

Cours de Traitement d'images

**rédigé par Catherine Achard,
sur la base des cours de J. Devars et M. Milgram**

2003/2004

Chapitre 0 : Introduction

L'imagerie est un domaine très vaste parmi lequel, on peut distinguer plusieurs catégories :

Traitement et interprétation des images

- Codage / compression (transmission / archivage)
- Amélioration d'images (subjectif / visuel)
- Vision par ordinateur
 - + Métrologie (segmentation)
 - + Contrôle (binarisation)
 - + Identification (2D / 3D / RdF)
 - + Interprétation (RdF, ...)

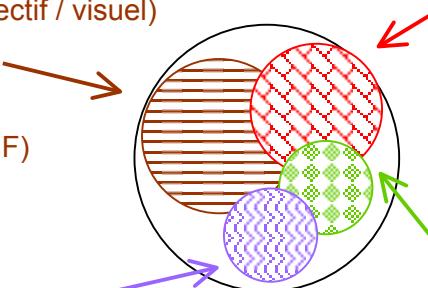
Capteurs d'images

- Vidéo, thermique
- Imagerie X
- Imagerie d'écho (radar / sonar)
- Imagerie de comptage

Image calculée

- Graphisme 2D et 3D
- Infographie
- Réalité virtuelle

Systèmes de traitement



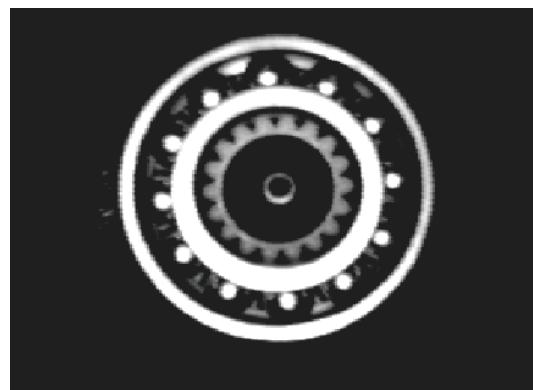
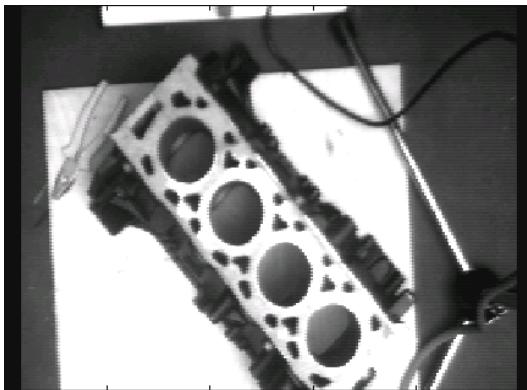
C'est aussi un domaine pluridisciplinaire qui fait appel à :

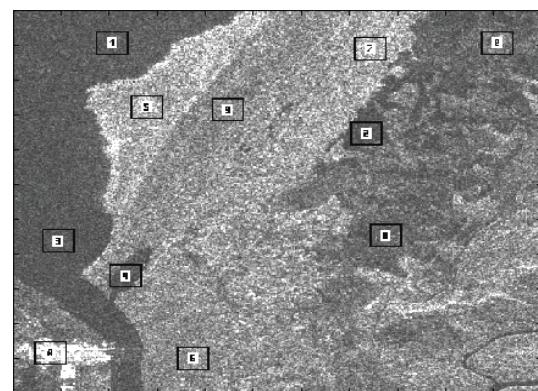
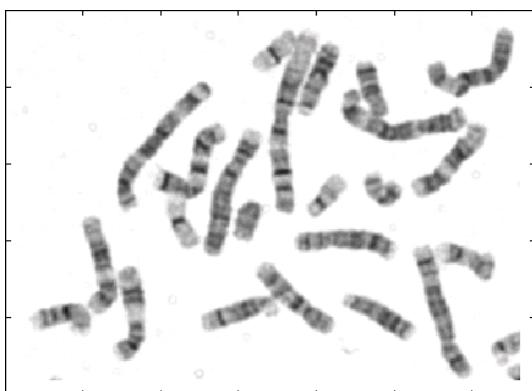
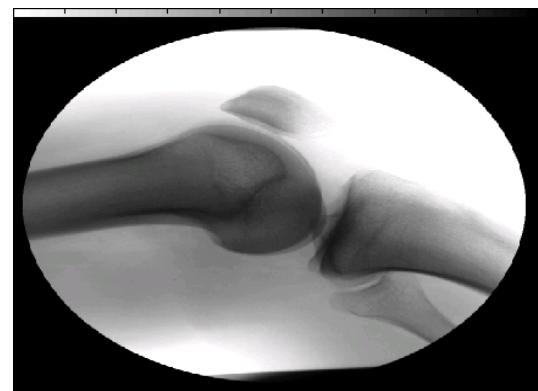
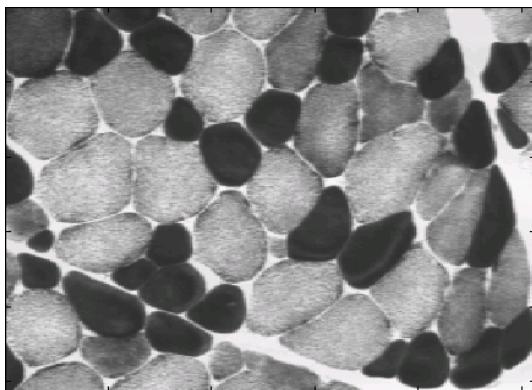
- Traitement du signal
- Analyse numérique
- Informatique
- Théorie de l'information
- Etude statistique
- Optique
- Electronique

Les applications sont aussi très variées et s'étendent de jour en jour :

- Compression d'images
- Amélioration d'images en vu d'un meilleur rendu visuel (télévision)
- Imagerie biomédicale : compter des cellules dans des images microscopiques
- Identification de personnes : reconnaître des empreintes digitales, des visages
- Authentification (de billets de banque par exemple...)
- Aide à la conduite : guidage latéral de véhicule
- Surveillance de la qualité d'une chaîne de production,
- Reconnaissance de l'écriture,
- Imagerie satellitaire
- Indexation d'images : organiser automatiquement des grandes bases d'images en fonction de leur contenu
- ...

Ci dessous sont présentées quelques images correspondant à ces divers domaines d'application :





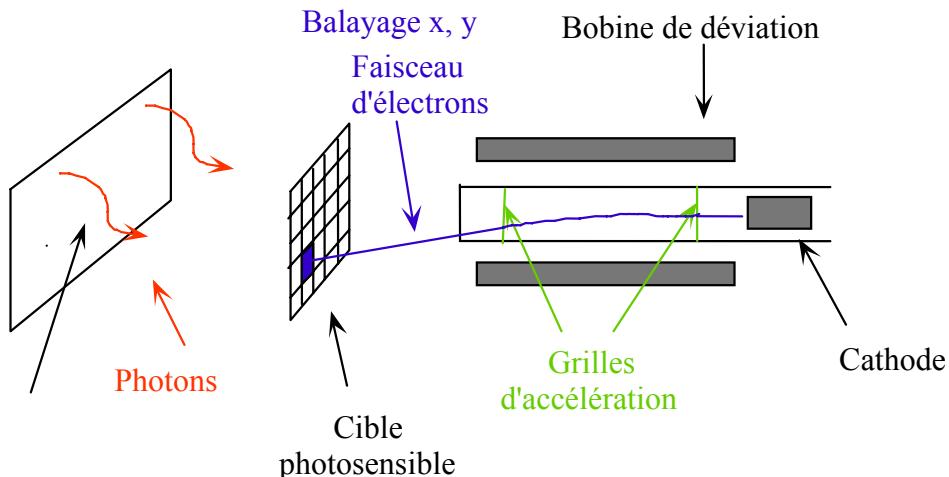
A new image contrast based on experimental in a background of that follows a power between the proposed function and histogram Tests on different images better results. © 199

Chapitre 1 : Acquisition d'images

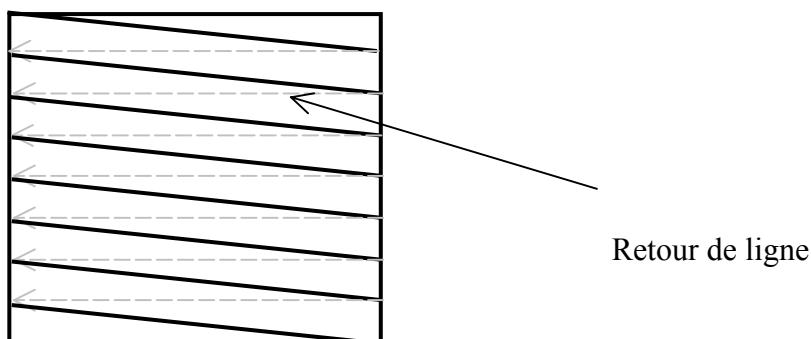
Tout système de traitement d'images peut être vu comme la combinaison de deux étapes : l'acquisition et le traitement proprement dit. La qualité des résultats du système dépend bien sur de l'algorithme mis en place et de son adéquation au problème posé, mais aussi de la qualité initiale des images.

I. Caméra à tubes

Ces capteurs sont relativement anciens et font appel à l'archéologie électronique puisqu'ils ont été inventés en 1931 par Vladimir Zworykin. Leur principe est le suivant : la scène est projetée sur une cible photosensible qui convertit une quantité de lumière en une quantité de charge. Il se forme ainsi un relief de potentiel correspondant à l'image analysée sur la cible. Celle-ci est l'anode du tube dont la source électronique est la cathode. Les électrons issus de la cathode sont attirés par l'anode polarisée positivement. Ce flux d'électrons est dévié par l'ensemble des bobines de déviation afin de scruter l'ensemble de la cible rectangulaire à analyser. Le courant issu de la cible correspond à l'image projetée. L'amplitude du courant est fonction de l'éclairement.

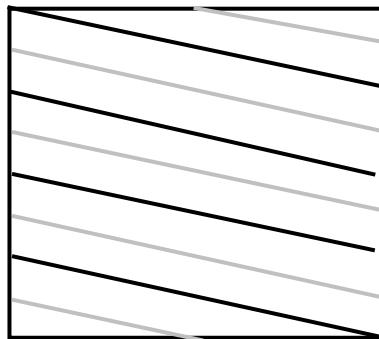


Un balayage dans les deux dimensions de la zone photosensible a pour but de remplir tout l'écran correspondant à l'image. Il est fait de la manière suivante :



La durée d'acquisition d'une ligne ($64 \mu\text{s}$) et le nombre de lignes par image (625) amènent à 25 acquisitions d'image par seconde. Cette fréquence n'est pas suffisante, il apparaît un phénomène de scintillement qui donne une série d'images saccadées et discontinues dans le

temps. Ce problème peut-être résolu sans augmenter le nombre d'images par une technique de balayage entrelacé : chaque image est analysée au moyen de deux trames à raison de 50 trames par seconde :

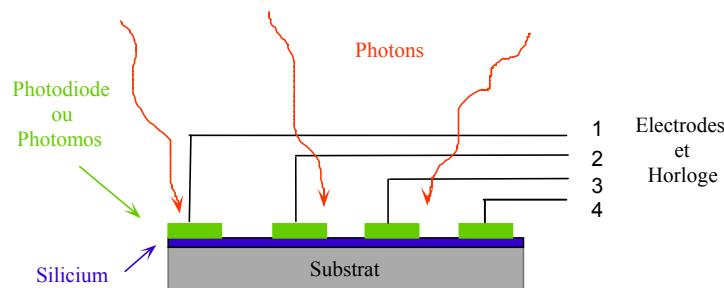


A ce jour, ces tubes ont été remplacés par des capteurs de type CCD (Charge Couple Devices). En effet, même si les tubes vidéo présentent certains avantages (bonne sensibilité spectrale de certains tubes dans l'infrarouge), ils ont aussi une durée de vie plus courte que les caméras CCD, ils produisent des images avec de fortes distorsions géométriques et ont un encombrement beaucoup plus important que celui des capteurs CCD.

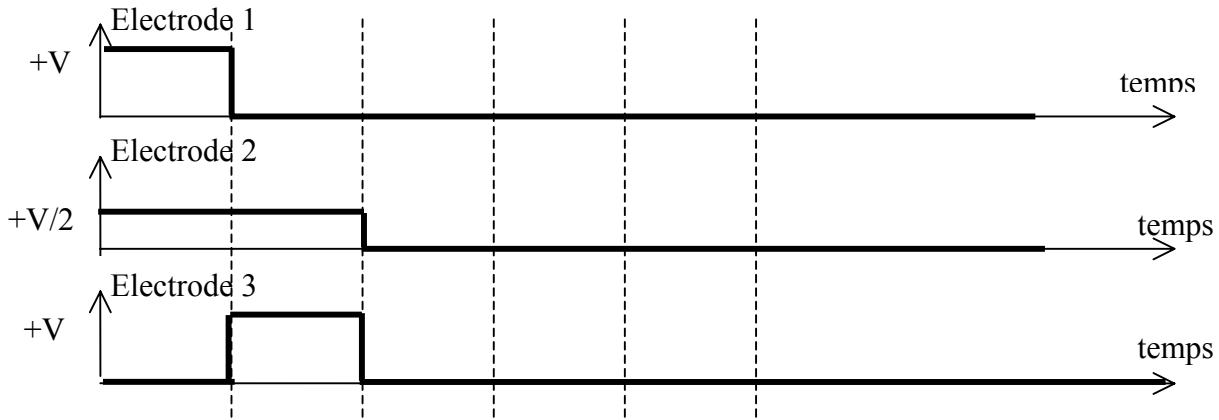
II. Caméras CCD

Ces capteurs sont relativement récents. Leur principe est le suivant : la scène est projetée au moyen d'un objectif sur un réseau de capteurs discret, ce qui réalise un échantillonnage spatial de l'image. Chaque photo-élément convertit l'énergie lumineuse en énergie électrique. Il existe deux types de capteurs : les photodiodes ou les condensateurs MOS de type P. Dans les deux cas, l'arrivée de photons conduit à la formation d'une charge électrique sous la photodiode ou le photomos.

1. Principe du transfert de charge

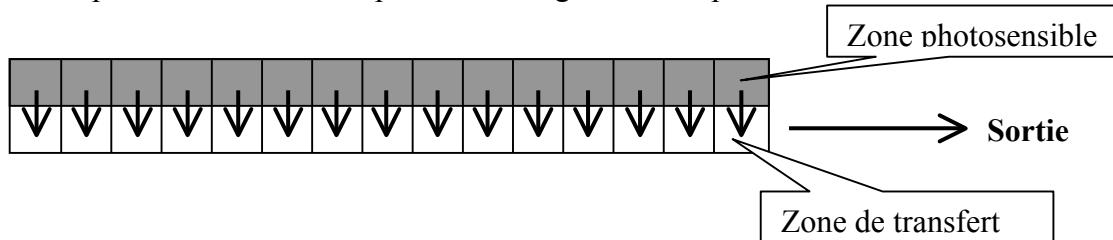


Pour réaliser le transfert de charge, les condensateurs MOS sont associés les uns à la suite des autres. Le substrat des semi-conducteurs ainsi que l'isolant est commun. Les grilles métalliques sont indépendantes pour chaque capteur. Le but consiste à transférer successivement les charges électriques présentes sous les différentes grilles vers la sortie. Plusieurs types de transfert sont possibles, nous allons étudier le plus simple : le transfert biphasé. Supposons que des charges électriques soient présentes sous la grille 1 (électrode 1 polarisée). Pour les déplacer, on va successivement polariser les électrodes 2, 3, 4, ... Pour optimiser le déplacement des charges, une phase intermédiaire où deux électrodes sont simultanément polarisées est mise en place :



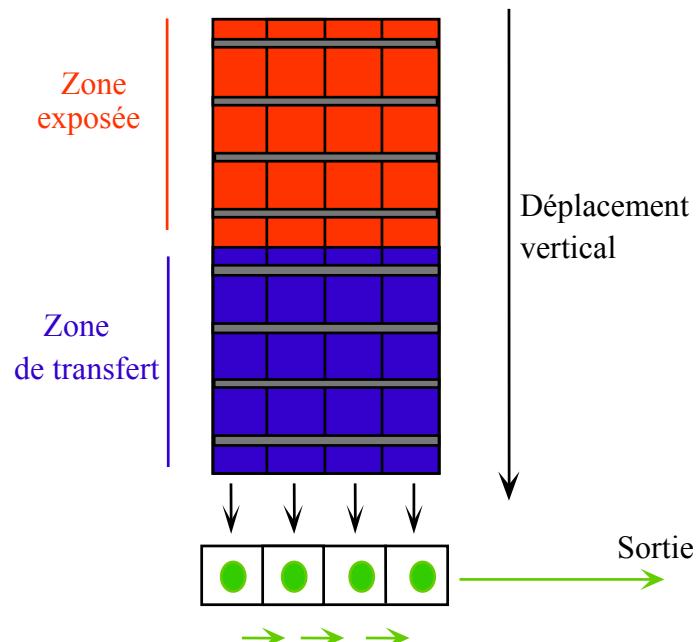
2. Les CCD linéaires

La zone photosensible est composée d'un alignement de photo-éléments :

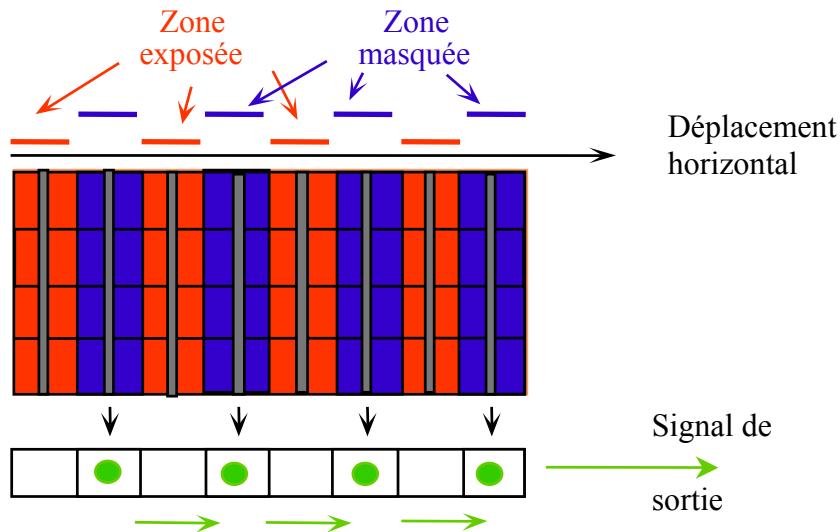


Les charges de la zone photosensible sont transmises toutes en même temps en zone de transfert. Elles sont ensuite évacuées séquentiellement vers la sortie.

3. Les CCD matricielles



Le capteur possède deux zones : une composée des capteurs photosensibles, l'autre est une zone de transfert. Après la saisie de l'image, elle est transférée par un déplacement vertical en zone de transfert puis évacuée ligne par ligne en zone mémoire. Les charges élémentaires sont alors récupérées séquentiellement en sortie. Le problème de cette technologie est que lors du transfert en zone mémoire, les cellules photosensibles restent actives. La première ligne image passe donc devant toutes les autres cellules durant son transfert, ce qui amène des variations.



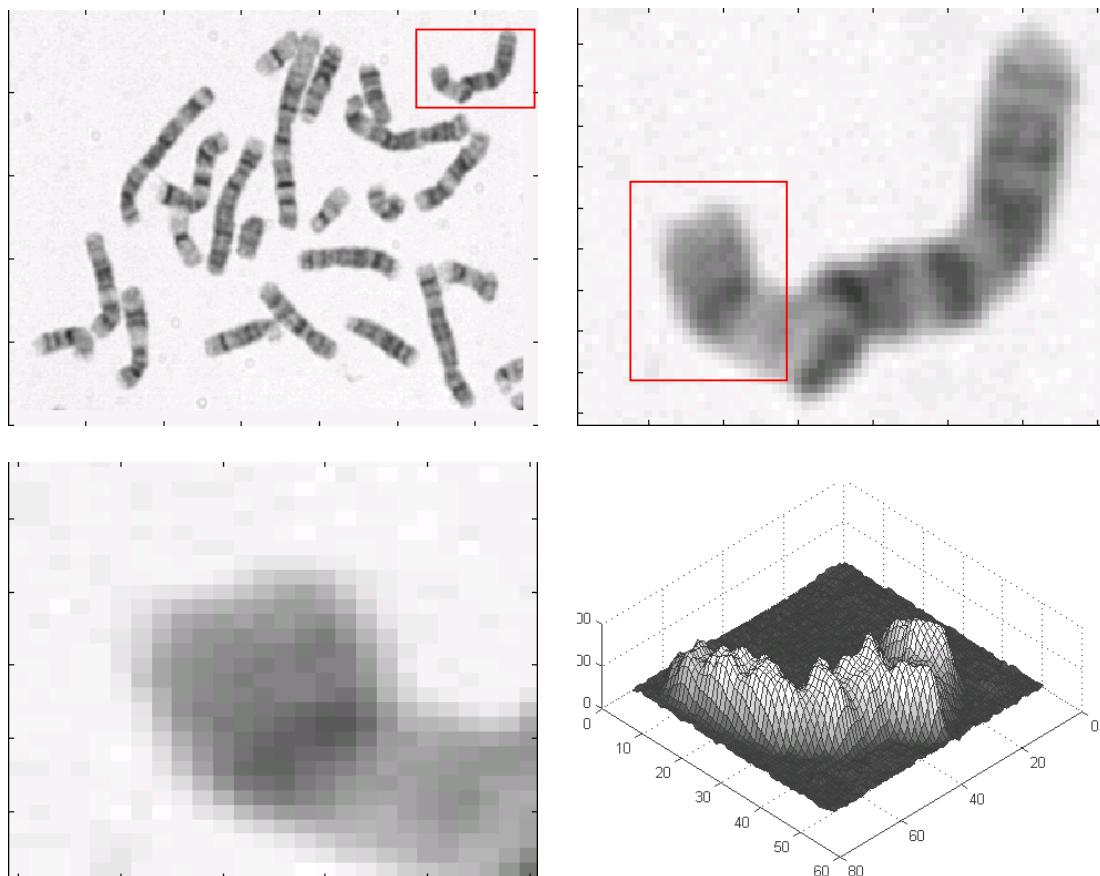
Avec cette nouvelle technologie, la zone photosensible est transférée directement en zone masquée par un déplacement vertical. Ensuite, les charges sont transférées ligne par ligne en zone mémoire avant d'être évacuées séquentiellement. L'inconvénient ici est que l'on aura une mauvaise résolution spatiale dans l'image finale.

Chapitre 2 : Présentation des images

I. Introduction : image niveaux de gris

Une image est représentée par une matrice de dimension « nombre de lignes » x « nombre de colonnes ». Chaque élément de la matrice, nommé pixel, représente l'intensité lumineuse comprise entre 0 et 255, soit 256 niveaux de gris. Le niveau de gris 0 correspond au noir tandis que 255 est représenté en blanc.

Ci dessous sont représentées une image de chromosome et des zooms d'une partie de cette image.



On peut constater, en zoomant sur ces images l'effet de la discréétisation : des pixels carrés apparaissent. On peut également visualiser les images comme une surface 3D, les axes x et y représentant la position spatiale des images tandis que l'axe des z représente la luminance (par soucis de visibilité, on affiche l'inverse vidéo de l'image pour cet affichage).

NB : Pour lire des images sous Matlab, on utilise la commande `imread()` :

```
I=imread('chromosome.tif');
```

Pour afficher une image en niveau de gris :

```
imagesc(I)
```

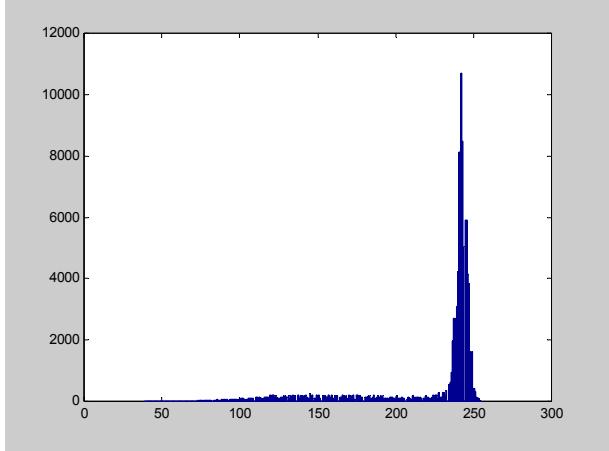
Pour afficher des images comme des surfaces :

```
surf(I)
```

De ces images peuvent être extraits l'histogramme et l'histogramme cumulé.

1. Histogramme d'une image

L'histogramme est un vecteur de dimension 256. Chaque élément du vecteur $h(i)$ représente le nombre de pixels de l'image possédant le niveau de gris i . On peut donc assimiler l'histogramme à la densité de probabilité des intensités lumineuses (à un facteur de normalisation près). Ci-dessous est représenté l'histogramme de l'image de chromosome.



On peut observer que cette image est composée en grande partie de pixel de niveaux de gris entre 230 et 256 représentants le fond. Les pixels représentant l'objet forment un petit amas dans la zone de niveaux de gris autour de 150. Bien sûr, ceux-ci sont bien moins important en nombre que les pixels du fond.

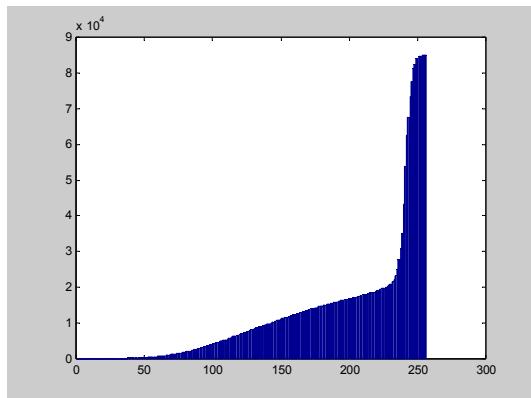
NB : Sous matlab, on calcule l'histogramme avec :
`hist(I(:),[0 :255])`

2. Histogramme cumulé d'une image

C'est également un vecteur de dimension 256. Chaque élément $hc(i)$ représente le nombre de pixels de l'image possédant un niveau de gris inférieur ou égal à i . L'histogramme cumulé peut donc être assimilé, à un facteur de normalisation près, à la fonction de répartition des niveaux de gris. Celui-ci peut-être estimé à partir de l'histogramme en faisant une somme discrète :

$$hc(i) = \sum_{j=0}^i h(j)$$

Ci-dessous est représenté l'histogramme cumulé de la même image.



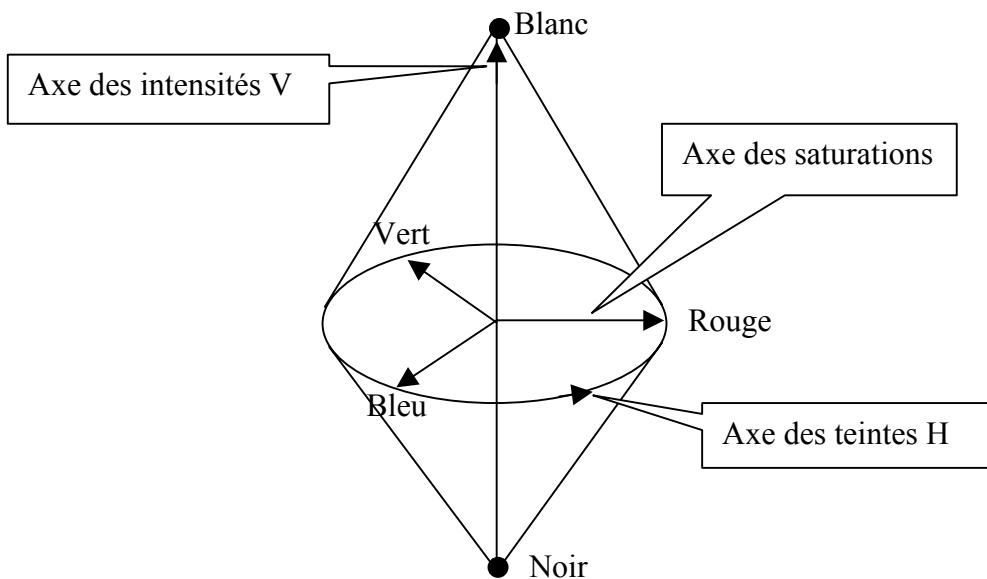
II. Image couleur

Une image couleur contient 3 plans couleur. La plupart des caméras amènent aux trois plans Rouge, Vert, Bleu (R,V,B). Chaque plan est codé comme une image niveaux de gris, avec des valeurs allant de 0 à 255. Lorsque R=V=B, la couleur associée est un niveau de gris. D'autre part, pour passer d'une image couleur à une image niveau de gris, on réalise :

$$I(y, x) = \frac{R(y, x) + V(y, x) + B(y, x)}{3}$$

De la même manière que précédemment, on peut calculer l'histogramme couleur de chaque plan.

De nombreux autres systèmes de représentation des couleurs existent parmi lesquels on peut citer le système HSV (Hue, Saturation, Value) :



L'intérêt de cette représentation est qu'elle permet une interprétation facile des grandeurs :

- H est la teinte qui varie entre 0 et 2π ,
- S, la saturation varie entre 0 et 1 ainsi, une couleur très saturée qui possède une faible proportion de blanc se trouvera loin de l'axe des intensités
- V représente l'intensité lumineuse.

On passe du système RVB au système HSV grâce aux équations non linéaires suivantes :

$$L = \frac{R + V + B}{3}$$

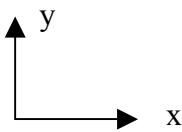
$$H = \begin{cases} a \cos \left(\frac{(R-B)+(R-V)}{2\sqrt{(R-V)^2 + (R-B)(V-B)}} \right) & \text{si } B < V \\ 2\pi - a \cos \left(\frac{(R-B)+(R-V)}{2\sqrt{(R-V)^2 + (R-B)(V-B)}} \right) & \text{si } B \geq V \end{cases}$$

$$S = 1 - \frac{\min(R, V, B)}{L}$$

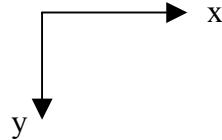
III. Convention

Le système de convention des axes est le suivant :

En analogique :



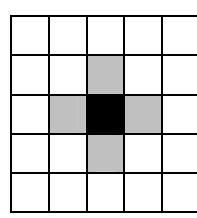
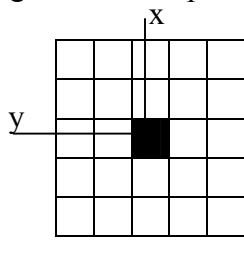
En discret :



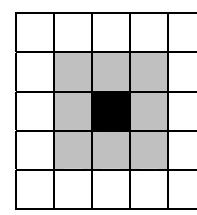
Ainsi, la ligne 0 d'une image est celle située en haut. Ceci implique une réadaptation de toutes les définitions de traitement du signal (exemple : convolution 2D)

IV. Notion de topologie

L'image est donc représentée par un ensemble de pixels disposés sous la forme d'une grille :



4-connectivité



8-connectivité

Quels sont les pixels voisins du pixel (y, x) ?

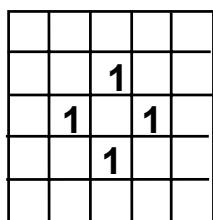
Selon les applications, on pourra se placer en 4 connexités ou en 8 connexités. Par contre, dès que l'on parle de contours et de régions conjointement, le choix s'impose.

Dans \mathbb{R}^2 , le degré de connexité (ou nombre d'Euler) est défini par :

$$C = \text{nombre de composantes connexes} - \text{nombre total de trous}$$

et vérifie : $C(E) + C(\sim E) = 1$ (relation d'Euler)

Dans l'espace image \mathbb{N}^2 , notons E l'ensemble des 1 et $\sim E$ l'ensemble des 0 et considérons l'imagette suivante :



Dans le cas de la 4-connectivité,

$$\begin{cases} C4(E) = 4 - 0 = 4 \\ C4(\sim E) = 2 - 1 = 1 \end{cases} \Rightarrow C4(E) + C4(\sim E) = 5$$

Dans le cas de la 8-connectivité,

$$\begin{cases} C8(E) = 1 - 1 = 0 \\ C8(\sim E) = 1 - 4 = -3 \end{cases} \Rightarrow C8(E) + C8(\sim E) = -3$$

Il faut donc utiliser $C4(E)$ et $C8(\sim E)$ pour conserver une relation topologique ($C4(E) + C8(\sim E) = 1$). On conserve donc la 8-connectivité des contours pour garder une 4-connectivité des régions lors des traitements.

Chapitre 3 : Restauration/Amélioration d'images et diminution du bruit

Dans ce chapitre, nous introduisons successivement l'amélioration d'images, la restauration d'images et la diminution du bruit dans les images. Avant d'aller plus loin, nous définissons ces trois opérations.

Amélioration d'images : le but est de modifier l'apparence d'une image pour qu'un observateur puisse plus facilement extraire des informations. On peut par exemple renforcer le contraste des objets (éventuellement de manière plus prononcée que dans la réalité).

Restauration d'images : elle a pour but de diminuer les dégradations dues

- au milieu de propagation (turbulence,...)
- aux imperfections du capteur (distortion,...)
- à l'échantillonnage (résolution, bruit de quantification,...)

Le but est d'inverser le phénomène dégradant (supposé connu et modélisé) afin de produire une image la plus proche possible de la réalité.

Diminution du bruit dans les images : Le but est de réduire le bruit présent dans les images sans avoir de connaissance a priori sur celui-ci (aucun modèle). Ce pré-traitement a pour but de faciliter les traitements ultérieurs dans la chaîne.

Avant d'aller plus loin, nous allons faire quelques rappels de traitement du signal en introduisant le filtrage deux dimensions.

I. Rappel de traitement du signal

La convolution 2D de l'image I avec un filtre F est donnée par :

$$I(x, y) * F(x, y) = IF(x, y) = \sum_{n=-\infty}^{+\infty} \sum_{m=-\infty}^{+\infty} I(x-n, y-m).F(n, m)$$

où $F(x,y)$ est la réponse impulsionnelle du filtre.

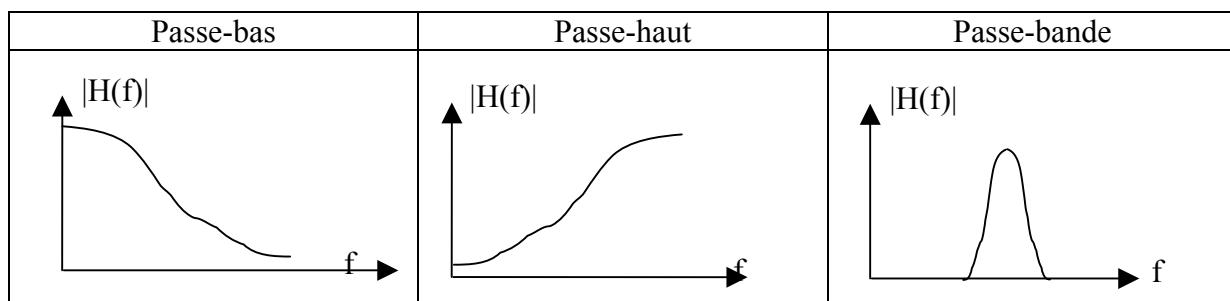
En fréquence, on a :

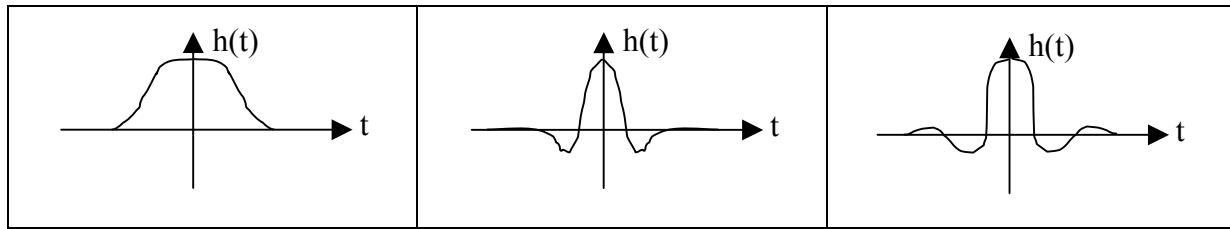
$$TF\{IF(x, y)\} = TF\{I(x, y) * F(x, y)\} = TF\{I(x, y)\}.TF\{F(x, y)\}$$

Il existe trois types de filtres :

- les filtres passe-bas
- les filtres passe-haut
- les filtres passe-bande

Le module de leur fonction de transfert ainsi que leur réponse impulsionnelle sont présentés ci-dessous.





Le filtre passe-bas élimine le bruit mais atténue aussi les détails.

Le filtre passe-haut accentue les détails et le bruit.

Le passe-bande laisse passer certaines fréquences et en supprime d'autres.

La convolution peut être représentée graphiquement par un cumul des produits terme à terme des niveaux de gris de l'image et des coefficients du filtre, à condition toutefois de prendre garde à inverser les coefficients du filtre :

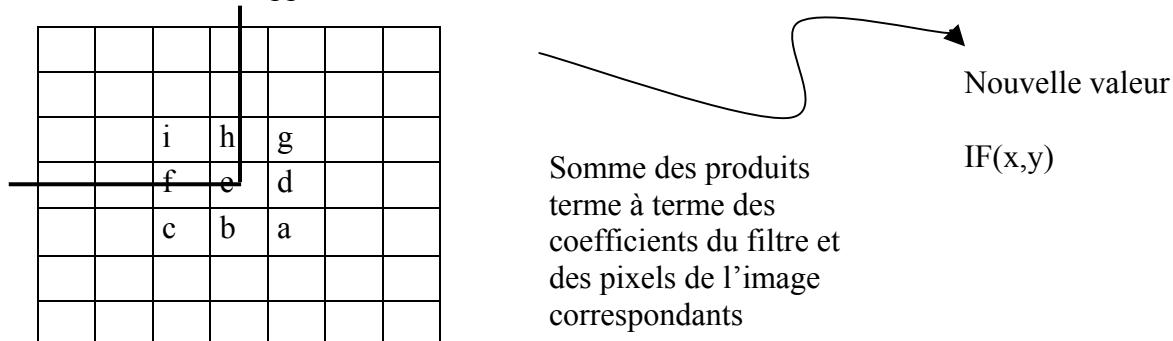
Supposons que l'on veuille filtrer (donc convoluer) une image avec le filtre :

$$F = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

où l'élément central e correspond à la position $(0,0)$ dans la matrice. Avant de réaliser la somme des produits terme à terme, il faut inverser les coefficients du filtre :

$$F = \begin{bmatrix} i & h & g \\ f & e & d \\ c & b & a \end{bmatrix}$$

Pixel d'application du filtre



En traitement d'image, par abus d'écriture, on présente directement le masque sous forme graphique, déjà inversé.

La convolution introduit un effet de bord auquel il faudra prendre garde. D'autre part, par principe, l'image finale a une taille plus grande que l'image de départ. En traitement d'image, on prend uniquement la partie centrale de l'image résultante de manière à conserver la même taille d'image.

En général, on travaille avec des masques de taille impair de manière à assurer la présence d'un pixel central.

NB : sous Matlab, la convolution se fait avec :

`IF=conv2(I,F,'same');`

L'option 'same' permet à l'image IF d'avoir la même dimension que l'image I

II. Amélioration d'images

1. Extension de dynamique

Cette méthode consiste à utiliser au mieux la dynamique de niveaux de gris. Ainsi, si une image possède des niveaux de gris entre G_{\min} et G_{\max} , on va étendre la plage des niveaux de gris pour ramener à une dynamique comprise entre 0 et 255.

Cette étendue réduite de niveaux de gris peut survenir suite à un temps de pose incorrecte ou à un éclairage de la scène trop faible.

La transformation mise en place est :

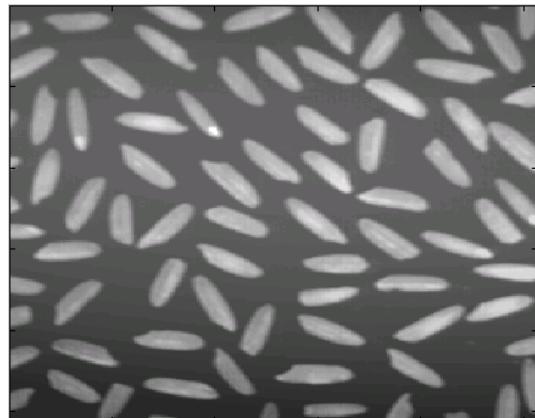
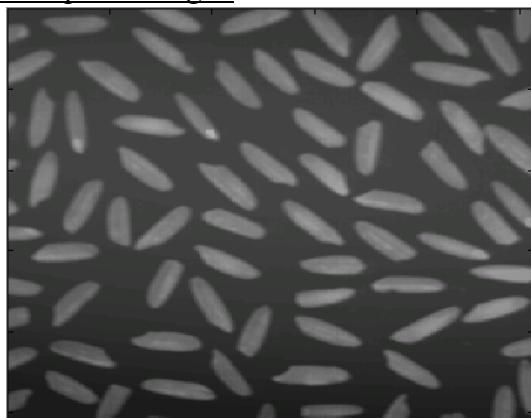
$$I'(x, y) = \frac{I(x, y) - G_{\min}}{G_{\max} - G_{\min}}$$

NB : sous Matlab, si I est l'image de départ, la transformation se fait avec :

```
mini=min(I(:));
maxi=max(I(:));
I1=(I-mini)/(maxi-mini)
```

Cette transformation ne fait qu'améliorer la qualité visuelle de l'image, elle ne change pas l'information présente dans le signal.

Exemple d'image :



2. Correction d'exposition

On peut aussi être amené à vouloir renforcer certaine plage de niveaux de gris, au détriment d'autres plages pour mettre certains objets en valeur. Dans ce cas, la transformation des luminances n'est plus linéaire.

Exemple pour renforcer la présence des éléments d'un circuit électrique, on décide de renforcer la gamme des niveaux de gris foncés :

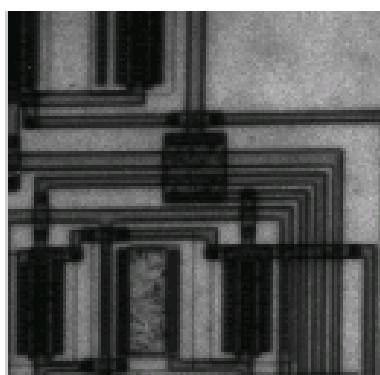
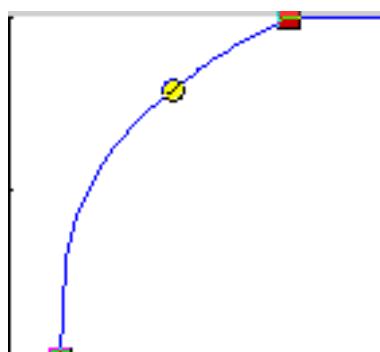


Image de départ



Transformation des luminances

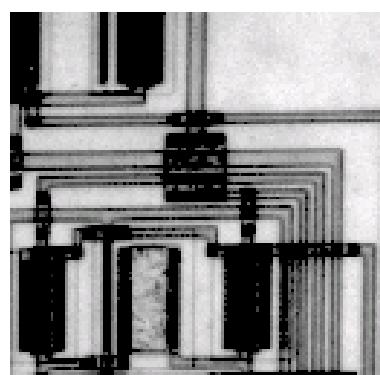


Image d'arrivée

3. Egalisation d'histogramme

Cette opération à pour but de rendre l'histogramme le plus plat possible. On souhaite ainsi que chaque niveau de gris soit également représenté dans l'image.

Soit G , le niveau de gris d'un pixel de départ, le niveau de gris de l'image d'arrivée sera :

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

Exemple sur une image :



Image de départ



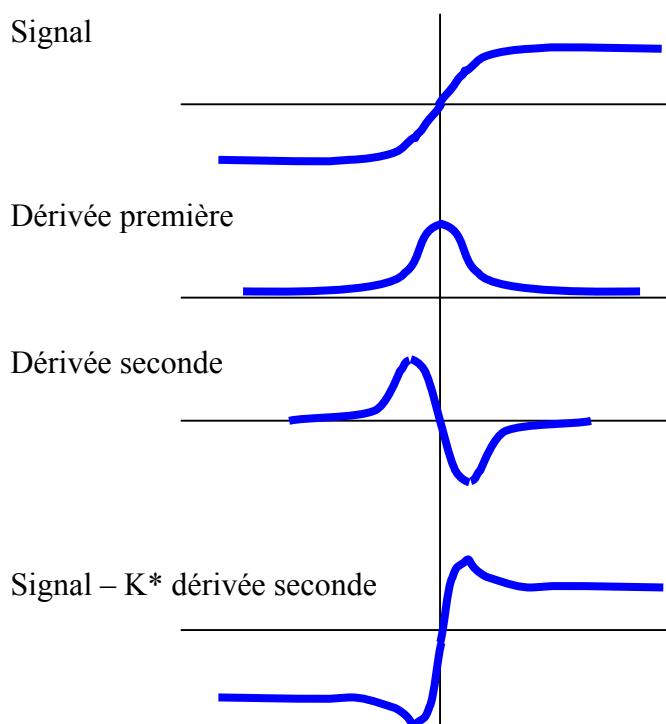
Image égalisée

4. Renforcement de contraste

Contrairement aux autres méthodes, cette opération agit grâce à un traitement local sur les images (traitement prenant en compte les pixels du voisinage).

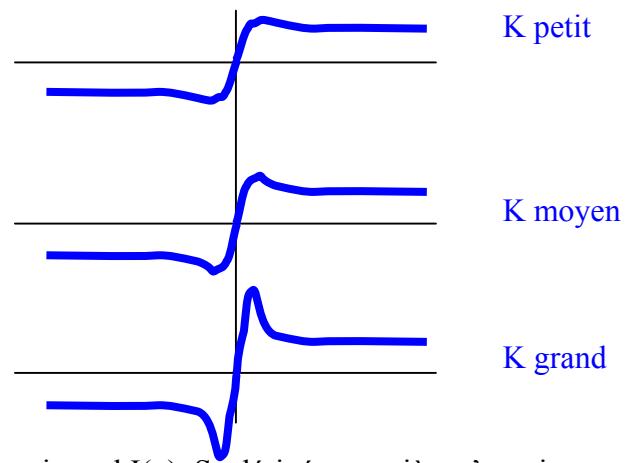
Le problème que nous nous proposons de résoudre est d'essayer de supprimer un effet de flou dû par exemple à un bouger lors de la prise de vue. On ne veut donc pas toucher aux zones homogènes des images mais par contre, essayer de retrouver des contours francs. En effet, le flou est caractérisé par une transition douce entre deux zones de niveaux de gris différents. Il va falloir réaliser un traitement qui amène à une transition la plus raide possible.

Nous allons dans un premier temps expliquer le principe 1D de la méthode.



En réalisant l'opération Signal – K^* dérivée seconde, on ne touche pas au signal dans les zones homogènes (car la dérivée seconde est nulle), par contre, on renforce les zones de contraste. Le signal ainsi obtenu possède une transition plus abrupte.

Selon les valeurs du paramètre K, on peut accentuer plus ou moins le phénomène :



Considérons un signal discret monodimensionnel $I(x)$. Sa dérivée première s'exprime par :

$$I'\left(x - \frac{1}{2}\right) = I(x) - I(x-1)$$

et

$$I'\left(x + \frac{1}{2}\right) = I(x+1) - I(x)$$

Sa dérivée seconde s'exprime par :

$$I''(x) = I'\left(x + \frac{1}{2}\right) - I'\left(x - \frac{1}{2}\right) = I(x+1) - 2I(x) + I(x-1)$$

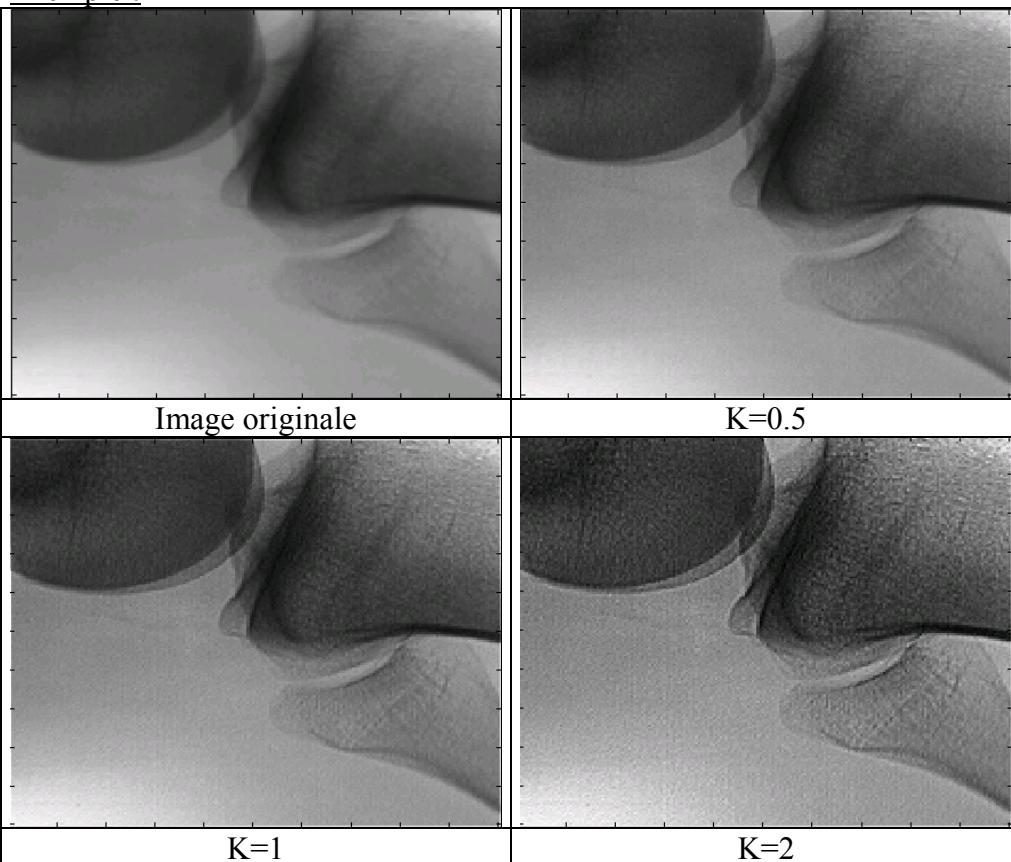
Dans le cas d'un signal à deux dimensions, on utilise le Laplacien défini par :

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

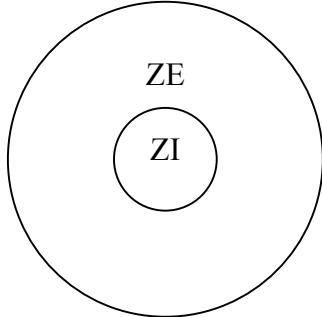
Ce qui correspond à une convolution avec le masque :

0	1	0
1	-4	1
0	1	0

Exemple :



Un des problèmes du laplacien est que celui-ci est très sensible au bruit, ce qui fait apparaître sur les images précédentes un effet de grain. Pour le remédier, on peut utiliser des masques de plus grande dimension pour son calcul. L'idée est d'utiliser des zones circulaires de plus grande dimension :



On aura ainsi une zone externe composée de coefficients positifs et une zone interne composée de coefficients négatifs. La somme des coefficients positifs et négatifs doit être nulle.

Exemple de masque :

1	1	1
1	-8	1
1	1	1

1	1	1	1	1
1	1	-3	1	1
1	-3	-8	-3	1
1	1	-3	1	1
1	1	1	1	1

III. Restauration d'images

1. Les sources de dégradation

Les dégradations peuvent provenir de différents phénomènes qu'il s'agit de bien comprendre puisqu'il va falloir les modéliser afin de pouvoir restaurer l'image.

Une dégradation très fréquente est le bougé. Il ne peut pas être modélisé a priori puisque chaque bougé est différent.

Des dégradations sont également introduites par des capteurs de mauvaise qualité. Citons par exemple les distorsions géométriques en croissant ou bâillet ou les distorsions photométriques. A partir de l'appareil, il est possible de modéliser ces distorsions, ce qui permet de renverser le phénomène et de reconstruire une image correcte.

On peut enfin citer les distorsions dues à l'environnement : présence de turbulence, de poussière ou d'un éclairement non uniforme

2. Modèles de dégradation

Chaque dégradation est un cas particulier et correspond à un modèle différent.

Distorsions géométriques

Pour modéliser les distorsions géométriques, on peut par exemple observer l'image d'une mire à travers la caméra et en déduire la géométrie des distorsions :

Mire

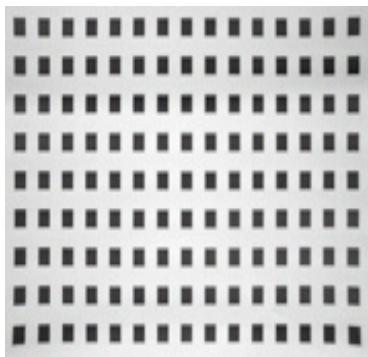
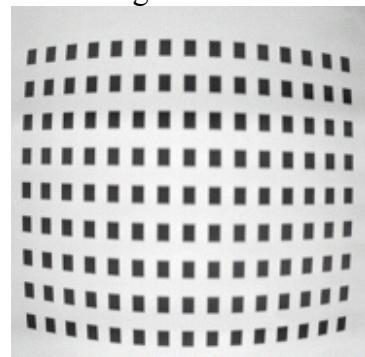


Image de la mire



Modèle de bruit

On peut obtenir un modèle de bruit en étudiant sur l'image les zones homogènes de la scène. Cette opération nécessite d'une part que la scène présente des zones homogènes et d'autre part une intervention humaine pour délimiter ces zones dans l'image.

Changement d'éclairage

Comment faire pour « éclaircir » ou « noircir » une image ? Beaucoup de logiciels standards de traitement d'image utilisent juste l'addition d'une constante (positive ou négative). Evidemment, on obtient l'effet désiré mais celui-ci ne correspond pas à un modèle physique : on n'obtient pas l'image qui aurait été acquise si l'éclairage avait été plus intense par exemple.

En tous points de la scène, la réponse du capteur est donnée par :

$$L(x, y) = \int E(x, y, \lambda) S(x, y, \lambda) R(\lambda) d\lambda$$

où :

- $E(x, y)$ représente l'éclairage.
- $S(x, y, \lambda)$ est la réflectance de la surface, fonction de la longueur d'onde λ
- $R(\lambda)$ est la sensibilité du capteur qui, pour simplifier est supposé répondre à une seule longueur d'onde : $R(\lambda) = \delta(\lambda - \lambda_k)$

On aura alors : $L(x, y) = E(x, y) S(x, y, \lambda_k)$

Supposons que l'on acquiert deux images de la même scène sous deux éclairages différents :

$$L_1(x, y) = E_1(x, y) S(x, y, \lambda_k) \quad L_2(x, y) = E_2(x, y) S(x, y, \lambda_k)$$

Et donc, $L_2(x, y) = L_1(x, y) \frac{E_2(x, y)}{E_1(x, y)} = cte. L_1(x, y)$

Pour éclaircir une image, il ne faut pas lui ajouter une constante mais multiplier ses niveaux de gris par une valeur supérieure à 1.

Problèmes liés aux éclairages non uniformes

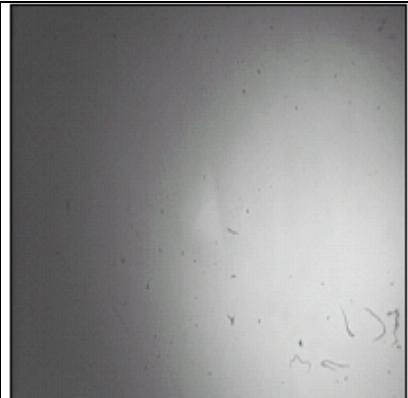
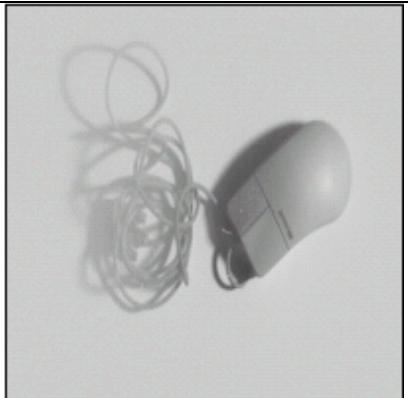
Les distorsions dues aux éclairages non uniformes (sur un banc de mesure par exemple) peuvent être facilement caractérisées en acquérant l'image du fond (qui ne sera pas uniforme si l'éclairage ne l'est pas).

L'image du fond aura pour expression : $I_0(x, y) = F(x, y, \lambda_k) E(x, y)$

L'image acquise sous le même éclairage, composée du fond et de l'objet sera : $I_1(x, y) = S(x, y, \lambda_k) E(x, y)$

En soustrayant systématiquement l'image du fond aux images acquises, on obtient :

$$I_d(x, y) = I_1(x, y) - I_0(x, y) = [S(x, y, \lambda_k) - F(x, y, \lambda_k)] E(x, y)$$

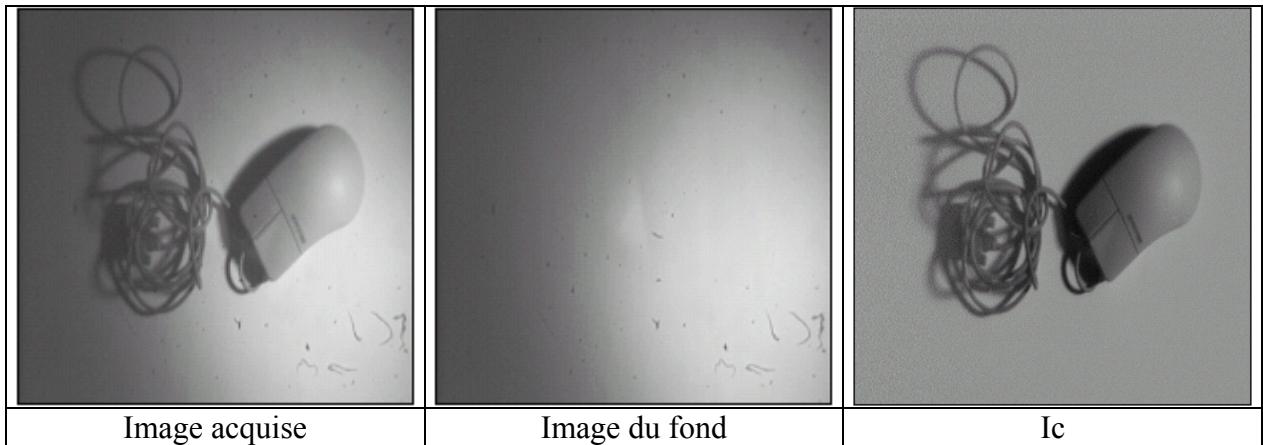
		
Image acquise	Image du fond	Image acquise - Image du fond + cte

En utilisant cette dernière expression, on voit que là où il n'y a pas d'objet, l'image de différence vaut zéro (on a bien reconstruit une image du fond indépendante de l'éclairage). Par contre, là où il y a l'objet, l'image de différence dépend toujours de l'éclairage.

Pour corriger l'image, il vaut mieux faire :

$$I_c(x, y) = \frac{I_1(x, y)}{I_0(x, y)} = \frac{S(x, y, \lambda_k)}{F(x, y, \lambda_k)}$$

On obtient une valeur qui est indépendante de l'éclairage et qui, si le fond est homogène, est directement proportionnelle à la réflectance de l'objet.



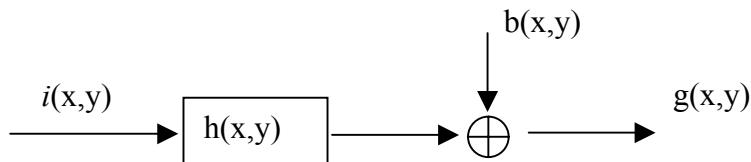
Systèmes linéaires

Beaucoup de dégradations peuvent être modélisées par une transformation linéaire répondant à l'équation :

$$g(x, y) = h(x, y) * i(x, y) + b(x, y)$$

où

- $i(x, y)$ est l'image avant dégradation
- $b(x, y)$ est un bruit supposé additif
- $h(x, y)$ est la réponse impulsionnelle du système modélisant la dégradation
- $g(x, y)$ est l'image dégradée à restaurer



Comment retrouver $h(x, y)$?

- à partir de la réponse du système à une impulsion de Dirac (le bruit peut être supprimer en moyennant plusieurs images par exemple)
 - grâce à un modèle de distorsion .On a vu que les dégradations sont modélisées par :
- $$g(x, y) = h(x, y) * i(x, y) + b(x, y)$$

soit en fréquence,

$$G(u, v) = H(u, v) \cdot I(u, v) + B(u, v)$$

Si on connaît une forme paramétrique de $h(x, y)$ (donc de $H(u, v)$), et que $H(u, v)$ présente des passages par zéro, on peut retrouver les paramètres de $H(u, v)$ grâce à ses passages par zéros qui correspondent aussi à des passages par zéro de $G(u, v)$.

Exemple 1 :

L'erreur de mise au point d'une lentille circulaire est donnée par :

$$h(x, y) = \begin{cases} 0 & \text{si } \sqrt{x^2 + y^2} > r \\ 1/\pi r^2 & \text{sinon} \end{cases}$$

Les zéros de $H(u,v)$ et donc certains zéros de $G(u,v)$ se trouvent sur des cercles concentriques, centrés à l'origine et périodique de période r .

Exemple 2 :

Un flou uniforme 2D peut être modélisé par :

$$h(x, y) = \begin{cases} 1/L^2 & \text{si } -L/2 \leq x, y \leq L/2 \\ 0 & \text{sinon} \end{cases}$$

Les zéros sont espacés de n/L en x et y .

3. Restauration par filtrage inversea.Approche directe

Reprendons les équations modélisant la distorsion :

$$g(x, y) = h(x, y) * i(x, y) + b(x, y) \quad \text{et} \quad G(u, v) = H(u, v).I(u, v) + B(u, v)$$

Si on considère le bruit négligeable, on peut directement estimer $I(u, v)$ par :

$$\hat{I}(u, v) = \frac{G(u, v)}{H(u, v)},$$

Ce qui revient à filtrer l'image dégradée par un filtre de fonction de transfert :

$$F(u, v) = \frac{1}{H(u, v)}$$

Ceci pose néanmoins plusieurs problèmes :

- lorsque $H(u, v)$ est nul ou petit, $\hat{I}(u, v)$ n'est pas défini ou très grand.
- les imprécisions sur l'estimation de $h(x, y)$ et donc, de $H(u, v)$ ont des conséquences très importantes car $H(u, v)$ intervient au niveau d'une division.
- Le bruit présent sur $g(x, y)$, qui n'est jamais rigoureusement nul, produit de grandes variations sur l'estimation de $\hat{I}(x, y)$. En effet, on a :

$$\hat{I}(u, v) = \frac{G(u, v)}{H(u, v)} = \frac{H(u, v).I(u, v) + B(u, v)}{H(u, v)} = I(u, v) + \frac{B(u, v)}{H(u, v)}$$

Ainsi, quand $H(u, v)$ devient très petit (souvent aux hautes fréquences), le bruit devient prépondérant.

Une solution consiste à exploiter les connaissances statistiques et spectrales des images. C'est le filtre de Wiener.

b.Filtre de Wiener

On recherche le filtre $F(u, v)$ qui minimise l'erreur quadratique moyenne :

$$e = E \left\{ |\hat{I}(u, v) - I(u, v)|^2 \right\} = E \left\{ |\hat{I} - I|^2 \right\} \text{ (par simplification d'écriture)}$$

Or on a $\hat{I} = F.G = F(HI + B)$

L'erreur vaut donc :

$$e = E \left\{ |F(HI + B) - I|^2 \right\}$$

$$\begin{aligned}
e &= E \left\{ |I(FH - 1) + FB|^2 \right\} \\
e &= E \left\{ (I(FH - 1) + FB)(I(FH - 1) + FB)^* \right\} \\
e &= E \left\{ (I(FH - 1) + FB)(I^*(F^* H^* - 1) + F^* B^*) \right\} \\
e &= E \left\{ |I|^2 |FH - 1|^2 + |FB|^2 + 2\Re e(I(FH - 1)FB) \right\}
\end{aligned}$$

En supposant que l'image et le bruit sont décorrélés, on a une somme de deux variables aléatoires indépendantes et donc,

$$\begin{aligned}
e &= E \left\{ |I(FH - 1)|^2 + |FB|^2 \right\} \\
e &= |FH - 1|^2 E \left\{ |I|^2 \right\} + |F|^2 E \left\{ |B|^2 \right\}
\end{aligned}$$

Notons $W_I = E \left\{ |I|^2 \right\}$, $W_B = E \left\{ |B|^2 \right\}$, $F = \gamma e^{j\theta}$ et $H = \eta e^{j\Phi}$

$$\begin{aligned}
e &= |\gamma \eta e^{j(\theta+\Phi)} - 1|^2 W_I + \gamma^2 W_B \\
e &= |\gamma \eta [\cos(\theta+\Phi) + j \sin(\theta+\Phi)] - 1|^2 W_I + \gamma^2 W_B \\
e &= [(\gamma \eta \cos(\theta+\Phi) - 1)^2 + \gamma^2 \eta^2 \sin^2(\theta+\Phi)] W_I + \gamma^2 W_B \\
e &= [\gamma^2 \eta^2 + 1 - 2\gamma \eta \cos(\theta+\Phi)] W_I + \gamma^2 W_B \\
e &= (\eta^2 W_I + W_B) \gamma^2 - 2\eta \cos(\theta+\Phi) W_I \gamma + W_I
\end{aligned}$$

Pour minimiser cette somme (qui est positive par définition), il faudra maximiser le terme négatif et donc, faire en sorte que le cosinus vaille 1. On devra donc avoir :

$$\theta = -\Phi$$

L'erreur est alors :

$$e = (\eta^2 W_I + W_B) \gamma^2 - 2\eta W_I \gamma + W_I$$

Dérivons cette erreur par rapport à l'inconnue γ :

$$\frac{\partial e}{\partial \gamma} = 2\gamma (\eta^2 W_I + W_B) - 2\eta W_I = 0$$

$$\text{Soit, } \gamma = \frac{\eta W_I}{\eta^2 W_I + W_B} = \frac{\eta}{\eta^2 + W_B/W_I}$$

D'où l'expression du filtre :

$$F = \gamma e^{j\theta} = \frac{\eta e^{-j\Phi}}{\eta^2 + W_B/W_I} = \frac{H^*}{|H|^2 + W_B/W_I}$$

Très souvent, le rapport W_B/W_I est approximé par une constante fonction de la variance des signaux.

IV. Réduction du bruit dans les images

On peut considérer une image comme étant constituée de plusieurs zones homogènes représentant les objets. Dans la réalité, des fluctuations des niveaux de gris sont présentes à

cause du bruit. On cherchera donc à diminuer l'amplitude de ces perturbations, sans toucher aux zones de transitions.

1. Filtre moyenneur

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

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

La taille du masque est un paramètre variable. Plus le masque sera de grande dimension, plus l'effet du filtrage sera fort. De manière générale, on essayera de réaliser un filtrage isotopique (même effet dans toutes les directions). Pour cela, le voisinage considéré devra avoir une forme circulaire :

$$\frac{1}{29} * \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Afin de conserver la dynamique des niveaux de gris, il faut que la somme des éléments du masque soit égale à 1.

On remarque ici le problème lié à la définition d'un cercle sur un maillage carré.

NB : sous Matlab,
IF=conv2(I,ones(3,3)/9,'same');

Exemple de filtrage :



Image originale



Image filtrée avec
le masque 3x3



Image filtrée avec
le masque 7x7

Bien que cette méthode soit très simple à mettre en œuvre, elle possède un inconvénient majeur : le filtrage introduit un effet de flou, les contours sont dégradés.

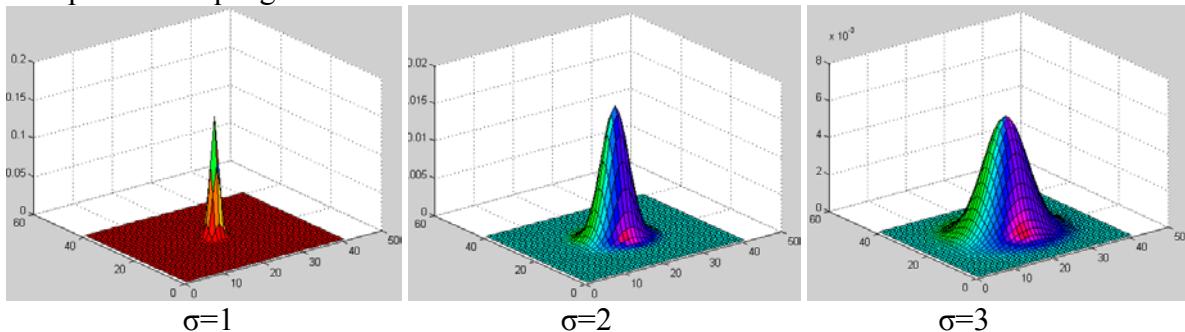
2. Filtrage gaussien

L'expression de la gaussienne en deux dimensions est donnée par :

$$g(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

L'intérêt du filtre gaussien est que l'on règle très facilement le degré de filtrage à travers le paramètre σ .

Exemple de masque gaussien :



Tous ces masques ont été calculés sur un voisinage de taille 41x41 mais on voit sur les figures précédentes que la taille du masque peut-être réduite ou augmentée en fonction de σ . En théorie, la gaussienne a une étendue infinie, mais en pratique, on limite cette étendue $[-3\sigma, 3\sigma]$ car la plupart de la puissance est dans cette zone.

Par rapport au filtre moyenneur, le filtre gaussien accorde une grande importance aux pixels proches du pixel central, et diminue cette importance au fur et à mesure que l'on s'éloigne de celui-ci.

On constate par contre le même défaut que le filtre moyenneur : il dégrade les contours en introduisant un flou.

L'inconvénient des deux filtres linéaires présentés précédemment est que le filtrage s'accompagne d'un étalement des transitions. La détermination des coefficients du filtre résulte ainsi d'un compromis filtrage/dégradation. Ce problème peut être contourné en utilisant des filtres non linéaires

3. Filtre médian

Ce filtre étant non linéaire, il ne peut pas être réalisé avec une convolution 2D de l'image. Considérons un voisinage rectangulaire autour du pixel d'intérêt. Le filtre médian consiste à prendre la valeur de niveaux de gris séparant la population en deux effectifs égaux.

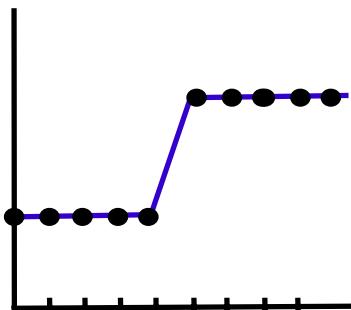
Exemple :

7	10	12	7	9	9	9
12	12	12	7	9	9	10
9	9	10	10	7	10	12
10	12	9	12	10	12	12
12	12	12	10	7	7	10
10	7	10	12	12	10	7
10	7	9	12	10	9	12

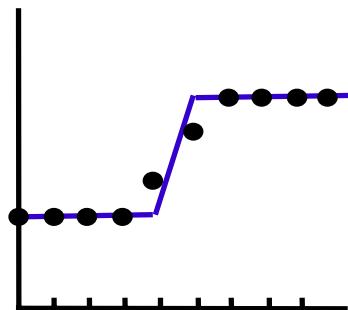
7-7-7-7-7-7-7-7-9-9-9-9-9-9-9-10-10-10-10-10-10-10-10-10-10-10-10-10-10-12-12-
12-12-12-12-12-12-12-12-12-12-12-12-12

→ valeur médiane : 10

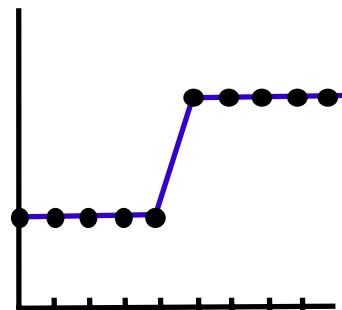
Etude comparative :



Signal initial



Moyenne



Médiane

On voit sur cet exemple l'effet de la moyenne : des niveaux de gris intermédiaires apparaissent au moment de la transition. Ce n'est pas le cas pour le signal filtré grâce à la médiane. Ce filtre donne de très bons résultats de part son principe sur le bruit impulsionnel (type poivre et sel).

Il a par contre l'inconvénient de supprimer les détails fins. Nous allons examiner dans la suite d'autres types de filtres permettant :

- de conserver les détails fins
- de réduire les coûts en temps de calcul imposés par la médiane (tri)
- de conserver une bonne qualité au niveau des transitions

4. Filtre à moyenne seuillée

Le principe consiste à réaliser un filtrage uniquement en dehors des zones de transitions. Celles-ci sont repérées grâce à leur variance élevée.

Principe :

Autour de chaque pixel, calcul de la variance des niveaux de gris.

- si elle est inférieure à un seuil (on est dans une zone homogène), calcul de la moyenne des niveaux de gris
- sinon, on garde le niveau de gris d'origine

5. Filtre à moyenne pondérée

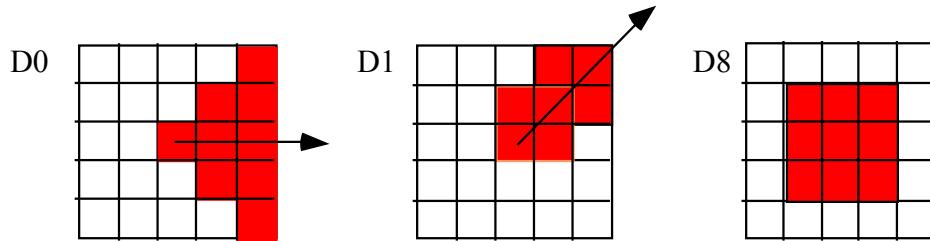
Ce filtre est moins binaire que le précédent (filtrage/non filtrage) mais nécessite une estimation de la variance du bruit. Cette estimation peut être réalisée en prélevant une portion de l'image correspondant à une zone homogène et en calculant la variance des niveaux de gris de cette fenêtre. Une fois la variance du bruit VB estimée, les pixels sont modifiés à l'aide de la relation :

$$G = (1-K) \cdot \text{Moyenne} + K \cdot G_{\text{origine}}$$

avec $K = (\text{Var}-\text{VB})/\text{Var}$

6. Filtre de Nagao

Ce filtre est également appelé filtre à sélection de voisinage. Chaque fenêtre 5x5 est divisée en 9 domaines :



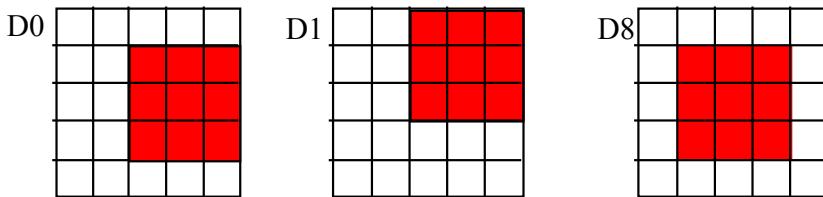
D2, D4 ,D6 sont déduits de D0 par rotations de 90 degrés. D3, D5, D7 sont déduits de D1 de la même façon..

Pour chaque domaine Di, on calcule la moyenne et la variance : $moy(i)$ et $var(i)$. On va ensuite rechercher le domaine où la variance est la plus faible (aucun contour ne passe par ce domaine) et on affecte la moyenne des niveaux de gris de ce domaine au pixel central :

$$k / Var(k) = \text{Min Var}(i) \rightarrow G = Moy(k)$$

Ce filtre permet de limiter les pixels hors norme (bruit impulsionnel). D'autre part, il préserve et même améliore le contraste. Les zones avant et après la transition sont plus homogènes.

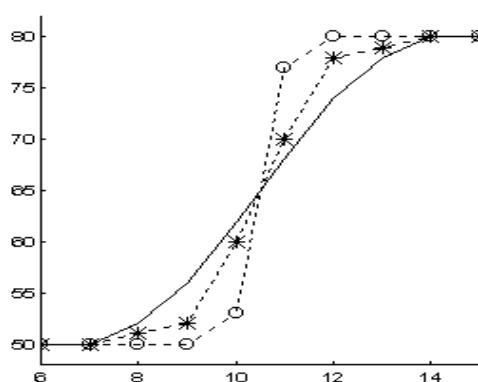
Une première amélioration de ce filtre consiste à régulariser la forme géométrique des domaines :



Deux autres améliorations sont envisageables :

- Utiliser l'étendue $\text{max}(\text{Gris})-\text{Min}(\text{Gris})$ comme paramètre de dispersion plutôt que la variance (gain en temps de calcul)
- Utiliser la médiane comme paramètre de centrage pour une meilleure robustesse vis à vis du bruit

7. Etude comparative



Sur une zone de transition, nous affichons ci-dessous :

- la moyenne 5x5
- la médiane 5x5
- ° Nagao

On peut constater que du point de vue raideur de la transition, c'est l'opérateur de Nagao qui donne les meilleurs résultats

Ces résultats sont encore plus visibles en affichant les images sous forme surfacique :

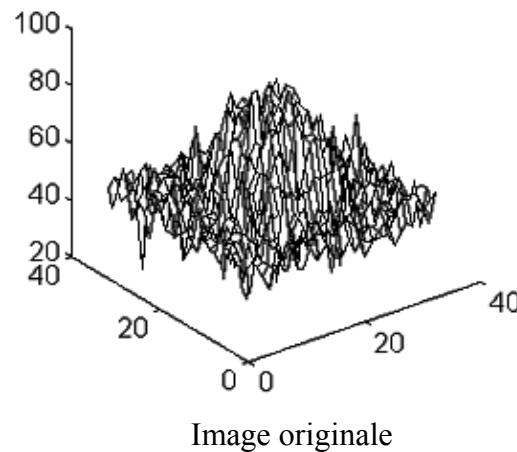
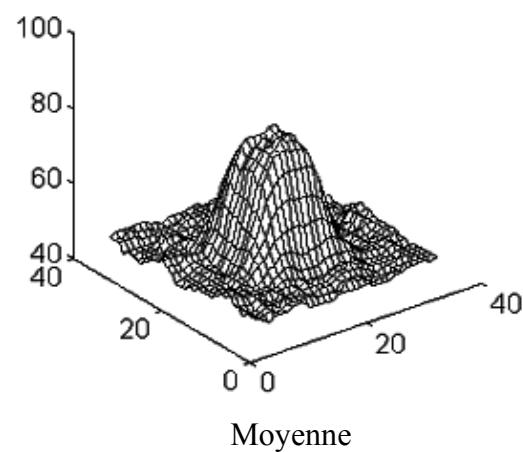
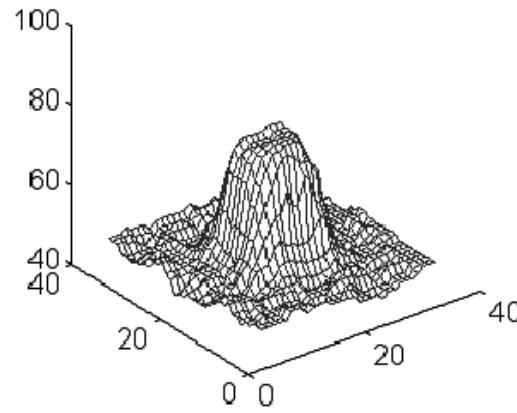


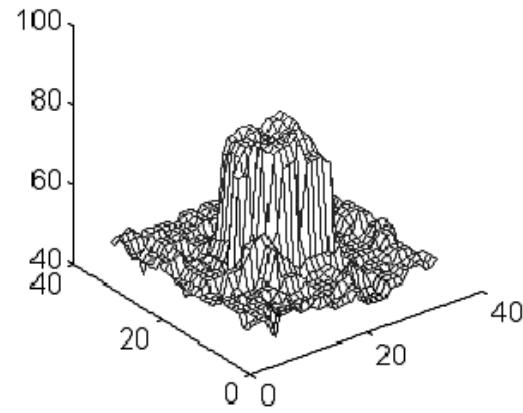
Image originale



Moyenne



Médiane



Nagao

Chapitre 4 : Détection de contour

Une image est généralement composée de plusieurs zones de niveaux de gris différents correspondants aux différents objets de la scène. Il nous faut maintenant trouver une manière de repérer ces objets dans l'image afin de pouvoir passer leurs caractéristiques à un éventuel processus de reconnaissance des formes.

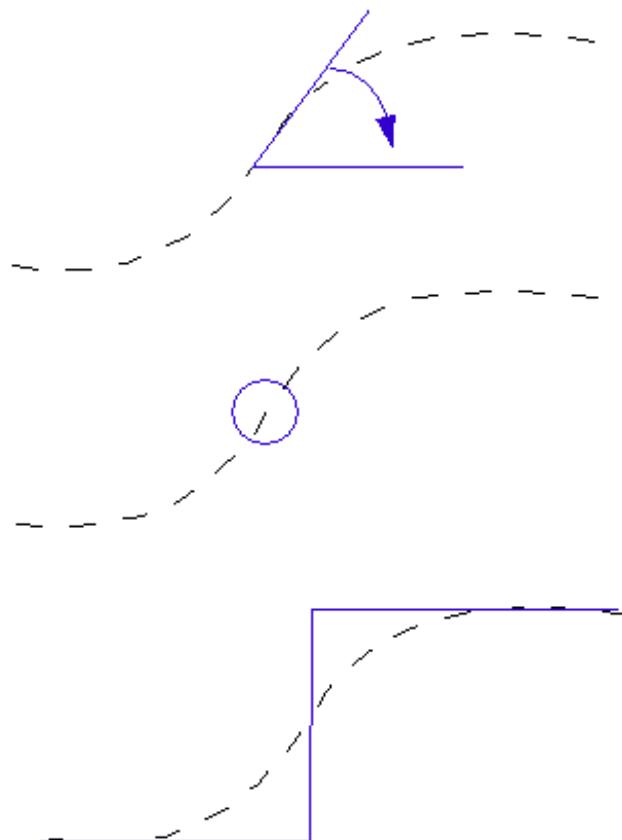
Deux grandes approches peuvent être envisagées pour extraire les zones pertinentes des images :

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

Dans ce chapitre, nous allons nous intéresser à cette dernière méthode, la segmentation en région sera présentée dans le chapitre suivant.

I. Introduction

Les contours correspondent aux lieux géométriques où le signal présente une forte discontinuité. Regardons sur un signal à une dimension comment retrouver ces discontinuités.



Elles correspondent à une pente élevée de la transition
 → recherche des maxima locaux du gradient

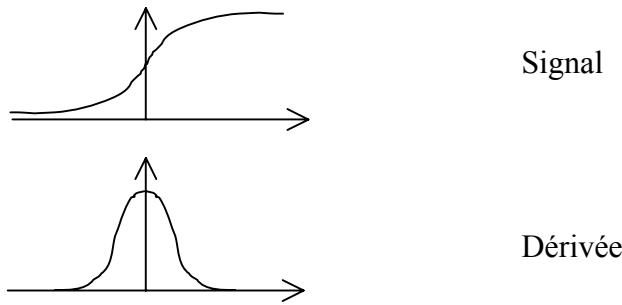
Elles correspondent à un changement de concavité
 → recherche des passages par zéros de la dérivée seconde (du laplacien pour les images)

Elles correspondent à une forme particulière du signal
 → adaptation à un gabarit

Les deux premières approches sont des méthodes dérivatives tandis que la troisième est une méthode d'adaptation à un gabarit

II. Les approches dérivatives

1. L'approche gradient



La recherche de la transition peut être réalisée en recherchant les maxima locaux du gradient. Dans le cas 2D des images, le vecteur gradient est défini au point de coordonnées (x,y) par :

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

Le module du gradient est défini par :

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

tandis que son orientation est définie par :

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

L'orientation du gradient est perpendiculaire au contour et va de la partie foncée à la partie claire.



Image originale

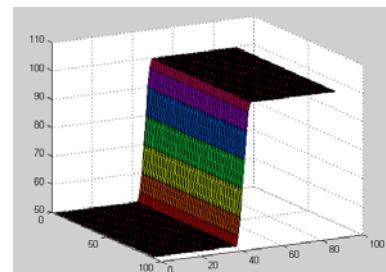


Image originale visualisée sous forme surfacique

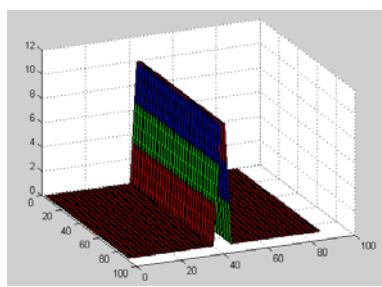


Image du module du gradient

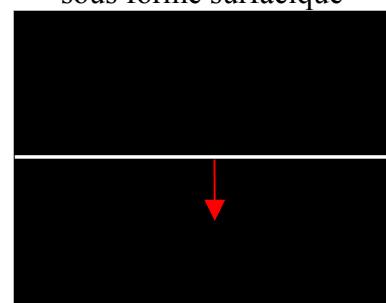


Image du gradient seuillé et orientation

Le calcul des dérivées peut être réalisé de plusieurs façons.

a) calcul direct

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

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

$$Iy(y,x) = I(y+1,x) - I(y,x) = (I^*Gy)(y,x)$$

Ce qui correspond à une convolution avec les masques :

$$Gx = \begin{bmatrix} -1 & 1 \end{bmatrix} \text{ et } Gy = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Malheureusement, ces dérivées sont très sensibles au bruit, on aura donc à faire un filtrage préalable.

b) Masque de Prewitt

Les masques dérivateur sont maintenant :

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad \text{et} \quad Gy = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \text{ ou } \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} : \text{filtrage des lignes ou des colonnes}$$

$$\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \text{ ou } \begin{bmatrix} 0 \\ 1 \end{bmatrix} : \text{dérivée sur les lignes ou les colonnes}$$

Ce type de masque combine donc à la fois un filtrage et une dérivée, il est donc moins sensible au bruit que le calcul direct des dérivées.

c) Masque de Sobel

C'est le même principe que le masque précédent sauf que le filtrage préalable n'est pas réalisé de la même manière :

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad \text{et} \quad Gy = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

d) Masque dérivée de gaussienne

Le filtrage précédent la dérivation peut être réalisé grâce à un masque gaussien G. Dans ce cas, le calcul de la dérivée en x se fait par :

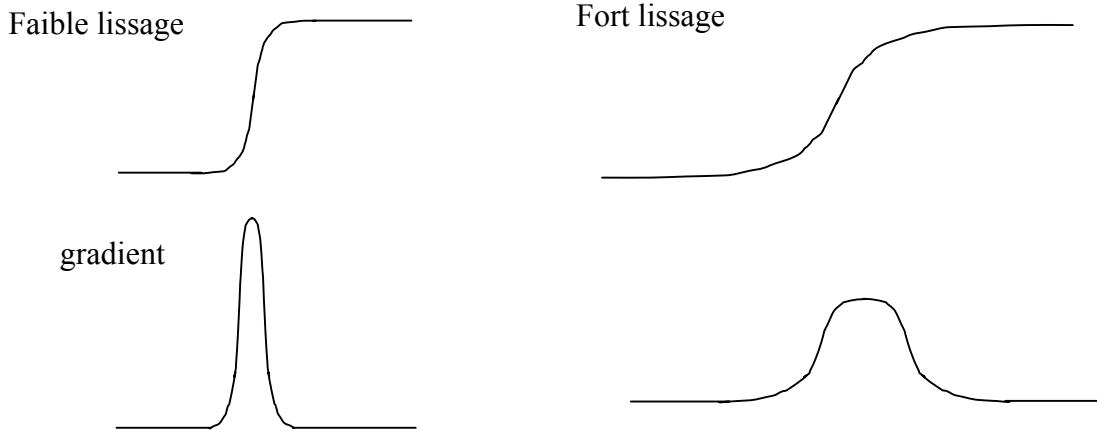
$$Ix = \frac{\partial(I^*G)}{\partial x} = I * \frac{\partial G}{\partial x}$$

Plutôt que de calculer deux produits de convolution sur toute l'image (coûteux en temps de calcul), on calcul le produit de convolution de l'image et de la dérivée en x du filtre. Le même raisonnement peut bien sûr être tenu pour la dérivée en y. On :

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \text{ soit } \frac{\partial G(x,y)}{\partial x} = -\frac{x}{2\pi\sigma^4} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

Le paramètre σ règle le degré de lissage.

La dualité que l'on a vue dans le chapitre précédent (filtrage/raideur de la transition) se retrouve ici :



Un faible lissage amène à un pic de haute amplitude et étroit, donc à des contours fins. Par contre, si le signal est bruité, il restera beaucoup de bruit dans la dérivée. Un fort lissage amène lui à un pic de plus faible amplitude et beaucoup plus large, ce qui conduira à un contour épais. Par contre, le gradient d'une image bruité sera beaucoup moins perturbé par le bruit.

e) Calcul de la norme du gradient

Quel que soit le calcul de I_x et I_y , il faut déterminer la norme du gradient pour pouvoir la comparer à un seuil et déterminer où sont les contours. Une simplification est exigée pour ne pas obtenir des temps de calcul trop longs (contrainte temps réel par exemple).

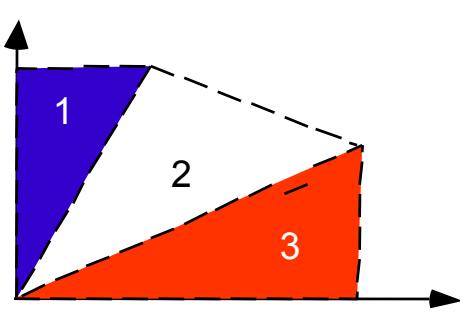
De manière générale, la norme s'exprime par :

$$\|\overrightarrow{\text{grad}}\| = \left(|I_x|^N + |I_y|^N \right)^{1/N}$$

- $N=2$: distance euclidienne, trop long en temps de calcul car calcul d'une racine carrée
- $N=1$: somme des valeurs absolues
- $N=\infty$: maximum des valeurs absolues

Ces deux dernières distances sont utilisables du point de vue temps de calcul mais amènent à des approximations importantes.

Une autre idée consiste à changer de norme selon la répartition angulaire dans laquelle on se trouve :



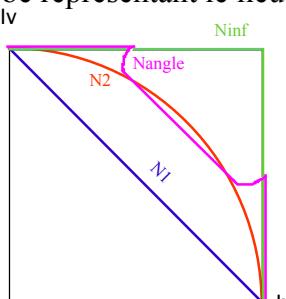
Zone 1 : $|I_y| > 2|I_x|$
 \rightarrow norme = I_y

Zone 3 : $|I_x| > 2|I_y|$
 \rightarrow norme = I_x

Zone 2 : le reste
 \rightarrow norme = $\sqrt{2} \cdot (\sqrt{|I_x|^2 + |I_y|^2}) / 2$
ou norme = $\sqrt[4]{|I_x|^2 + |I_y|^2}$

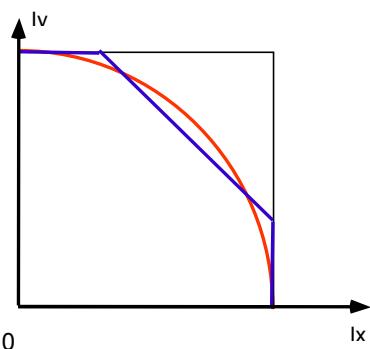
Nous appellerons cette distance Nangle

La dernière distance présentée, qui est aussi la moins approximative résulte d'une étude de la courbe représentant le lieu de toutes les distances depuis l'origine égales à 1 :



On constate que les normes N1 et Ninf sont très approximatives tandis que la norme Nangle est plus proche de la vérité.

De cette courbe, on peut déduire une nouvelle distance résultant de l'approximation du cercle par trois droites



$$\text{Norme} = \max(G_x, G_y, 3/4(G_x+G_y)).$$

On obtient ainsi une meilleure approximation du cercle

f) Seuillage des distances

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

La méthode la plus simple consiste à considérer un seuil fixe S. Tous les pixels possédant une norme supérieure à S sont déclarés appartenir à un contour. Tout le problème réside alors dans le choix du seuil : un seuil trop bas amène à des sur-détections : on détecte beaucoup de bruit, tandis qu'un seuil trop élevé amène à des contours non fermés.

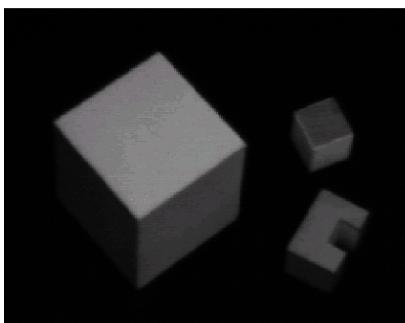
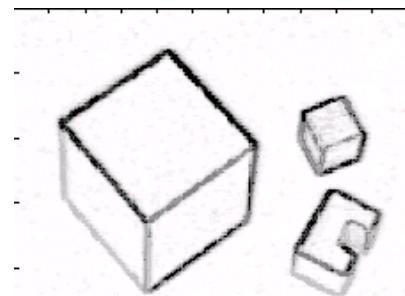
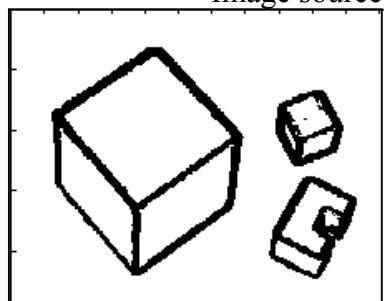


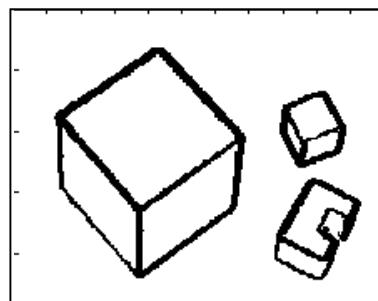
Image source



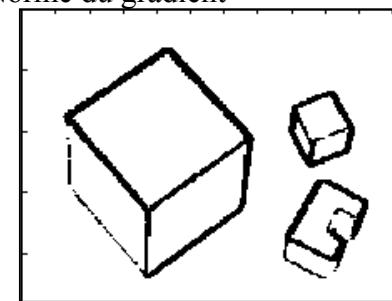
Norme du gradient



Seuil=10
Contour très épais



Seuil =15
Encore des contours épais



Seuil = 20
Contours épais par endroits et
il manque des contours à
d'autres endroits

Seuillage par hystérésis

Une façon de diminuer l'importance du seuil est d'utiliser un seuillage par hystérésis. Avant, il y avait un seuil unique qui amenait soit à des fausses alarmes, soit à des lacunes. Il était très difficile, voire impossible, de trouver un seuil adapté à toute l'image.

On introduit maintenant 2 seuils : un seuil haut et un seuil bas.

- Si norme > seuil haut \rightarrow contour sur 1
- Si norme < seuil bas \rightarrow pas de contour 0
- Si seuil bas < norme < seuil haut \rightarrow contour de fermeture 2

Les contours hypothétiques de fermeture sont transformés en contours sûrs s'ils sont adjacents à un contour déjà codé à 1. Cette recherche d'adjacence s'effectue à partir des points d'extrémités des contours sûrs (soit par suivi, soit par étiquetage des voisins connexes).

Exemple :

1	0	1	0	2	0	1	0	0	2
2	3	3	4	0	2	3	1	0	0
0	2	2	1	1	2	3	3	1	2
0	0	3	2	3	2	2	0	2	0
9	10	11	11	7	9	10	9	9	10
8	11	9	11	9	10	9	11	11	10
10	7	8	10	9	7	11	11	7	8
8	8	10	10	8	10	11	8	8	10

Image de départ

Calcul de la dérivée :

$$\begin{aligned} & |I(y+1,x) - I(y,x)| \\ & + |I(y,x+1) - I(y,x)| \end{aligned}$$

4	2	5	6	4	3	3	1
3	1	4	1	1	1	2	3
2	4	2	3	1	1	5	3
11	9	9	8	9	9	10	7
4	4	2	4	2	2	4	2
7	2	3	1	5	6	0	8
1	4	0	3	5	1	6	1

Norme du gradient

0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
1	0	0	0	1	1	0	1
0	0	0	0	1	0	1	0

Seuillage simple S=5

0	0	2	2	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	2	0
1	1	1	1	1	1	1	2
0	0	0	0	0	0	0	0
2	0	0	0	2	2	0	1
0	0	0	0	2	0	2	0

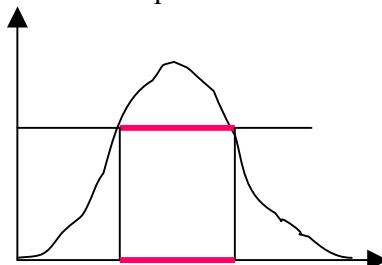
Seuillage par hystérésis, Seuil bas=5, Seuil haut=8

Le seuillage par hystérésis permet de limiter le nombre de fausses alarmes et de lacunes mais il amène aussi à des contours épais lorsqu'un fort filtrage a été réalisé.

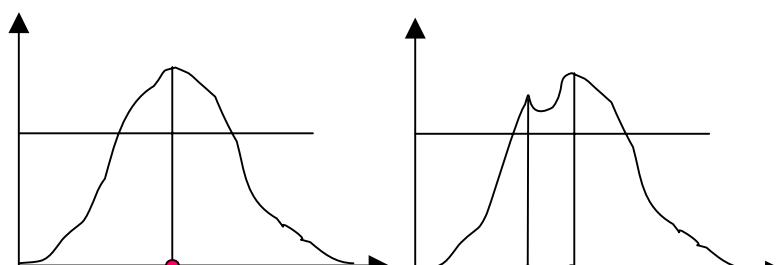
Lignes de crête

Pour revenir à des contours plus fins, on procède à une recherche des lignes de crête. Un point est gardé comme contour s'il est supérieur au seuil et s'il est maximum local dans une direction.

Seuillage classique

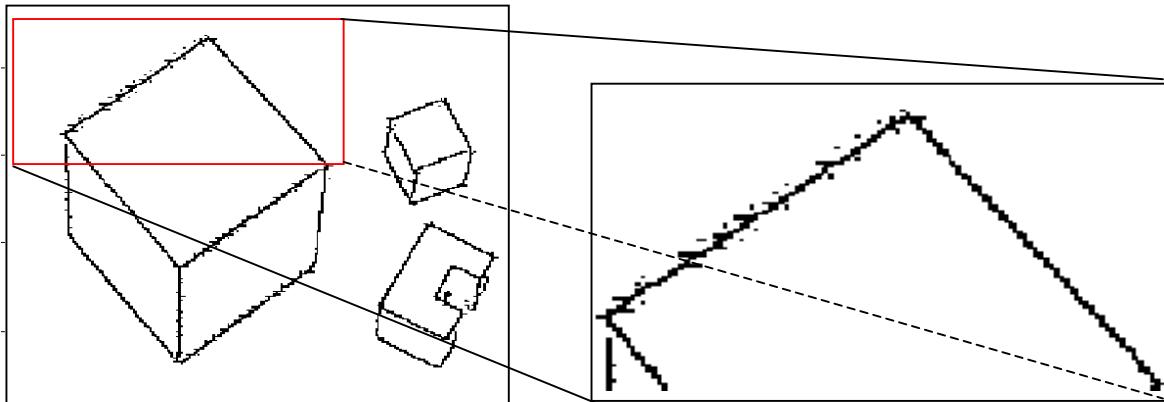


Problème des lignes de crête

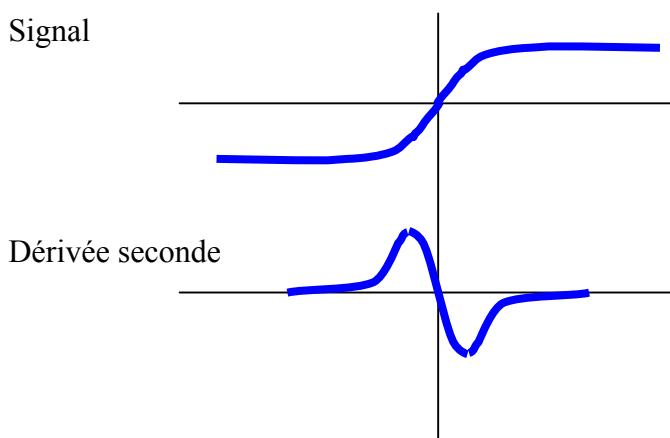


Les lignes de crête ne résolvent pas tous les problèmes. D'une part, elles peuvent amener à des contours parasites. D'autre part, elles ne permettent pas de garder des contours fermés, même si le seuil choisi est tel que les contours l'étaient.

Exemple de lignes de crête :



2. L'approche laplacien



La recherche des contours dans l'image peut aussi être réalisée en recherchant les passages par zéro du laplacien. Comme nous l'avons vu dans le chapitre précédent, celui-ci peut-être estimé grâce à une convolution avec le masque :

$$\begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array}$$

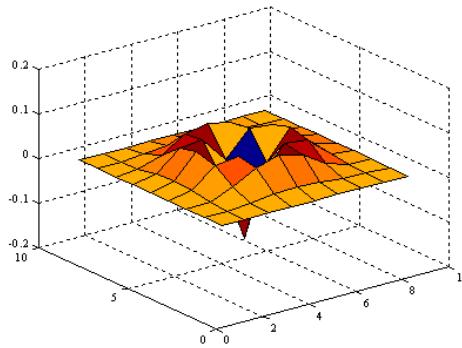
Comme la dérivée seconde est très sensible au bruit, on préfère utiliser le laplacien à partir de dérivées de gaussiennes :

$$\nabla^2(IM \otimes G) = IM \otimes \nabla^2 G \quad \text{avec} \quad G(x, y) = \frac{1}{2\pi\sigma^2} * \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

$$\nabla^2 G(x, y) = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2} = -\frac{1}{\pi\sigma^4} * \frac{1-x^2-y^2}{2\sigma^2} * \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

Comme pour le filtrage, la variance σ^2 règle le degré de lissage et fixe la dimension du masque.

Exemple : $\sigma = 1$, taille du masque : -4σ à 4σ



Les points de contours sont ensuite extrait comme étant les passages par zéro significatifs du laplacien.

III. Approche dérivative précédée d'un filtrage non linéaire

Le filtre de Nagao vu dans le chapitre précédent se prête très bien à une détection de contour : il supprime le bruit sans toucher à la raideur de la transition. Une façon de détecter les contours est donc de réaliser un filtrage de Nagao, suivi d'un calcul de dérivée simple :

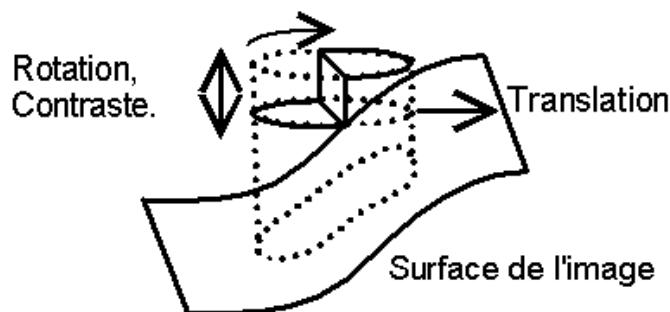
$$Gx = \begin{bmatrix} -1 & 1 \end{bmatrix} \text{ et } Gy = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Cette méthode amène à de très bons résultats de détection.

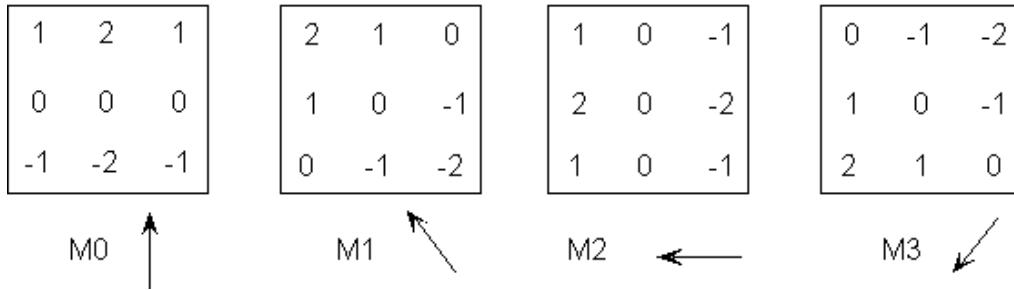
IV. Adaptation à un gabarit

1. Forme élémentaire

On définit un gabarit de contour dans une fenêtre d'analyse circulaire par pondération des pixels (isotropie) et on recherche, pour chaque pixel de l'image, si son voisinage a une forme qui correspond à la forme du gabarit. La décision d'acceptation est réalisée sur critère(s) de contraste suffisant et/ou de rapport contraste/bruit suffisant. L'information d'orientation est implicite dès que l'on a trouvé la position du gabarit qui amène à l'erreur quadratique minimale.



On définit le gabarit du contour M0 et trois rotations de celui-ci par pas de 45° . L'image est alors convoluée avec ces quatre masques et en chaque pixel, on regarde quel est le masque qui a amené à la plus grande réponse en valeur absolue. Si cette réponse est supérieure à un seuil (seuil de contraste), le pixel est déclaré comme pixel contour. Son orientation est donnée par celle du masque qui a le plus répondu si cette réponse est positive et par l'orientation du masque + 180 si la réponse était négative.



2. Généralisation

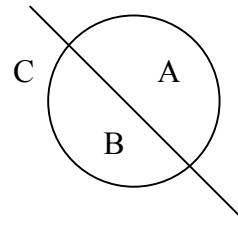
A l'intérieur d'une fenêtre d'analyse C, nous définissons deux zones complémentaires A et B telles que :

$$A \cup B = C$$

$$\forall i, \forall j, a_{ij} + b_{ij} = c_{ij}$$

$$\sum_{i,j \in A} a_{ij} = \sum_{i,j \in B} b_{ij} = \text{cte}$$

$$\sum_{i,j \in C} c_{ij} = \sum_{i,j \in (A \cup B)} a_{ij} + b_{ij} = 2 * \text{cte}$$



Une fois le modèle de transition défini, on déduit toutes ses rotations possibles, ce qui amène à différentes fenêtres A_k, B_k , où k code l'orientation.

On peut alors rechercher :

- l'orientation k qui amène au contraste maximum :

$$\underset{k}{\text{Max}} |IA_k - IB_k| \text{ ou } \underset{k}{\text{Max}} (IA_k - IB_k)^2$$

où $IA_k = \sum_{i,j \in A_k} a_{ij} I_{ij}$ et $IB_k = \sum_{i,j \in B_k} b_{ij} I_{ij}$ représentent la convolution de l'image avec les masques A_k et B_k respectivement.

- l'orientation k qui amène à l'écart quadratique minimum :

$$\varepsilon^2 = \underset{k}{\text{Min}} \left[\sum_{i,j \in A_k} a_{ij} \cdot (I_{ij} - IA_k / \text{cte})^2 + \sum_{i,j \in B_k} b_{ij} \cdot (I_{ij} - IB_k / \text{cte})^2 \right]$$

$$= \underset{k}{\text{Min}} \left[\sum_{i,j \in A_k} a_{ij} I_{ij}^2 + a_{ij} \cdot \frac{IA_k^2}{\text{cte}^2} - 2a_{ij} I_{ij} \frac{IA_k}{\text{cte}} + \sum_{i,j \in B_k} b_{ij} I_{ij}^2 + b_{ij} \cdot \frac{IB_k^2}{\text{cte}^2} - 2b_{ij} I_{ij} \frac{IB_k}{\text{cte}} \right]$$

$$= \underset{k}{\text{Min}} \left[\sum_{i,j \in A_k} a_{ij} I_{ij}^2 + \frac{IA_k^2}{\text{cte}^2} \sum_{i,j \in A_k} a_{ij} - 2 \frac{IA_k}{\text{cte}} \sum_{i,j \in A_k} a_{ij} I_{ij} + \sum_{i,j \in B_k} b_{ij} I_{ij}^2 + \frac{IB_k^2}{\text{cte}^2} \sum_{i,j \in B_k} b_{ij} - 2 \frac{IB_k}{\text{cte}} \sum_{i,j \in B_k} b_{ij} I_{ij} \right]$$

En posant $Ic = \sum_{i,j \in C} c_{ij} I_{ij}$ et $I2c = \sum_{i,j \in C} c_{ij} I_{ij}^2$,

$$\varepsilon^2 = \underset{k}{\text{Min}} S2C + \frac{IA_k^2}{\text{cte}} - 2 \frac{IA_k}{\text{cte}} IA_k + \frac{IB_k^2}{\text{cte}} - 2 \frac{IB_k}{\text{cte}} IB_k$$

$$\varepsilon^2 = \underset{k}{\text{Min}} S2C - \frac{IA_k^2}{\text{cte}} - \frac{IB_k^2}{\text{cte}}$$

Comme $Ic = IA_k + IB_k$

$$\varepsilon^2 = \underset{k}{\text{Min}} S2C - \frac{IA_k^2}{\text{cte}} - \frac{(IC - IA_k)^2}{\text{cte}}$$

$$\varepsilon^2 = \underset{k}{\operatorname{Min}} \ S2C - \frac{2IA_k^2 + IC^2 - 2IC \cdot IA_k}{cte}$$

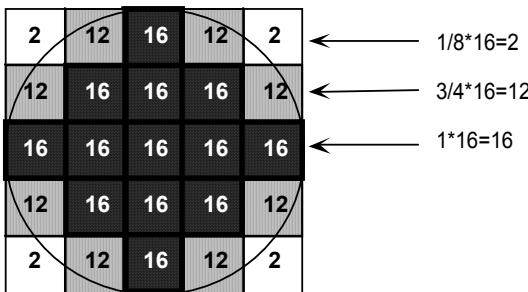
$$\varepsilon^2 = \underset{k}{\operatorname{Min}} \ S2C - \frac{IC^2}{cte} + \frac{2IA_k(1C - IA_k)}{cte}$$

Comme $S2C - \frac{IC^2}{cte}$ est constant, cela revient à minimiser $IA_k(1C - IA_k)$

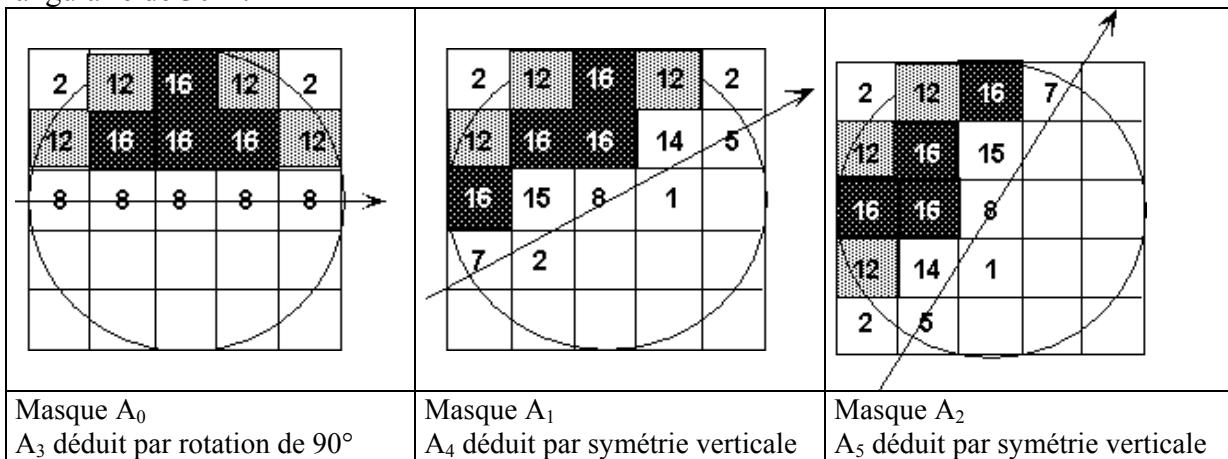
On obtient ainsi l'orientation k qui sépare la fenêtre de l'image en deux zones les plus homogènes possible.

Exemple de mise en pratique.

On choisit comme fenêtre circulaire une fenêtre de taille 5x5 :



On choisit également de prendre 6 orientations possibles, ce qui amène à une résolution angulaire de 30° :



On calcule ensuite IC et IA_k , pour k variant de 0 à 5.

En chaque pixel, on recherche la valeur de k qui amène au minimum de

$$IA_k \cdot (1C - IA_k)$$

On étiquette ensuite le pixel comme contour si

$$|IB_k - IA_k| > S \Rightarrow |1C - 2IA_k| > S$$

où S est un seuil de contraste donné.

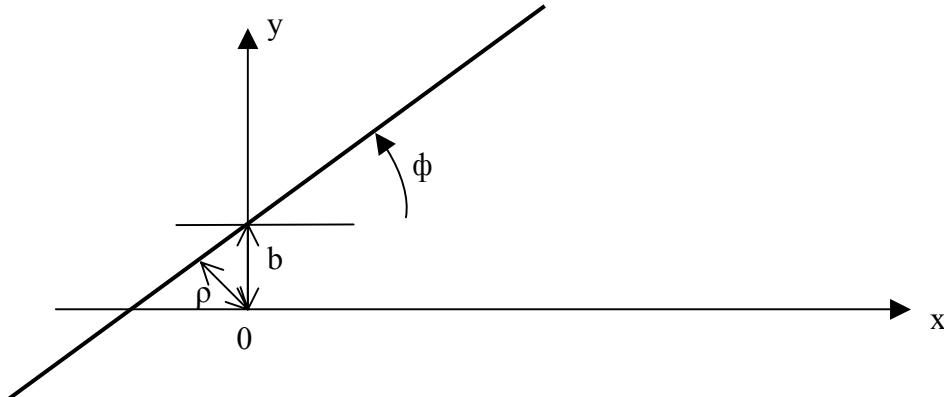
Si $1C < 2IA_k$, l'orientation retenue est l'orientation k, sinon, on ajoute 180° à l'orientation.

V. Transformée de Hough

1. Forme générale

Cette méthode a pour but de détecter des droites lorsque l'on a les points de contour. Considérons une droite dans le plan (x,y) . Elle a pour équation :

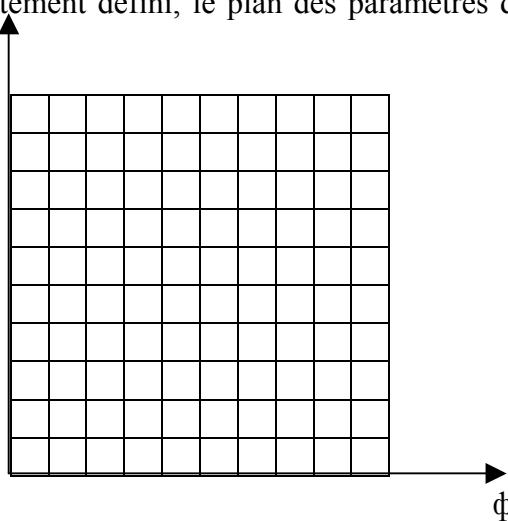
$$y = a x + b \quad \text{ou} \quad x \sin \phi + y \cos \phi = \rho \quad (\text{équation 1})$$



Si $\phi \in [0, \pi]$, les paramètres de la droite (ϕ, ρ) sont uniques.

La transformée de Hough est une transformation qui permet de passer du plan image (x,y) au plan des paramètres.

Le plan image est parfaitement défini, le plan des paramètres devra être discréétisé en $N \times M$ cellules :



Par chaque point de contour, on peut en théorie faire passer une infinité de droite. Ici, comme on a discréétisé les orientations, on pourra seulement faire passer M droites.

Considérons un point contour (x_i, y_i) . Pour chaque orientation ϕ , on peut déterminer avec l'équation un le ρ (et donc la droite d'orientation ϕ passant par ce point). On incrémentera alors de un la case (ϕ, ρ) correspondante dans l'espace des paramètres. Chaque point de contour vote donc pour M points dans l'espace des paramètres (un pour chaque orientation). Si une droite est présente dans l'image, chaque point de contour de cette droite va voter pour la même case de l'espace des paramètres provoquant ainsi une accumulation. En la détectant, on a les coordonnées de la droite.

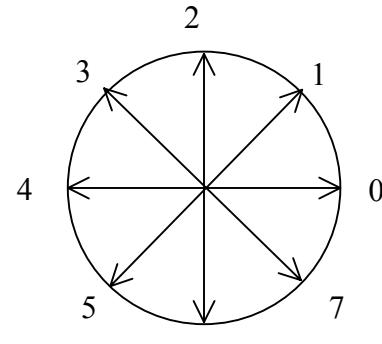
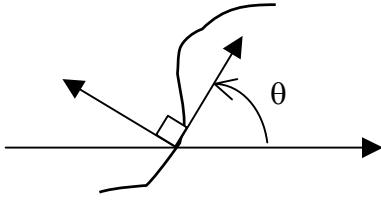
2. Transformée de Hough généralisée

Une amélioration peut-être amenée si en plus de connaître les points de contour (image binaire), on connaît l'orientation de ces points (facile à avoir : c'est l'orientation du gradient).

Pour chaque point contour, on calcule ρ uniquement pour l'orientation perpendiculaire à celle du gradient. L'accumulation dans l'espace des paramètres est alors beaucoup plus nette.

Cette méthode peut être étendue à des courbes de forme quelconque.

VI. Codage des orientations et suivi de contours



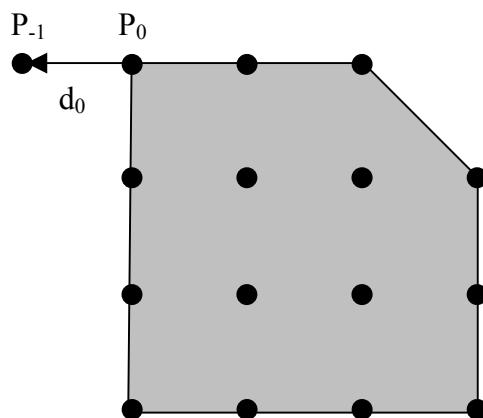
L'orientation codée est celle de la tangente au contour, normale à l'orientation du gradient. Une résolution angulaire de 45° correspond à un codage de Freeman à 8 états.

Un contour est décrit par la succession des codes de Freeman de ses pixels (chaînes de Freeman). Ce code est invariant :

- strictement en translation
- à une constante additive près, modulo 8, en rotation
- à la longueur des sous-chaînes près en homothétie.

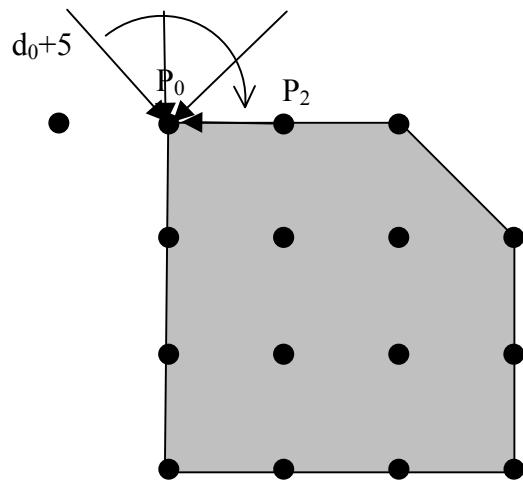
Ce codage exige d'une part d'être en présence d'un contour d'épaisseur unitaire et fermé et d'autre part, de réaliser un suivi de contour.

Pour cela, on parcourt l'image de haut en bas et de la gauche vers la droite pour trouver le premier point de contour P_0 . On connaît aussi le point du fond que l'on a parcouru juste avant d'obtenir P_0 : P_{-1} .



Appelons d_0 la direction qui permet de passer de P_0 à P_{-1} .

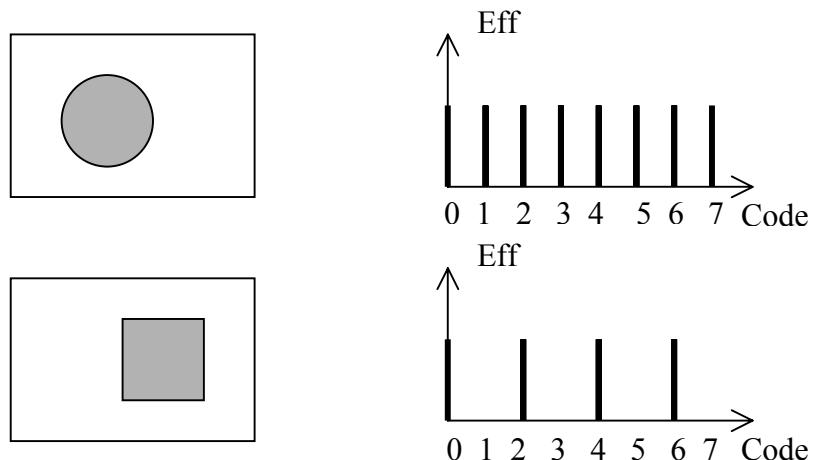
Pour obtenir le point P_2 , on parcourt les voisins de P_0 , dans le sens trigonométrique, à partir de la direction (d_0+5) modulo 8, et ce, jusqu'à arriver à un voisin qui soit point de contour. Ce voisin est alors le point P_2 :



On recommence de même pour obtenir P_3 : on parcourt les voisins de P_2 , dans le sens trigonométrique, à partir de la direction (d_1+5) modulo 8, et ce, jusqu'à arriver à un voisin qui soit point de contour. d_1 est la direction reliant le point P_2 au point P_1 .

On continue ainsi tant qu'il y a des points de contour.

Un autre codage de forme, dérivé des chaînes de Freeman, mais ne faisant aucune hypothèse sur le contour est l'histogramme des orientations :



Chapitre 5 : Segmentation en régions

Nous allons aborder ici le principe de plusieurs méthodes de segmentation en régions :

- par agrégation de pixel (fusion)
- par division
- par division-fusion
- par quadtree

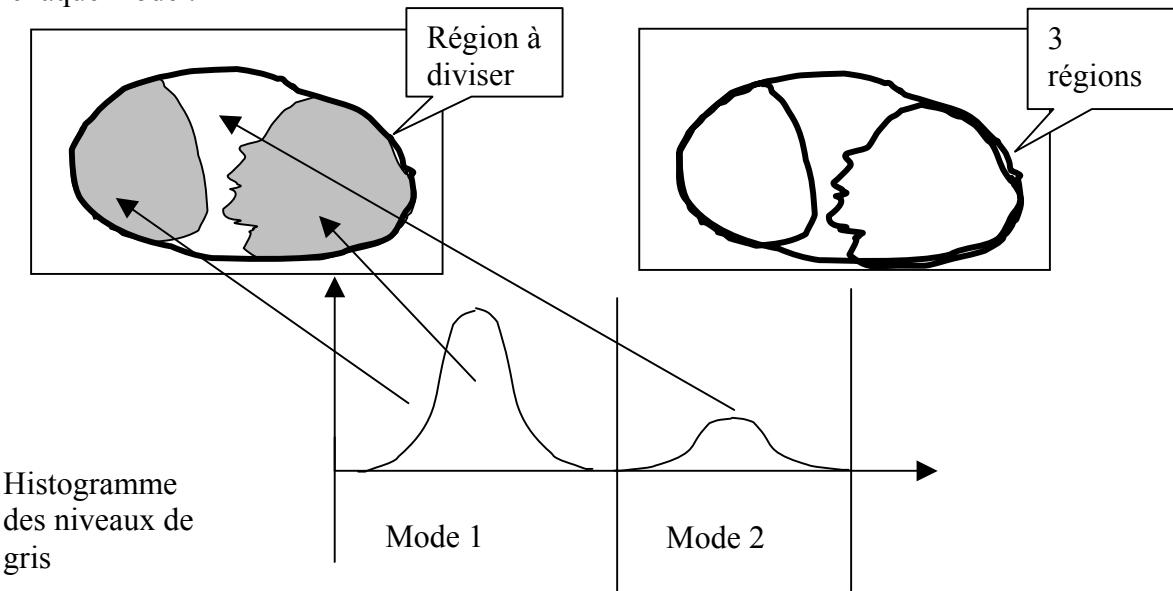
I. Segmentation par agrégation de pixels

L'initialisation de cette méthode consiste à considérer chaque pixel comme une région. On va alors essayer de les regrouper entre elles avec un double critère de similarité des niveaux de gris et d'adjacence. Le critère de similarité peut par exemple être : la variance des niveaux de gris de la région R est inférieure à un seuil.

Le principe de l'agrégation de pixel est le suivant : on choisit un germe et on fait croître ce germe tant que des pixels de son voisinage vérifient le test d'homogénéité. Lorsqu'il n'y a plus de pixel candidat dans le voisinage, on choisit un nouveau germe et on itère le processus.

II. Segmentation par division

Au contraire de la méthode précédente, celle-ci suppose au départ que tous les pixels appartiennent à la même région. Si la région n'est pas homogène (critère d'homogénéité), elle est divisée en plusieurs régions, sinon, le processus se termine. Chaque région nouvellement créée est potentiellement redivisée en plusieurs régions si elle n'est pas homogène. Le critère d'homogénéité peut être le même que précédemment (variance des niveaux de gris inférieure à un seuil). Lors de la division, on peut rajouter un critère d'arrêt sur la taille des régions. La division peut être réalisée de plusieurs façons. Une technique consiste à déterminer les différents modes de l'histogramme des niveaux de gris et à affecter une ou plusieurs région à chaque mode :

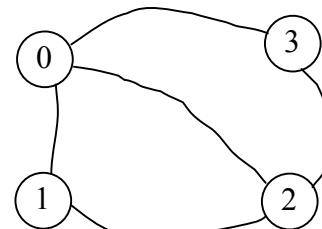
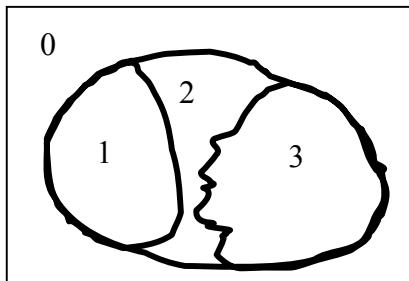


III. Segmentation par division-fusion

Cette méthode consiste à reprendre le résultat de la division et à essayer de fusionner des régions qui ont été malencontreusement séparées. Pour ce faire, on utilise un outil très précieux : le graphe d'adjacence des régions.

IV. Graphe d'adjacence

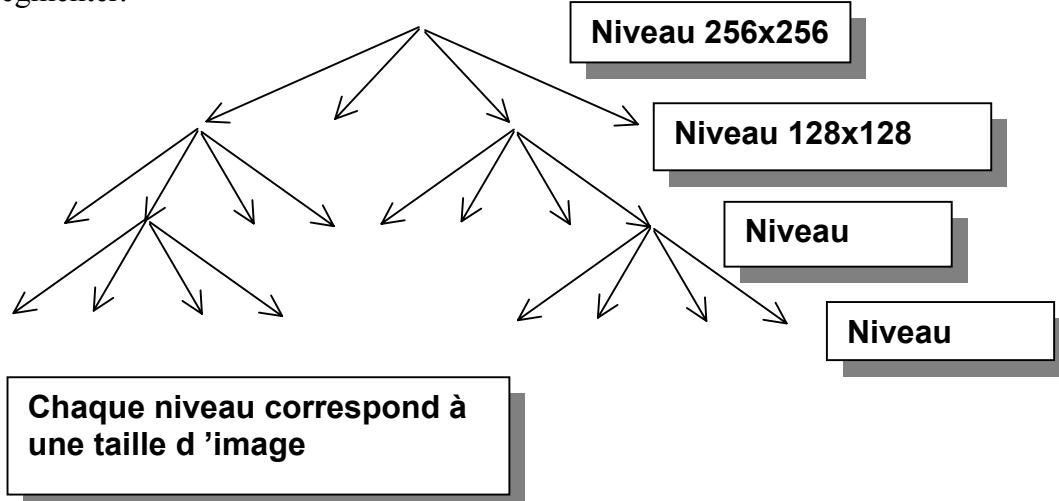
Le graphe d'adjacence est un graphe non orienté où chaque nœud représente une région et chaque branche représente l'adjacence entre les régions :



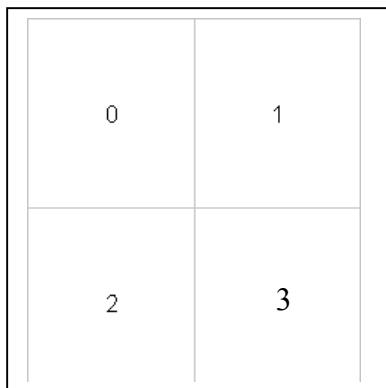
Grâce au graphe d'adjacence, on n'a pas à remonter au pixels de l'image pour fusionner les régions.

V. Quad-tree

C'est une méthode récursive de découpage d'une image. Au départ, l'image est constituée d'un seul rectangle qui est découpé en 4 quartiers si il n'est pas jugé homogène. Les 4 nouveaux rectangles sont ses « fils » dans un arbre (quadtree). Chacun des fils est de nouveau découpé en 4 s'il n'est pas homogène. Cette méthode peut servir à compresser une image ou à la segmenter.



Dans le quad-tree linéaire, chaque feuille du quadtree est codée de manière à pouvoir retrouver sa position. Principe de codage retenu :



00	01	10	11	000	001	010	011	100	101	110	111
				002	003	012	013	102	103	112	113
02	03	12	13	020	021	030	031	120	121	130	131
				022	023	032	033	122	123	132	133
20	21	30	31	200	201	210	211	300	301	310	311
				202	203	212	213	302	303	312	313
22	23	32	33	220	221	230	231	320	321	330	331
				222	223	232	233	322	323	332	333

À chaque étape de la division, on code les 4 nouveaux fils avec la même séquence : 0,1,2,3.

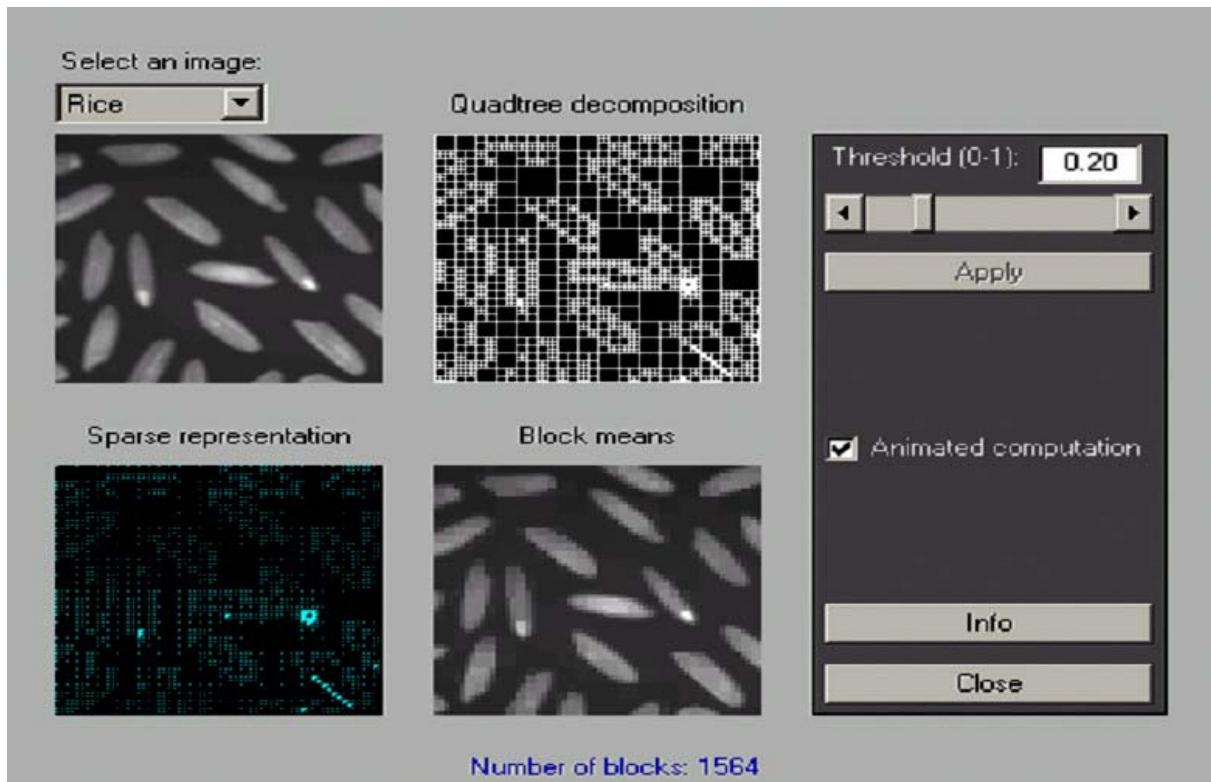
A partir des coordonnées d'un pixel, on pourra alors retrouver très facilement le numéro de la feuille et vice-versa. Il suffit d'intercaler les numéros des lignes et des colonnes :

010 (numéro de colonne en binaire)

000	001	010	011	100	101	110	111
002	003	012	013	102	103	112	113
020	021	030	031	120	121	130	131
022	023	032	033	122	123	132	133
200	201	210	211	300	301	310	311
202	203	212	213	302	303	312	313
220	221	230	231	320	321	330	331
222	223	232	233	322	323	332	333

011 et 010 0 0 1 1 1 0
 0 3 2

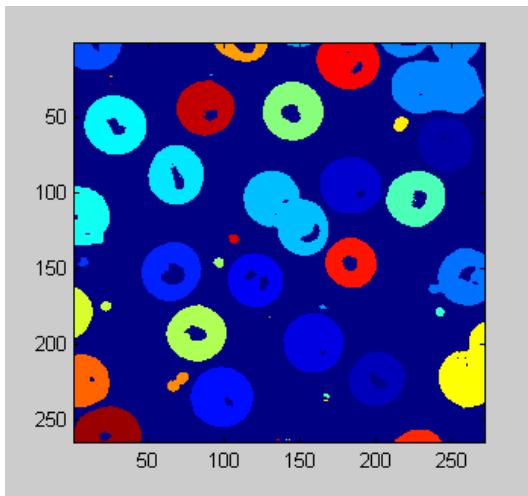
Exemple sous matlab :



VI. Etiquetage des régions

La plupart des méthodes citées ci-dessus requièrent un algorithme d'étiquetage en composantes connexes : tous les pixels connexes possédant le même attribut doivent être affectés à la même région.

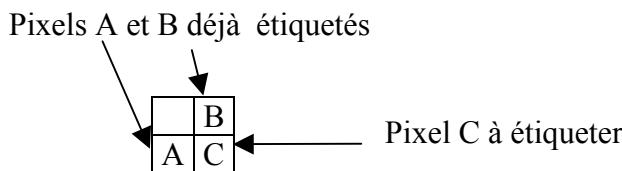
Exemple :



Il existe plusieurs algorithmes permettant de réaliser l'étiquetage.

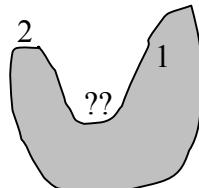
1. Algorithme intuitif

Supposons qu'à chaque pixel de l'image est affecté un attribut. On veut regrouper tous les pixels connexes possédant le même attribut (noté att dans la suite). Pour cela, on va balayer l'image et donner une étiquette (image notée E) à chaque nouveau pixel C.



On déplace un masque en L

1. Si $\text{att}(C) = \text{att}(A)$ et $\text{att}(C) \neq \text{att}(B)$
 → $E(C) = E(A)$
2. Si $\text{att}(C) = \text{att}(B)$ et $\text{att}(C) \neq \text{att}(A)$
 → $E(C) = E(B)$
3. Si $\text{att}(C) \neq \text{att}(B)$ et $\text{att}(C) \neq \text{att}(A)$
 → $E(C) = \text{nouvelle étiquette}$
4. Si $\text{att}(C) = \text{att}(B)$ et $\text{att}(C) = \text{att}(A)$ et $E(A) = E(B)$
 → $E(C) = E(B)$
5. Si $\text{att}(C) = \text{att}(B)$ et $\text{att}(C) = \text{att}(A)$ et $E(A) \neq E(B)$
 → $E(C) = E(B)$ et remonte dans l'image et change toutes les $E(A)$ en $E(B)$



			1	1
	2	??		
	2			
3	??			
3				

Cet algorithme amène à un étiquetage correct de l'image, mais en combien de temps ? A chaque fois qu'une configuration telle que les précédentes se présente, il faut rebalayer toute l'image.

2. Algorithme évolué

Pour éviter ce problème, on fait appel à une table de correspondance. Celle-ci est un vecteur ou à chaque étiquette, on affecte une autre étiquette qui lui correspond. Par exemple, dans le cas précédent, on aurait $T[2]=1$.

Initialisation : $T[i] = i$ pour tout i

Premier balayage :

On déplace un masque en L de la même façon que précédemment,

1. Si $\text{att}(C) = \text{att}(A)$ et $\text{att}(C) \neq \text{att}(B)$
 → $E(C)=E(A)$
2. Si $\text{att}(C) = \text{att}(B)$ et $\text{att}(C) \neq \text{att}(A)$
 → $E(C)=T[E(B)]$
3. Si $\text{att}(C) \neq \text{att}(B)$ et $\text{att}(C) \neq \text{att}(A)$
 → $E(C)=$ nouvelle étiquette
4. Si $\text{att}(C) = \text{att}(B)$ et $\text{att}(C) = \text{att}(A)$ et $E(A)=E(B)$
 → $E(C)=E(A)$
5. Si $\text{att}(C) = \text{att}(B)$ et $\text{att}(C) = \text{att}(A)$ et $E(A) \neq E(B)$
 → $E(C)=\min(T[E(B)], E(A))$
 $T [E(C)] = E(C)$
 $T [E(A)] = E(C)$
 $T [\max(T[E(B)], E(A))] = E(C)$

Mise à jour de la table de correspondance :

Toutes les étiquettes telles que $T[i]=i$ représentent de vraies régions. On leur affecte donc un numéro de région en numérotant dans l'ordre croissant.

Pour toutes les étiquettes qui ne valident pas l'hypothèse ci-dessus, on fait $T[i]=T[T[i]]$.

Deuxième balayage :

A chaque pixel d'étiquette i , on lui affecte l'étiquette $T[i]$.

Cette méthode ne nécessite donc que 2 balayages de l'image.

Exemple :

1 1 1	2 2 2	Table d'équivalence T	Mise à jour de la table	1 1 1	1 1 1
1 1 1 1 1 1 1		1 1	1 1	1 1 1	1 1 1
		2 1	2 1	1 1 1	1 1 1
3		3 3	3 2	2 1 1	2 1 1
3		4 3	4 3	2 1 1	2 1 1
3 3 3 3 3 3 3		5 4	5 4	2 2 2	2 2 2
3 3 3 3 3 3 3				2 2 2	2 2 2

VII. Caractéristiques de Forme extraites des régions

Supposons que lors de traitements antérieurs, on ait réussi à isoler des régions. On veut maintenant caractériser ces régions pour envoyer des vecteurs les représentant à un futur

processus de reconnaissance des formes. Soit E l'image correspondant aux étiquettes et I, l'image de luminance d'origine.

1. Taille

Un des paramètres le plus intuitif est la taille de la région représentée dans l'image par son nombre de pixels. Celui-ci s'obtient sous Matlab, pour la région « i » avec :

$E=(E==i)$; %tous les pixels de l'étiquette i sont à 1, les autres sont à 0
 $Taille=\text{sum}(E(:))$;

2. Position

La position de la région (barycentre) est extraite avec :

$[y,x]=\text{find}(E==i)$;
 $ymoy=\text{mean}(y)$;
 $xmoy=\text{mean}(x)$;

3. Direction principale

Elle correspond au premier vecteur propre de la matrice :

$$\begin{pmatrix} \sigma_x^2 & \sigma_x \sigma_y \\ \sigma_x \sigma_y & \sigma_y^2 \end{pmatrix} \quad \text{où} \quad \left\{ \begin{array}{l} \sigma_x^2 = \frac{1}{N} \sum_{i=1}^N (x_i - x_{moy})^2 \\ \sigma_y^2 = \frac{1}{N} \sum_{i=1}^N (y_i - y_{moy})^2 \\ \sigma_x \sigma_y = \frac{1}{N} \sum_{i=1}^N (x_i - x_{moy})(y_i - y_{moy}) \end{array} \right. \quad \text{et } N \text{ est le nombre de pixels de la région}$$

L'axe principal de la région est défini par l'angle θ qu'il fait avec l'axe des x :

$$\theta = \frac{1}{2} \arctan\left(\frac{2\sigma_x \sigma_y}{\sigma_x^2 - \sigma_y^2}\right)$$

Sous Matlab, pour la région i,

$[y,x]=\text{find}(E==i)$;
 $ymoy=\text{mean}(y)$;
 $xmoy=\text{mean}(x)$;
 $sx2=\text{mean}((x-xmoy).^2)$;
 $sy2=\text{mean}((y-ymoy).^2)$;
 $sxy=\text{mean}((x-xmoy).* (y-ymoy))$;
 $theta=0.5*\text{atan}(2sxy/(sx-sy))$

4. Moments géométriques

Les moments géométriques 2D d'ordre $(p+q)$, pour une fonction $f(x,y)$ sont définis par :

$$M_{pq} = \iint x^p y^q f(x, y) dx dy$$

où $p, q = 0, 1, \dots, \infty$.

L'ensemble des moments d'ordre n consiste en tous les M_{pq} tels que $p+q \leq n$. Cet ensemble contient $\frac{1}{2}(n+1)(n+2)$ éléments.

Masse et aire

La définition du moment M_{00} d'une fonction $f(x,y)$ est :

$$M_{00} = \iint f(x, y) dx dy$$

Il représente la masse d'une fonction ou image $f(x,y)$. Pour une image binaire, M_{00} représente l'aire totale d'une image.

Centre de masse

Les deux premiers moments de l'image sont :

$$M_{10} = \int \int x f(x,y) dx dy \text{ et } M_{01} = \int \int y f(x,y) dx dy$$

Les coordonnées du centre de masse sont définies par :

$$\bar{x} = \frac{M_{10}}{M_{00}} \text{ et } \bar{y} = \frac{M_{01}}{M_{00}}$$

Ils représentent la position dans l'image $f(x,y)$ qui peut être utilisée comme point de référence. Pour une image binaire, ce point représente le centre de gravité de l'image.

Moments centrés

Les moments centrés de $f(x,y)$ sont définis par :

$$\mu_{pq} = \int \int (x - \bar{x})^p (y - \bar{y})^q f(x,y) dx dy$$

Ils ont la propriété d'être invariant en translation.

Orientation

Les moments du second ordre (aussi appelés moment d'inertie) $\{M_{20}, M_{11}, M_{02}\}$ peuvent être utilisés pour déterminer l'orientation qui décrit les directions principales de l'image :

$$\theta = \frac{1}{2} \arctan\left(\frac{2\mu_{11}}{\mu_{20} - \mu_{02}}\right)$$

θ est l'angle de la direction principale par rapport à l'axe des x et varie entre $-\pi/4 < \theta < \pi/4$.

Moments invariants

En utilisant la théorie algébrique des invariants, Hu a défini un ensemble de combinaisons de moments qui sont invariants en position, échelle et orientation. Ces combinaisons seront donc particulièrement utilisées en reconnaissance des formes :

$$\Phi_1 = \mu_{20} + \mu_{02}$$

$$\Phi_2 = (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2$$

$$\Phi_3 = (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2$$

$$\Phi_4 = (\mu_{30} + \mu_{12})^2 + (\mu_{21} + \mu_{03})^2$$

$$\Phi_5 = (\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{12}) \left[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2 \right]$$

$$+ (3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03}) \left[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2 \right]$$

$$\Phi_6 = (\mu_{20} - \mu_{02}) \left[(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2 \right] + 4\mu_{11}(\mu_{30} + \mu_{12})(\mu_{21} + \mu_{03})$$

$$\Phi_7 = (3\mu_{21} - 3\mu_{03})(\mu_{30} + \mu_{12}) \left[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2 \right]$$

$$- (\mu_{30} - 3\mu_{12})(\mu_{21} + \mu_{03}) \left[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2 \right]$$

5. Compacité

On peut également définir la compacité d'une région comme :

$$C = 4\pi \frac{\text{Surface}}{\text{Périmètre}^2}$$

Propriétés de la compacité:

- invariante en rotation
- Invariante en translation
- Invariante au changement d'échelle

Exemple

- Pour le disque $C = 4\pi \frac{\pi R^2}{(2\pi R)^2} = 1$
- Pour le carré $C = 4\pi \frac{l^2}{(4l)^2} = \frac{\pi}{4} = 0.78$
- Pour le rectangle ($L=2$, $l=0.5$) $C = 4\pi \frac{lxL}{4(l+L)^2} = 4\pi \frac{1}{4(2.5)^2} = \frac{4\pi}{25} = 0.5$

Autrement dit, plus une forme a une allure compacte et plus sa compacité sera grande. La plus grande valeur (1) correspond au cercle (forme la plus compacte qui soit).

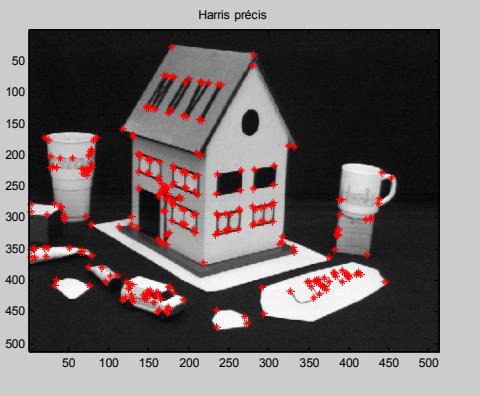
6. Moyenne et écart type des niveaux de gris de la région

On peut aussi représenter la région par la moyenne et l'écart type de ces niveaux de gris. Avec Matlab, toujours pour la région « i »,

```
Pix=find(E==i) ;
Moyenne=mean(I(Pix)) ;
Ecarttype=std(I(Pix)) ;
```


Chapitre 6 : Détection de points d'intérêt

Les points d'intérêt sont également une primitive pertinente que l'on peut extraire des images. Ils sont largement utilisés pour faire de l'appariement d'images, du calibrage de cameras, des mosaïques d'images, de la mise en correspondance,....
Nous présentons ci-dessous les coins extraits d'une image de maison.



La détection peut être réalisée de deux façons :

- à partir des contours
- à partir du signal

I. Détection des points d'intérêt à partir des contours

Dans un premier temps, une extraction des points de contour est réalisée. Ensuite, trois approches peuvent être considérées :

- Calculer la courbure en chaque point de contour et extraire ceux qui ont une grande courbure
- Une fois les contours extraits, faire une approximation polygonale de ceux-ci (on remplace les contours par des segments). Toute intersection de segment est alors interprétée comme un point d'intérêt
- Réaliser un chaînage des points de contour et coder l'orientation des gradients en chaque point. On recherche ensuite des ruptures dans la chaîne des orientations.

Le problème de toutes les méthodes basées sur les contours est que le gradient est très mal défini près des coins :

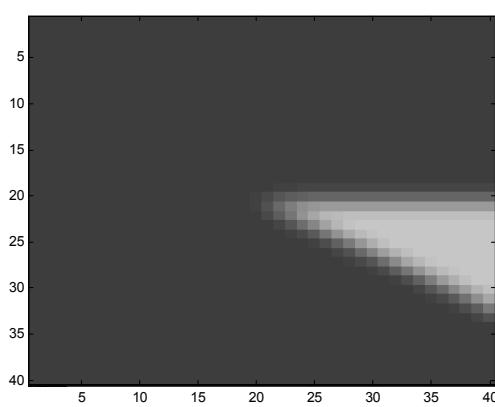
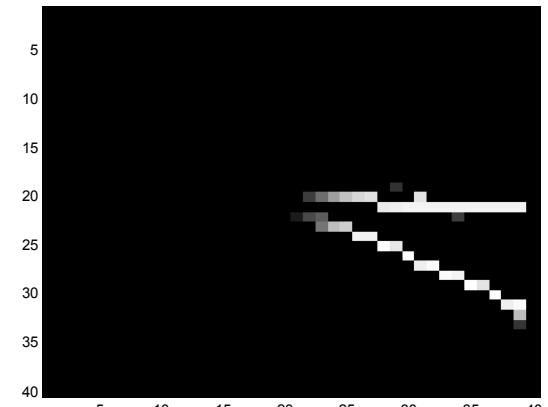


Image de synthèse d'un coin



Module du gradient

Ainsi, les points correspondant à l'angle ne seront bien souvent même pas détectés lors de la détection de contour (module du gradient trop faible), ce qui invalide la méthode.

II. Détection directe des points d'intérêt à partir du signal

1. Méthode de Beaudet

En chaque point de l'image est calculée :

$$DET = I_{xx}I_{yy} - I_{xy}^2$$

où I_{xx} , I_{yy} et I_{xy} représentent les dérivées secondes de l'image par rapport à x , y , et xy . La géométrie différentielle montre que cette mesure, en valeur absolue, est grande près des coins. On recherchera donc les maxima locaux de DET .

Le problème qui apparaît avec cette méthode est qu'elle est basée sur des calculs de dérivées secondes, lesquelles sont très sensibles au bruit.

2. Détecteur de Harris

Harris nous propose d'utiliser comme mesure :

$$C = \langle I_x^2 \rangle \langle I_y^2 \rangle - \langle I_x I_y \rangle - \lambda (\langle I_x^2 \rangle + \langle I_y^2 \rangle)^2$$

où $\langle X \rangle$ représente la convolution de X par un masque gaussien.

Il montre que C est grand que si l'on est situé sur un coin. La détection est donc faite en recherchant les maxima locaux de C .

L'avantage de cette méthode est que l'on n'utilise que des dérivées premières.

3. Détecteur basé sur le gradient

On considère que près d'un coin, la norme du produit vectoriel entre deux vecteurs gradient est grande, alors qu'il est petit ailleurs : si on est dans une zone homogène, le module des vecteurs gradient est petit et si on est situé sur un contour rectiligne, l'angle entre les vecteurs est petit, ce qui amène à une faible norme du produit vectoriel. Ainsi, on calcule la moyenne des produits vectoriels entre le gradient du point étudié et celui de ses voisins :

$$\begin{aligned} k &= \sum_{j \in V_i} \left\| \overrightarrow{\text{grad } P_i} \wedge \overrightarrow{\text{grad } P_j} \right\|^2 \\ k &= \sum_{j \in V_i} \left\| \overrightarrow{\text{grad } P_i} \right\|^2 \left\| \overrightarrow{\text{grad } P_j} \right\|^2 \cdot \sin^2(\overrightarrow{\text{grad } P_i}, \overrightarrow{\text{grad } P_j}) \end{aligned} \quad (\text{équation 1})$$

Dans cette expression,

$$\left\| \overrightarrow{\text{grad } P} \right\|^2 = I_x^2 + I_y^2 \quad \text{et} \quad \sin(\overrightarrow{\text{grad } P}) = \frac{I_y}{\sqrt{I_x^2 + I_y^2}}$$

En développant l'expression précédente, on arrive à :

$$k = I_x^2 \langle I_y^2 \rangle + I_y^2 \langle I_x^2 \rangle - 2I_x I_y \langle I_x I_y \rangle$$

où $\langle \rangle$ dénote la convolution par le masque :

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

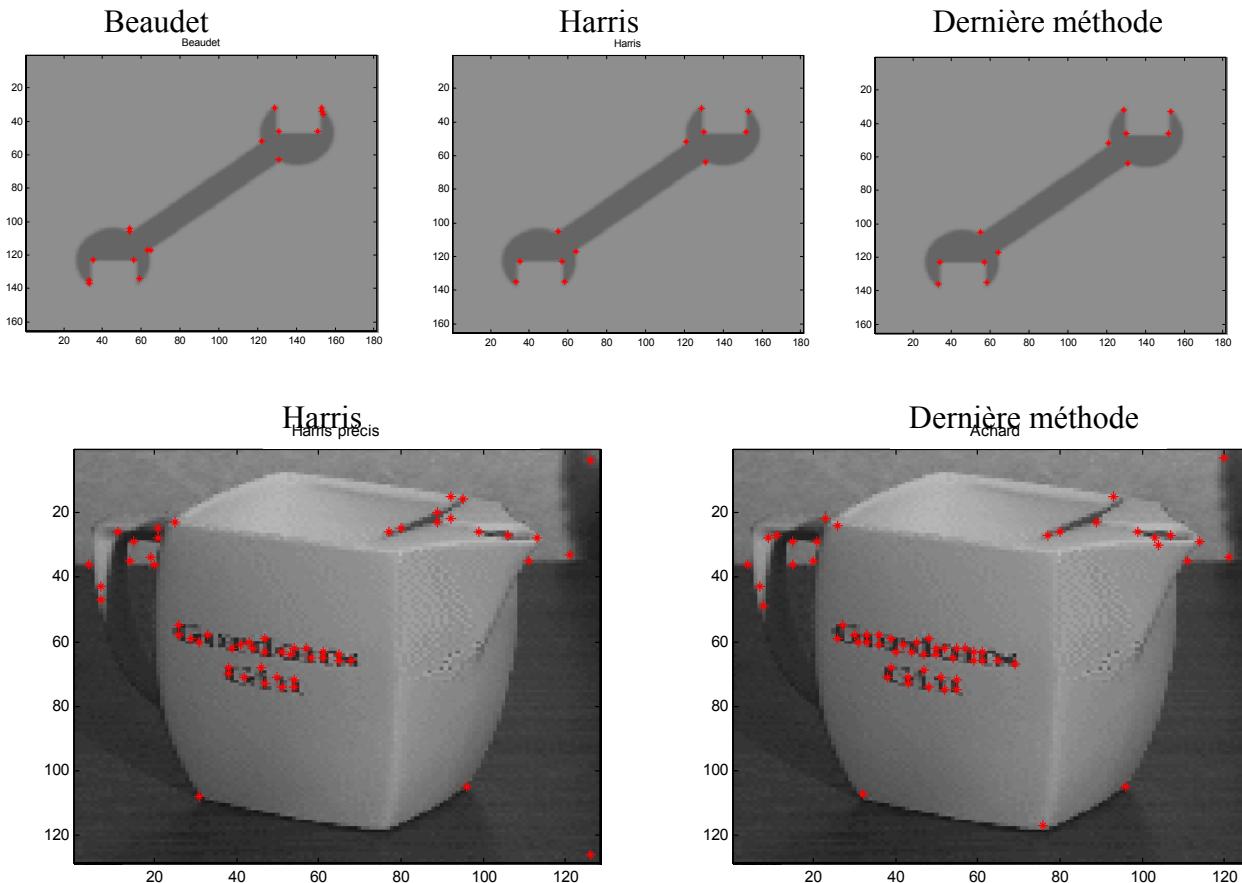
k est similaire à la norme du gradient puissance 4 multipliée par le carré du sinus de l'angle entre les gradients (équation 1). Si on souhaite favoriser autant le gradient que l'angle, il faut diviser k par le carré du gradient :

$$k = \frac{Ix^2 < Iy^2 > + Iy^2 < Ix^2 > - 2IxIy < IxIy >}{< Ix^2 > + < Iy^2 >}$$

De la même façon que précédemment, on recherche ensuite les maxima locaux de k .

4. Comparaison des méthodes

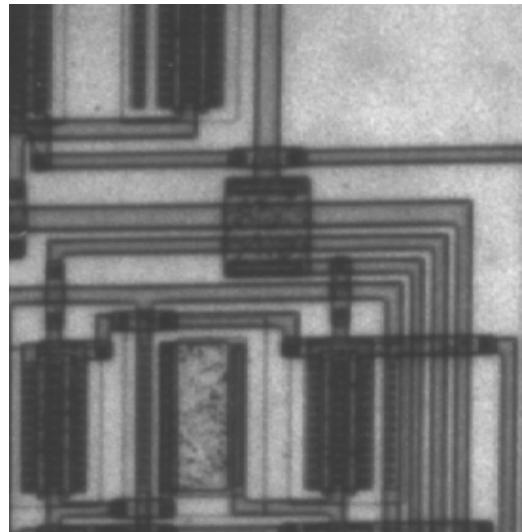
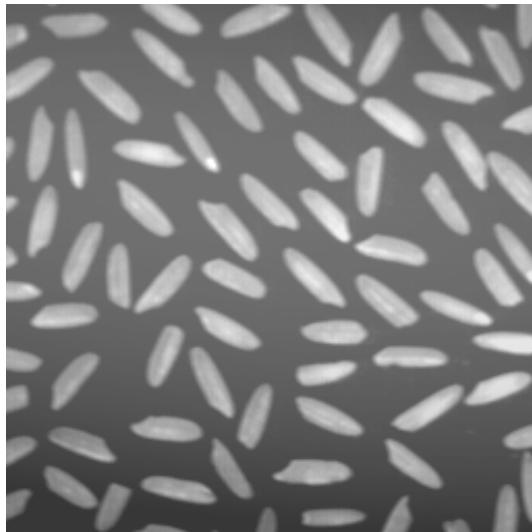
De manière générale, le détecteur de Beaudet est très sensible au bruit. Le détecteur de Kitchen , ainsi que la dernière méthode amènent à des résultats plus robustes. Une étude un peu plus fine montre que le détecteur de Harris à tendance à ne pas détecter les angles obtus.



Chapitre 7 : Morphologie mathématique

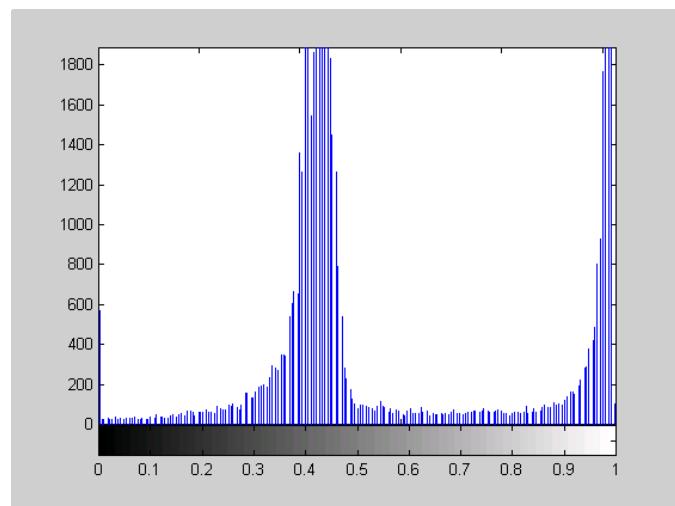
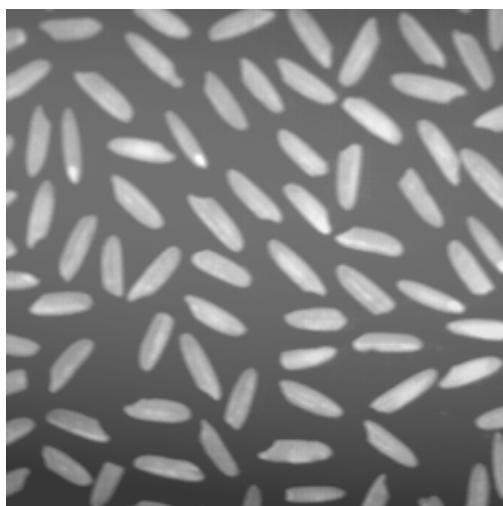
Il existe deux grands types de morphologie mathématique : la morphologie binaire (ou ensembliste) et la morphologie multi-niveaux (ou fonctionnelle). La première nécessite en entrée une image binaire, la première étape consistera donc à binariser les images. Bien sûr, toutes les images ne se prêtent pas à ce type de traitement, c'est néanmoins souvent le cas des images industrielles possédant deux grandes classes de niveaux de gris : une correspondant à l'objet et une correspondant au fond). La morphologie multi-niveaux s'applique elle directement sur les images.

I. Binarisation des images



Certaines images se prêtent très bien à la binarisation, ce sont les images bimodales (voir ci-dessus). Le problème de la binarisation consiste à déterminer un seuil pour séparer les différents objets de la scène : tous les pixels ayant un niveau de gris inférieur au seuil seront mis à 0, les autres seront mis à 1. Il n'est pas envisageable pour certaines applications, de déterminer le seuil à la main, il va donc falloir le trouver de manière automatique.

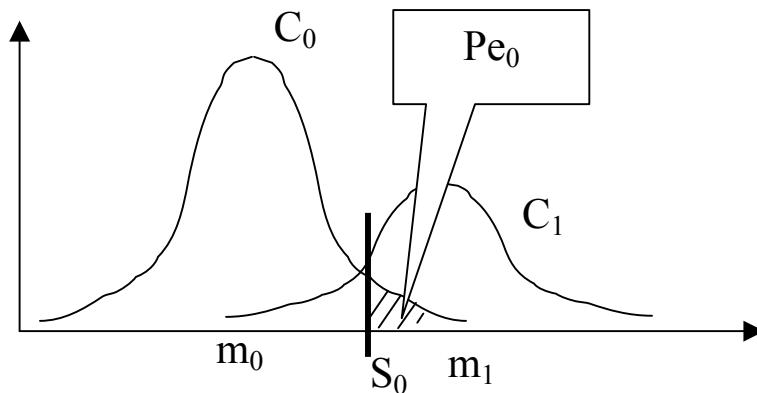
1. Binarisation avec l'histogramme



L'histogramme des images bimodales possède deux modes. On calcule donc l'histogramme de l'image en excluant les zones frontières (gradient élevé). Puis on lisse l'histogramme et on extrait le min entre les deux pics qui correspond au seuil de binarisation.

2. Binarisation en utilisant un critère bayésien

Hypothèse : On suppose que les niveaux de gris sont répartis en deux classes C0 et C1. Chaque classe est modélisée par une loi gaussienne de moyenne m_0 et m_1 et d'écart type V. Les deux classes possèdent la même variance caractérisant le bruit. Celui-ci est considéré stationnaire et affecte donc de la même façon tous les niveaux de gris. Chaque classe a pour effectif (nombre de pixel appartenant à la classe) N_0 et N_1 . On recherche donc le seuil S_0 qui minimise la probabilité d'erreur :



Cette probabilité d'erreur est donnée par :

$$P_{\text{erreur}} = Pe_0 \cdot P(C_0) + Pe_1 \cdot P(C_1)$$

avec :

$$Pe_0 = \frac{1}{\sqrt{2 \pi V}} \int_{S_0}^{\infty} \exp\left(-\frac{(S-m_0)^2}{2V}\right) dS \quad \text{et} \quad Pe_1 = \frac{1}{\sqrt{2 \pi V}} \int_{-\infty}^{S_0} \exp\left(-\frac{(S-m_1)^2}{2V}\right) dS$$

On minimise cette probabilité d'erreur par rapport à S, soit :

$$\frac{dP_{\text{erreur}}}{dS} = \frac{1}{\sqrt{2 \pi V}} \left[-P(C_0) \exp\left(-\frac{(S_0-m_0)^2}{2V}\right) + P(C_1) \exp\left(-\frac{(S_0-m_1)^2}{2V}\right) \right] = 0$$

donc,

$$\ln(P(C_0)) - \frac{(S_0-m_0)^2}{2V} = \ln(P(C_1)) - \frac{(S_0-m_1)^2}{2V}$$

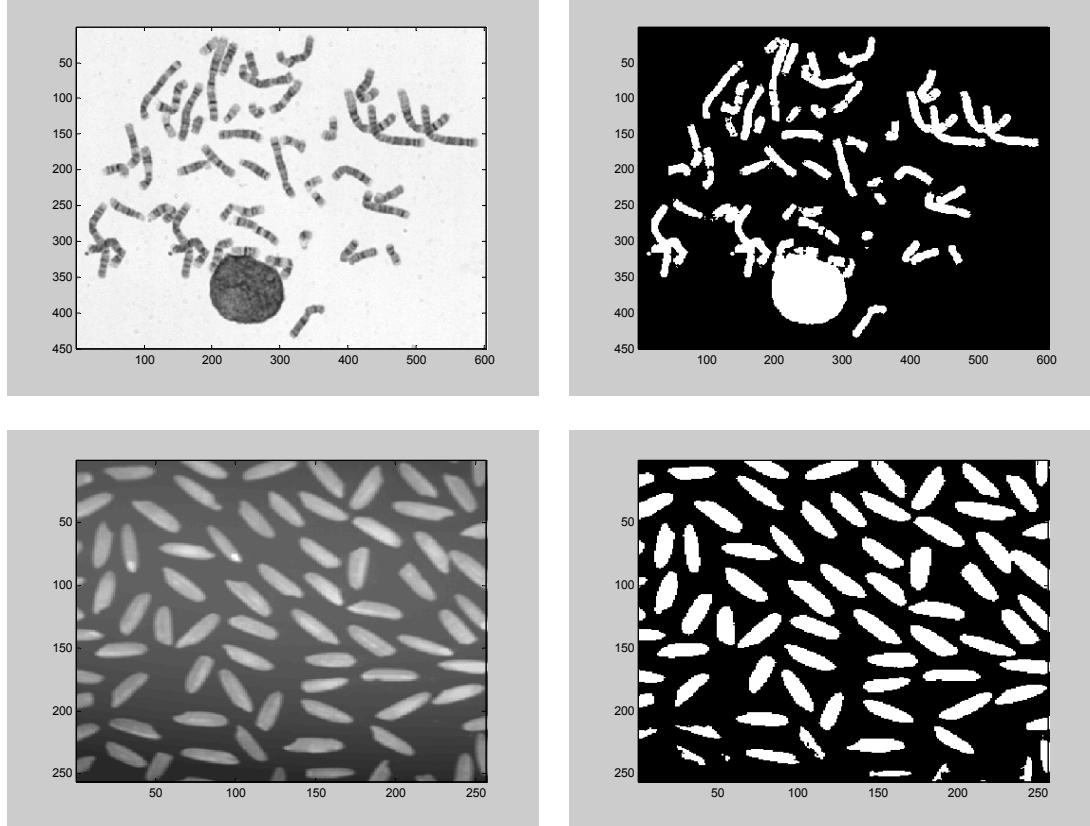
et le seuil S_0 :

$$S_0 = \frac{m_0 + m_1}{2} + \frac{V}{m_1 - m_0} \ln\left(\frac{P(C_0)}{P(C_1)}\right)$$

On peut estimer m_i , $P(C_i)$ et S_0 par itérations successives. Voici l'algorithme :

2. On initialise S_0 =moyenne des ndg de l'image.
3. On réalise le seuillage à deux classes C0 et C1.
4. On en déduit m_0 , m_1 , $P(C_0)$, $P(C_1)$ et V
5. A partir de ces nouvelles valeurs, on re-calcule S_0 .
6. Retour en 2 jusqu'à stabilisation

Exemples de binarisation :



3. Binarisation en utilisant un critère structurel

On va rechercher un seuil de manière à avoir des régions stables, avec une bonne régularité topologique. Cette notion de régularité est obtenue grâce à la compacité :

$$C = 4\pi \frac{\text{Surface}}{\text{Périmètre}^2}$$

Propriétés de la compacité:

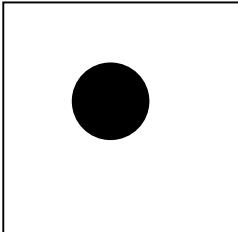
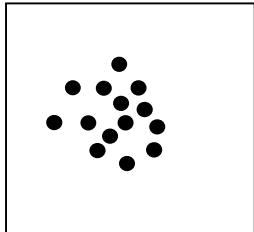
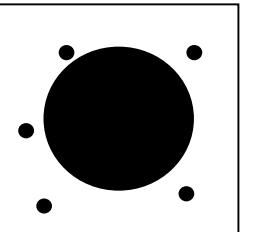
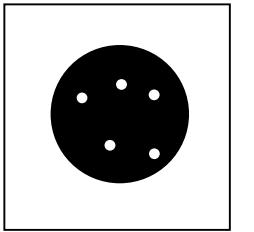
- invariante en rotation
- Invariante en translation
- Invariante au changement d'échelle

Exemple

- Pour le disque $C = 4\pi \frac{\pi R^2}{(2\pi R)^2} = 1$
- Pour le carré $C = 4\pi \frac{l^2}{(4l)^2} = \frac{\pi}{4} = 0.78$
- Pour le rectangle ($L=2$, $l=0.5$) $C = 4\pi \frac{l \times L}{4(l+L)^2} = 4\pi \frac{1}{4(2.5)^2} = \frac{4\pi}{25} = 0.5$

Autrement dit, plus une forme a une allure compacte et plus sa compacité sera grande. La plus grande valeur (1) correspond au cercle (forme la plus compacte qui soit). La binarisation utilisera ce critère : on seuille avec chaque niveau de gris et on retient comme seuil définitif celui qui amène à la plus grande compacité.

Etude de la compacité dans différents cas :

Forme compacte	Pixels isolés	Fauuses alarmes	Lacunes
			
Cercle de N pixels V. <u>$C=1$</u>	N pixels isolés $C = 4\pi \frac{N\pi}{(N2\pi)^2}$	Cercle de rayon R Et N fausses alarmes $C_f = 4\pi \frac{\pi R^2 + \pi N}{(2\pi R + 2\pi N)^2}$ $C_f = \frac{R^2 + N}{(R + N)^2}$	Cercle de rayon R Et N lacunes $Cl = 4\pi \frac{\pi R^2 - N\pi}{(2\pi R + N2\pi)^2}$ $Cl = \frac{R^2 - N}{(R + N)^2}$

Autrement dit, la compacité aura tendance à favoriser les formes très compactes (cercle). Si ce n'est pas possible (niveaux de gris entremêlés à cause du bruit), elle privilégierra les fausses alarmes par rapport aux lacunes : $Cl < Cf < 1$.

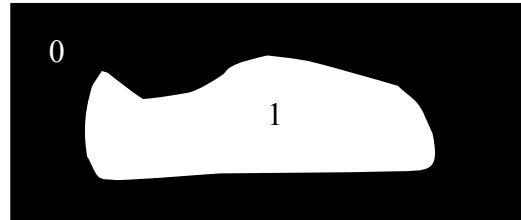
II. Introduction à la morphologie mathématique binaire

1. Introduction

C'est une théorie ensembliste.

Notations et définitions :

- Objets : $X = \{ x / x = 1 \}$
- Fonds : $\sim X = \{ x / x = 0 \}$

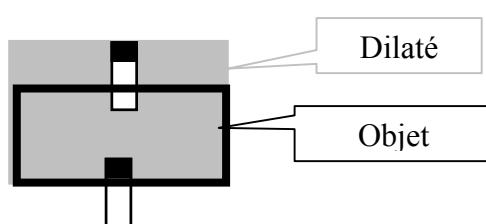


- Élément structurant :
Sx caractérisé par ses dimensions et sa forme
Son centre x est le point d'application de l'opération locale



2. Dilatation morphologique

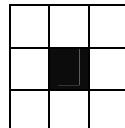
$$\begin{aligned} X \oplus Sx &= \{ x / Sx \cap X \neq \emptyset \} \\ &= \{ x / (Sx \cap X) \subset X \} \end{aligned}$$



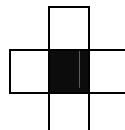
Il y a forcément une relation d'inclusion : $X \subset X \oplus S$

Interprétation avec un élément structurant particulier :

Considérons un élément structurant S_x centré en x , de rayon 1. Il peut prendre deux formes :
Si on travaille en 8 connexités,



Si on travaille en 4 connexités,



La dilatation peut alors être interprétée comme une croissance de 1 pixel :

$\forall x = 0$, s'il existe un point à un dans le voisinage, alors $x=1$.

On peut programmer très facilement la dilatation avec une convolution :

$$X \oplus S = \{ x / S_x \cap X \neq \emptyset \}$$

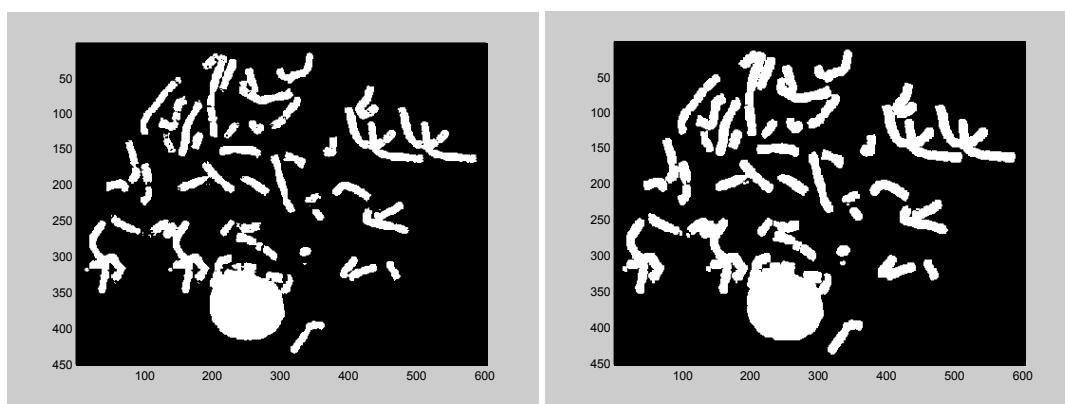
$$X \oplus S = (S * \text{Image}) \neq 0$$

Sous matlab, on utilisera les lignes de commande :

Dil=conv2(X,ones(3,3),'same')>0 si on travaille en huit connexités

Dil=conv2(X,[0 1 0 ;1 1 1 ;0 1 0]','same')>0 si on travaille en quatre connexités

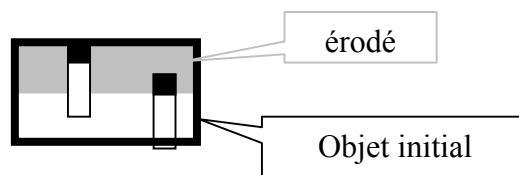
Un exemple :



On voit que la dilatation bouche les trous.

3. Erosion morphologique

$$\begin{aligned} X \ominus S &= \{ x / S_x \subset X \} \\ &= \{ x / (S_x \cap \sim X) \neq \emptyset \} \end{aligned}$$



Il y a forcément une relation d'inclusion : $X \ominus S \subset X$

Interprétation avec un élément structurant particulier :

L'érosion peut être interprétée comme une contraction de 1 pixel :

$\forall x = 1$, s'il existe un point à 0 dans le voisinage, alors $x=0$.

On peut programmer très facilement l'érosion avec une convolution :

$$X \Theta S = \{x / Sx \subset X\}$$

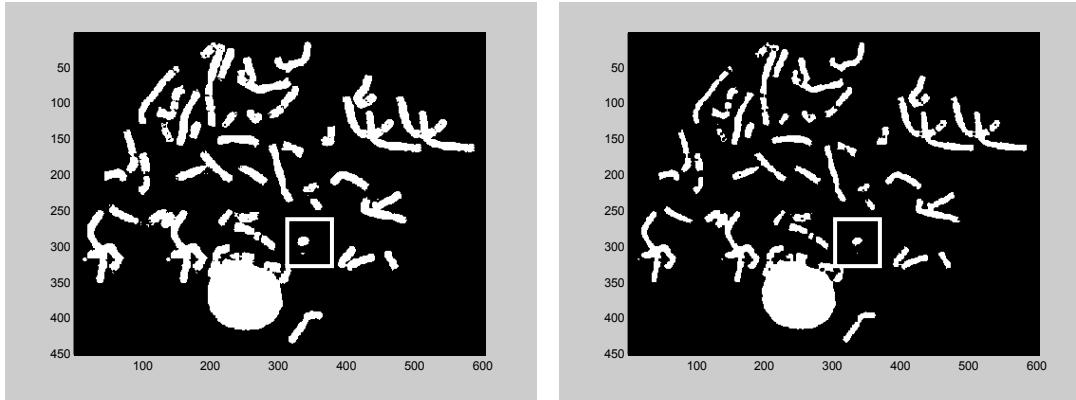
$$X \Theta S = (S * \text{Image}) = \text{card}(S)$$

Sous matlab, on utilisera les lignes de commande :

Dil=conv2(X,ones(3,3),'same')= 9 si on travaille en huit connexités

Dil=conv2(X,[0 1 0 ;1 1 1 ;0 1 0]'same')= 5 si on travaille en quatre connexités

Un exemple :



On voit que l'érosion supprime les fausses alarmes.

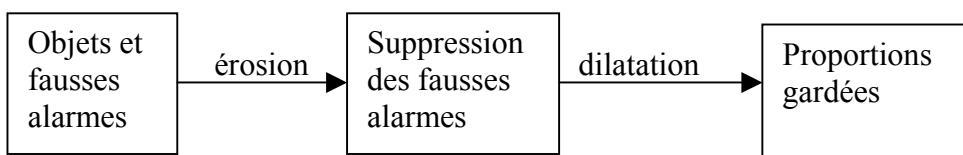
4. Cascade d'opérations

A partir de l'érosion et de la dilatation, on peut définir deux nouvelles opérations : l'ouverture et la fermeture.

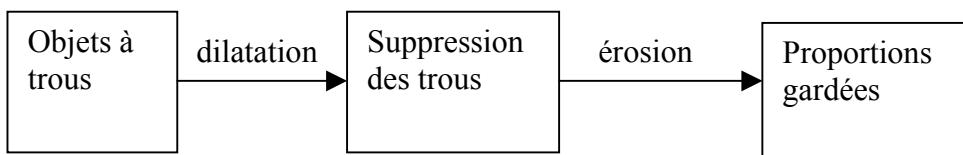
Ouverture : $X \circ S = (X \ominus S) \oplus S$

Fermeture : $X \bullet S = (X \oplus S) \ominus S$

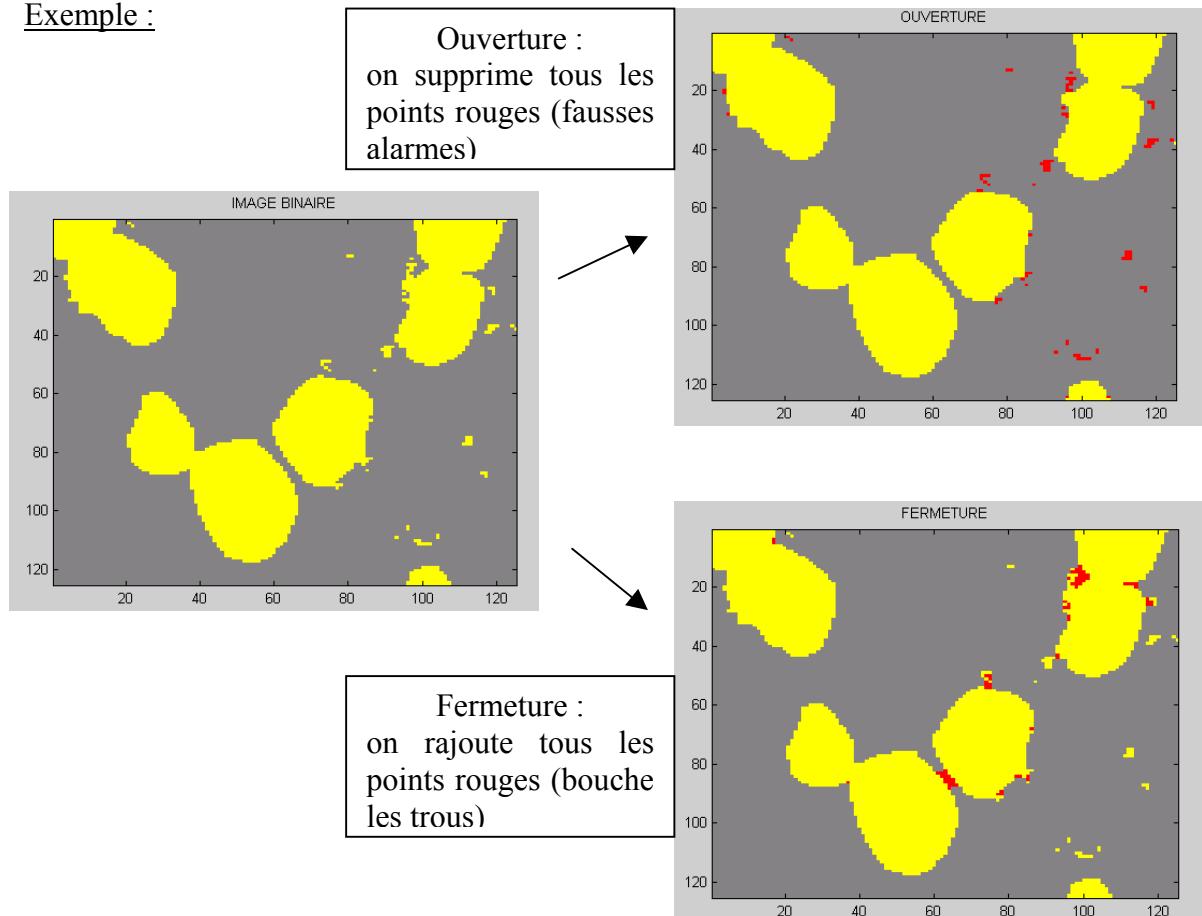
L'ouverture servira à supprimer les fausses alarmes :



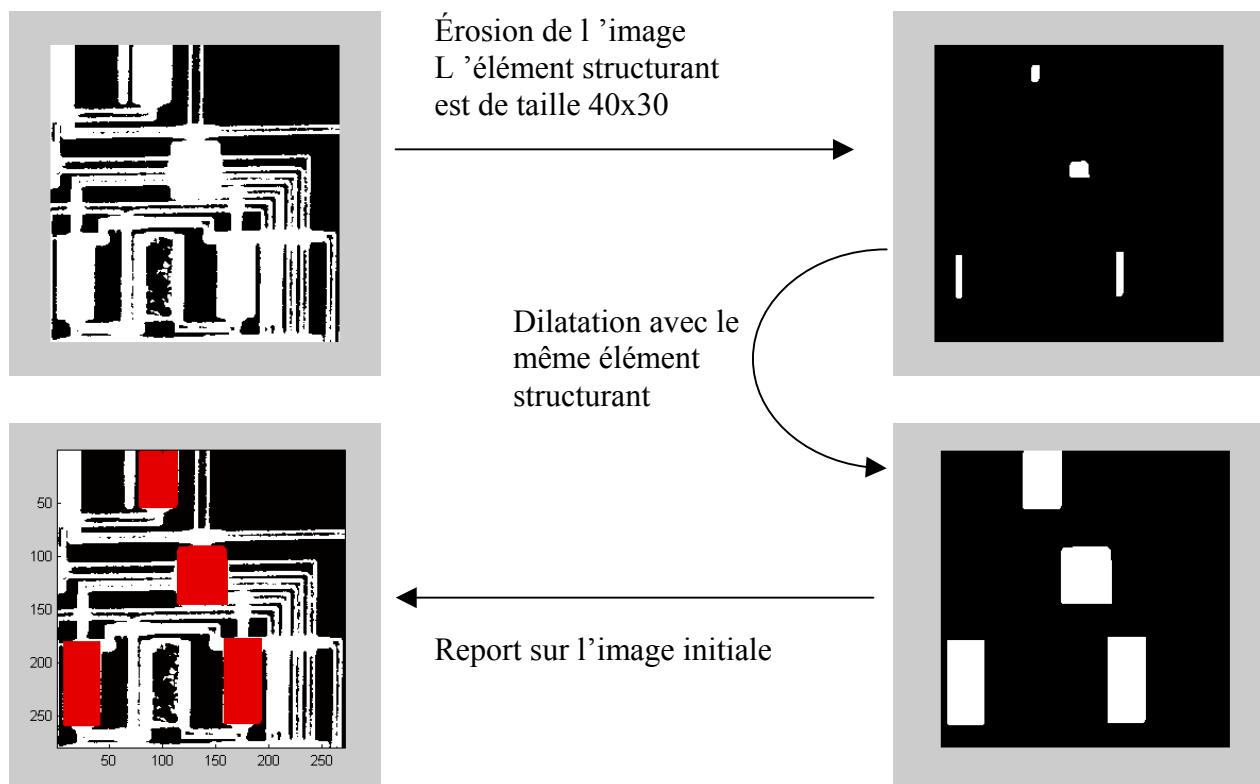
La fermeture supprimera les trous :



Exemple :

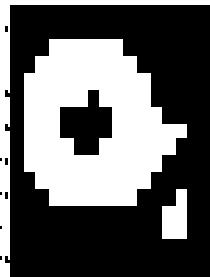


Application : détection de composants

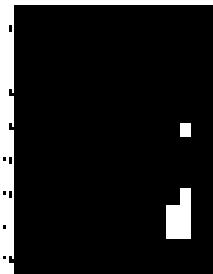


Autre application : Sélection d'objets de petite taille

$$X - (X \circ S)$$



$$S = V4$$



5. Contours d'une image binaire

Les contours intérieurs s'obtiennent grâce à :

$$X - (X \ominus S)$$

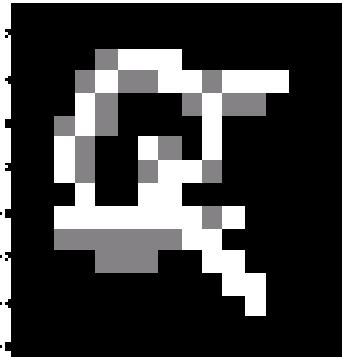
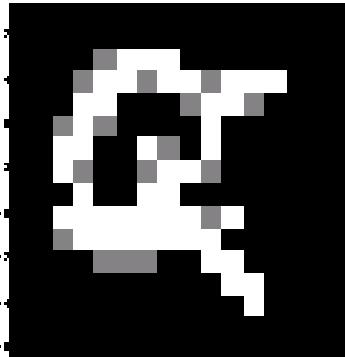
En effet, l'érosion supprime un pixel aux bords de l'objet. En faisant Image-érodé, on retrouve donc un contour de 1 pixel de large. On peut aussi déterminer les contours extérieurs avec :

$$(X \oplus S) - X$$

6. Amincissement

L'amincissement est en fait une érosion avec conservation de la connexité. Il peut servir à améliorer les contours d'objets fins mais sa grande utilisation sera la squelettisation.

Application : amélioration de contours d'objets minces (caractères,...)



Gris : images source

Blanc : 1 puis 2 amincissements

Principe : pour un point de bordure nord (avec un zéro au dessus), l'élimination du pixel central, dans les deux configurations données ci-dessous, ne modifie pas la connexité.

V8	<table border="1"> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>x</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	0	0	0	x	0	0	1	1	1	<table border="1"> <tr><td>x</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> <tr><td>x</td><td>1</td><td>x</td></tr> </table>	x	0	0	1	1	0	x	1	x
0	0	0																		
x	0	0																		
1	1	1																		
x	0	0																		
1	1	0																		
x	1	x																		

x : état indifférent (0 ou 1)

Donc, pour tous les pixels à 1, de bordure nord, vérifiant une des configurations précédentes, on pourra mettre le pixel central à 0.

Les autres configurations (est, sud, ouest) se déduisent de la configuration nord par rotation de 90°.

7. Squelettisation

Le squelette est un graphe qui caractérise la forme d'un objet binaire et qui peut permettre de le reconstituer (transformation réversible). Il sera très utile pour faire de la reconnaissance d'objets. Il existe deux définitions du squelette qui sont similaires dans \mathbb{R}^2 :

Lieu des centres des disques de rayon maximum inscrits dans l'objet. Le rayon en chaque point permet de reconstruire l'objet.	Axe médian, ensemble des points équidistants de 2 bords de l'objet. Cette définition correspond à l'état stable obtenu par amincissements itératifs

Le squelette est très sensible aux détails mais aussi, aux lacunes et à l'irrégularité des bords.

Exemple : gris : image source, blanc : squelette

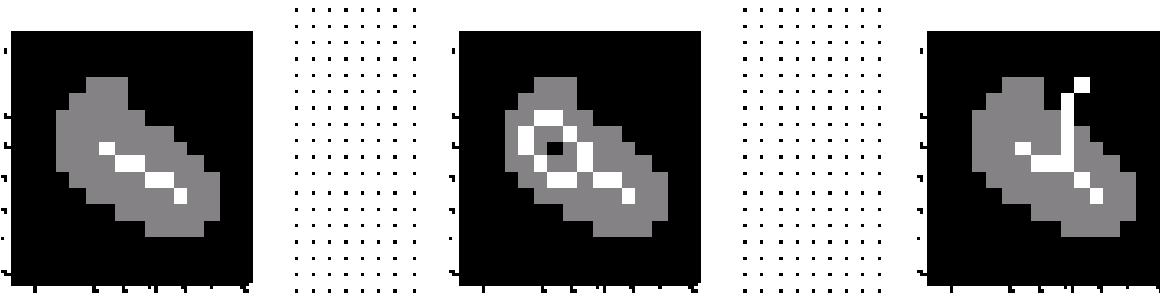


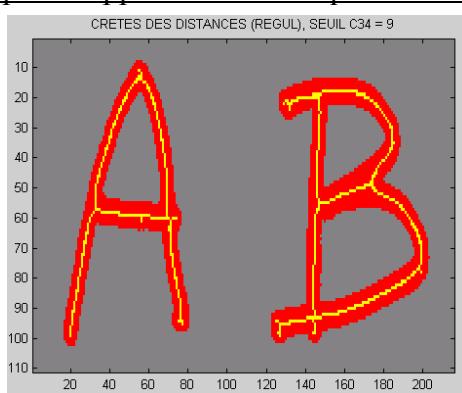
Image de départ

Même image avec un trou

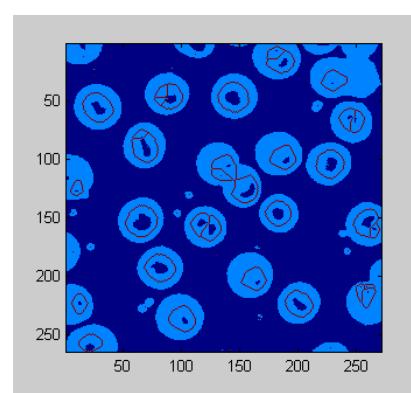
Même image avec un bord irrégulier

Ainsi, avant de faire la squelettisation, on sera souvent amené à réaliser un pré-traitement de l'image.

Exemples d'application de la squelettisation :



Reconnaissance de l'écrit



Reconnaissance des composantes sans trou (réduite à 1 pixel) et des composantes avec trou

a. Pré-traitement d'une image binaire

Pour éliminer les trous, ou lacunes, on mettra à 1 le pixel central dans les configurations suivantes :

x	x	x
1	0	1
x	1	x

x	1	x
1	0	1
x	x	x

x	1	x
x	0	1
x	1	x

x	1	x
1	0	x
x	1	x

(Les trois derniers masques se déduisent du premier par rotation de 90°).

Le lissage des points saillants se fera en mettant à 0 le point central dans les configurations suivantes (bordure nord) :

0	0	0
0	1	0
x	x	0

0	0	0
0	1	0
x	1	x

0	0	0
0	1	0
0	x	x

b. Algorithme parallèle de squelettisation

Cet algorithme est qualifié de parallèle car le traitement est réalisé avec une image source et une image d'arrivée. L'algorithme consiste à réaliser des itérations d'amincissement jusqu'à stabilisation (plus aucun changement).

Lors des itérations d'amincissement, il faudra prendre garde aux bandes de largeur 2 pixels (0-1-1-0) qui seront supprimées. Une solution consiste à les détecter et à ne pas modifier le pixel central dans ces configurations.

c. Algorithme séquentiel de squelettisation

Les modifications se font directement dans l'image source, ce qui introduit une causalité lors du balayage. Ici aussi, l'algorithme est itératif et se poursuit jusqu'à stabilisation (le nombre d'itérations dépend de l'épaisseur de l'objet).

Voici, pour un point de bordure est, les seules configurations où il faut conserver le pixel central :

x	x	x
0	1	x
1	0	x

1	0	x
0	1	x
x	x	x

x	0	0
0	1	1
x	0	0

x	x	x
0	1	0
z	z	z

si tous les x et z à 0	point isolé à conserver
si tous x à 0 et au moins un z à 1	point d'extrémité
si au moins un x à 1	l'élimination du point central rompt la connexité locale
si x et z ont au moins un 1	rupture de connexité

Les points non essentiels (qui ne correspondent pas à une de ces configurations) sont marqués et sont éliminés en fin d'itération.

Déroulement des itérations :

- ✓ Initialisation : iter=1,
- ✓ Itérations tant que des points sont modifiés, 3 sous – cycles
 - test des points bordure E et W, essentiels = iter,
non essentiels = Mrk
 - test des points non modifiés N et S, essentiels = iter,
non essentiels = Mrk
 - points d'étiquette Mrk éliminés (0), iter = iter+1
- ✓ en fin d'itération, squelette = point > 0

- ✓ La valeur d'un point squelette indique l'itération à laquelle il fut déclaré essentiel, donc, l'épaisseur locale de l'objet (possibilité de reconstruction)

d. Squelettisation par recherche des maxima locaux des distances entre les pixels de l'objet et le fond

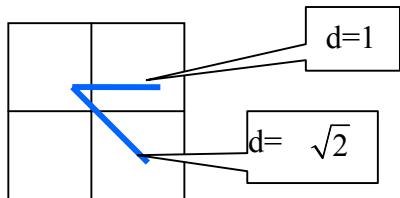
Cette méthode consiste à calculer, en chaque point de l'objet, sa distance au fond. On recherche ensuite les maxima locaux directionnels (ligne de crête) de l'image des distances.

0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0
0	0	0	1	2	1	0	0	0
0	1	1	1	2	1	1	0	0
0	1	2	2	2	2	1	0	0
0	1	2	3	3	2	1	0	0
0	1	2	2	2	2	1	0	0
0	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0

Cette méthode est très rapide car le calcul des distances ne nécessite que 2 parcours de l'image. Le temps de calcul du squelette est indépendant de la dimension de l'objet, contrairement aux autres algorithmes. Par contre, avec cet algorithme, le squelette n'est plus forcément connexe (On peut avoir deux squelettes pour un objet, voir l'exemple ci-contre). Ceci n'était pas le cas avec les deux autres méthodes.

Algorithme de Chamfer pour calculer en deux parcours de l'image, la distance des points de l'objet au fond.

On procède de proche en proche. Lorsque l'on se déplace horizontalement ou verticalement, on ajoute 1 et lorsqu'on se déplace en biais, on ajoute $\sqrt{2}$



Pour simplifier, on se ramène à des valeurs entières :

$$\begin{cases} 1 \rightarrow 2 \\ \sqrt{2} \rightarrow 3 \end{cases} \Rightarrow 2 \text{ fois la distance euclidienne} \Rightarrow \text{erreur de 13\% par défaut}$$

$$\begin{cases} 1 \rightarrow 3 \\ \sqrt{2} \rightarrow 4 \end{cases} \Rightarrow 3 \text{ fois la distance euclidienne} \Rightarrow \text{erreur de 8\% par excès}$$

L'algorithme procède en deux balayages : un en sens vidéo puis un en inverse vidéo. Il cumule le chemin de coût minimum de proche en proche :

Balayage en sens vidéo :

Cd0	Cv1	Cd2
Ch3	p	-->

$$Dp = \min_{i=0 \dots 3} (D(i) + C(i))$$

avec Cv=Ch=2 ou 3 respectivement
Cd=3 ou 4 respectivement

Balayage en sens inverse vidéo :

<--	p	Ch3
Cd2	Cv1	Cd0

$$Dp = \min_{i=0 \dots 3} (Dp, D(i) + C(i))$$

avec Cv=Ch=2 ou 3 respectivement
Cd=3 ou 4 respectivement

Exemple :

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	0	0	0	0
0	0	0	1	1	1	1	0	0	0	0
0	0	0	1	1	1	1	0	0	0	0
0	0	0	1	1	1	1	0	0	0	0
0	0	0	1	1	1	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

Image binaire

∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	1	1	1	1	2	4	6	
∞	∞	3	1	1	1	1	2	4	6	
∞	6	3	1	1	1	1	2	4	6	
9	6	3	1	1	1	1	2	4	6	
9	6	3	2	2	2	2	3	5	7	
9	6	5	4	4	4	4	5	6	8	
9	8	7	6	6	6	6	7	8	9	

Balayage sens vidéo

9	8	7	6	6	6	6	7	8	9
8	6	5	4	4	4	4	5	6	8
7	5	3	2	2	2	2	3	5	7
6	4	2	1	1	1	1	2	4	6
6	4	2	1	1	1	1	2	4	6
6	4	2	1	1	1	1	2	4	6
6	4	2	1	1	1	1	2	4	6
7	5	3	2	2	2	2	3	5	7
8	6	5	4	4	4	4	5	6	8
9	8	7	6	6	6	6	7	8	9

Balayage sens inverse vidéo

III. Introduction à la morphologie multi-niveaux

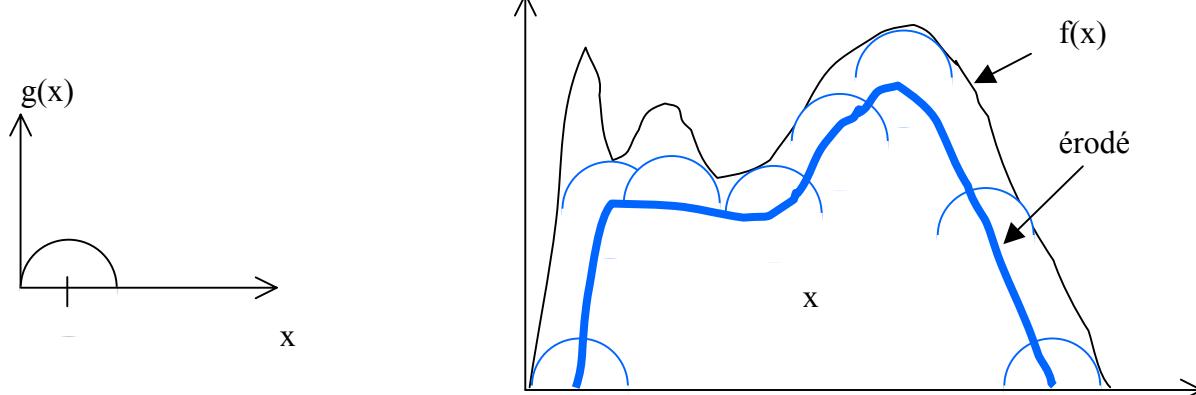
La morphologie mathématique a aussi été définie pour des images codées en niveaux de gris. Elle procède de la même façon que la morphologie binaire : un élément structurant est déplacé en chaque pixel de l'image. L'image est considérée comme un objet volumique, l'élément structurant est lui-même volumique. Par simplicité, nous allons définir les opérations morphologiques dans le cas 1D, ce qui revient à considérer une ligne de pixels.

Soit $f(x)$ la fonction représentant cette ligne de pixels. Elle donne, pour chaque position x , la valeur du niveau de gris.

1. Erosion

Soit $g(x)$ un élément structurant. L'érodé de f par g est défini par :

$$(f \Theta g)(x) = \min_y \{f(x) - g(x+y)\}$$



Cette transformation a pour propriétés de réduire les "pics" de niveaux de gris (valeurs claires) et d'élargir les "vallées", donc à assombrir l'image en augmentant la taille des objets les sombres. Un exemple d'érosion est présenté ci-dessous :

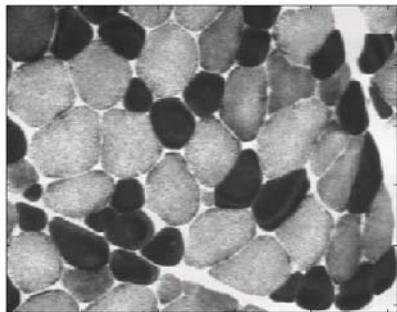


Image originale

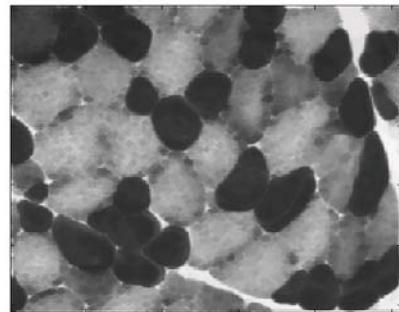
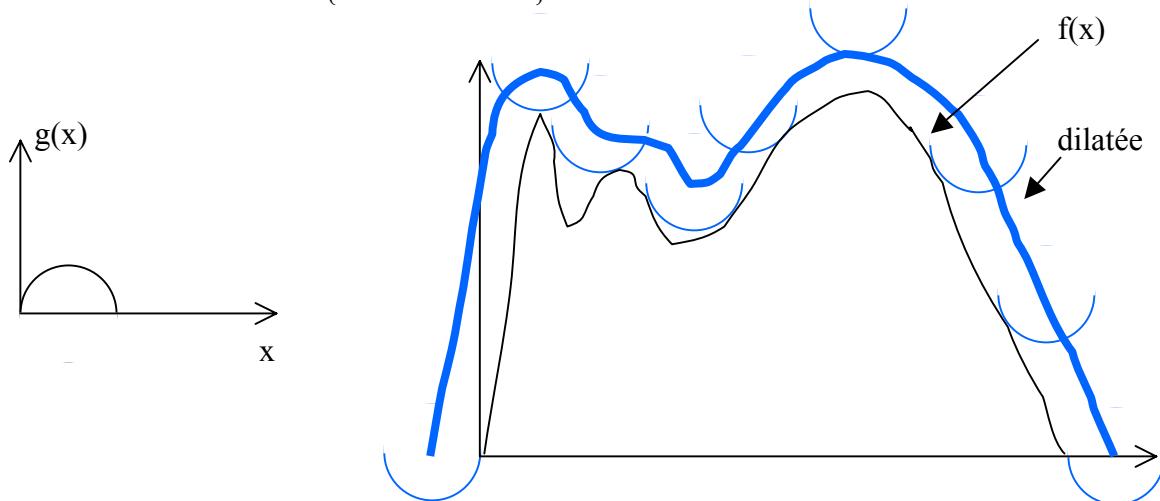


Image érodée

2. Dilatation

Soit $g(x)$ un élément structurant. Le dilaté de f par g est défini par :

$$(f \oplus g)(x) = \max \{ f(x) + g(x+y) \}$$



Cette transformation a les propriétés inverses de l'érosion : elle augmente les "pics" de niveaux de gris (valeurs claires) et diminue les "vallées", elle éclaircie donc l'image en augmentant la taille des objets clairs. Un exemple de dilatation est présenté ci-dessous :

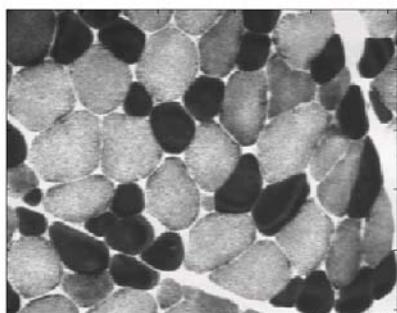


Image originale

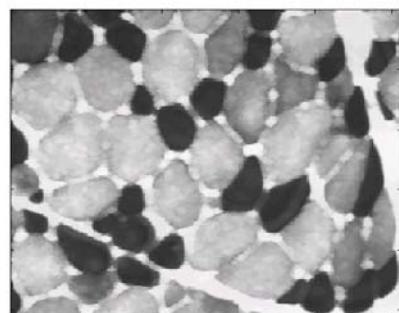


Image dilatée

3. Gradient

Le gradient est défini par :

$$\text{gradient} = \text{dilaté} - \text{érodé} - 2 * \max_x(g(x))$$

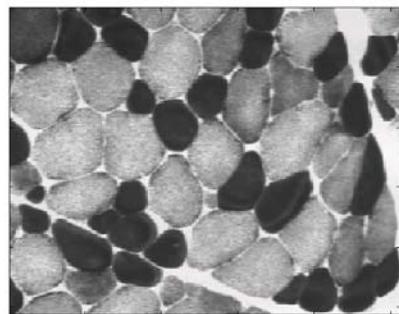
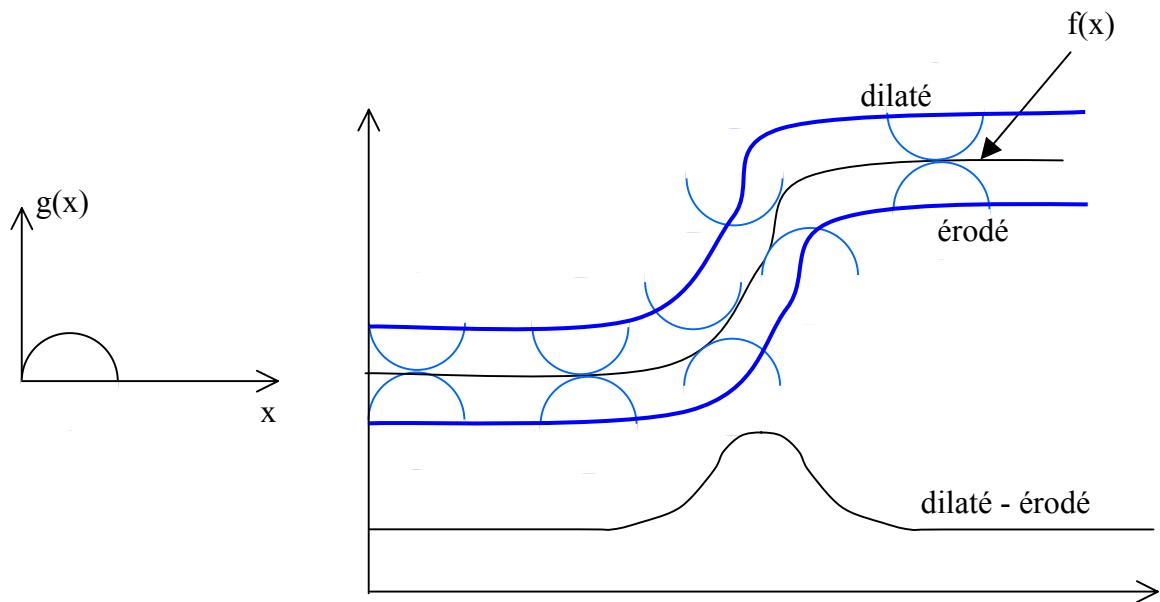


Image originale

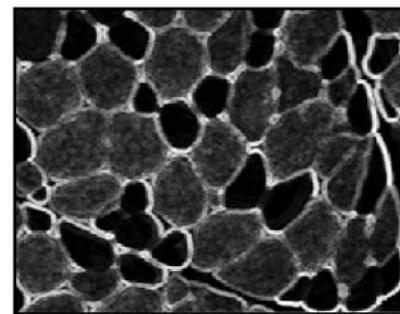


Image dilatée

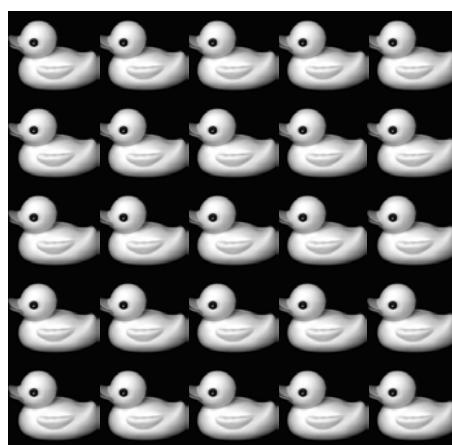
Chapitre 8 : Texture

I. Introduction

1. Introduction

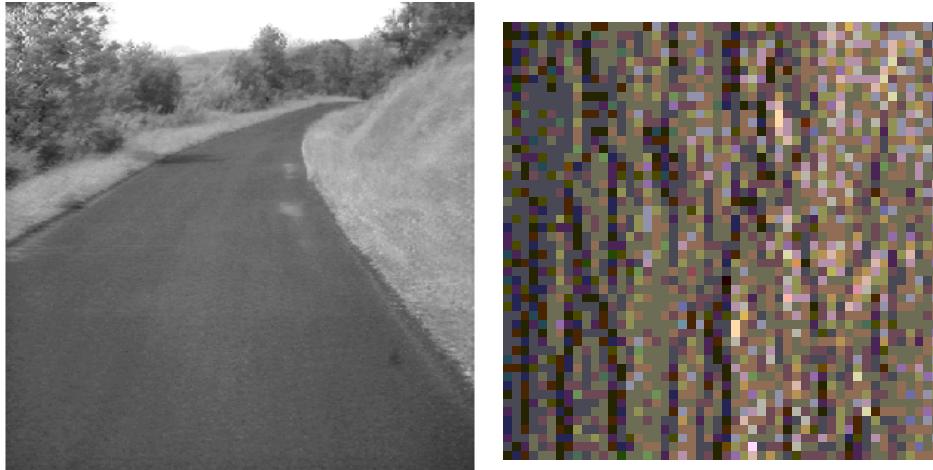
Dans le but de segmenter une image en régions, nous ne pouvons pas nous limiter à l'utilisation seule des niveaux de gris. En effet, beaucoup d'images, et plus particulièrement les images naturelles, possèdent des zones texturées, non homogènes au sens des niveaux de gris. Nous voyons donc la nécessité, d'introduire des paramètres de texture dans les algorithmes de segmentation, dans le souci de travailler sur des images les plus générales possibles (ne pas se limiter aux images contenant des objets homogènes au sens des niveaux de gris). Malheureusement, il n'existe pas de définition universelle pour la texture et bien que celle-ci soit visible de façon naturelle pour l'œil humain, aucune règle mathématique générale capable de quantifier les différentes classes de texture n'a été mise en évidence à ce jour. De nombreux travaux ont cependant été réalisés sur ce sujet, mettant en avant deux grandes approches d'analyse de texture : l'approche structurale et l'approche statistique.

- L'approche structurale considère que les textures sont formées d'un ou plusieurs motifs se répétant de façon régulière dans l'espace. A titre d'exemples, nous pouvons considérer un mur de briques ou un tissu. Une description structurale de la texture consistera alors à repérer dans l'image les éléments de base ou motifs, ainsi que leurs arrangements. Cette méthode exige que l'image soit formée d'une texture régulière et se heurte à de nombreux problèmes. Tout d'abord, il est généralement difficile de repérer les motifs élémentaires, d'autant plus que les propriétés de ces motifs ne sont pas nécessairement constantes dans l'image (variation de l'éclairage,...). Un autre problème est dû à la répartition des primitives qui n'est pas forcément régulière (un tissu avec des trames plus serrées par endroits), ce qui entraîne l'introduction d'un élément aléatoire. Une certaine souplesse peut cependant être apportée aux algorithmes, permettant de les rendre moins sensibles au bruit. Jusqu'à présent, l'approche structurale ne semble pas être en mesure de conduire à une représentation unifiée de la texture, son utilisation se limite à des applications très précises.



- L'approche statistique s'applique en particulier aux textures ne possédant pas de primitives élémentaires autres que le pixel ou des voisinages de pixels. Ces textures présentent un aspect désordonné, mais sont néanmoins homogènes. Cette approche statistique est bien adaptée aux textures naturelles. Elle est d'autre part plus facile à mettre en oeuvre dans un algorithme de segmentation d'images que l'approche

structurale. Les méthodes statistiques sont donc les plus utilisées, nous en détaillons quelques unes, ainsi que les paramètres de texture auxquels elles donnent accès, dans le but de la segmentation d'images.



De nombreux travaux utilisant des caractéristiques de texture existent. Citons, pour illustration, les applications :

- segmentation d'images aériennes (détection des arbres, des bâtiments,...) ;
- segmentation d'images volcaniques ;
- recherche de fruits dans un arbre ;
- segmentation de scènes routières pour asservir un véhicule sur la route ;
- détection de défauts sur des tôles découpées ;

II. Histogramme des différences de niveaux de gris

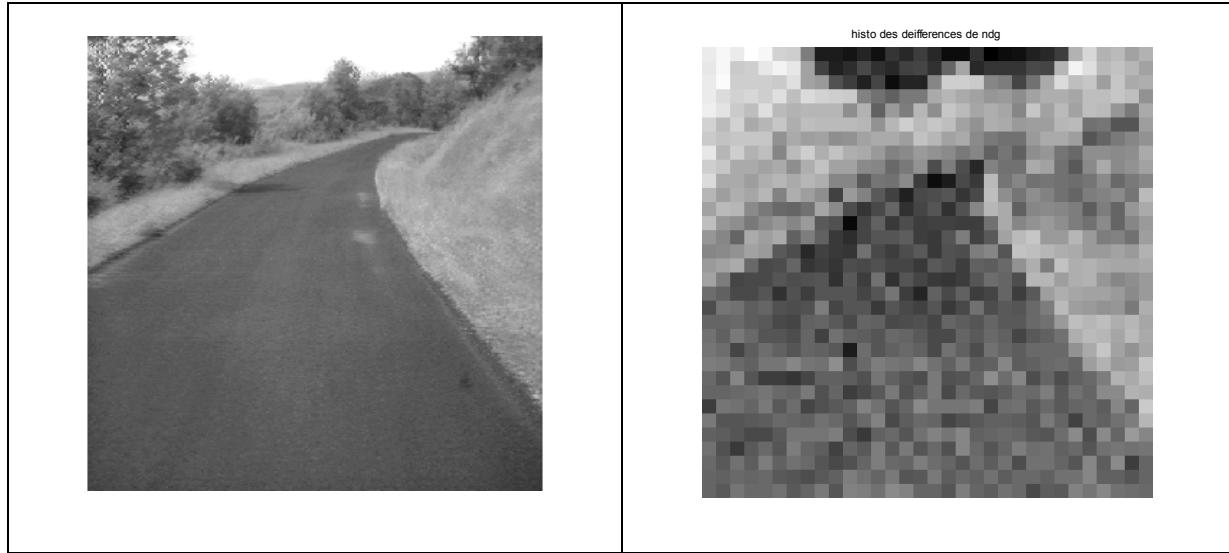
Pour chaque point (x,y) de la fenêtre étudiée, nous calculons :

$$F(x, y) = |f(x, y) - f(x + \Delta x, y + \Delta y)|$$

où f est la fonction luminance.

Si l'image possède N niveaux de gris, alors F varie entre 0 et $N-1$. Le décompte du nombre de fois, où $F(x,y)$ prend la valeur i , fournit un vecteur à N dimensions duquel des paramètres tels que la valeur maximale, la moyenne, la variance, les moments,... sont extraits. Ceux-ci caractérisent la texture. Un gros inconvénient de cette méthode est qu'elle nécessite le calcul de plusieurs histogrammes (plusieurs directions et plusieurs distances) pour chaque fenêtre.

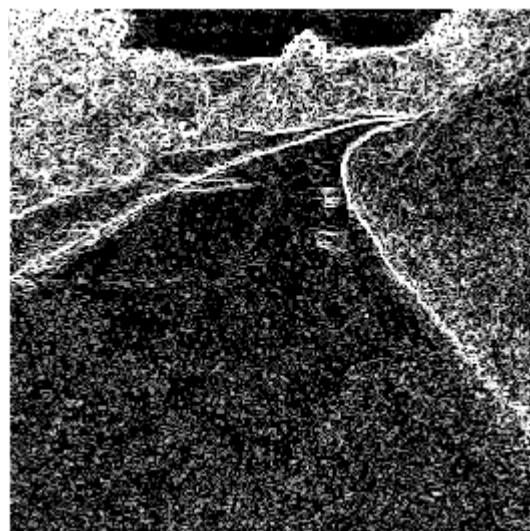
A titre d'illustration, un exemple de recherche de paramètres de texture sur une image de route sans marquage au sol est présenté. Afin de diminuer les temps de calcul, la gamme des niveaux de gris a été réduite, de manière linéaire à 64. De plus, comme il est impossible de calculer la texture d'un point, l'image a été divisée en fenêtres de 16x16 pixels. Le paramètre de texture estimé pour cette zone lui est affecté et est représenté sur l'image résultat. On aurait aussi pu affecter ce paramètre au pixel central et ainsi, disposer d'un paramètre de texture pour chaque pixel de l'image. Bien évidemment, les temps de calcul seraient alors beaucoup plus longs. Nous avons calculé un seul histogramme avec $\Delta x = 1$ et $\Delta y = 0$. De plus, le paramètre de texture représenté est la valeur maximale de l'histogramme.



On voit que les paramètres de texture associés à la route ou au ciel sont plus faibles que ceux associés aux bas-cotés ou aux arbres, ce qui traduit bien les zones que visuellement, on juge moins texturées.

III. Méthode fondée sur le calcul du gradient de la fonction luminance

De nombreuses méthodes d'extraction de paramètres de texture reposent sur le calcul du gradient de la fonction luminance. En effet, les contours extraits d'une image semblent caractériser la texture de celle-ci :



Plusieurs méthodes permettent d'obtenir des paramètres de texture à partir des contours, nous en présentons une ici.

Dans un premier temps une extraction, puis un chaînage des points contours sont réalisés. A partir des points ainsi retenus, on caractérise la texture par :

- nombre de petits segments obtenus,
- longueur moyenne des segments,
- ...

IV. Filtre de Gabor

Un filtre de Gabor h , à deux dimensions, peut être représenté comme une gaussienne modulée par une onde plane sinusoïdale :

$$h(x,y) = \exp\left(-\frac{1}{2}\left[\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right]\right) \cos(2\pi u_0 x + \Phi)$$

Où :

- u_0 et Φ sont respectivement la fréquence et la phase de l'onde plane sinusoïdale.
- σ_x σ_y caractérisent l'étendue spatiale du filtre.

En faisant varier les différents paramètres, plusieurs filtres de réponse impulsionnelle h_j sont obtenus. Une image résultatat f_j est déterminée à la sortie de chacun de ces filtres :

$$f_j(x,y) = f(x,y) * h_j(x,y)$$

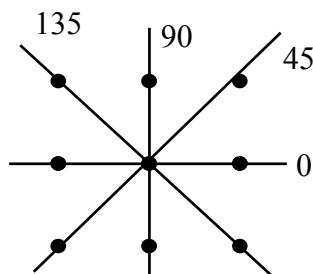
Plusieurs paramètres peuvent ensuite être extraits de ces images comme, par exemple, la moyenne absolue de la déviation :

$$m_j = \frac{1}{n^2} \sum_{x,y=0}^{n-1} |f_j(x,y)|$$

V - Les matrices de cooccurrences, ou de transitions

Une méthode fréquemment utilisée en pratique pour la discrimination des textures repose sur l'emploi des matrices de cooccurrences. Celles-ci sont déterminées à partir de caractéristiques représentant les relations spatiales entre paires de niveaux de gris de pixels.

Plusieurs définitions des matrices de cooccurrences existent dans la littérature, nous considérons ici la définition des matrices de cooccurrences symétriques. Elles déterminent la fréquence d'apparition de paires de niveaux de gris à une distance δ donnée, dans une direction θ donnée :



La définition des matrices de cooccurrences est la suivante. Soit $f(x,y)$ la fonction luminance. La matrice de cooccurrences $P_{\delta,\theta}$ pour une distance δ donnée et une orientation θ fixée est définie par :

$$P_{\delta,\theta}(i,j) = \sum_x \sum_y \mathbf{1}_{[f(x,y)=i]} \mathbf{1}_{[f(x',y')=j]}$$

Les directions les plus usitées sont 0, 45, 90 et 135 degrés. La somme des matrices de cooccurrences dans les différentes directions peut être effectuée :

$$P_\delta = P_{\delta,0} + P_{\delta,45} + P_{\delta,90} + P_{\delta,135}$$

Afin d'illustrer le calcul des matrices de cooccurrences, nous présentons un exemple, et calculons celle-ci pour un secteur codé sur sept niveaux de gris. La matrice de cooccurrences de ce secteur est déterminée pour une distance de un pixel ($\delta = 1$) et une orientation de zéro degré ($\theta = 0^\circ$).

secteur	matrice																																																																								
<table border="1"> <tr><td>0</td><td>6</td><td>0</td><td>6</td></tr> <tr><td>6</td><td>0</td><td>6</td><td>0</td></tr> <tr><td>0</td><td>6</td><td>0</td><td>6</td></tr> <tr><td>6</td><td>0</td><td>6</td><td>0</td></tr> </table>	0	6	0	6	6	0	6	0	0	6	0	6	6	0	6	0	<table border="1"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>12</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>12</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </table>	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0
0	6	0	6																																																																						
6	0	6	0																																																																						
0	6	0	6																																																																						
6	0	6	0																																																																						
0	0	0	0	0	0	0	12																																																																		
0	0	0	0	0	0	0	0																																																																		
0	0	0	0	0	0	0	0																																																																		
0	0	0	0	0	0	0	0																																																																		
0	0	0	0	0	0	0	0																																																																		
0	0	0	0	0	0	0	0																																																																		
12	0	0	0	0	0	0	0																																																																		
<table border="1"> <tr><td>0</td><td>0</td><td>5</td><td>5</td></tr> <tr><td>0</td><td>0</td><td>5</td><td>5</td></tr> <tr><td>0</td><td>0</td><td>5</td><td>5</td></tr> <tr><td>0</td><td>0</td><td>5</td><td>5</td></tr> </table>	0	0	5	5	0	0	5	5	0	0	5	5	0	0	5	5	<table border="1"> <tr><td>8</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>4</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>4</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>8</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </table>	8	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0
0	0	5	5																																																																						
0	0	5	5																																																																						
0	0	5	5																																																																						
0	0	5	5																																																																						
8	0	0	0	0	0	4	0																																																																		
0	0	0	0	0	0	0	0																																																																		
0	0	0	0	0	0	0	0																																																																		
0	0	0	0	0	0	0	0																																																																		
0	0	0	0	0	0	0	0																																																																		
4	0	0	0	0	0	8	0																																																																		
0	0	0	0	0	0	0	0																																																																		
<table border="1"> <tr><td>0</td><td>2</td><td>5</td><td>2</td></tr> <tr><td>4</td><td>2</td><td>3</td><td>6</td></tr> <tr><td>3</td><td>0</td><td>1</td><td>4</td></tr> <tr><td>6</td><td>4</td><td>2</td><td>1</td></tr> </table>	0	2	5	2	4	2	3	6	3	0	1	4	6	4	2	1	<table border="1"> <tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>1</td><td>2</td><td>2</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>2</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>0</td><td>2</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td></tr> </table>	0	1	1	1	0	0	0	0	1	0	1	0	1	0	0	0	1	1	0	1	2	2	0	0	1	0	1	0	0	0	1	0	0	1	2	0	0	0	0	1	0	0	2	0	0	0	0	0	0	0	0	1	1	0	0	0
0	2	5	2																																																																						
4	2	3	6																																																																						
3	0	1	4																																																																						
6	4	2	1																																																																						
0	1	1	1	0	0	0	0																																																																		
1	0	1	0	1	0	0	0																																																																		
1	1	0	1	2	2	0	0																																																																		
1	0	1	0	0	0	1	0																																																																		
0	1	2	0	0	0	0	1																																																																		
0	0	2	0	0	0	0	0																																																																		
0	0	0	1	1	0	0	0																																																																		

Les matrices de cooccurrences contiennent une grande quantité d'informations, difficilement exploitable dans son intégralité (pour une image à 256 niveaux de gris, la matrice a comme dimension 256x256). Il faudra donc réduire cette quantité d'informations pour garder uniquement des données pertinentes, relatives à la texture de la scène.

Haralick a défini quatorze paramètres représentatifs de la texture à partir des matrices de cooccurrences. Nous en citons deux :

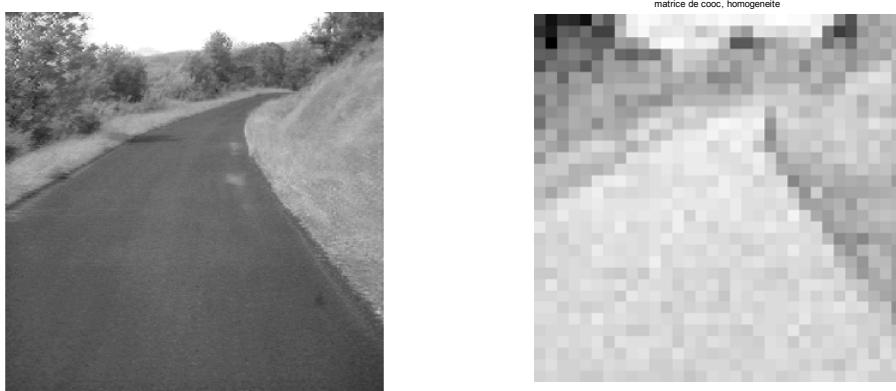
- Moment second ou énergie

$$\sum_i \sum_j [P(i,j)]^2$$

- Moment des différences inverses ou homogénéité

$$\sum_i \sum_j \frac{1}{1+(i-j)^2} P(i,j)$$

De manière générale, ces paramètres sont discriminants et facilement extractibles. Nous reprenons le même exemple que précédemment qui est celui de la route sans marquage au sol. L'image est toujours divisée en fenêtres de 16x16 pixels, et nous travaillons sur 64 niveaux de gris. Ceci a pour conséquence directe de réduire considérablement la taille des matrices de cooccurrence puisqu'elles passent de 256x256 à 64x64, soit une réduction de taille d'un facteur 16. Le paramètre que nous représentons est le moment des différences inverses. On peut considérer qu'il représente l'homogénéité des niveaux de gris de la zone étudiée.



VI - Les matrices de voisinage

La matrice de voisinage V à une distance d donnée est définie par :

$$V_d(i,j) = \text{nbre de pixels ayant un niveau de gris } i \text{ et } j \text{ voisins de niveaux de gris } i \pm d.$$

Une fois la matrice obtenue, les mêmes paramètres que ceux définis par Haralick sur les matrices de cooccurrences peuvent être extraits. Cette méthode donne des matrices de taille beaucoup plus faible que les matrices de cooccurrences et de plus, présente une invariance en rotation de par la nature même du voisinage centré. Par contre, un inconvénient majeur est que les caractéristiques extraites de la matrice varient énormément en fonction de d. Ceci dénote une certaine instabilité, et oblige à calculer plusieurs matrices, pour des distances algébriques différentes entre les niveaux de gris.

VII - Les matrices de longueur de plage

Les matrices de longueur de plage, consistent à rechercher les plages (ou segments de droite) de même niveau de gris. Ainsi, pour différentes orientations θ (principalement 0, 45, 90 et 135 degrés), une matrice de longueur de plage P_θ est déterminée. Celle-ci est caractérisée par ses éléments :

$$P_{\theta}(i,j) = \text{nombre de fois où l'image contient une plage de niveaux de gris } i, \text{ de longueur } j \text{ dans la direction } \theta.$$

Les matrices sont de dimension *nombre de niveaux de gris x longueur maximale d'une plage*. Une fois la matrice déterminée, les mêmes paramètres que ceux définis par Haralick à partir des matrices de cooccurrences peuvent être extraits. De nouveaux paramètres peuvent

également être définis comme par exemple << l'accentuation des petites longueurs >> qui varie en fonction du nombre de segments uniformes.

VIII - Fonction d'autocorrélation

La fonction d'autocorrélation est définie dans le cas discret par :

$$F(x,y) = \frac{1}{n} \sum_{x,y} f(x,y)f(x+\Delta x, y+\Delta y)$$

où :

- n représente les tailles horizontale et verticale de l'image ;
- f est la fonction de luminance ;
- Δx et Δy sont les déplacements horizontaux et verticaux.

La fonction d'autocorrélation permet de mettre en avant les dépendances spatiales existant entre certaines paires de pixels. Cette fonction mesure une distance entre une image originale et sa translatée. Nous présentons ci-dessous les coefficients obtenus avec $\Delta x=1$ et $\Delta y=0$.



IX - Transformée de Fourier

La transformée de Fourier bidimensionnelle peut être utilisée pour caractériser la texture. Celle-ci est définie dans le cas discret par :

$$F(u,v) = \frac{1}{n} \sum_{l,m} f(l,m) \exp(-2i\pi(ul+vm))$$

où :

- n représente les tailles horizontale et verticale de l'image ;
- f est la fonction de luminance.

La transformée de Fourier opère un changement d'espace de représentation. On passe ainsi du domaine spatial au domaine fréquentiel. Certaines textures, représentées dans cet espace, vont produire des concentrations d'énergie dans le spectre de Fourier, traduisant ainsi une périodicité horizontale et verticale.

X - Approches multi-résolution

Une autre méthode pour caractériser des textures consiste à utiliser la dimension fractale. En effet, des études ont mis en évidence la correspondance entre la dimension fractale et l'évaluation humaine de la rugosité. De plus, si une surface naturelle est fractale, alors l'image

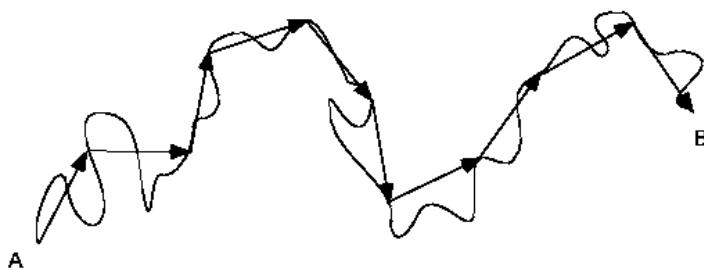
d'intensité lumineuse de cet objet l'est aussi. La dimension fractale d'un objet peut donc être calculée en utilisant les niveaux de gris qui lui correspondent. Trois grandes méthodes, permettant de calculer cette dimension fractale, existent :

- La méthode des boîtes;
- La méthode des variations;
- La méthode du spectre de puissance.

Nous ne parlerons ici que de la première méthode, mais avant tout, nous allons introduire ce qu'est une fractale.

La fin des années 1970 a vu se développer la notion de géométrie fractale dans de nombreux domaines de la physique, et en particulier, en traitement d'images. Elle est le complément qui manquait à la géométrie euclidienne. Comme l'a fait remarquer Benoit Mandelbrot, les nuages ne sont pas des sphères, les montagnes des cônes. Les objets fractals, encore appelés fractales, sont définis comme étant des objets dont la dimension de Hausdorff-Besicovich est fractionnaire et strictement supérieure à leur dimension topologique. Dans le cas d'une courbe, la dimension fractale d est toujours comprise entre un et deux. Si d est proche de un, la courbe est assimilable à une droite, tandis que si d est proche de deux, la courbe sera très << froissée >> dans le sens où elle occupera une large partie du plan et sera donc assimilable à une surface.

Afin de mieux comprendre la nécessité des dimensions non entières, nous allons mesurer la longueur L d'une côte maritime entre deux points A et B :



On est rapidement confronté à une difficulté : L dépend de la longueur du pas p choisi et croît infiniment quand p décroît. En effet, pour un pas p_1 , la longueur sera $Lp_1=N_1 p_1$ où N_1 est le nombre de pas nécessaire pour parcourir la côte. Pour un pas $p_2 < p_1$, on aura $Lp_2 = N_2 p_2 > Lp_1$. La mesure est donc relative à la résolution et, est différente de la << vraie >> longueur L_{AB} . Mais comment calculer cette vraie longueur L_{AB} ?

En 1910, Richardson établit la loi empirique reliant la longueur au pas de mesure :

$$L = p^{-a}$$

où a est un paramètre qui caractérise la côte ($-1 < a < 0$).

En 1977, Benoit Mandelbrot introduit la notion de dimension fractale d :

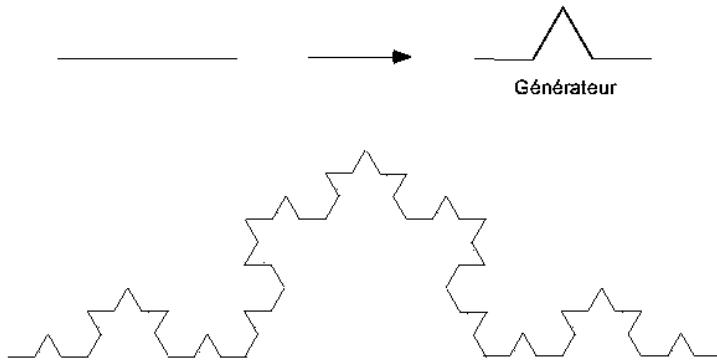
$$d=n-a$$

où n est la dimension topologique ($n=1$ pour une côte maritime).

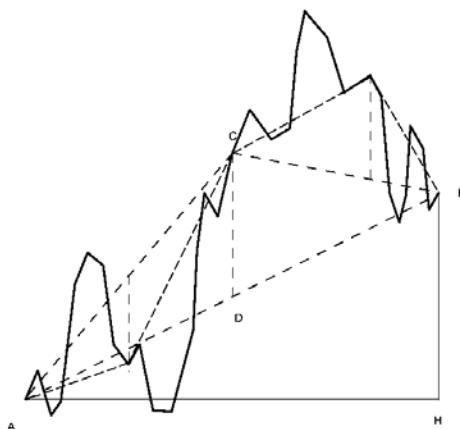
Ces deux équations conduisent au résultat :

$$L=p^{n-a}$$

La notion de géométrie fractale est étroitement liée aux propriétés d'invariance par changement d'échelle : une structure fractale est la même de près comme de loin. A titre d'exemple, nous présentons la courbe de Koch (1904). Chaque segment de longueur l est remplacé par une ligne brisée (générateur), formée de quatre segments de longueur $l/3$. On montre que la courbe ainsi obtenue a une dimension fractale $d = \ln 4 / \ln 3 = 1.2618$.



Les fractales sont maintenant très utilisées en imagerie de synthèse. Grâce à elles, on peut générer des formes naturelles comme des montagnes. Suivons cet exemple : on part d'un grand triangle rectangle, dont on prend le côté vertical comme unité. Au milieu de son hypoténuse AB, on porte verticalement la distance $1^{\sqrt{2}} (CD \cdot BH^{\sqrt{2}})$. On procède de même avec les nouveaux segments AC et BC, mais en déplaçant le milieu de AC vers le bas et celui de BC vers le haut. La courbe obtenue au bout de quatre étapes a une allure alpestre :



Etudions maintenant la mesure de la dimension fractale pour les images et, plus particulièrement, la méthode des boîtes. Considérons un objet fractal A dans l'espace de dimension n. A est inclus dans une boîte de cet espace de taille R. Si on réduit le partitionnement d'un rapport $1/r$, alors il faudra $N(r) = r^d$ boîtes de taille rR pour contenir l'objet A.

Dans le cas de la dimension fractale d'une image, l'objet A correspond à la surface des niveaux de gris et est dans un espace à trois dimensions. Si l'image possède $n_y \times n_x$ pixels

$$d = \lim_{p \rightarrow 0} \frac{\ln(N(p))}{\ln(1/p)}$$

quantifiés sur n_g niveaux de gris, alors la boîte renfermant A est un volume de taille $n_y \times n_x \times n_g$. En découplant ce parallélépipède en petits cubes d'arrête p, on obtient N(p) cubes qui ont une intersection non vide avec la surface A considérée. La dimension fractale est alors :

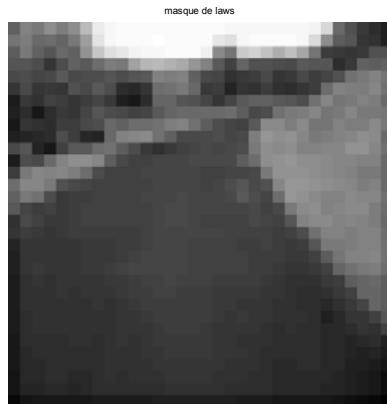
La pente de la droite dans le diagramme bi-logarithmique $\ln(N(p))$ vs $\ln(1/p)$ donne la dimension fractale d .

XI - Masque de Laws

Laws nous propose un ensemble de masques 3x3 ou 5x5 trouvés de manière expérimentale en effectuant des tests sur un ensemble d'images de textures naturelles. L'idée consiste à représenter les textures par des mesures d'énergie réalisées à la sortie d'un banc de filtres. L'image originale I est donc convoluée par ces filtres, donnant lieu à plusieurs images résultat I_j . Les caractéristiques de texture sont ensuite obtenues en sommant en valeur absolue les pixels des images résultats dans les différentes fenêtres de travail. L'intérêt de cette méthode est qu'elle est simple à mettre en oeuvre, et qu'elle est directement exploitable pour réaliser une segmentation en régions. Nous pouvons citer à titre d'exemples un des masques 3x3 proposés par Laws :

1	2	1
2	4	2
4	2	4

Bien que cette méthode repose uniquement sur des tests expérimentaux, elle donne de bons résultats comme on peut le constater sur l'image ci-dessous



XII - Conclusion

Nous venons de voir plusieurs méthodes permettant d'obtenir des paramètres de texture. Il faut retenir qu'aucune d'entre elles ne peut être a priori considérée comme meilleure. En effet, selon les applications certaines seront plus robustes que d'autres. Un critère important à prendre en compte pour des applications réelles est le temps de calcul. Ce paramètre fait que certaines méthodes, très lourdes en temps de calcul, sont peu employées dans la pratique.

Il est aussi important de rappeler ici qu'il n'existe pas de paramètre universel permettant de décrire les textures dans leur globalité. Il faut toujours raisonner au cas par cas, avec, qui plus est, des critères de sélection assez subjectifs. Pourquoi utiliser un paramètre plutôt qu'un autre ? Les paramètres sont-ils complémentaires ou redondants ? Autant de questions qui ne seront résolues que par des tests sur les images de la base d'application.

Chapitre 9 : Introduction à la stéréovision

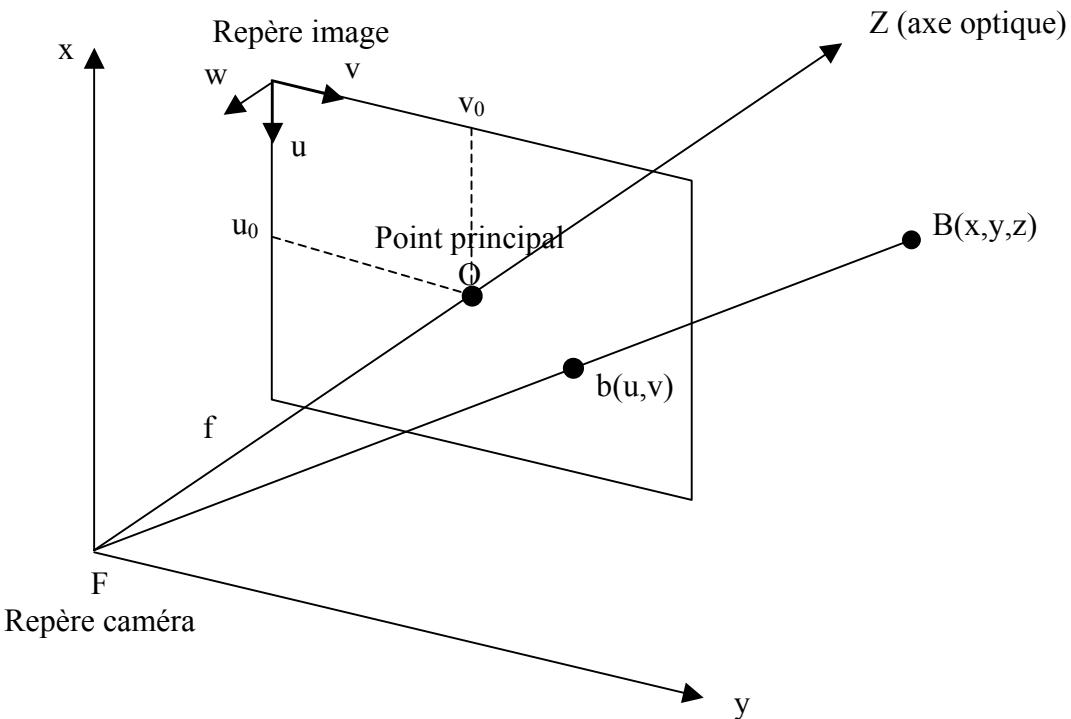
La stéréovision est destinée à connaître la troisième dimension de l'environnement. En traitement d'images, elle permet de résoudre de nombreux problèmes : distance d'un obstacle, estimation des dimensions des pièces, ... La troisième dimension peut être obtenue de plusieurs manières :

- Grâce à des mouvements oculaires.
- Grâce à l'accommodation : l'œil n'accorde pas de la même façon selon que l'objet d'attention est près ou loin.
- Grâce à l'utilisation de plusieurs capteurs (œil ou caméra), ce qui permet de trouver la troisième dimension à partir d'une triangulation. C'est la vision stéréoscopique.

Voyons dans un premier temps la géométrie mise en place lors de l'acquisition d'une image.

I - Transformation d'un point de l'espace en un point de l'image

1. La projection perspective



Soit $B(x,y,z)$, un point de l'espace exprimé dans le repère caméra. Il se projette en $b(x',y',z')$. La distance (O,F) est la distance focale, F étant le centre optique de la caméra. On peut, dans le repère caméra, exprimer les coordonnées de b en fonction de celles de B :

$$\begin{cases} x'/x = f/z \\ y'/y = f/z \\ z' = f \end{cases} \quad \text{soit} \quad \begin{cases} x' = xf/z \\ y' = xf/z \\ z' = f \end{cases}$$

Ces trois équations peuvent se mettre sous forme matricielle, en introduisant les coordonnées homogènes du point $B(x,y,z,1)$:

$$\begin{pmatrix} sx' \\ sy' \\ sz' \\ s \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{pmatrix}}_P \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Les coordonnées cartésiennes du point b sont $(sx'/s, sy'/s, sz'/s)$.

2. Transformation repère caméra / repère image

Dans le repère image, le point b a pour coordonnée (u, v) . La transformation s'écrit :

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} k_u & 0 & 0 \\ 0 & k_v & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} + \begin{pmatrix} u_0 \\ v_0 \\ w_0 \end{pmatrix}$$

- La première matrice correspond à un changement d'échelle. k_u et k_v sont les facteurs d'échelle vertical et horizontal. Ils ne sont pas forcément égaux car un pixel n'est pas obligatoirement carré. k_u et k_v sont mesurés en pixels/mm.
- La seconde matrice représente une rotation du système d'axe.
- u_0, v_0 et w_0 correspondent à des translations. Ce sont les coordonnées de F dans le repère image.

La troisième composante, w est toujours nulle. On peut donc la supprimer et arriver à :

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \underbrace{\begin{pmatrix} -k_u & 0 & 0 & u_0 \\ 0 & k_v & 0 & v_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_K \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

3. Transformation point de l'espace / point image

Il suffit maintenant de combiner les deux équations précédentes :

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \underbrace{\begin{pmatrix} -k_u & 0 & 0 & u_0 \\ 0 & k_v & 0 & v_0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_K \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{pmatrix}}_P \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{pmatrix} -k_u & 0 & u_0/f & 0 \\ 0 & k_v & v_0/f & 0 \\ 0 & 0 & 1/f & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

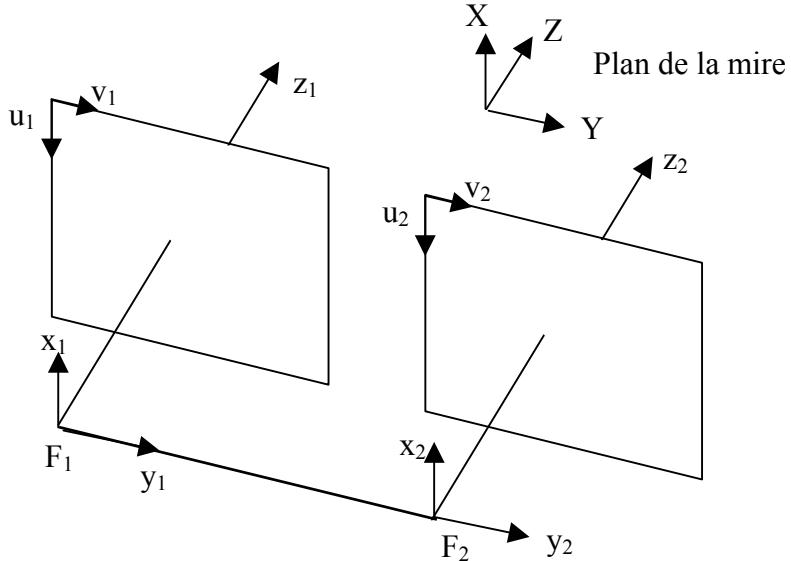
En multipliant tous les coefficients de la matrice par f , on obtient :

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \underbrace{\begin{pmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{Ic} \begin{pmatrix} x \\ y \\ z \\ s \end{pmatrix}$$

Les paramètres α_u, α_v, u_0 et v_0 sont appelés paramètres intrinsèques.

III - Cas simple de stéréovision

On considère une configuration particulière des deux caméras : leurs plans image sont dans le même plan. D'autre part, on introduit un nouveau repère : celui d'une mire (objet connu) que l'on appelle (X, Y, Z) . Dans la pratique, on n'a pas accès au repère (x, y, z) , on se placera donc dans le repère de la mire. Ici encore on simplifie le problème en supposant qu'il n'y a pas de rotation entre le repère de la mire et le repère de la caméra.



Transformation point de l'espace / point image

De la même façon que précédemment, pour une caméra, la transformation repère caméra/repère image s'écrit :

$$\begin{pmatrix} su_1 \\ sv_1 \\ s \end{pmatrix} = \underbrace{\begin{pmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{I_c} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ s \end{pmatrix} \text{ soit } \begin{cases} u_1 = \alpha_u \frac{x_1}{z_1} + u_0 \\ v_1 = \alpha_v \frac{y_1}{z_1} + v_0 \end{cases}$$

Passage repère mire / repère image

On suppose ici que seules des translations existent entre le repère de la mire et le repère de la caméra :

$$\begin{cases} x_1 = X + t_x \\ y_1 = Y + t_y \\ z_1 = Z + t_z \end{cases}$$

Côté stéréo

Le passage d'un repère caméra à l'autre est composé d'une seule translation selon l'axe y :

$$\begin{cases} x_2 = x_1 \\ y_2 = y_1 - b_y \\ z_2 = z_1 \end{cases}$$

D'où

Pour la caméra 1

$$\begin{cases} u_1 = \alpha_u \frac{x_1}{z_1} + u_0 \\ v_1 = \alpha_v \frac{y_1}{z_1} + v_0 \end{cases}$$

En exprimant tout dans le repère caméra 1,

$$\begin{cases} u_1 = \alpha_u \frac{x_1}{z_1} + u_0 \\ v_1 = \alpha_v \frac{y_1}{z_1} + v_0 \end{cases}$$

Pour la caméra 2

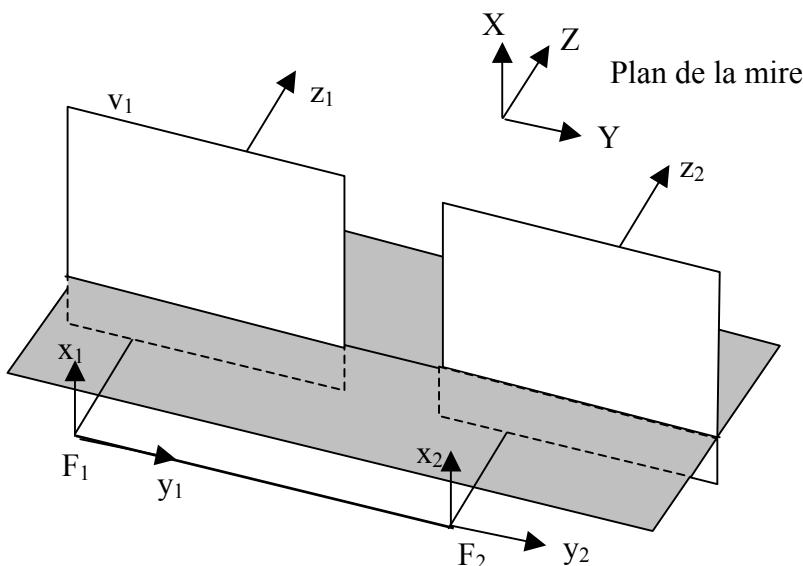
$$\begin{cases} u_2 = \alpha_u \frac{x_2}{z_2} + u_0 \\ v_2 = \alpha_v \frac{y_2}{z_2} + v_0 \end{cases}$$

$$\begin{cases} u_2 = \alpha_u \frac{x_1}{z_1} + u_0 \\ v_2 = \alpha_v \frac{y_1 - b_y}{z_1} + v_0 \end{cases}$$

On retrouve, si les deux caméras sont identiques,

$$u_1 = u_2 \text{ et } v_2 = v_1 - \frac{\alpha_v b_y}{z_1}$$

En fait, il est logique de trouver le même u si on raisonne graphiquement :



Dans le plan de la mire, on a:

$$v_2 = v_1 + \frac{\alpha_v b_y}{Z + t_z} \text{ soit } (v_2 - v_1)Z + (v_2 - v_1)t_z + \alpha_v b_y = 0$$

En prenant plusieurs points de la mire où Z , v_1 et v_2 sont connus, on peut facilement estimer les deux inconnues t_z et b_y .

Lors de la reconstruction, dès que la mise en correspondance est faite, on obtient Z .

Chapitre 10 : Initiation à la compression d'images

Aujourd’hui, des supports de données de grande capacité (clés USB, DVD, disques durs de plusieurs giga) existent permettant de stocker de nombreuses images. Comme, en parallèle, le nombre d’images ne fait qu’augmenter (appareils photo numériques) et que l’on utilise de plus en plus internet (qui nécessite des vitesses de transfert élevé des images) la compression d’images est toujours d’actualité.

I – Généralités

La plus petite unité de stockage est le bit (0 ou 1). Toutes les données informatiques sont codées en bits. Une photo, une musique ou bien un texte est une suite plus ou moins longue de 0 et de 1, qu'un programme saura interpréter pour lui rendre son apparence d'origine.

Les bits sont regroupés en paquets de 8 pour former des octets (byte en anglais). Un octet peut prendre une valeur comprise entre 00000000 et 11111111 (base 2), soit entre 0 et 255 en base 10.

Ces unités ne sont que peu fréquemment employées, on utilise plutôt leur multiple :

- $2^{10} = 1024$ octets = 1 ko (kilo octet), soit 8192 bits.
- $2^{20} = 1\ 048\ 576$ octets = 1 Mo (Méga octet).
- $2^{30} = 1\ 073\ 741\ 824$ octets = 1 Go (Giga octet).

Une image est composée de plusieurs pixels (dépendant de sa résolution). Chaque pixel possède trois composantes (R, V, B) et chaque composante est codée sur un octet.

La taille d'une image de 600x800 pixels sera :

- $3 \times 8 \times 600 \times 800$ bits = 1410 ko pour une image couleur
- $8 \times 600 \times 800$ bits = 470 ko pour une image couleur
- 600×800 bits = 60 ko pour une image niveaux de gris

II– Notion de théorie de l'information

La compression consiste à garder l'information contenue dans l'image (le dessin qui nous apparaît) dans un fichier qui représente beaucoup moins d'octets que l'image brute.

Prenons un exemple de compression très simple et considérons la suite de valeurs d'octets suivante : **0 0 0 255 255 255 255 0 0 0 0 254**. Cette suite est codée sur 14 octets.

Mais on aurait aussi pu la coder avec le nombre de fois où un octet apparaît, suivi de cet octet : **0 4 255 5 0 1 254** (On a donc 4 fois 0, 4 fois 255, 5 fois 0 et 1 fois 254), ce qui fait 8 octets au total. D'un message qui demandait 14 octets, on est passé à 8 octets, et ce tout en pouvant reconstituer le message initial !

On introduit alors le taux de compression :

$$\tau = 100 - \frac{\text{nombre de bits de l'image compressée}}{\text{nombre de bits de l'image originale}} * 100$$

Mais bien sûr, ce format de codage très simple et ne marche pas à tous les coups. Prenons par exemple les octets suivants : **255 254 253 252 251 250** (soit 6 octets). La même méthode aboutit à : **1 255 1 254 1 253 1 252 1 251 1 250**, soit 12 octet ! On constate dès à présent que le taux de compression d'une image dépend de son contenu.

Ceci nous conduit à une discipline appelée la théorie de l'information qui permet de déterminer la quantité d'information contenue dans une image (travaux de Shannon, 1940). Cette quantité amène à la taille minimale du fichier compressé que l'on pourra obtenir, sans perdre d'information.

Considérons une série d'images numériques de même type, c'est à dire de mêmes dimensions, nature et nombre de niveaux de gris. Après lecture de ces images nous avons reçu une plus ou moins grande quantité d'informations dépendant de ce que l'image représentait. L'objectif de Shannon et de la théorie de l'information est de définir puis de quantifier (c'est à dire chiffrer) la quantité d'information reçue .

Considérons une image numérique « niveaux de gris » de N points. Chaque point de l'image est considéré comme une variable aléatoire pouvant prendre 256 valeurs: n_i (les niveaux de gris). L'image est donc représentée comme une suite de N variables aléatoires que l'on suppose indépendante.

Soit $p(n_i)$ la probabilité du niveau de gris n_i . La quantité d'information contenue dans l'image (aussi appelée entropie) est définie par :

$$H = - \sum_{i=1}^N p(n_i) \log_2(p(n_i))$$

Exemple :

- tous les pixels sont identiques. Une des probabilité vaut 1 tandis que les autres valent 0. La quantité d'information contenue dans cette image est nulle.
- chaque niveau de gris est équiprobable $p(n_i)=1/256$. Son entropie sera : $H = - \sum_{i=1}^N p(n_i) \log_2(p(n_i)) = 8$. Il faudra donc 8 bits pour coder chaque niveau de gris.

La quantité d'information nous donne une limite théorique que l'on ne peut pas dépasser. On ne peut pas réduire une donnée plus que l'information qu'elle contient. La plupart des algorithmes de compression actuels s'approchent déjà des limites théoriques.

C'est pourquoi, pour gagner encore en taux de compression, la compression avec perte est apparue (par opposition à la compression sans perte).

Une méthode dite "sans perte" s'il n'y a aucune perte d'information lors de la compression. On sera alors capable de retrouver l'image initiale à partir de l'image compressée. Le taux de compression est généralement assez faible.

La compression avec perte est un compromis entre qualité d'image et taux de compression (on n'est maintenant plus capable de retrouver l'image d'origine, il y a eu perte d'information. Plus le fichier compressé est petit et plus la qualité de l'image est mauvaise.

III – Les différents types de compression

1 – Le code RLE (Run Length Encoding)

Cette méthode (aussi notée *RLC* pour Run Length Coding) est utilisée par de nombreux formats d'images (BMP, PCX, TIFF). Elle est basée sur la répétition d'éléments consécutifs.

Le principe de base consiste à coder un premier élément donnant le nombre de répétitions d'une valeur puis le compléter par la valeur à répéter.

Exemple 1 :

« 19 19 19 19 19 19 5 5 5 5 5 5 5 5 5 5 5 » devient « 6 19 14 5 » soit

$$\tau = 100 - \frac{4}{20} * 100 = 80\%$$

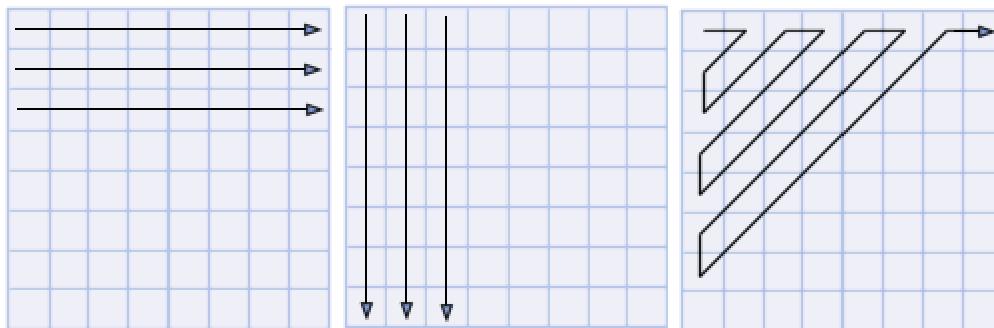
Exemple 2 :

« 5 2 2 4 4 2 8 2 6 9 » devient « 1 5 2 2 2 4 1 8 1 2 1 6 1 9 » soit $\tau = 100 - \frac{14}{10} * 100 = -40\%$.

La compression s'avère ici très coûteuse, avec un gain négatif ! En réalité la compression RLE est régie par des règles particulières permettant de compresser lorsque cela est nécessaire et de laisser la chaîne telle quelle lorsque la compression induit un gaspillage. Par exemple :

- Lorsque trois éléments ou plus se répètent consécutivement alors la méthode de compression RLE est utilisée. Sinon un caractère de contrôle (00) est inséré, suivi du nombre d'éléments de la chaîne non compressée.

Ainsi la compression RLE n'a du sens que pour les données possédant de nombreux éléments consécutifs redondants, notamment les images possédant de larges parties uniformes. Cette méthode a toutefois l'avantage d'être peu difficile à mettre en œuvre. Il existe des variantes dans lesquelles l'image est encodée par pavés de points, selon des lignes, ou bien même en zigzag.



2 – Le code de Huffman

David Huffman a proposé en 1952 une méthode statistique qui permet d'attribuer un symbole aux différentes valeurs à compresser. La longueur de chaque symbole n'est pas identique : plus un symbole est fréquent, plus il sera codé sur peu de bits. On parle de **codage à longueur variable préfixé** car aucun code ne sera le préfixe d'un autre. Avec ce codage, la longueur moyenne des symboles sera plus petite que pour un code à longueur fixe.

Exemple :

Considérons la suite de pixels :

« 3 7 1 1 4 8 9 5 3 2 5 1 2 0 3 6 E »

Les fréquences d'apparition des différents symboles seront :

Ngd	1	2	3	4	5	6	7	8	9	0
fréquence	3	2	2	2	2	1	1	1	1	1

La méthode de Huffman permet, à partir de ces fréquences, de déterminer le symbole optimal de chaque niveau de gris, pour avoir une longueur moyenne de message minimale. Ici, on obtient :

N d g	1	2	3	4	5	6	7	8	9	0
fréquence	3	2	2	2	2	1	1	1	1	1
symbole	00	100	110	010	011	1110	1111	1010	10110	10111

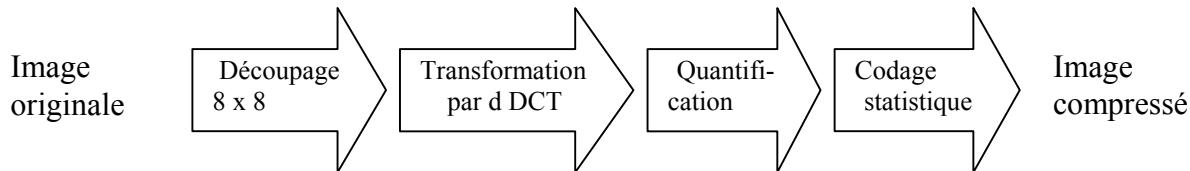
On remarque que :

- le symbole le plus fréquent (1) est codé sur peu de bits (2).
- Aucun code n'est préfixe d'un autre.

3 – La compression JPEG (Joint Photographic Expert 1974)

Contrairement aux autres compressions vues jusqu'à maintenant, il s'agit d'une compression avec pertes, ce qui lui donne un des meilleurs taux de compression.

Le principe de l'algorithme JPEG est le suivant :



Considérons comme exemple la matrice 8 x 8 suivante :

255	255	255	0	0	0	0	0
36	255	100	100	36	36	36	36
73	255	100	73	100	73	73	73
109	255	100	100	100	100	100	109
146	146	100	146	100	146	146	146
182	182	100	182	100	100	100	182
218	218	218	218	100	218	218	218
255	255	255	255	100	100	100	255

La première étape consiste à soustraire 128 à chaque valeur puis à appliquer la DCT : Discret Cosinus Transform :

$$F(u,v) = \frac{2}{N} c(u)c(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I(x,y) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right]$$

avec $c(0)=1/\sqrt{2}$ et $c(u)=1$ pour $u=1, \dots, N-1$

La DCT peut s'apparenter à la transformée de Fourier 2D. Pour la matrice précédente, nous arrivons à :

Composante continue	50	258	135	-120	-73	-76	-57	-103
	-319	148	-9	28	-124	-101	-55	63
	68	170	52	-89	-15	31	80	35
	-21	25	40	44	16	68	60	30
	22	81	46	-9	14	14	55	47
	66	33	39	33	-54	4	29	38
	-31	68	30	-10	54	-29	13	33
Contours horizontaux	41	-30	0	34	-31	37	36	29
Contours verticaux								
Hautes fréquences								

On remarque que les coefficients possédant les valeurs absolues les plus fortes se trouvent en haut à gauche de la matrice.

L'étape suivante est celle de la quantification. On remplace chaque valeur de la matrice par le quotient de cette valeur par le pas de quantification. Comme l'œil est plus sensible aux basses fréquences qu'aux hautes fréquences, on fait appel à une matrice de quantification (il existe

plusieurs matrice de quantification, en fonction du degré de perte choisi pour l'algorithme) où les coefficients sont plus faibles aux hautes fréquences :

Matrice de quantification

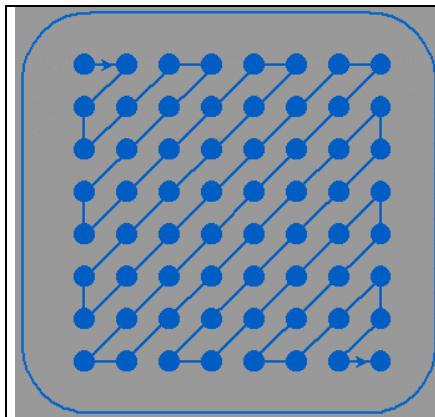
6	11	16	21	26	31	36	41
11	16	21	26	31	36	41	46
16	21	26	31	36	41	46	51
21	26	31	36	41	46	51	56
26	31	36	41	46	51	56	61
31	36	41	46	51	56	61	66
36	41	46	51	56	61	66	71
41	46	51	56	61	66	71	76

C'est lors de cette étape de quantification qu'il y a perte d'information

Nouveau secteur

8	23	8	-5	-2	-2	-1	-2
-29	9	0	1	-4	-2	-1	1
4	8	2	-2	0	0	1	0
-1	0	1	1	0	1	1	0
0	2	1	0	0	0	0	0
2	0	0	0	-1	0	0	0
0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0

De nombreux zéros apparaissent dans la matrice que l'on va de nouveau compresser (sans perte). La norme JPEG traite les 0 d'une manière particulière, en raison de leur nombre important. La matrice est balayée en zigzag de pour obtenir les suites de 0 les plus importantes possibles :



Il s'ensuit :

- un codage RLE pour profiter des longues plages de zéros
- un codage de Huffman

FIN

Lorsque l'on pousse le taux de la compression (avec la matrice de quantification), on obtient une détérioration sensible de la qualité :

- des tâches qui n'existaient pas apparaissent, elles sont dues à un effet de blocs (secteur 8x8)
- des bavures apparaissent au niveau des zones de transition. Elles sont dues à la quantification très importante des hautes fréquences.