{vmatrix}$$

-1	4	-1		-2	4	-2
0	-1	4		1	-2	1

image filtrée (premier filtre)

image filtrée (deuxième filtre)



5.1.4.3 Filtres à réponse impulsionnelle séparable

L'objectif poursuivi ici est toujours de convoluer l'image par un filtre pour obtenir gradient ou laplacien par exemple , mais on veut accélérer les calculs...

Prenons l'exemple d'un image de taille N x N
et soient un filtre de taille p X p

Alors le calcul de convolution correspondant nécessitera:

- $p^2 \times N^2$ multiplications

- $(p^2 - 1) \times N^2$ additions

On peut alors réfléchir à des filtres séparables , c'est à dire qu'un filtre 2D (par exemple un masque sur une fenêtre 3 X 3) qu'on applique en une passe sur une image serait « séparé » en deux filtres 1D (voisinage de 3 points alignés par exemple) qui seraient appliqués successivement sur l'image en produisant le même effet que le filtre 3D. Le nombre d'opérations serait nettement diminué.

puisqu'on obtient :

- $p \times N^2$ multiplications et $(p-1) \times N^2$ additions — pour le premier filtre appliqué ligne par ligne

- $p \times N^2$ multiplications et $(p-1) \times N^2$ additions — pour le second filtre appliqué colonne par colonne sur le résultat du filtrage précédent

soit au total

- $2 \times p \times N^2$ multiplications et $2 \times (p-1) \times N^2$ additions — pour le premier filtre appliqué ligne par ligne

exemple : image 512 x 512 et filtre 3 x 3

- filtre non séparable

2 359 296 multiplications
2 097 152 additions

- filtre séparable

1 572 864 multiplications
1 048 576 additions

Approche théorique de la séparabilité des filtres 2D :

Un filtre dont la réponse impulsionnelle est la fonction f(x,y) est dit séparable si

$$f(x,y)=f_1(x) \times f_2(y)$$

L'intérêt est bien sûr l'amélioration du temps de calcul, la possibilité d'adapter les filtres si il y a des bruits directionnels dans l'image, c'est aussi la possibilité de filtres récursifs.

On cherche donc:

$$I * f(x,y) = I* (f_1(x) \times f_2(y))$$

or $$I* (f_1(x) \times f_2(y)) = (I*f1(x)) * f2(y)$$

Ce qui signifie qu'on applique successivement le filtre f1 ligne par ligne , puis le filtre f2 colonne par colonne.

Démonstration :

$$I*\left(f_1(x)\times f_2(y)\right)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} I(x',y')\times f_1(x-x')\times f_2(y-y')\times dx'\times dy'$$

$$=\int_{-\infty}^{+\infty}\left(\int_{-\infty}^{+\infty} I(x',y')\times f_1(x-x')\times dx'\right)\times f_2(y-y')\times dy'$$

$$=\int_{-\infty}^{+\infty}\left(I*f_1(x)\right)\times f_2(y-y')\times dy'$$

$$=\left(I*f_1(x)\right)*f_2(y)$$

Dans la plupart des applications on trouve un bruit homogène dans toutes les directions, alors on peut choisir

f1 = f2 = s

d'où

I * f(x,y) = I * s(x) * s(y)

Voyons alors comment calculer gradient et laplacien dans ce cas particulier
Notons :

$$Gx = \frac{\partial(I * f)}{\partial x} = I * \frac{\partial f}{\partial x}$$

$$or \quad \frac{\partial f}{\partial x} = \frac{\partial(s(x) \times s(y))}{\partial x} = s'(x) \times s(y) + s(x) \times 0$$

$$soit \quad Gx = I * \left(s'(x) \times s(y)\right)$$

de même on a

$$G_{xx} = \frac{\partial^2 (I * f)}{\partial x^2} = \frac{\partial\left(I * \left(s'(x) \times s(y)\right)\right)}{\partial x} = I * \left(s''(x) \times s(y)\right)$$

on en déduit:

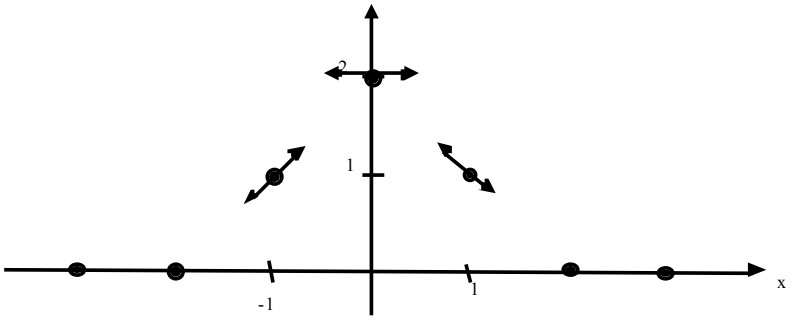
$$\nabla^2 I = I * \left(s''(x) \times s(y) + s(x) \times s''(y)\right)$$

Exemples d'opérateurs séparables :

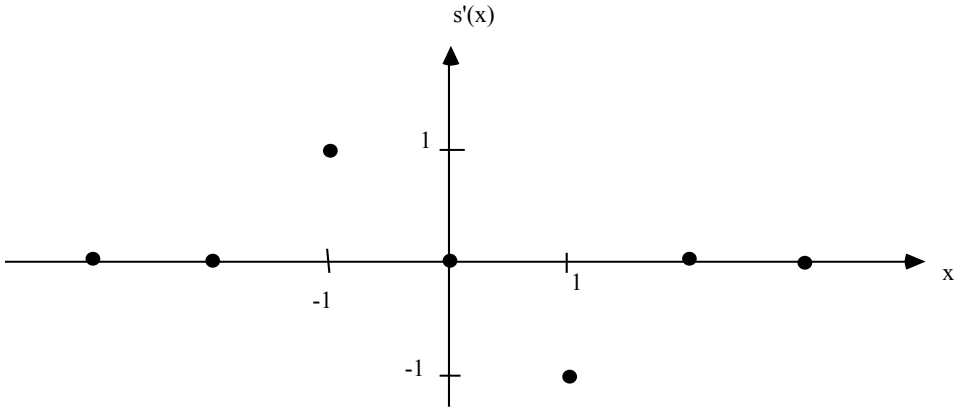
Les masques de Sobel

$$SOBEL1 = \begin{vmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{vmatrix} \qquad SOBEL2 = \begin{vmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{vmatrix}$$

Soit s(x) = (1 2 1)



Alors $s'(x) = (1 \quad 0 \quad -1)$



De même

$$s(y)= \begin{vmatrix} 1 \\ 2 \\ 1 \end{vmatrix} \qquad s'(y)= \begin{vmatrix} 1 \\ 0 \\ -1 \end{vmatrix}$$

alors

SOBEL1=s'(x) * s(y)	SOBEL2=s(x) * s'(y)

remarque : s'(x) est un filtre de type gradient alors que s(y) est un filtre de type lissage

Le calcul de l'image résultat de la convolution par le masque SOBEL1 soit

$$\begin{vmatrix} 1 & 0 & -1 \end{vmatrix}$$

2 0 -2
1 0 -1

donne le même résultat que l'image convoluée ligne par ligne par le masque s'(x) = | 1 0 -1 |, l'image obtenue étant alors convoluée par le masque

$$s(y) = \begin{vmatrix} 1 \\ 2 \\ 1 \end{vmatrix}$$

preuve par le calcul :

$$I * (s' * s) = \sum_{x'=-1}^{+1} \sum_{y'=-1}^{+1} I(x - x', y - y') \times s'(x') \times s(y')$$

I*(s'*s) = I(x-1,y-1) x 1 x1
+ I(x-1,y-0) x 1 x 0
+ I(x-1,y+1) x 1 x (-1)
+ I(x,y-1) x 2 x 1
+ I(x,y) x 2 x 0
+ I(x,y+1) x 2 x (-1)
+ I(x+1,y-1) x 1 x 1
+ I(x+1,y) x 1 x 0
+ I(x+1,y+1) x 1 x (-1)

= I(x-1,y-1)
+ 2 x I (x-1,y)
+ I(x-1,y+1)
- I(x+1,y-1)
- 2 x I(x+1,y)
- I(x+1,y+1)

Ce qui est identique au résultat de la convolution de l'image I par le masque SOBEL1

I(x-1,y-1) **1**	I(x,y-1) **0**	I(x+1,y-1) **-1**
I(x+1,y) **2**	I(x,y) **0**	I(x+1,y) **-2**
I(x-1,y+1) **1**	I(x,y+1) **0**	I(x+1,y+1) **-1**

5.1.5 Améliorations dans la recherche des points contours à l'aide des gradient et laplacien

Jusqu'à maintenant les méthodes proposées consistaient à calculer le gradient (par exemple) en tout pixel de l'image , puis de définir un seuil S global sur l'amplitude G du gradient permettant de classer les points en deux paquets.

- G > S ---> le pixel est un point candidat contour
- G <= S ---> le pixel n'est pas un point contour

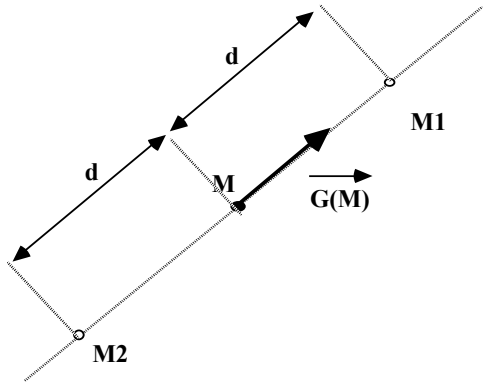
En partant du point de vue que plus l'amplitude du gradient est élevée, plus on a de chances que le point soit un point contour.

Mais le contraste (sur les niveaux de gris, si le paramètre discriminant choisi est le niveau de gris !) dans l'image va varier d'une zone de l'image à une autre zone (ce qui fait que les discontinuités ne seront pas aussi marquées sur tous les contours) . Dans ce cas il est clair qu'un traitement global tel que celui que nous avons décrit risque fort d'oublier certains contours peu marqués.

D'où l'idée de mettre en oeuvre un traitement local où la détection tiendra compte des particularités locales sur le paramètre de discrimination.

Alors plutôt que de chercher des pixels de gradient supérieur à un seuil, on va chercher à mettre en évidence les maxima locaux de l'amplitude du gradient.

Exemple de méthode de ce type:



d étant fixé (par exemple 1)

alors le pixel M correspond à un maximum local si

$$G(M) > G(M1) \quad \text{et} \quad G(M) > G(M2)$$

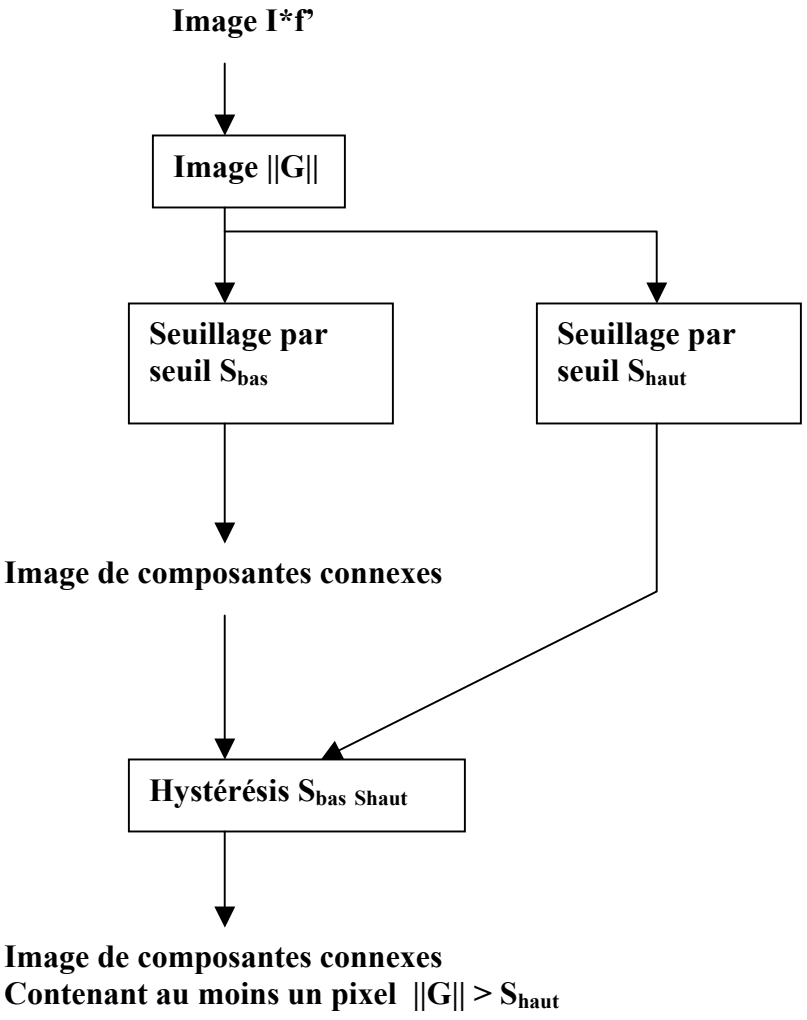
Puis on sélectionne certains des maxima en définissant deux seuils S_{bas} et S_{haut} (avec $S_{bas} < S_{haut}$)

1- On construit alors un image binaire I_{bas} telle que tous les points de l'image tels que $G(M) > S_{bas}$ sont mis à 1 et les autres à 0

2- On construit une autre image binaire I_{haut} telle que tous les points de l'image tels que $G(M) > S_{haut}$ sont mis à 1 et les autres à 0.

On cherche alors toutes les composantes connexes de I_{bas} comportant au moins un point de I_{haut}

On appelle cette méthode **seuillage par Hystérésis**.



amélioration possible:

On réalise l'expansion des composantes connexes dans la direction perpendiculaire au gradient $\vec{G}(M)$ et on choisit un seuil S_{bas} plus petit.

En ce qui concerne l'approche par laplacien:

On calcule donc le laplacien $\nabla^2 (I * f)$ en chaque point à l'aide d'un masque de convolution puis théoriquement on sélectionne les points où le laplacien est nul comme étant des points contour.

Cependant dans la mesure où il peut y avoir du bruit on peut préférer détecter les changements de signe du laplacien

exemple de méthode :

1- calcul d'une image binaire Ip

telle que Ip(M) = 0 si $\nabla^2 (M) > 0$

=1 si $\nabla^2 (M) <= 0$

2- détection des points où le laplacien change de signe

d'où production d'une image Iz

telle que Iz(M) =1 si M appartient
à une transition (0,1) où (1,0)

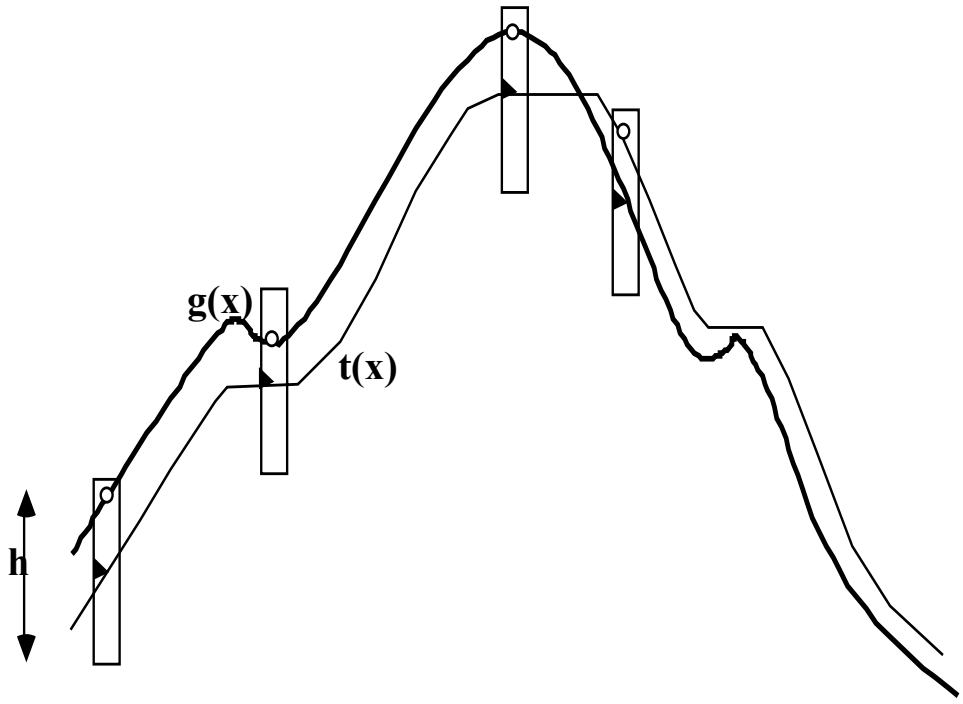
3- choix des points contour en éliminant dans l'image Iz
les points ayant une amplitude de gradient trop faible.

On peut ensuite faire un seuillage par Hystérésis....sur les gradients des points sélectionnés.

On a l'ensemble des points considérés comme des points contours (points de l'image Iz) , on a des points contour « définitifs ».

On garde alors toutes les composantes connexes passant par ces points définitifs....

Compréhension du lissage Hystérésis (voir Duda et Hart pp354) :



C’est une méthode qui permet de repérer les maxima globaux (réels d’une fonction) et d’éviter les maxima locaux.

La différence entre les deux dépendant d’un paramètre (seuil h).

Plus h diminue plus le nombre de maxima considérés comme globaux va grandir.

fonctionnement : on prend la barrette avec un curseur pouvant monter ou descendre dans la barrette de longueur h. Le curseur se déplace sans frottement dans la barrette et la barrette ne se déplace globalement que quand le curseur est bloqué en haut ou en bas. on a alors le cheminement de la barrette indiqué ci-dessus

On voit qu’aux endroits où la fonction t(x) passe de croissant à décroissant on trouve un maximum global !..

6 SEGMENTATION D’IMAGES

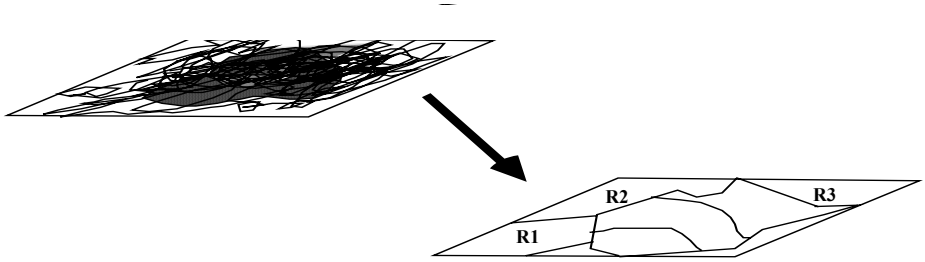
6.1 INTRODUCTION

Il s’agit d’une étape importante dans l’analyse d’une image.

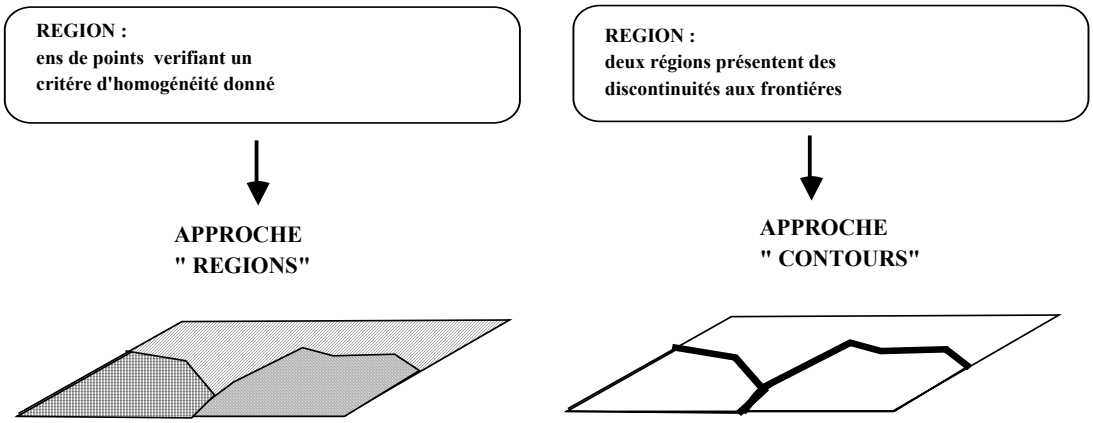
La segmentation va consister à regrouper les pixels de l'image en régions (composantes connexes).

Ces régions vérifiant un critère d'homogénéité (par exemple sur les niveaux de gris ou sur la texture... On cherche par ce traitement à obtenir une description compactée de l'image en régions.

Le traitement suivant consistera probablement à mesurer la forme des régions , certaines de leurs caractéristiques et d'autres part les relations spatiales entre régions par exemple.



On voit que la segmentation peut être abordée de deux points de vue dans la mesure où une région peut être définie par l'ensemble des pixels la composant (approche région de la segmentation) ou bien par les contours de la région approche contour de la segmentation).



6.2 SEGMENTATION: APPROCHE PAR LES CONTOURS

En général cette recherche se décompose en deux étapes:

- détection des points contours
  mais en général ces points sont peu connectés et donc on n'obtient pas de contours fermés pour les régions.

- liaison des composantes connexes obtenues à l'étape précédente : il s'agit de la « fermeture des contours ».

6.2.1 Détection des points contours

La partie détection des contours a déjà été largement décrite lors de l'étude du filtrage linéaire spatial .

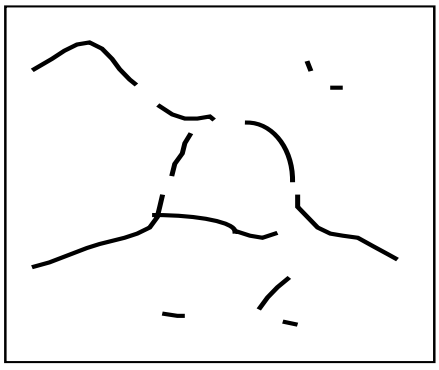
L'image est convoluée par des masques permettant d'obtenir des informations telles que l'amplitude du gradient, ou le laplacien en tout point.

A la fin de cette étape on obtient un classement des pixels de l'image en points contour et points non contour. Cependant pour tous les points on dispose d'informations (par exemple amplitude du gradient et direction du gradient)

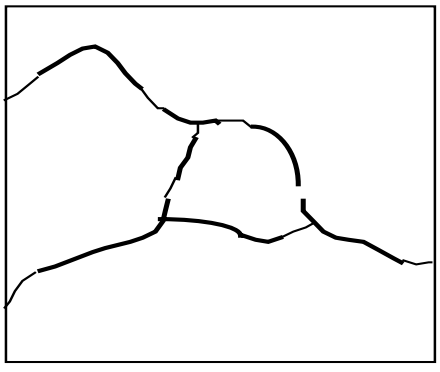
Tous les points considérés comme des points contours forment en fait des composantes connexes qui sont des régions (ce ne sont pas forcément des lignes d'épaisseur « un pixel »). Comme les contours que nous recherchons doivent être d'épaisseur un pixel nous allons procéder à la squeletisation (voir le chapitre sur la squeletisation plus loin !) de ces régions connexes. On obtient maintenant une image de lignes connexes considérées comme les embryons des contours de régions qui doivent ^tre des lignes connexes fermées.

6.2.2 Fermeture des contours

Il s'agit de chercher à fermer les contours initiaux , c'est à dire de continuer les lignes connexes obtenues à l'étape précédente. Eventuellement il faudra supprimer les composantes connexes non significatives.

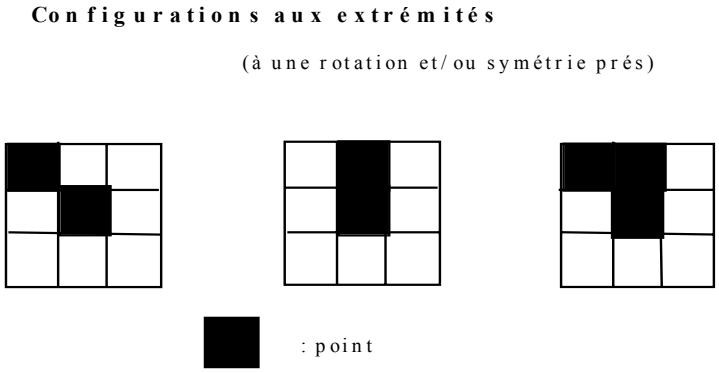


points contour



fermeture des contours

la fermeture de contour a donc pour objectif de rétablir la connexité au niveau des frontières d'une image. Le principe de ces méthodes consiste en général à prolonger les extrémités des chaînes ouvertes (composantes connexes) générées par l'extraction des points contour de l'étape précédente.



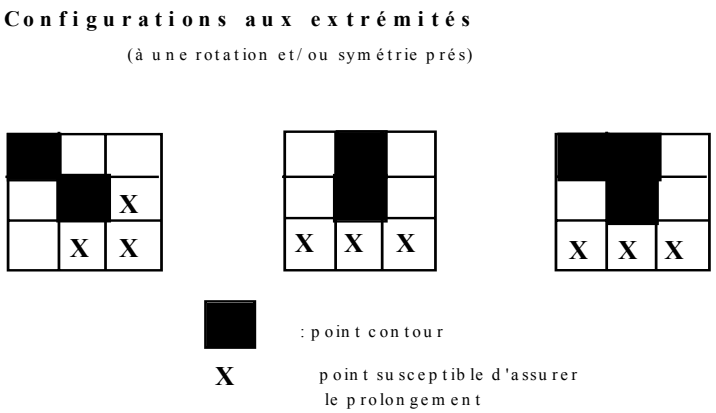
Pour réaliser, la continuation du point contour on va utiliser l'information disponible dans le voisinage de l'extrémité.

Ainsi de façon classique on peut utiliser :

- la direction estimée du contour au point extrémité (cette direction peut être brutalement l'orientation du gradient $+\pi/4$ du seul pixel extrémité, ou bien une orientation calculée sur n points d'une terminaison.

- les modules des gradients au voisinage de ce point associé éventuellement

La direction estimée du contour au point extrémité sert à déterminer les points voisins candidats au prolongement du contour. Ces points sont ceux qui perturbent le moins cette direction.



Notons que les points candidats indiqués représentent une possibilité. Par exemple dans le troisième cas on pourrait rajouter un point pour la continuation du contour.

En supposant maintenant qu'on se restreint aux trois points possibles indiqués, on pourra alors appliquer une méthode de choix basée sur les remarques suivantes:

Un critère de décision peut être de minimiser une fonction de coût , celle ci étant calculée en fonction de l'amplitude du gradient du point candidat par exemple (on peut aussi moduler par la direction du gradient,...).

L'algorithme général de fermeture des contours va donc consister à rechercher dans l'image les extrémités des composantes connexes de points contour et à les prolonger suivant le critère indiqué précédemment. On pourra éventuellement éliminer les points contour isolés
en considérant qu'ils ne peuvent être représentatifs que du bruit.

Exemple de fonction de coût plus élaborée :

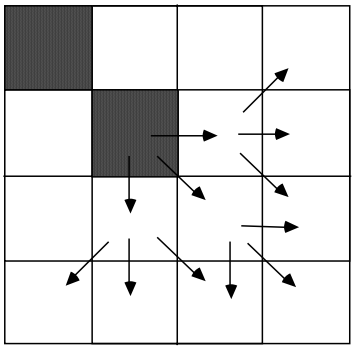
Dans la méthode décrite ci-dessus la fonction de coût est en fait l'inverse de l'amplitude du gradient du point candidat (un des 3 indiqués). Cette méthode n'est pas très robuste car le calcul du coût se fait sur un voisinage trop restreint et de ce fait la prolongation par fermeture peut se faire dans une direction erronée (ceci est bien clair dans l'exemple suivant).

			4
		11	8
	20	19	12
10	8	9	25

les valeurs marquées sont
les amplitudes des gradients

ici la fermeture va donner 20
puis 10 alors que le mieux
semble être 19 25

On peut améliorer la technique de fermeture des contours en décidant de faire intervenir plusieurs niveaux successifs de voisins d'un point extrémité suivant la logique ci après (dans le cas de deux niveaux de points):



Et on calcule une fonction de coût calculée sur deux pixels « consécutifs » basée sur l'amplitude du gradient en chaque point.

On peut d'autre part rajouter des contraintes sur l'amplitude du gradient des pixels concernés pour « fermer » (par exemple un seuil sur l'amplitude proportionnel à l'amplitude du point extrémité)

6.2.3 Fermeture de contour avec "backtracking"

Dans la méthode décrite précédemment on estime qu'un tentative de fermeture de contour échoue quand on tombe sur un pixel permettant de continuer (pixel minimisant une fonction de coût) mais dont l'amplitude de gradient est inférieure à un seuil donné. Dans ce cas on élimine toute la chaîne que l'on avait commencé à construire.

En fait cet échec est peut-être du à une mauvaise orientation du chemin à un moment de la construction de la chaîne (choix d'un pixel de coût minimum localement

mais qui entraîne sur une fausse piste!...) Il serait donc intéressant de pouvoir revenir en arrière si on arrive dans un cul de sac et d'essayer des pixels non optima. On revient donc en arrière pour chercher d'autres chemins.

On peut alors procéder de la façon suivante:

A chaque itération , plutôt que de conserver uniquement le point candidat de coût minimum, on garde tous les candidats vérifiant la condition sur le seuil du gradient donc qui sont susceptibles de conduire à la fermeture de la chaîne.

On développe alors partiellement un arbre de recherche où les noeuds correspondent à des bifurcations possibles dans la construction de la chaîne de fermeture et dont les branches sont les candidats qui orientent la chaîne dans les différentes bifurcations.

Pour chaque noeud , les candidats sont classés par ordre croissant de coût. Ceci signifie que l'on oriente tout d'abord le prolongement de la chaîne vers le candidat de meilleur coût local.

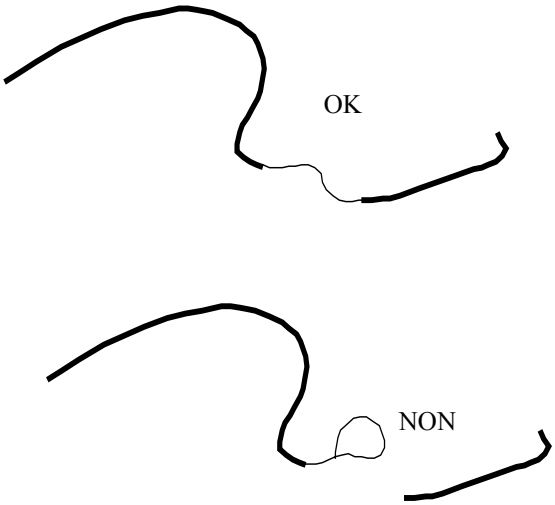
Dans le cas où le chemin emprunté mène à l'échec, alors on remonte au dernier noeud rencontré durant le parcours de l'arbre, en éliminant la chaîne construite jusque là et on redémarre la construction avec le pixel de meilleur coût en second, puis éventuellement avec le troisième candidat (cas où la deuxième tentative pour ce noeud échoue également, et où il y a effectivement trois candidats potentiels qui y sont rattachés).

On voit qu'ainsi le parcours de l'arbre est largement optimisé.

La recherche ne s'arrêtera que dés que le contour aura été fermé.

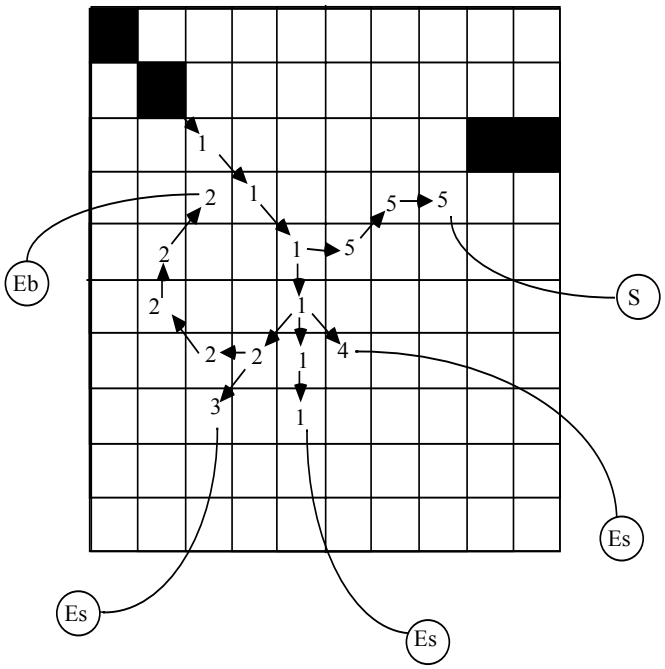
Si tous les chemins examinés mènent à un échec, on va donc remonter jusqu'à la racine (le pixel extrémité qui a déclenché le processus de fermeture) et donc parcourir tout l'arbre de recherche. Alors la fermeture sera considérée comme impossible.

Cependant cette méthode plus souple peut amener plus facilement à réaliser des boucles (le contour prolongé va se refermer sur lui même) ce qui est à éviter. On peut donc rajouter un test qui vérifie qu'on ne crée pas de boucle dans la partie de chaîne construite durant la fermeture .



En effet le but poursuivi n'est pas de créer de nouveaux contours mais d'établir des connexions entre composantes connexes existantes.

Le schéma suivant donne un aperçu de la méthode



Les méthode »s que nous venons de décrire sont donc en fait des techniques d'exploration de graphes où chaque pixel de l'image est considéré comme un nœud dans un

graphe et chaque liaison d'un pixel à l'un de ses voisins est un arc du graphe. La fermeture de contour devient en fait la recherche d'un chemin optimal dans un graphe

Chaque nœud du graphe a des propriétés associées qui peuvent être norme du gradient et direction du gradient.

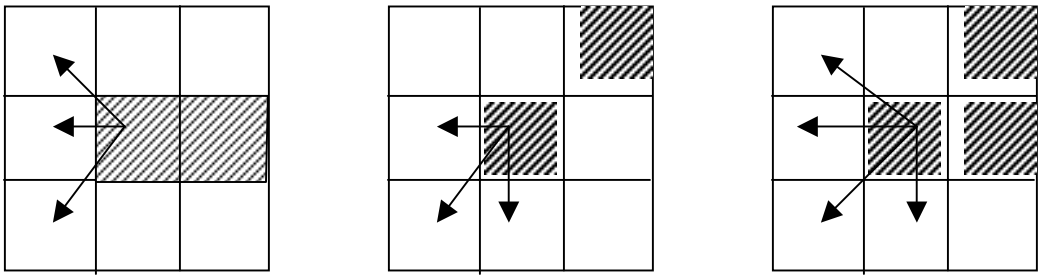
Nous avons décrit une « continuation de contour » à partir d'une extrémité, la condition d'arrêt étant de rejoindre un autre contour ou bien on peut définir une profondeur maximale de recherche au dela de laquelle on considérera la méthode comme en échec.

6.2.4 Exemple de méthode de fermeture de contour (ref……)

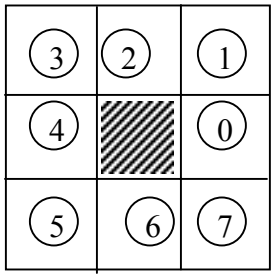
La méthode décrite ici ne comporte pas l'option de « backtracking »

La méthode consiste à considérer le point extrémité d'une ligne connexe et à considérer l'état de ses huit voisins . Suivant la configuration trouvée (en terme de voisins déjà contours ou non contours) on définira au départ les points candidats à examen pour choisir un nouveau point qui sera élu comme nouvelle extrémité.

Quelques exemples de configuration et des points considérés comme à « examiner » :



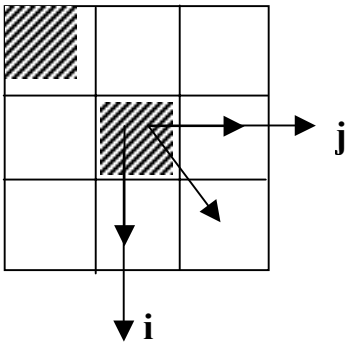
Et prenons les notations suivantes pour « nommer » les huit voisins du point « extrémité » suivant le codage de Freeman (voir le chapitre sur le codage des contours :.



Et notons $V = \sum_{i=0}^{7} x_i \times 2^i$ avec $x_i = = 1$ si i est un point « contour » sinon $x_i = = 0$

On suppose avoir préalablement construit une table définissant les continuations possibles en fonction de la valeur de V (soit en fonction de la configuration des huit voisins du point extrémité.

exemple



Ici V vaut $1 \times 2^3 = = 8$

Les points possibles pour la continuation du contour ayant les coordonnées relatives (0 , 1) (1 , 1) (1 , 0)

pour V = = 8 on notera T[V][0] = = 1 signifiant qu'on peut « continuer »

T[V][1] = = (0 , 1)

T[V][2] = = (1 , 1)

T[V][3] = = (1 , 0)

T[V][4] = = NULL

La table T est donc une table à 256 entrées avec 5 paramètres .

L'algorithme de fermeture des contours prend donc en compte l'image des lignes connexes « contours initiaux » et la table V :

```
DEBUT
    POUR  tous les points (i,j) de l'image  FAIRE
        SI  (i,j) est un point « contour »   ALORS
            m=i ; n=j ;
            V = Σ_{k=0}^{7} x_k × 2^k
            SI  T[V][0] = =  1   ALORS
                Continuer = 1 ; k=longmax ;  /* profondeur max de
                                                fermeture */
                TANTQUE  ((k>0) && Continuer)   FAIRE
                    k -- ;
                    /* recherche meilleur candidat */
                    (m,n)=recherche ( m,n,V,T) ;
                    SI ( (m,n) = = « contour » )  ALORS
                        Continuer = 0 ;
                    SINON
                        (m,n) marqué point « contour » ;
                    FINSI
                FINTANTQUE
            FINSI
        FINSI
    FINPOUR
FIN
```

6.3 FERMETURE DE CONTOURS PAR « CONTOURS ACTIFS »

6.4 SEGMENTATION: APPROCHE PAR LES REGIONS

6.4.1 Segmentation en régions par seuillage

Dans ce cas on fait l'hypothèse que l'objet que l'on cherche à mettre en évidence (région) est défini par une caractéristique d'homogénéité (caractéristique , par exemple le niveau de gris, appartenant à un intervalle défini par un seuil-bas et un seuil-haut).

Exemple:

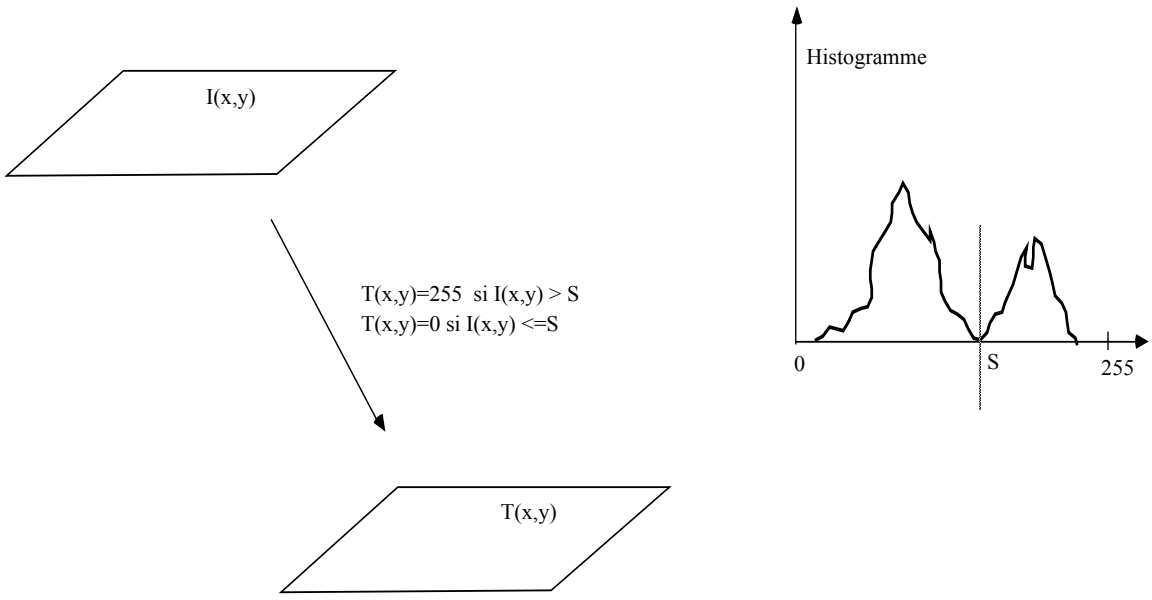
Dans une image SPOT XS3 , on supposera que l'eau correspond aux caractéristiques suivantes:

$$0 <= \text{niveau gris} <= 30$$

L'approche la plus simple est celle où on ne fait intervenir qu'un seuil (on dira que l'image est alors formée d'un fonds et d'un objet)

Le résultat obtenu par seuillage global sur l'image sera alors une image binaire.

On étudiera à priori l'histogramme des niveaux de gris (si le paramètre d'homogénéité est le niveau de gris) et on en déduira un seuil (en partant du point de vue que l'objet doit avoir une distribution des niveaux de gris relativement distincte de la partie « non-objet » : le fonds)



Le gros inconvénient de cette méthode rapide c'est qu'on ne tient pas compte des relations spatiales entre pixels d'une région et par conséquent rien ne permet d'assurer que les pixels sélectionnés seront contigus.

D'autre part des pixels du fonds pourront être intégrés dans la région et des pixels de la région classés en fonds , ceci particulièrement au voisinage du contour et pour les régions bruitées (où le niveau de gris pris en compte n'a pas alors une valeur normale)

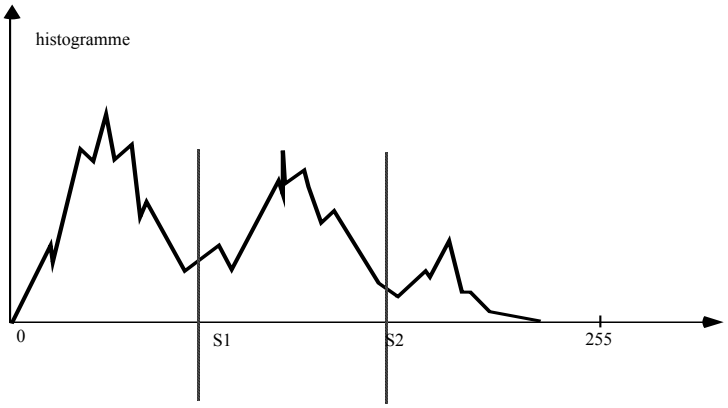
Un autre inconvénient de la méthode de seuillage global sur l'image vient du fait que l'illumination n'est pas forcément constante sur l'image . On peut alors envisager un seuillage local , ce qui signifie que le seuil en tout point de l'image est fonction de l'illumination dans le voisinage.

problème du calcul des seuils:

Le gros problème est bien sur la détection automatique des seuils.

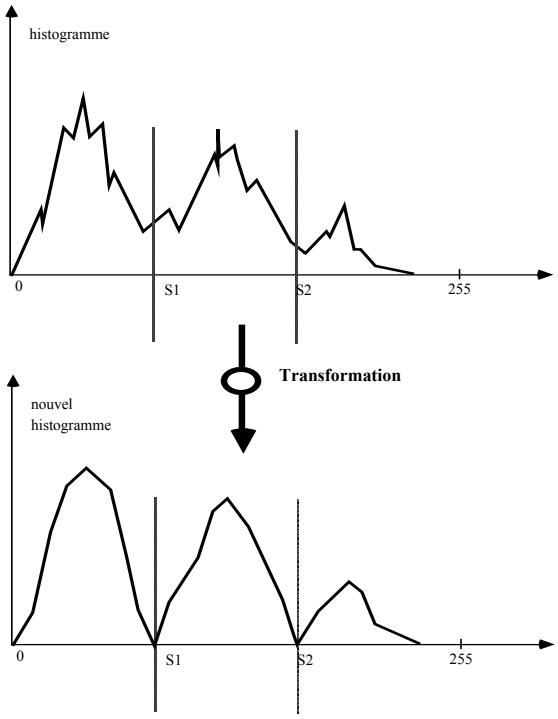
Pour cela on part de l'analyse de l'histogramme de la caractéristique sur laquelle on travaille (en général le niveau de gris) et on cherche alors des modes (sommets) dans l'histogramme qu'on suppose caractéristiques des régions cherchées (puisqu'une région est considérée comme un ensemble de pixels ayant des niveaux de gris voisins).

6.4.1.1 Méthode par recherche de sommets (ou de vallées)



On voit ici que le problème de détection automatique des seuils est difficile car il n'y a pas de véritables vallées (il y a recouvrement des distributions de niveaux de gris), on cherche donc en général des vallées .... mais alors il peut y avoir une multitude de petites vallées.

Une approche intéressante consiste à mettre en évidence les grandes vallées, à les creuser et à « supprimer » les petites vallées sans signification par un traitement directement sur l'histogramme



Pour réaliser ce type de transformation de l'histogramme et mettre en évidence les vallées susceptibles de correspondre à des seuils on peut faire l'hypothèse raisonnable suivante:

- les sommets significatifs de l'histogramme correspondent à priori aux pixels les plus intérieurs de la région (à priori les plus nombreux)

- les vallées significatives de l'histogramme correspondent plutôt aux pixels de la frontière des régions .

On peut alors penser à distinguer les pixels de contour (qui auront donc des amplitudes de gradients élevés) avec les pixels intérieurs des régions (qui auront eux des amplitudes de gradients faibles).

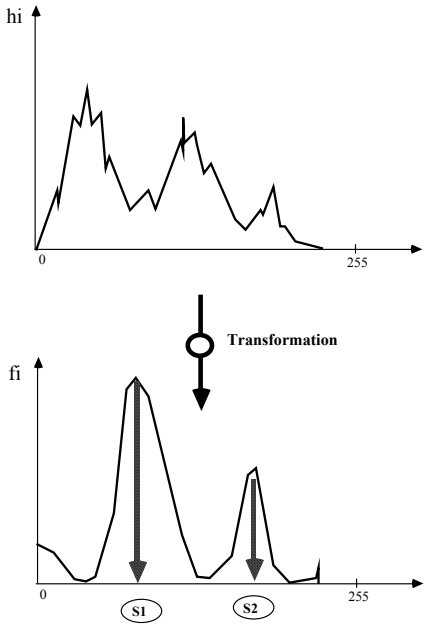
On va donc calculer pour chaque pixel l'amplitude du gradient et appliquer le principe suivant sur l'image I:

$$f_i = \sum_{(x,y) \mid I(x,y)=i} G(x,y)$$

G(x,y) étant l'amplitude du gradient au point de coordonnées (x,y)

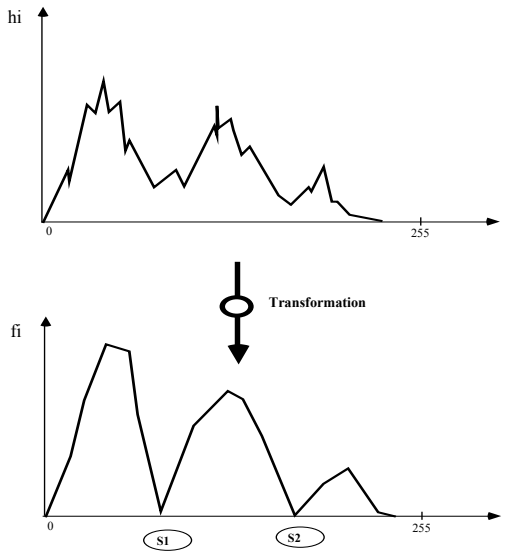
Ainsi pour le niveau de gris i on remplacera h(i) la valeur de l'histogramme pour le niveau i (soit le nombre de pixels ayant le niveau i dans l' image) par la somme des amplitudes de gradient des mêmes points .

On voit immédiatement que si ce sont des points de fort gradient la valeur f'i sera augmentée (et diminuée dans le cas contraire).



D'autres formules basées sur la même logique sont évidemment possibles

par exemple: $$f_i' = \sum_{(x,y)\,/\,I(x,y)=i} \frac{1}{1+G(x,y)}$$



ou bien encore

$$f_i = \sum_{(x,y)\,/\,I(x,y)=i} \Theta(x,y)$$

$$\text{avec} \qquad \Theta(x,y) = 0 \quad \text{si} \quad G(x,y) \geq Cte$$

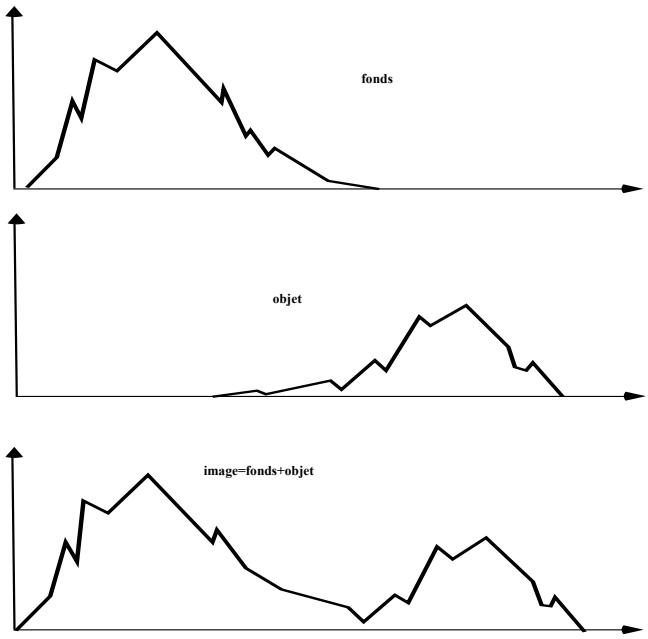
$$= 1 \quad \text{sinon}$$

6.4.1.2 Méthode de détection de seuil par segmentation de l'histogramme

Cette méthode (développée par Otsu) ne s'applique que dans le cas de la segmentation d'image en deux catégories (le fonds et les objets).

On suppose être dans le cas d'un histogramme où les deux populations de pixels se recouvrent partiellement. L'idée va être alors de chercher un seuil permettant d'obtenir les deux populations en minimisant une fonction de coût.

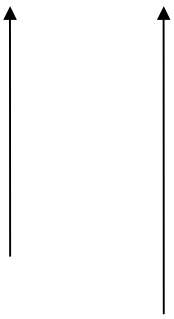
On part du point de vue que l'histogramme de l'image est en fait la somme de deux histogrammes (celui des points du fonds et celui des autres points)



On va alors essayer diverses valeurs de seuil et choisir celui qui sépare l'histogramme de façon optimale en deux segments (qui maximise la variance intersegments ou bien qui minimise une mesure de variance intrasegment. Ce qui peut s'exprimer ainsi:

En supposant que le nombre de niveaux de gris est 256 et que l'histogramme est noté h(i) (nombre de pixels ayant le niveau de gris i)

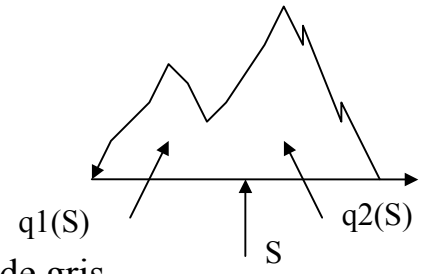
Alors on peut définir une mesure de variance intrasegment par:



$$\sigma^2_{\text{int } ra}(S) = q_1(S) \times \sigma_1^2(S) + q_2(S) \times \sigma_2^2(S)$$

avec

$$q_1(S) = \sum_{i=0}^{S-1} h(i) \quad et \quad q_2(S) = \sum_{i=S}^{255} h(i)$$



$\sigma_1^2(S)$: variance des pixels dont le niveau de gris est strictement inférieur à S

$\sigma_2^2(S)$: variance des pixels dont le niveau de gris est supérieur ou égal à S

$$\sigma^2_{\text{intra}} = \sum_{i=0}^{S-1} h(i) \times (i - \mu_1)^2 + \sum_{i=S}^{255} (i - \mu_1)^2$$

On peut alors essayer toutes les valeurs du seuil S possibles et on garde celui qui rend $\sigma^2_{\text{intra}}(S)$ minimum. On peut aussi chercher une mesure de la « variance intersegments » en fonction du seuil S. On cherchera alors le seuil qui maximise cette mesure (c'est à dire le seuil qui sépare le « mieux » les deux segments)

Une mesure possible est la suivante :

$$\sigma^2_{\textbf{int}\, er} = \sigma^2 - \sigma^2_{\textbf{int}\, ra}$$

où σ^2 **est la variance globale**

ce qui donne

$$q_1(S)\left[(\mu_1(S) - \mu)^2\right] + q_2(S)\left[(\mu_2(S) - \mu)^2\right]$$

soit

$$q_1(S) \times q_2(S) \times \left[\mu_1(S) - \mu_2(S)\right]^2$$

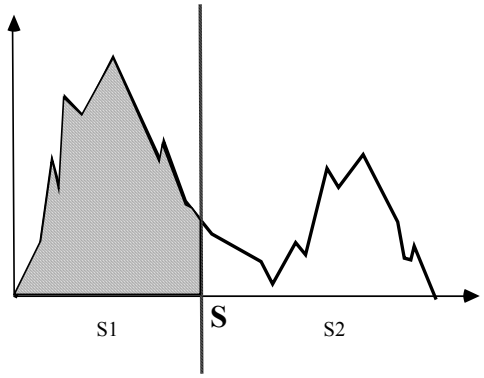
avec

$$\mu_1(S) = \frac{1}{q_1(S)} \sum_{i=0}^{S-1} h(i) \times i \quad et \quad \mu_2(S) = \frac{1}{q_2(S)} \sum_{i=S}^{255} h(i) \times i$$

On obtient alors la méthode suivante:

- choisir un seuil S

- calculer la moyenne de chaque segment sur l'histogramme



- calculer la différence des moyennes au carré

- multiplier par le produit du nombre de pixels du segment S1 et du nombre de pixels du segment S2

et chercher le seuil qui minimise cette valeur.

On peut optimiser le calcul car quand on passe d'un seuil S au suivant S+1 on n'est pas obligé de refaire tous les calculs , on met à jour q1 et q2 de même que les moyennes.

- Recherche du seuil S par une méthode paramétrique:

Dans cette approche on part du même point de vue que précédemment, c'est à dire que l'objet et le fonds sont considérés comme deux populations ayant deux distributions des niveaux de gris différentes. Mais ici on ajoute l'hypothèse supplémentaire suivante:

chacune de ces distributions va être considérée comme Gaussienne

$$N(\mu_1, \sigma_1) \quad N(\mu_2, \sigma_2)$$

On procède alors de la manière suivante:

On essaye plusieurs seuils S et pour chacun des segments obtenus sur l'histogramme on calcule moyenne et écart type et on considère qu'ils ont chacun des distributions gaussiennes.

On fait alors la « somme » de ces deux distributions

$$h_S(i) = \frac{q_1(S)}{\sqrt{2\pi}\sigma_1(S)} \times e^{-\frac{(i-\mu_1(S))^2}{2\times\sigma_1(S)^2}} + \frac{q_2(S)}{\sqrt{2\pi}\sigma_2(S)} \times e^{-\frac{(i-\mu_2(S))^2}{2\times\sigma_2(S)^2}}$$

et on estime la « différence » avec l'histogramme sous forme d'erreur quadratique moyenne :

$$e_S = \sum_{i=0}^{255} \left(h(i) - h_S(i)\right)^2$$

Le seuil S qui minimise cette erreur est considéré comme le bon seuil.

6.4.2 Segmentation par croissance de régions

L'idée de base est la suivante : on suppose disposer de points ou régions « amorces » et on va agréger à ces régions les pixels non encore affectés à une région.

Les points clé de cette approche sont les suivants:

- Choix des points ou des régions amorce (a priori c'est le « coeur des régions » , les pixels dont on est totalement sûrs!)

- règle d'agrégation des points voisins des amorces

prenons un exemple (extrait du livre de Gonzalez et Woods)

0	0	5	6	7
1	1	5	7	8
0	(1)	6	[7]	7
2	0	7	6	6
0	1	5	6	5

(○) : amorce région1

[□] : amorce région2

Prenons maintenant comme règle d'agrégation que l'on va agréger un point à une amorce si la différence de niveau de gris est inférieure à un seuil S. Et on suppose traiter d'abord la région1 jusqu'à blocage (règle non applicable)

ex avec S=3

0	0	5	6	7
1	1	5	7	8
0	1	6	7	7
2	0	7	6	6
0	1	5	6	5

ex avec S=8

0	0	5	6	7
1	1	5	7	8
0	1	6	7	7
2	0	7	6	6
0	1	5	6	5

Exemple de technique:

- calculer l'histogramme et repérer les modes significatifs
- utiliser les pixels de niveaux de gris voisins d'un mode comme amorces

possibilité de travailler sur un vecteur de paramètres (niveaux de gris, couleur, texture...)

possibilité de définir une règle d'agrégation prenant en compte l'évolution lors de la construction de la région (par exemple différence entre le niveau de gris du point voisin et la moyenne des niveaux de gris de la région à l'étape courante)

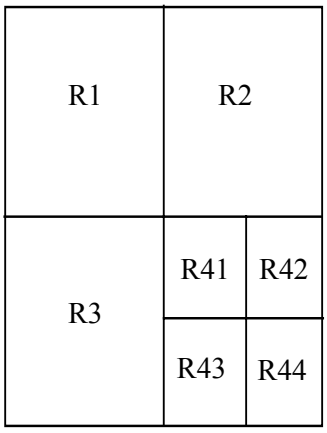
possibilité si l'on connaît le type et la forme, ... de la région cherchée de définir un critère d'arrêt sur l'agrégation.

Méthodes d'Agrégation division (split and merge) :

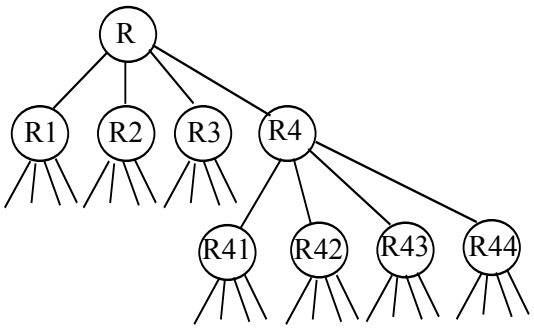
L'idée ici consiste à étendre la méthode précédente et à diviser initialement l'image en un ensemble de régions disjointes (par exemple 4 quadrants si l'image est carrée puis ensuite de diviser ou agréger les régions suivant que des critères de division ou d'agrégation sont vérifiés ou non.

Appelons R l'image entière et R1 R2 R3 R4 .... les régions obtenues par divisions successives .

Si l'on part de l'image R et que l'on divise jusqu'aux pixels on construit ce que l'on appelle une représentation de l'image en quadtree:



Quadtree:



Soit P(Ri) un prédicat logique sur un région Ri donnant comme résultat :

- Vrai si la région satisfait un critère donné d'homogénéité
  (par exemple tous les pixels ont même niveau de gris (durdur à satisfaire!)

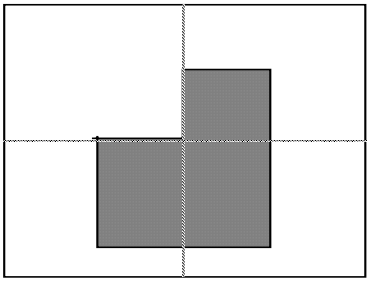
- Faux si la région ne satisfait pas ce critère

La méthode de « split and merge » peut alors s'écrire:

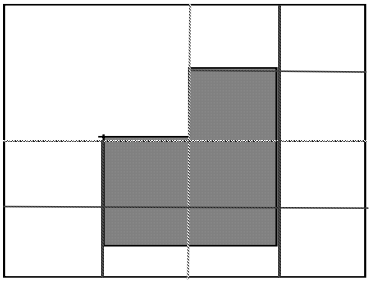
1. Diviser en quatre quadrants disjoints toute région Ri
   où l'on a P(Ri)=Faux

2. Fusionner toutes les régions adjacentes Rj et Rk
   pour lesquelles on a P(Rj U Rk)= Vrai

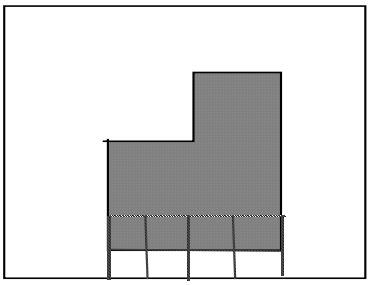
3. Arrêter quand on ne peut plus ni fusionner ni diviser .
   Sinon aller en 1



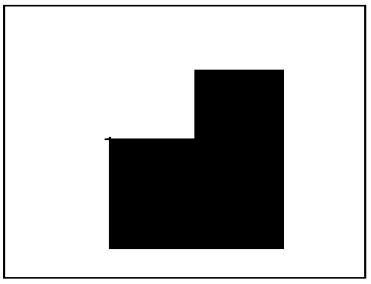
1 Image R
et découpage
en quadrants



2



3



4

7 CODAGE DES CONTOURS ET REGIONS

On suppose ici avoir segmenté l'image et donc avoir identifié les régions et leurs contours. Le problème posé est alors la représentation des objets.

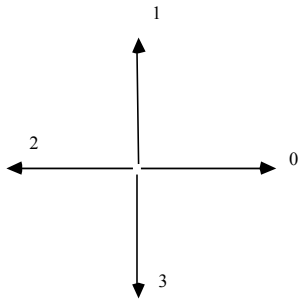
On peut alors représenter les régions en mémorisant la liste des points composant la région, mais c'est lourd ! ou bien la liste des points du contour.

Il existe cependant des méthodes de codage des contours et nous allons donner quelques techniques de codage de ces contours (donc des régions).

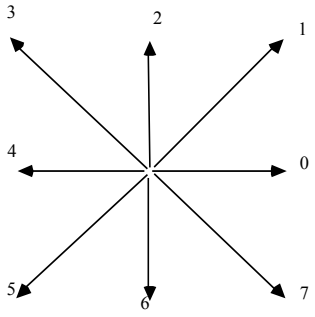
7.1 REPRESENTATION PAR CHAINES DE CODES

Dans cette méthode les contours sont représentées par une séquence connexe de segments de droite de longueur donnée et de direction donnée.

cette représentation est basée sur une 4-connexité ou une 8-connexité des segments. Chaque segment est codé suivant le schéma suivant (codage de Freeman) :



4-connexité

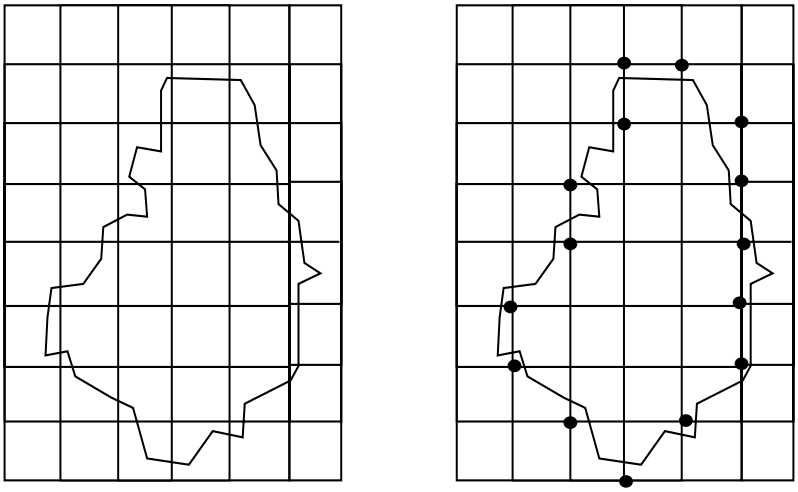


8-connexité

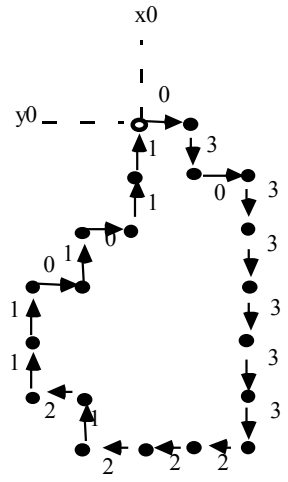
Pour coder un contour on peut évidemment choisir un pixel de départ sur le contour (dont on note les coordonnées image) puis on suit le contour par exemple dans le sens des aiguilles d'une montre et on note les différents codes de segments.

Cette méthode est très sensible au bruit et erreurs sur le contour. Une solution peut consister à passer à une résolution inférieure et à coder sur cette nouvelle grille.

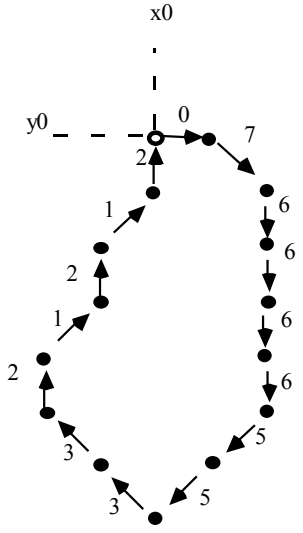
On place donc une grille relativement « grossière » sur l'image et chaque point contour est placé sur le point de la grille le plus proche. On code alors les points de la grille obtenus en 4-connexité ou 8-connexité.



Les résultats suivant que l'on considère la 4-connexité ou la 8-connexité sont les suivants:



4-connexité



8-connexité

soit les chaînes:

4-connexité :(x0,y0) , **03033333222121101011**

8-connexité : (x0,y0), **076666553321212**

On remarque bien sûr que la chaîne dépend du point de départ choisi. Si l'on veut normaliser le résultat on peut procéder par exemple ainsi :

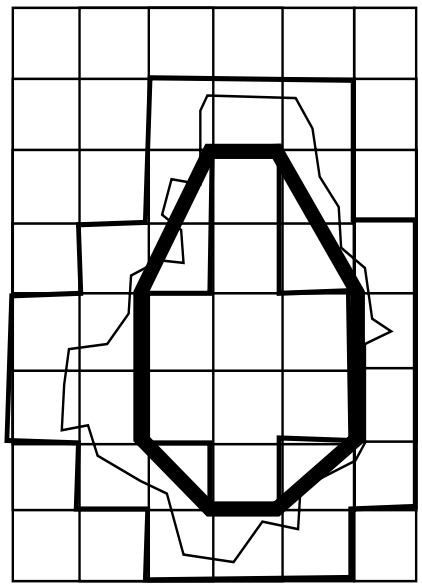
essayer tous les points de départ possibles et choisir la chaîne qui donne le « nombre entier » le plus petit., on évite ainsi le problème de mémoriser le point de départ.

7.2 APPROXIMATION POLYGONALE

Il va s'agir ici de remplacer un contour par une approximation polygonale, ce qui permettra de coder le contour par une suite de vecteurs.

Une possibilité pour réaliser cela est de considérer un quadrillage comme pour le codage précédent puis de noter tous les carrés contenant au moins un point contour. Le principe est alors de faire comme si on mettait un élastique dans cette suite de carrés et de voir quelle position il prend.

Voir ci-dessous:



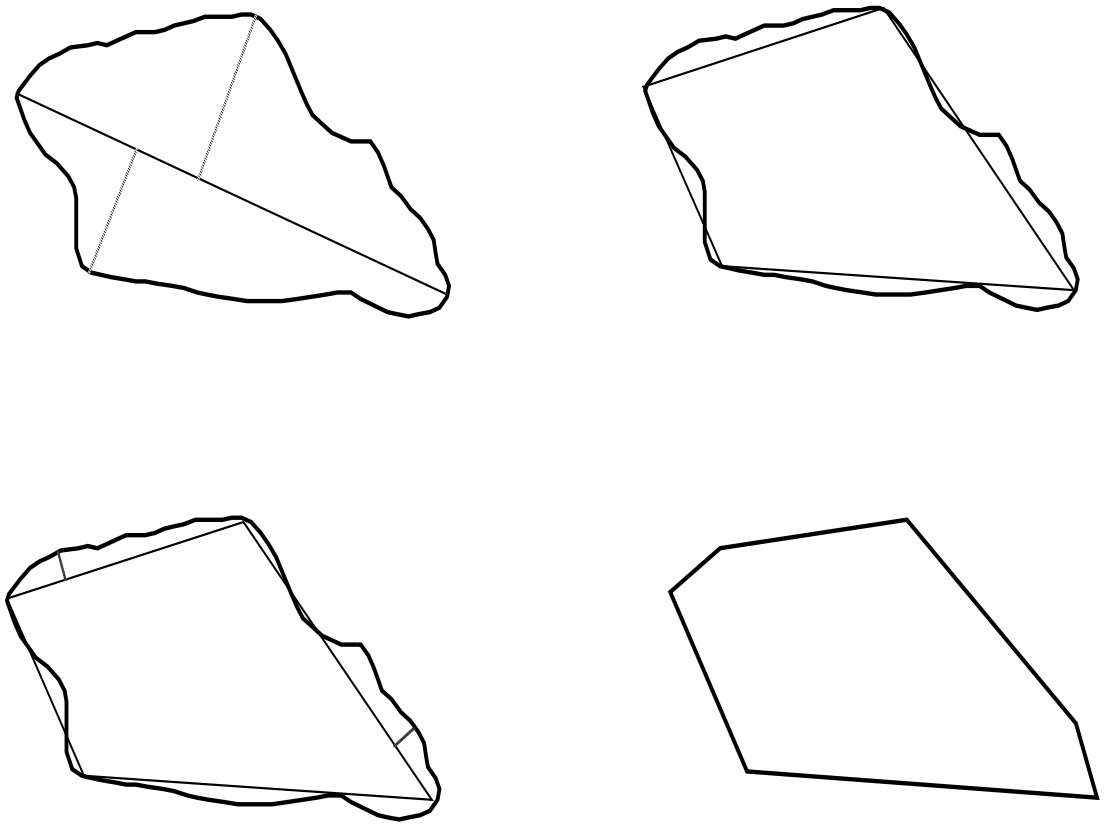
Il est clair que plus le quadrillage est grossier plus l'approximation polygonale sera grossière.

Cependant cette méthode n'est pas très simple à programmer

Une autre approche consiste à partir d'un segment (par exemple le plus long possible joignant deux points du contour. Ce segment définit une droite qui découpe la région en 2. Dans chaque demi-plan obtenu on cherche le segment perpendiculaire à la droite , partant de cette droite et joignant un point du contour le plus long possible.

Si cette longueur dépasse un seuil fixé à priori alors on choisit ce point contour comme nouvelle extrémité « polygonale ». On itére la méthode jusqu'à stabilité.

L'exemple ci-dessous illustre cette méthode:

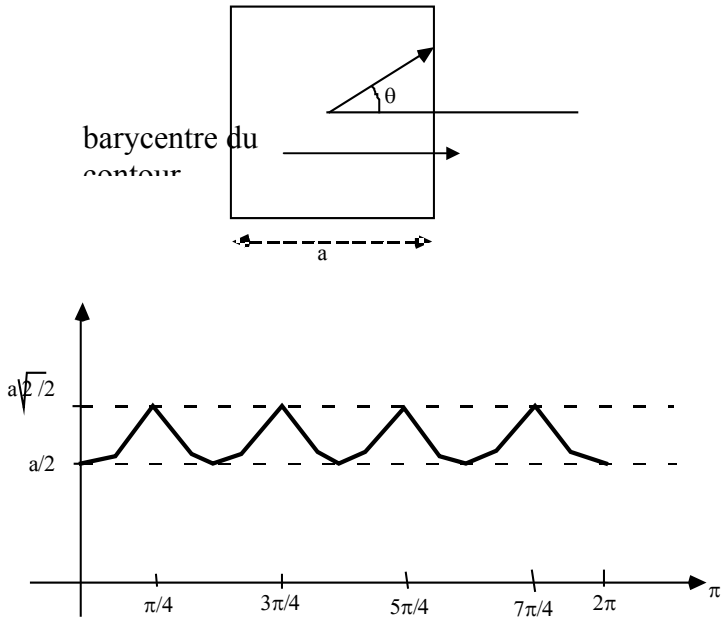


Le résultat est l'ensemble des coordonnées des points extrémités.

Plus on fixe un seuil bas plus on sera prés du contour initial , mais aussi plus on aura de sommets et donc de données à mémoriser.

7.3 SIGNATURE D'UN CONTOUR

Une autre approche consiste à représenter un contour 2D par une « signature » 1D. Le principe consiste à calculer la distance du barycentre du contour à tous les points de ce contour en procédant par rotation centrée sur ce point comme le montre l'exemple ci-dessous:



La signature obtenue est invariante en translation mais pas en rotation et en changement d'échelle

Cependant en choisissant comme point de départ le point du contour le plus éloigné du barycentre on obtient une signature invariante en rotation.

On peut prendre aussi l'axe principal du contour (calculé à partir de la matrice de covariance des points du contour).

Pour rendre la signature indépendante de l'échelle, on peut normaliser la fonction obtenue de façon à ce que le minimum soit 0 et le maximum 1. Cela est sensible au bruit sur le contour on peut améliorer en divisant chaque distance par la variance sur les distances par exemple.

7.4 SEGMENTATION DE CONTOUR ET ENVELOPPE CONVEXE

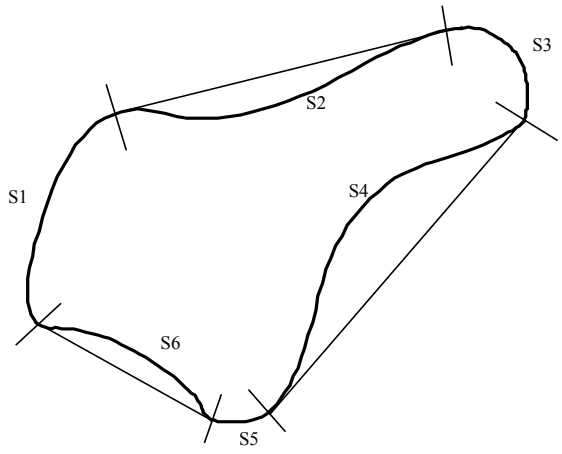
On se pose ici le problème de découper un contour en « segments » qui en général sont des segments de convexité donnée.

Pour cela on peut calculer l'enveloppe convexe du contour (on peut simplement comprendre cette approche en faisant comme si on mettait un élastique autour du contour).

La propriété de convexité correspond au fait que le segment reliant deux points quelconques de l'enveloppe convexe appartient à l'enveloppe.

Une autre définition est que l'enveloppe convexe est le plus petit ensemble convexe contenant le contour.

La segmentation du contour peut alors être réalisée en prenant en compte les points où l'enveloppe convexe touche le contour.



Cette méthode pose cependant de sérieux problèmes en présence de bruit ce qui fait qu'en général on la fait précéder d'un lissage (par exemple en calculant pour les coordonnées de chaque point du contour une moyenne des coordonnées des points contours voisins) ou bien en réalisant d'abord une approximation polygonale puis en calculant l'enveloppe convexe(ex de calcul pp 534 Gonzalez).

Notons que la différence de surface entre l'enveloppe convexe et la région donne aussi des informations sur la région elle-même.

7.5 SQUELETTISATION

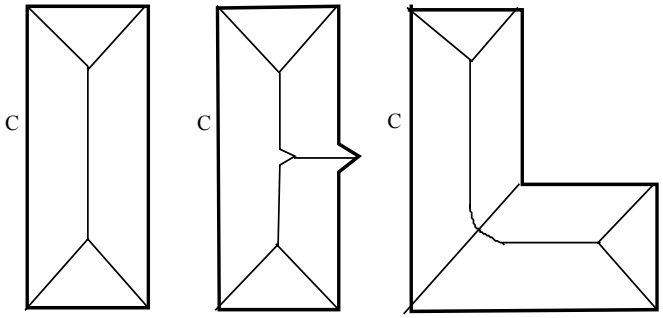
Cette approche consiste à « réduire » la région étudiée à un graphe représentant de façon simplifiée la forme de la région. Cette opération s'appelle une squelettisation de la région. La recherche d'un tel squelette de région peut être effectuée par la méthode de l'axe médian (Blum 67):

Soit une région R de contour C. Alors pour tout point p de la région on cherche le point du contour C le plus proche (au sens d'une distance particulière) et si il y a plusieurs points de ce type alors le point p est dit appartenir au squelette de la région R.

Le résultat de la squelettisation dépend de la distance choisie (City-block, Euclidienne,...).

Le temps de calcul est on s'en doute élevé car il faut calculer la distance de chaque point de la région à tous les points du contour c'est pourquoi on préférera des méthodes itératives consistant à amincir (éroder) petit à petit la région jusqu'au squelette !

exemple avec la distance Euclidienne:



Nous allons maintenant présenter un méthode d'amincissement d'une région permettant d'obtenir un squelette de la région relativement rapidement:

Il s'agit d'une méthode fonctionnant par itération, chaque itération étant composée de deux passes.

Les points traités à chaque passe étant les points contour (c'est à dire les points de la région ayant au moins un de leurs 8-voisins ne faisant pas partie de la région.

Pour décrire la méthode nous allons supposer que la région a ses pixels qui valent 1 et les pixels extérieurs à la région valent 0 (le fonds).

alors si l'on note les 8 voisins d'un point p1 tel qu'indiqué ci-dessous :

p9	p2	p3
p8	p1	p4
p7	p6	p5

avec p_i qui vaut 0 ou 1

Chaque itération prend alors la forme suivante:

- Etape1 :

un point contour est effacé (mis à 0) s'il satisfait les conditions suivantes:

2 <= N(p1) <= 6
S(p1) = 1
p2 . p4 . p6 = 0
p4 . p6 . p8 = 0

avec N(p1) nombre des voisins non nuls de p1

N(p1)=p2+p3+p4+p5+p6+p7+p8+p9

et S(p1) est le nombre de transitions 0-1 dans la séquence ordonnée p2,p3,p4,p5,p6,p7,p8,p9

exemple :

0 0 1

1 p 0 N(p)=4 et S(p)=3

1 0 1

- Etape 2 :

comme l'étape 1 mais les conditions N° 2 et 3 sont remplacées par

$$p2 \cdot p4 \cdot p8 = 0$$
$$p2 \cdot p6 \cdot p8 = 0$$

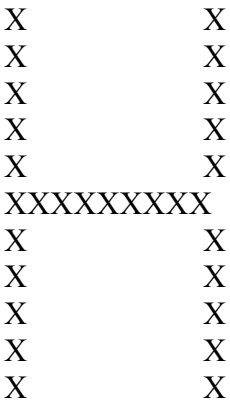
Lors de l'itération les points de contours qui peuvent être effacés sont simplement notés et ils ne sont mis à 0 que lorsque tous les points du contour auront été passés en revue.

résultats sur l'image suivante:

PASSE 1 Itération1:

```
OXXXXXXXO      OXXXXXXXO
XXXXXXXXO      XXXXXXXXO
XXXXXXXXO      XXXXXXXXO
XXXXXXXXO      XXXXXXXXO
XXXXXXXXO      XXXXXXXXO
XXXXXXXXO      XXXXXXXXO
XXXXXXXXXXXXXXXXXXXXO
XXXXXXXXXXXXXXXXXXXXO
XXXXXXXXXXXXXXXXXXXXO
XXXXXXXXXXXXXXXXXXXXO
XXXXXXXXXXXXXXXXXXXXO
XXXXXXXXXXXXXXXXXXXXO
XXXXXXXXXOOOXXXXXXXXO
XXXXXXXXO      XXXXXXXXO
XXXXXXXXO      XXXXXXXXO
XXXXXXXXO      XXXXXXXXO
XXXXXXXXO      XXXXXXXXO
XXXXXXXXO      XXXXXXXXO
OOOOOOOOOO      OOOOOOOOOO
```

résultat final:

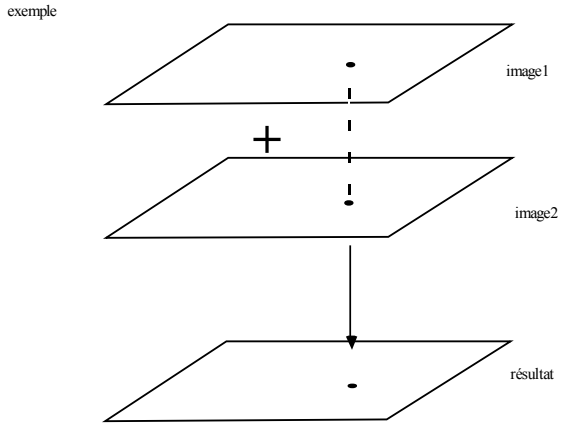

X X
X X
X X
X X
X X
XXXXXXXXX
X X
X X
X X
X X
X X

8 TEXTURE

9 ARITHMETIQUE DE L'IMAGE

L'arithmétique d'images consiste à appliquer les opérations arithmétiques standard et les opérations logiques à une ou plusieurs images. Les opérations sont en fait appliquées pixel par pixel. Il s'agit en fait des opérations les plus simples applicables à des images numériques.

L'image résultat d'une opération arithmétique ou logique est donc telle que la valeur de chaque pixel résultat ne dépend que des valeurs des pixels opérandes correspondants dans les images d'entrée.



les images opérandes doivent donc être de même taille, de plus une des images opérande peut être une image constante.

Des exemples d'applications concernent des images de scènes à des dates différentes (suppression du bruit en ajoutant des images temporelles successives, ou bien détection du mouvement en soustrayant deux images successives dans le temps).

9.1 ADDITION

$$R(x,y)=I1(x,y)+I2(x,y)$$

ou addition d'une constante

$$R(x,y)=I1(x,y)+Cte$$

Dans le cas de l'addition d'images couleur (3 bandes R,V,B) on additionne séparément chacun des canaux (R+R, V+V, B+B)

Si le résultat doit tenir sur un octet , alors il y a plusieurs possibilités:

- si il y a dépassement on prend 255 (saturation)
- suppression des bits de fort poids pour ne pas dépasser 255
  soit(256->0, 257->1,....)
- calcul du min et du max théorique et recadrage de dynamique sur 0 255
  (valeur=((255/(max-min))x(addition des niveaux))

9.2 SOUSTRACTION

$$R(x,y)=I1(x,y)-I2(x,y)$$

la soustraction peut aussi se faire en valeur absolue (par exemple deux images temporelles et la différence donne ce qui a bougé)

$$R(x,y)=abs(I1(x,y)-I2(x,y))$$

On peut aussi retrancher une constante à une image

$$R(x,y)=I1(x,y)-Cte$$

On peut généraliser à la couleur et aux images multispectrales en appliquant les formules précédentes sur chacun des canaux.

La façon d'implanter cet opérateur est liée à la manière de prendre en compte les résultats négatifs ou bien supérieurs à 255 (si l'on travaille sur des octets)

9.3 MULTIPLICATION

$$R(x,y)=I1(x,y)*I2(x,y)$$

mais aussi

$$R(x,y)=I1(x,y)*Cte$$

qui correspond en fait à un changement d'échelle sur les niveaux de gris

9.4 DIVISION

$$R(x,y)=I1(x,y) / I2(x,y)$$

ce qui pose le problème de la division par zéro

mais aussi

$$R(x,y)=I1(x,y) / Cte$$

Cet opérateur fonctionne, un peu comme l'opérateur de soustraction car il permet éventuellement de détecter des changements , mais surtout il permet de donner dans quel « rapport » il y a un changement (en plus ou en moins au niveau des niveaux de gris)

Une autre application concerne les images avec un éclairage non uniforme (par exemple des zones en pénombre) . En prenant deux images de ce type et en faisant le rapport on obtient une nouvelle image où l'effet d'éclairement non uniforme est surpassé.

9.5 COMBINAISON

Cette opération permet de réaliser une combinaison linéaire de deux images (de même taille). Les coefficients de la combinaison sont définis par l'utilisateur.

On peut faire en sorte que le résultat reste dans un intervalle de valeur acceptable:

$$R(x,y)= k \times I_1(x,y) + (1-k) \times I_2(x,y)$$

On voit ici que la valeur de k définit un rapport d' "importance" entre les deux images. Ex d'application :

Si on dispose d'une image des contours et de l'image initiale sur laquelle ces contours ont été calculés on peut combiner les deux images de façon a obtenir une image avec les contours en superposition plus ou moins marquée.

9.6 ET LOGIQUE

L'opération de « et logique » entre deux images peut s'appliquer sur des images binaires et alors le « et logique » est calculé entre les deux pixels directement ou bien sur des images de niveaux de gris et alors l'opération est réalisée bit par bit sur les huit bits de l'octet représentant le pixel.

L'intérêt de cet opérateur est par exemple de calculer l'intersection (et logique) de deux images. Si la deuxième image représente la même scène que la première mais avec des objets ayant bougé, le résultat du « et logique » entre les deux images mettra en évidence ce qui a bougé d'une vue à l'autre.

une autre utilisation possible est le masquage de zones dans une image: Il suffit de prendre une « deuxième » image contenant des pixels de niveau de gris saturé aux endroits d'intérêt et zéro ailleurs et de faire le « et logique » avec l'image 1 pour obtenir un résultat « masqué ».

On peut aussi réaliser du « bit-slicing » qui consistera à mettre en évidence dans une image les pixels ayant le bit n° n à 1

$$(\ R(x,y) = I_1(x,y) + (00001000)_2 \ )$$

9.7 OU LOGIQUE

L'opération ici est le « ou logique » sur deux pixels si les images sont binaires et sur les bits des octets de chacun des pixels si on travaille en niveaux de gris.

Une application est l'affichage combiné de deux images , par exemple l'image elle-même et d'autre part par exemple une image représentant l'histogramme.

9.8 OU EXCLUSIF

Opérations sur les pixels si les images sont binaires et sur les bits des octets si on travaille sur des images en niveaux de gris.

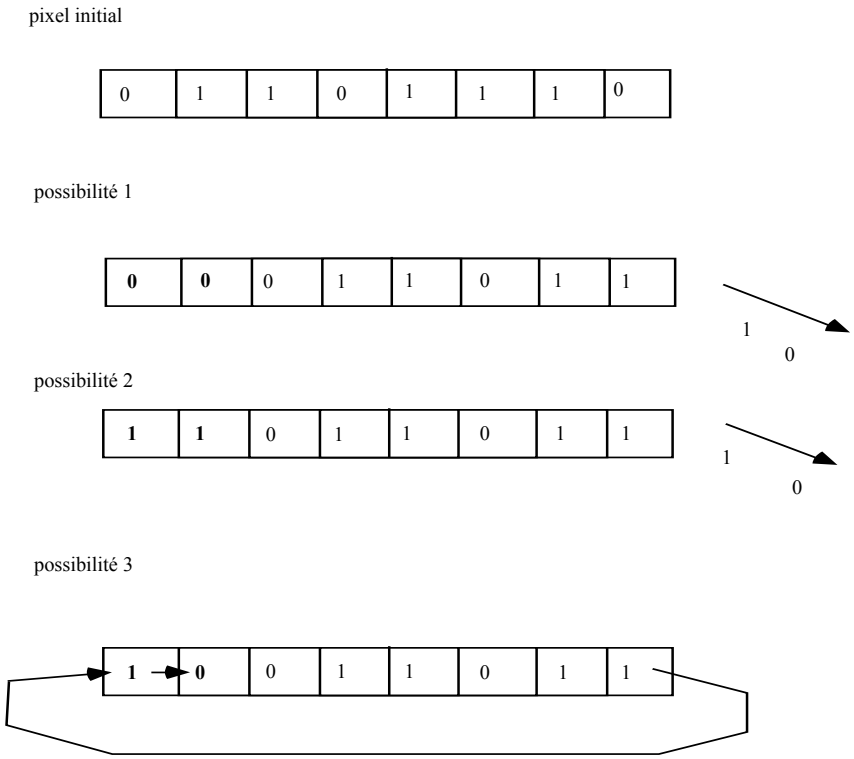
Une application concerne par exemple la détection des changements dans des séries temporelles d'images. En effet les pixels qui ne changent pas fourniront un résultat nul

9.9 DECALAGES BINAIRES

Cette opération ne fonctionne que sur des images en niveaux de gris et consiste à décaler la représentation binaire de chaque pixel d'un certain nombre de positions à droite ou à gauche. cela équivaut à multiplier par une puissance de 2.

possibilités:

ex : décalage de 2 positions à droite



On peut ainsi faire la somme de deux images en décalant chacune des images de 1 position avant d'additionner , on obtient ainsi un résultat normalisé.

On peut aussi augmenter le contraste d'une image en décalant à gauche...

10 GEOMETRIE DE L'IMAGE

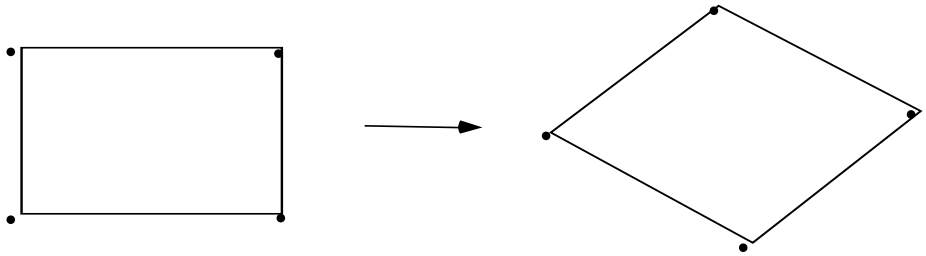
10.1 OPERATIONS GEOMETRIQUES

Une opération géométrique consiste à déplacer l'information en un pixel vers une autre position dans l'image transformée géométriquement.

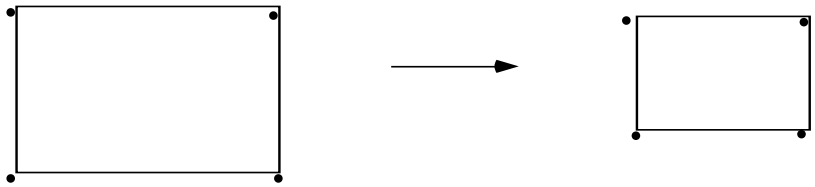
la valeur I1(x1,y1) est transférée à une position (x2,y2).

On peut donner une première palettes d'opérations géométriques sur les images:

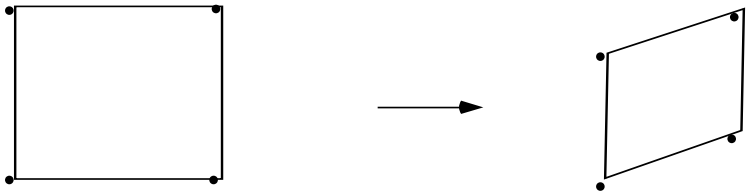
Euclidienne



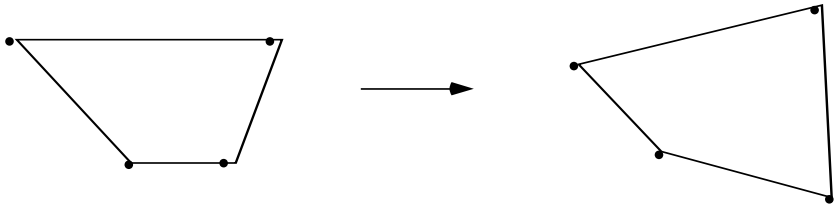
Homothétie



Affine



projective



Dans les transformations Euclidiennes , seules des rotations et translations sont permises... jusqu'aux transformations projectives où un carré peut être transformé en un quadrilatère quelconque.

Ces opérations peuvent être définies par des fonctions polynomiales du premier degré comme celles que nous verrons pour les corrections géométriques de certaines images de télédétection par exemple.

la fonction polynomiale d'ordre 1 prend la forme générale suivante:

$$\begin{vmatrix} x2 \\ y2 \end{vmatrix} = \begin{vmatrix} A11 & A12 \\ A21 & A22 \end{vmatrix} \times \begin{vmatrix} x1 \\ y1 \end{vmatrix} + \begin{vmatrix} b1 \\ b2 \end{vmatrix}$$

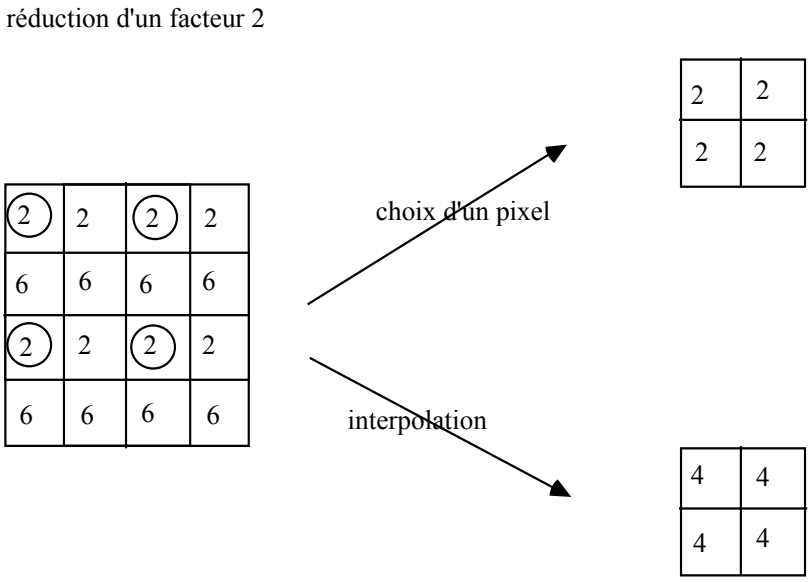
En jouant sur les quatre coefficients de la matrice A et sur les deux coefficients du vecteur B , on peut définir toutes les transformations décrites ci dessus.

10.1.1 Réduction et zoom

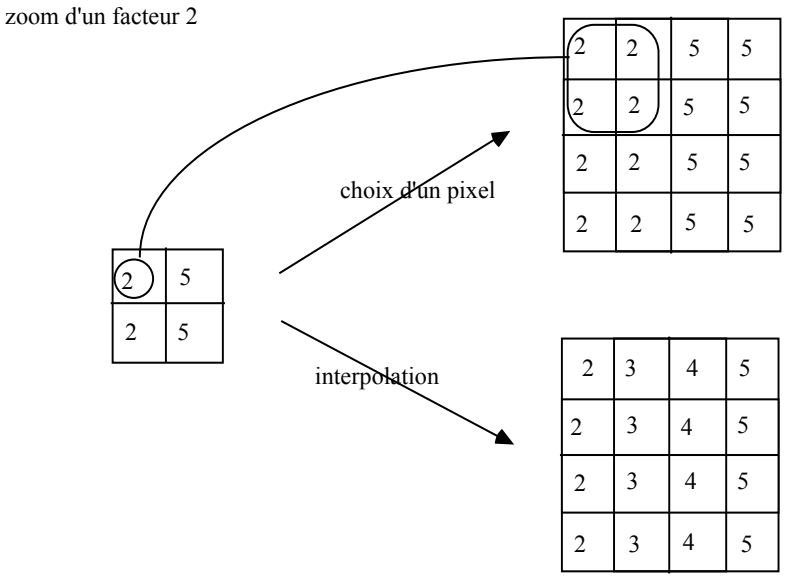
Ces opérations géométriques sont utilisées pour modifier la taille d'une image (ou d'une fenêtre d'image)

La réduction d'image va consister à remplacer un groupe de pixels par un seul pixel. Le niveau de gris correspondant pouvant être un des niveaux de gris d'un pixel du groupe (choisi par une heuristique donnée) ou par une valeur intermédiaire calculée à partir de tous les niveaux de gris des pixels du groupe.

Il s'agit d'un cas particulier de transformation affine (l'homothétie). Cette opération permet de changer l'échelle d'une image (en perdant de l'information) ...



De façon analogue on peut zoomer une image par duplication de chaque pixel ou bien par interpolation:



10.1.2 Rotation d'image

Une rotation est une opération qui déplace chaque pixel de la position (x1,y1) à la position (x2,y2) en réalisant une rotation définie par un centre de rotation (x0,y0) et un angle de rotation Θ .

L'opérateur de rotation peut alors être défini par la formule suivante:

$$\begin{pmatrix} x2 \\ y2 \end{pmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \times \begin{pmatrix} x1 - x_0 \\ y1 - y_0 \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$$

On voit que des pixels risquent de se retrouver rejetés hors des limites de l'image initiale et de même des points de l'image initiale peuvent ne pas être remplis (en général on les met à zéro).

D'autre part , si les coordonnées (x0,y0) et (x1,y1) sont bien entières (numéros de lignes et colonnes) , rien n'empêche les nouvelles valeurs x2 et y2 d'être réelles.

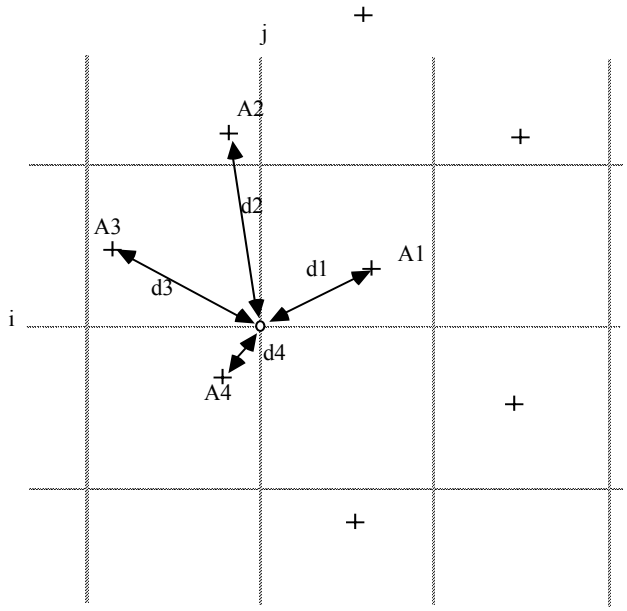
Pour obtenir une image classique il faut donc calculer les niveaux de gris en chaque point (de numéros de ligne et de colonne entiers !) . On peut pour cela procéder de l'une des façon suivantes:

- prendre la valeur du point (x2,y2) le plus proche

- ou bien prendre une combinaison linéaire d'un nombre fixé de voisins les plus proches (le poids affecté étant inversement proportionnel à la distance.

exemple :

en prenant les quatre voisins les plus proches,



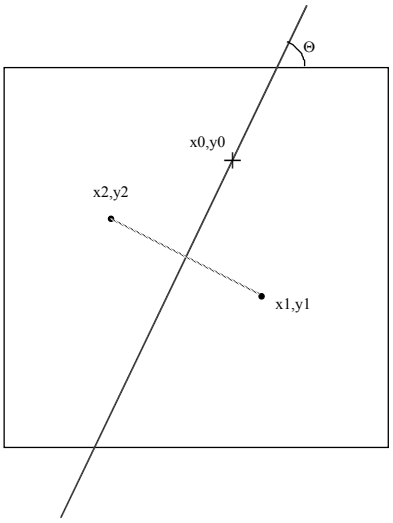
I(i,j) = A1x(1-D1) + A2x(1-D2) + A3x(1-D3) + A4x(1-D4)
puis recadrer la dynamique sur [0, 255].

10.1.3 Symétrie

Il s'agit ici de « transporter » chaque pixel de l'image sur le point symétrique par rapport à un axe de symétrie ou à un point de symétrie choisi.

Comme pour la rotation les points obtenus peuvent avoir des coordonnées réelles et l'image finale devra donc être « interpolée » suivant une heuristique déjà définie dans le paragraphe sur la rotation.

- symétrie par rapport à un axe:



axe vertical d'abscisse x0

$$x2 = -x1 + 2 \cdot x0$$
$$y2 = y1$$

axe horizontal d'ordonnée y0

$$x2 = x1$$
$$y2 = -y1 + 2 \cdot y0$$

axe quelconque passant par (x0,y0) et de direction Θ

$$x2 = x1 + 2 \cdot \Delta \cdot (-\sin\Theta)$$
$$y2 = y1 + 2 \cdot \Delta \cdot (\cos\Theta)$$

avec $\Delta = (x1-x0) \cdot \sin\Theta - (y1-y0) \cdot \cos\Theta$

- symétrie par rapport à un point:

le point symétrique de (x1,y1) par rapport au point de coordonnées (x0,y0) est donné par :

$$x2 = -x1 + 2 \cdot x0$$
$$y2 = -y1 + 2 \cdot y0$$

10.1.4 Translation d'image

Chaque pixel (x1,y1) est transféré à une position (x2,y2) par translation d'un vecteur de composantes (a,b)

$$x2 = x1 + a$$
$$y2 = y1 + b$$

10.1.5 Transformation affine

Souvent dans les images il y a des distorsions géométriques dues par exemple à une position perspective irrégulière du capteur.

C'est très souvent le cas dans les images satellitaires ou aériennes par exemple. Si on veut pouvoir corriger ces distorsions pour arriver à une image plus représentative de la réalité on peut utiliser les transformations affines qui pourront résoudre certains types de distorsions géométriques.

Ceci est appliqué dans le paragraphe sur les corrections géométriques d'images.

Cette transformation affine est une transformation linéaire à deux dimensions.

forme de la transformation affine:

$$\begin{vmatrix} x2 \\ y2 \end{vmatrix} = \begin{vmatrix} A11 & A12 \\ A21 & A22 \end{vmatrix} \times \begin{vmatrix} x1 \\ y1 \end{vmatrix} + \begin{vmatrix} b1 \\ b2 \end{vmatrix}$$

On peut alors retrouver certaines transformations vues précédemment comme des cas particuliers de cette forme générale.

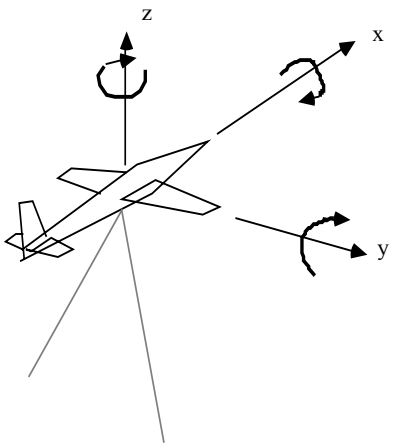
La transformation affine étant définie par 6 coefficients, peut donc être définie par la donnée de trois pixels et de leur homologues.

10.2 APPLICATION AUX CORRECTIONS GEOMETRIQUES

les systèmes optiques (capteurs) de même que leur vecteur (dans le cadre par exemple des images satellite, aériennes....) de par leur imperfection, mouvement, ... peuvent amener des déformations géométriques de l'image.

Il y a donc un problème de mise en correspondance avec un modèle .

Exemple d'un capteur à balayage embarqué dans un avion sujet au tangage et au roulis, sans compter les variations de vitesse de déplacement.



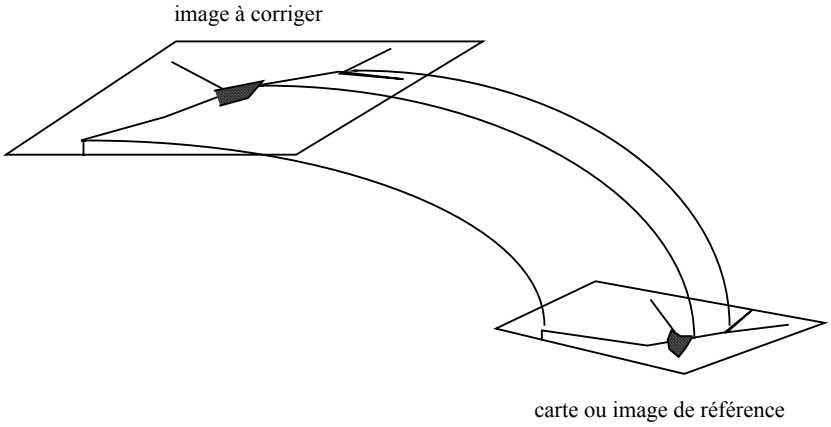
On peut alors se poser des problèmes tels que :

- corriger géométriquement l'image (donc la déformer!) pour la faire correspondre à une carte (avec un type de projection donné , par exemple Mercator)

- corriger géométriquement pour mettre en correspondance avec une autre image de la même scène (par exemple une image SPOT et une image radar).

Bien sûr il existe parfois la possibilité de corriger l'image directement si on a une modélisation enregistrée des paramètres de vol et de prise de vue (enregistrées lors de la prise de vue).

Mais dans la plupart des cas il va falloir faire ce travail par corrections géométriques faites à partir de points d'appuis (points reconnus dans le modèle et dans l'image en même temps) . On cherchera alors à obtenir le modèle mathématique de la déformation qui peut être plus ou moins complexe.

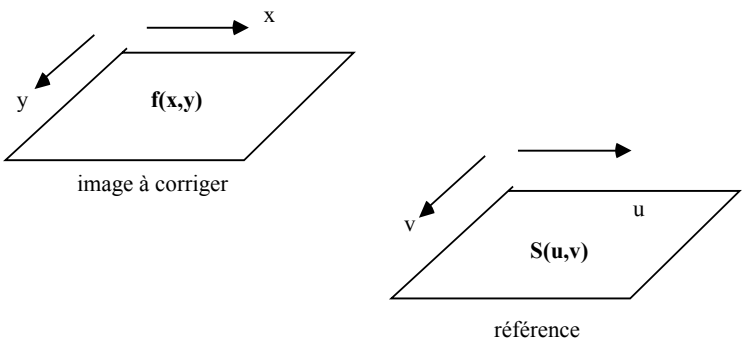


On suppose disposer de points d'appui (on comprend que plus les déformation géométrique sont complexes plus il faut de points d'appui).

Ces points d'appui sont des points homologues connus de façon précise à la fois sur la référence et sur l'image.

A partir de ces points d'appui on essaye de formaliser le modèle mathématique de la déformation . On pourra alors corriger géométriquement tous les points de l'image.

MODELE MATHEMATIQUE :



On suppose donc l'existence d'une transformation T (qui représente la déformation dont a été victime l'image par rapport à la référence telle que :

$$(x,y) = T\,(u,v)$$

On pourra donc théoriquement obtenir la correction géométrique nécessaire en appliquant T^{-1}

$$\text{et } (u,v) = T^{-1} (x,y)$$

Le problème à considérer maintenant est celui de l'estimation de la complexité de la fonction T. En effet une déformation géométrique peut aller d'une simple translation à une déformation élastique aléatoire par exemple.

Il faut donc faire une hypothèse sur le type de transformation mathématique.

On fera en sorte que ce choix permette de modéliser les distorsions géométriques habituelles au type de capteur et de vecteur considéré.

D'autre part le type de fonction choisi doit rester simple à manipuler.

D'où des choix possibles :

- modèle linéaire (avec des formes bilinéaires donc)
- modèle polynomial
- .....

Modèle linéaire :

On suppose alors que la transformation T s'exprime sous la forme suivante:

$$x = A . u + B . v + C$$
$$y = D . u + E . v + F$$

et donc

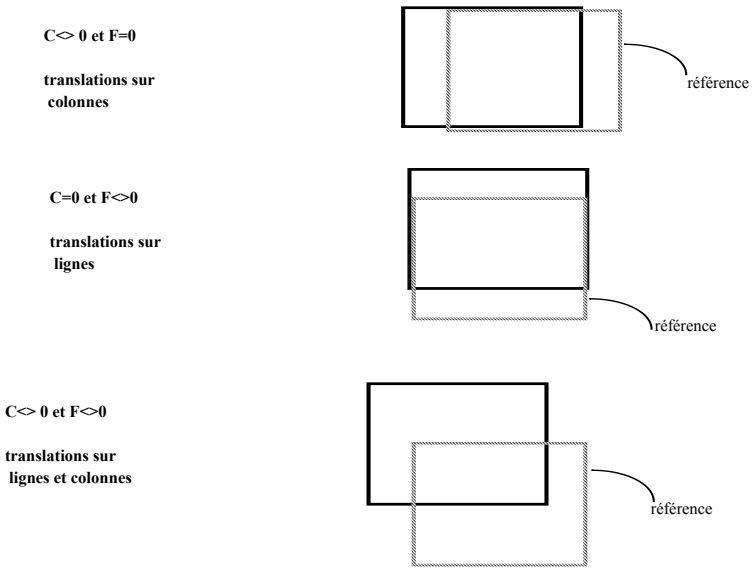
$$f (x,y) = f (A.u+B.v+C , D.u+E.v+F)$$

Le problème est alors de trouver les scalaires A B C D E F

* Cas d'une translation :

alors A=E=1
B=D=0

$$x = u + C$$
$$y = v + F$$

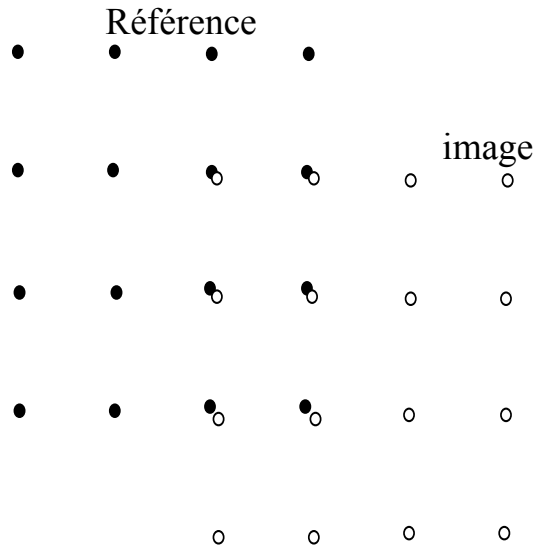


Dans ce cas extrêmement simple on voit immédiatement qu'il suffit de connaître un seul point de contrôle pour trouver C et F

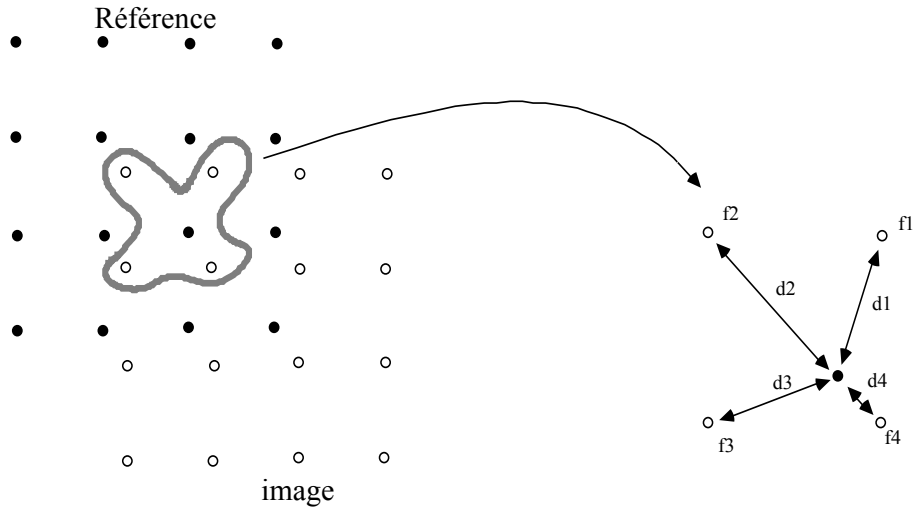
(en effet on a alors $x1 = u1 + C$ et $y1 = v1 + F$ si le point de contrôle image (x1,y1) correspond au point (u1,v1) de la référence)

Deux cas sont cependant à considérer si l'on veut construire l'image S(u,v) corrigée géométriquement:

C et F entiers , alors on est dans le cas de figure suivant:



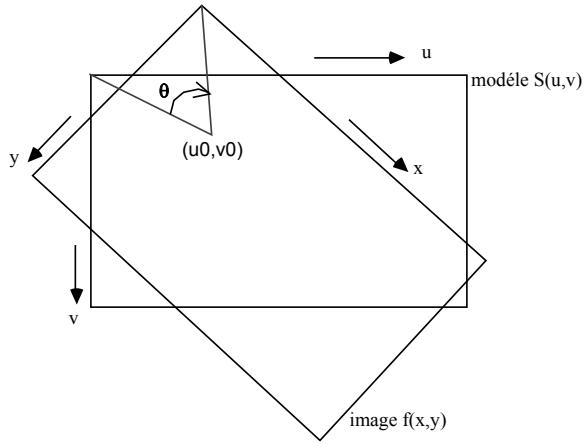
C et F réels :



alors S = fonct(f1,d1,f2,d2,f3,d3,f4,d4)

(par exemple (1-d1)f1+(1-d2)f2+(1-d3)f3+(1-d4)f4 a un recadrage de dynamique prés)

* cas d'une rotation



rotation : angle Θ

et centre C (u_0,v_0) alors $\begin{pmatrix} x \\ y \end{pmatrix} = R_{\Theta,C} \times \begin{pmatrix} u \\ v \end{pmatrix}$

soit,

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \times \begin{pmatrix} u - u_0 \\ v - v_0 \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \end{pmatrix}$$

or

$$\begin{cases} x = A \times u + B \times v + C \\ y = D \times u + E \times v + F \end{cases}$$

$d'où$

$$\begin{cases} x = \cos\Theta \times u - \sin\Theta \times v + (1 - \cos\Theta) \times u_0 + \sin\Theta \times v_0 \\ y = \sin\Theta \times u + \cos\Theta \times v + (1 - \cos\Theta) \times v_0 + \sin\Theta \times u_0 \end{cases}$$

pour trouver les paramètres d'une rotation (Θ, u_0, v_0)

il suffit de deux points de contrôle:

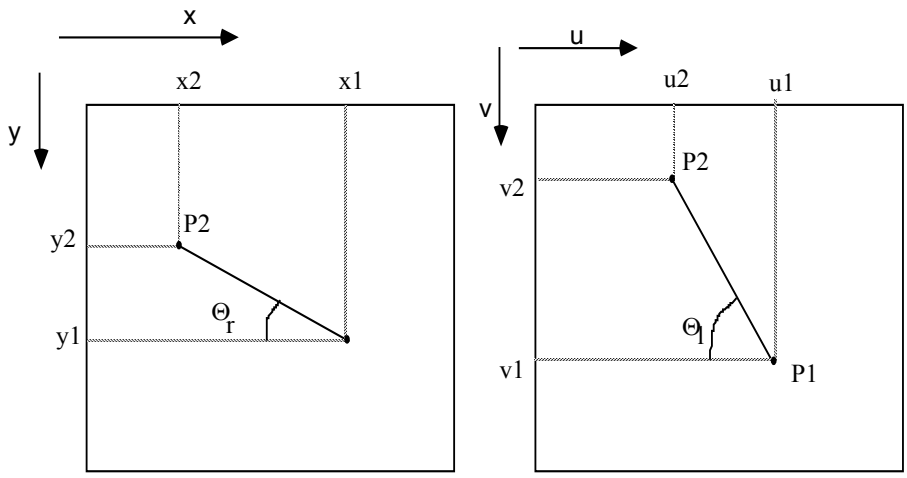


image modèle

d'où,

$$tg\Theta_r = \frac{y1 - y2}{x1 - x2} \qquad \Theta_r = arctg\left(\frac{y1 - y2}{x1 - x2}\right)$$

$$tg\Theta_l = \frac{v1 - v2}{u1 - u2} \qquad \Theta_l = arctg\left(\frac{v1 - v2}{u1 - u2}\right)$$

rotation **:**

$$\Theta = \Theta_l - \Theta_r$$

et donc

$$\Theta = arctg\left(\frac{v1 - v2}{u1 - u2}\right) - arctg\left(\frac{y1 - y2}{x1 - x2}\right)$$

ayant Θ on peut alors obtenir facilement u_0 et v_0

car

$$\begin{cases} x_1 = \mathbf{cos}\,\Theta \times u_1 - \mathbf{sin}\,\Theta \times v_1 + (1 - \mathbf{cos}\,\Theta) \times u_0 + \mathbf{sin}\,\Theta \times v_0 \\ y_1 = \mathbf{sin}\,\Theta \times u_1 + \mathbf{cos}\,\Theta \times v_1 + (1 - \mathbf{cos}\,\Theta) \times v_0 + \mathbf{sin}\,\Theta \times u_0 \end{cases}$$

11 FOURIER ET RESTAURATION D'IMAGES

12 COULEUR DANS LES IMAGES

13 ONT LARGEMENT PARTICIPE...

Digital image processing
R. Gonzalez et R.E. Woods Addison-Wesley

Vision par ordinateur : outils fondamentaux
R. Horaud et O. Monga Hermès

Hypermedia Image processing Reference
R.B. Fisher A. Walker S. Perkins E. Wolfart Wiley & sons

Digital Image Processing lecture
B. S. Morse

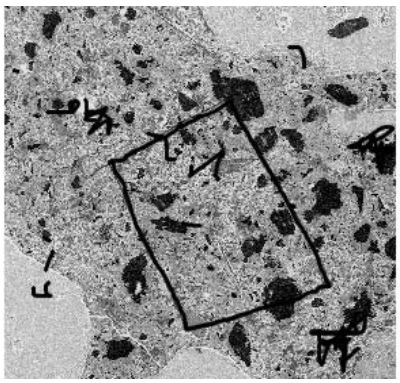
Computer image processing and recognition
E. Hall

Vision tutor
cours en ligne du logiciel APHELION

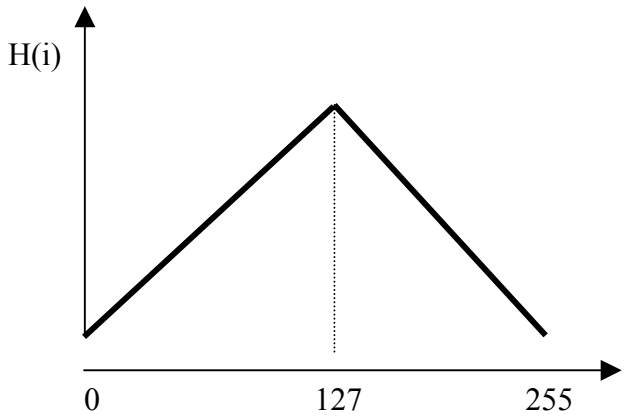
et les thésards de l'équipe Traitement et Compréhension des images

On donne l'image en niveaux de gris ci-dessous.

Décrire en détail la suite des opérations à réaliser pour obtenir à partir de cette image une image binaire où n'apparaisse que le contour du grand quadrilatère (on supposera que ce contour est de type vallée et d'épaisseur 1).



Soit une image 512 x 512, à 256 niveaux de gris. Détailler la méthode (formules permettant une programmation immédiate !) permettant d'obtenir la LUT qui, appliquée à cette image, permette d'obtenir une nouvelle image dont l'histogramme soit le plus proche possible de l'histogramme suivant :



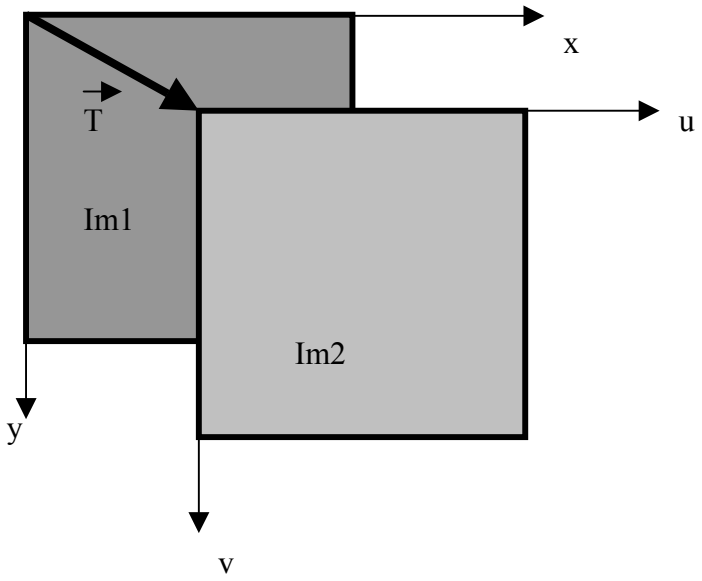
En utilisant des opérateurs de gradient (un vertical et un horizontal), proposez une méthode pour calculer l'orientation et la pente en chaque point d'un MNT (sauf les bords si nécessaire).

L'orientation est définie par un angle orienté relativement à la direction Nord. La pente est un angle orienté par rapport à la verticale.

Proposez une méthode permettant d'obtenir au mieux les grands versants sous forme de régions.

Note: un MNT modèle numérique de terrain est en fait une image ou la valeur en chaque point
est en fait l'altitude du point.

On suppose disposer de deux images satellite de résolution spatiale identique, mais décalées par rapport au sol. On suppose que la "déformation géométrique" entre les deux images est une translation T. Notons {x,y}le repère lié à l'image im1 et {u,v} le repère lié à l'image im2. Le vecteur T est défini par ses coordonnées u0 et v0 dans {x,y}, mais u0 et v0 pouvant être réels, indiquer comment pour un point de coordonnées x et y (entières) on peut obtenir le niveau de gris correspondant à partir de l'image im2 (éventuellement, ce peut être un niveau qui n'existe pas dans l'image im2 !).



Quelle technique permet de mettre en évidence des contours de type échelon, toit et vallée dans une image de niveaux de gris bruitée ?

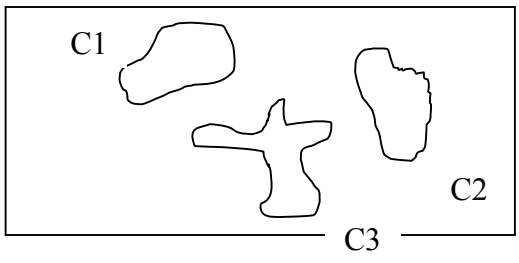
Donner toute la chaîne d'opérateurs de traitement d'images à enchaîner pour découvrir les contours de régions (homogènes en niveau de gris) dans une image monochrome.

Soit une portion d'image satellitaire (noir et blanc) représentant une zone de parcellaire agricole (parcelles supposées rectangulaires). En appliquant un seuillage sur l'image des modules du gradient on suppose avoir obtenu une image binaire où les points à 1 sont censés être des points de contour.

1) Dire comment on peut trouver l'équation des droites qui correspondent aux cotés des diverses parcelles rectangulaires par la méthode de Hough. Détailler l'algorithme correspondant

2) Indiquer ensuite comment obtenir exactement les segments cotés des parcelles.

Avec l'image binaire suivante où les points contours sont supposés à 1 et connus par (abscisse, ordonnée), indiquer comment vous feriez pour comparer automatiquement le contour C1 et les deux contours C2 et C3 et en conséquence pour décider de quelle forme C1 se rapproche le plus (à une rotation prés) ?



1) Proposer une méthode d'accentuation des contours des œufs présents dans cette image.
2) Proposer une chaîne complète d'opérateurs permettant d'obtenir les contours des œufs (prenant en compte ou non l'étape de la question 1)

Dans les deux questions donner tous les détails

Soit l'image suivante, de taille 10 x 10 :

12	16	16	15	15	15	21	20	20	17
16	12	13	13	15	21	20	20	21	20
15	12	13	13	15	21	21	21	18	20
15	15	13	15	16	21	15	18	19	20
14	14	14	15	17	15	15	18	14	16
13	12	14	16	17	15	15	14	14	14
14	14	14	16	15	16	15	19	14	14
13	12	15	16	15	16	15	17	14	14
15	13	15	16	15	16	16	17	14	17
15	15	15	16	16	14	16	17	17	17

1) Donner son histogramme.

2) Recadrer la dynamique de cette image sur [0, 255] et donner la LUT associée. Peut-on revenir facilement à l'image initiale ?

3) Donner l'histogramme et l'histogramme cumulé de la nouvelle image.

4) Donner la méthode permettant de passer de l'image obtenue à une nouvelle image ayant un histogramme égalisé et indiquer l'algorithme (le plus proche possible de la programmation) permettant de réaliser cela. Peut on revenir à l'image initiale ?

QUESTION 2 (4 pts)

Quelle technique permet de mettre en évidence des points-contours de type "rampe" et des points-contours de type "vallée" dans une image de niveaux de gris bruitée ?

QUESTION 3 (4 pts)

En utilisant des opérateurs de gradient (un vertical et un horizontal), proposez une méthode pour calculer l'orientation et la pente en chaque point d'un MNT (sauf les bords si nécessaire).

L'orientation est définie par un angle orienté relativement à la direction est. La pente est un angle orienté par rapport à l'horizontale.

QUESTION 4 (5 pts)

Pour l’image ci-dessous supposée être de taille 1024x1024, proposez une chaîne d'opérateurs avec le paramétrages correspondant pour détecter les tronçons de route rectiligne de longueur au moins égale à 400 pixels. On suppose que les routes sont de niveau de gris élevé.