

# Algorithmes pour le traitement de l'image

Mathias Ortner

18 mai 2004

# Table des matières

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Présentation du cours</b>                        | <b>5</b>  |
|          | Algorithmes pour le traitement de l'image . . . . . | 6         |
|          | Plan du cours . . . . .                             | 7         |
| <b>2</b> | <b>Courte introduction</b>                          | <b>8</b>  |
|          | Définition et historique . . . . .                  | 9         |
|          | Exemples d'images . . . . .                         | 10        |
|          | Exemples de traitements . . . . .                   | 11        |
|          | Quelques éléments simples d'une image . . . . .     | 12        |
| <b>3</b> | <b>Image Numérique</b>                              | <b>13</b> |
|          | Image Numérique . . . . .                           | 14        |
|          | L'image Numérique . . . . .                         | 15        |
|          | Notion de résolution . . . . .                      | 16        |
|          | Le codage des couleurs . . . . .                    | 17        |
|          | Codage des couleurs, illustration . . . . .         | 18        |
|          | Les formats de stockage . . . . .                   | 19        |
|          | Exemple d'un format de codage: PGM . . . . .        | 20        |
|          | Principaux formats . . . . .                        | 21        |
|          | L'image comme une fonction . . . . .                | 22        |
|          | L'image comme une surface . . . . .                 | 23        |
| <b>4</b> | <b>Histogrammes et contraste</b>                    | <b>24</b> |
|          | Histogramme d'une image . . . . .                   | 25        |
|          | Densité de probabilité . . . . .                    | 26        |
|          | Utilité de l'histogramme . . . . .                  | 27        |
|          | Histogrammes à $n$ niveaux . . . . .                | 28        |
|          | Transformation d'histogrammes . . . . .             | 29        |
|          | Exemples . . . . .                                  | 30        |
|          | Dynamique de l'image . . . . .                      | 31        |
|          | Égalisation d'histogrammes . . . . .                | 32        |
| <b>5</b> | <b>Détection de contours</b>                        | <b>34</b> |
|          | Introduction . . . . .                              | 35        |
|          | Discontinuités . . . . .                            | 36        |
|          | Différents types de contours . . . . .              | 37        |
|          | Détection d'une discontinuité . . . . .             | 38        |
|          | Gradient . . . . .                                  | 39        |
|          | Dans le cas discret . . . . .                       | 40        |
|          | Commentaires . . . . .                              | 41        |
|          | Masques de convolutions . . . . .                   | 42        |
|          | Masques de gradients . . . . .                      | 43        |
|          | Les opérateurs de Sobel . . . . .                   | 44        |
|          | Maxima du gradient . . . . .                        | 45        |
|          | Seuillage du gradient . . . . .                     | 46        |
|          | Problème du seuillage . . . . .                     | 47        |
|          | Recherche d'un maximum local . . . . .              | 48        |
|          | Contours et Laplacien . . . . .                     | 49        |
|          | Calcul du Laplacien . . . . .                       | 50        |
|          | Détection du passage par 0 . . . . .                | 51        |

|  |           |
|--|-----------|
| Problème du bruit . . . . .                              | 52        |
| Gradients et contours . . . . .                          | 53        |
| Un algorithme de numérotation des contours (1) . . . . . | 54        |
| Un algorithme de numérotation des contours (2) . . . . . | 55        |
| Notion de voisinage d'un pixel . . . . .                 | 56        |
| Mécanisme de diffusion(1) . . . . .                      | 57        |
| Mécanisme de diffusion(2) . . . . .                      | 58        |
| Un algorithme de numérotation des contours (3) . . . . . | 59        |
| <b>6 Détection de Droites (Hough) . . . . .</b>          | <b>60</b> |
| Détection de droites . . . . .                           | 61        |
| Méthode: . . . . .                                       | 62        |
| Comment caractériser une droite? . . . . .               | 63        |
| Paramétrisation de la droite . . . . .                   | 64        |
| Norme du gradient et droite . . . . .                    | 65        |
| Direction du gradient et droite . . . . .                | 66        |
| Rappel Contours et gradient . . . . .                    | 67        |
| Gradient et droites . . . . .                            | 68        |
| Mécanisme de vote . . . . .                              | 69        |
| Trouver les droites . . . . .                            | 70        |
| Identification des droites . . . . .                     | 71        |
| En pratique . . . . .                                    | 72        |
| Conclusion . . . . .                                     | 73        |
| Exercice . . . . .                                       | 74        |
| <b>7 Bruit et Filtrage . . . . .</b>                     | <b>75</b> |
| Le bruit . . . . .                                       | 76        |
| Le filtrage . . . . .                                    | 77        |
| Bruit Sel et poivre . . . . .                            | 78        |
| Bruit Gaussien (1) . . . . .                             | 79        |
| Bruit Gaussien (2) . . . . .                             | 80        |
| Filtrage Linéaire . . . . .                              | 81        |
| Filtrage linéaire itéré . . . . .                        | 82        |
| Filtrage Non linéaire . . . . .                          | 83        |
| Filtrage Non linéaire (2) . . . . .                      | 84        |
| <b>8 Morphologie Mathématique . . . . .</b>              | <b>85</b> |
| Introduction . . . . .                                   | 86        |
| Élément structurant . . . . .                            | 87        |
| Dilatation . . . . .                                     | 88        |
| Érosion . . . . .  | 89        |
| Rapport à l'élément structurant . . . . .                | 90        |
| Ouverture et Fermeture . . . . .                         | 91        |
| Ouverture . . . . .                                      | 92        |
| Fermeture . . . . .                                      | 93        |
| Exemple . . . . .  | 94        |
| Filtre Alterné séquentiel . . . . .                      | 95        |
| Exercices . . . . .                                      | 96        |
| Autre formulation . . . . .                              | 97        |
| Passage aux niveaux de gris . . . . .                    | 98        |

|  |            |
|--|------------|
| Illustration: cas 1D . . . . .             | 99         |
| Ouverture et Fermeture . . . . .           | 100        |
| Exemples . . . . .                         | 101        |
| Autres Opérateurs . . . . .                | 102        |
| Autres Opérateurs . . . . .                | 103        |
| Exercice . . . . .                         | 104        |
| <b>9 TP Watersheds</b>                     | <b>105</b> |
| Algorithme de watershed . . . . .          | 106        |
| Watershed? . . . . .                       | 107        |
| Définitions: les bases . . . . .           | 108        |
| Définitions: un minimum . . . . .          | 109        |
| Définitions: watershed . . . . .           | 110        |
| Bassin d'attraction . . . . .              | 111        |
| Rappel: connexité . . . . .                | 112        |
| Distance géodésique . . . . .              | 113        |
| Zone d'influence . . . . .                 | 114        |
| Illustration Zones d'influences . . . . .  | 115        |
| Immersion . . . . .                        | 116        |
| Illustration . . . . .                     | 117        |
| Premier cas . . . . .                      | 118        |
| Deuxième cas . . . . .                     | 119        |
| Troisième cas . . . . .                    | 120        |
| Itérations et résultat . . . . .           | 121        |
| <b>10 Segmentation des images</b>          | <b>125</b> |
| Introduction . . . . .                     | 126        |
| Définition . . . . .                       | 127        |
| Dans ce chapitre . . . . .                 | 128        |
| Définition . . . . .                       | 129        |
| Division de région (1) . . . . .           | 130        |
| Division de région (2) . . . . .           | 131        |
| Arbre de division . . . . .                | 132        |
| Croissance de région . . . . .             | 133        |
| Commentaires . . . . .                     | 134        |
| Variantes . . . . .                        | 135        |
| Critères d'homogénéité . . . . .           | 136        |
| Critères d'homogénéité . . . . .           | 137        |
| Autres possibilités . . . . .              | 138        |
| Fusion de régions dans un graphe . . . . . | 139        |
| Problématique . . . . .                    | 140        |
| Graphe d'adjacence . . . . .               | 141        |
| Graphe de fusion . . . . .                 | 142        |
| Stratégies de fusion . . . . .             | 143        |
| Stratégies de fusion 1 . . . . .           | 144        |
| Stratégie de fusion 2 . . . . .            | 145        |
| Quadtree . . . . .                         | 146        |
| Quadtree (2) . . . . .                     | 147        |
| Arbre et coupure . . . . .                 | 148        |
| Coupure et segmentation . . . . .          | 149        |

|  |            |
|--|------------|
| Approches pyramidales . . . . .                              | 150        |
| Pyramide de graphes par Fusion . . . . .                     | 151        |
| Notion de stable . . . . .                                   | 152        |
| Illustration . . . . .                                       | 153        |
| Stable en pratique . . . . .                                 | 154        |
| Illustration . . . . .                                       | 155        |
| Segmentation par corrélation d'histogrammes locaux . . . . . | 156        |
| Approximations . . . . .                                     | 157        |
| Propriétés de l'histogramme local . . . . .                  | 158        |
| H. locaux et relaxation . . . . .                            | 159        |
| Relaxation: illustration . . . . .                           | 160        |
| H. Locaux et propagation . . . . .                           | 161        |
| H. Locaux et propagation(2) . . . . .                        | 162        |
| Propagation: illustration . . . . .                          | 163        |
| <b>11 Champs de Markov</b>                                   | <b>164</b> |
| Modèle probabiliste . . . . .                                | 165        |
| Le modèle aléatoire le plus simple . . . . .                 | 166        |
| Notion de voisinage . . . . .                                | 167        |
| Cliques . . . . .  | 168        |
| Dépendance . . . . .   | 169        |
| Champs de Markov:local . . . . .                             | 170        |
| Champs de Markov: global . . . . .                           | 171        |
| Le modèle d'Ising . . . . .                                  | 172        |
| Le modèle d'Ising (2) . . . . .                              | 173        |
| Le modèle de Potts . . . . .                                 | 174        |
| Exemple de modèle de texture . . . . .                       | 175        |
| Le triangle d'or . . . . .                                   | 176        |
| Champs de Markov Gaussiens . . . . .                         | 177        |
| Exemple . . . . .  | 178        |
| Ajouter une texture à une image . . . . .                    | 179        |
| Simuler un champ de Markov . . . . .                         | 180        |
| Metropolis-Hasting et Propriété de Markov . . . . .          | 181        |
| <b>12 Stéréovision</b>                                       | <b>182</b> |
| Stéréovision . . . . .                                       | 183        |
| Les étapes de la stéréovision . . . . .                      | 184        |
| Comment ça marche? . . . . .                                 | 185        |
| Commentaires . . . . .                                       | 186        |
| Caméra et Géométrie . . . . .                                | 187        |
| La projection . . . . .                                      | 188        |
| Du repère Monde au repère Caméra . . . . .                   | 189        |
| Du repère Caméra a l' Image . . . . .                        | 190        |
| Triangulation . . . . .                                      | 191        |
| Entre deux images . . . . .                                  | 192        |
| Des Images au Monde . . . . .                                | 193        |
| Les épipoles . . . . .                                       | 194        |
| La contrainte épipolaire . . . . .                           | 195        |
| Les lignes épipolaires . . . . .                             | 196        |
| La configuration parallèle . . . . .                         | 197        |

|   |     |
|---|-----|
| La rectification . . . . .                            | 198 |
| Rectifier une image . . . . .                         | 199 |
| Rectification de l'image 1 . . . . .                  | 200 |
| Rectification de l'image 2 . . . . .                  | 201 |
| Mise en correspondance . . . . .                      | 202 |
| Image de disparité . . . . .                          | 203 |
| Exemple . . . . .                                     | 204 |
| Occlusions . . . . .                                  | 205 |
| Corrélation . . . . .                                 | 206 |
| Commentaires . . . . .                                | 207 |
| Premier algorithme . . . . .                          | 208 |
| Les problèmes . . . . .                               | 209 |
| Les solutions . . . . .                               | 210 |
| Aller retour . . . . .                                | 211 |
| Ordre de projection . . . . .                         | 212 |
| Contrainte d'ordre . . . . .                          | 213 |
| Occlusions . . . . .                                  | 214 |
| Occlusions et ordre de parcours . . . . .             | 215 |
| Appariement simultané . . . . .                       | 216 |
| Appariement simultané et contrainte d'ordre . . . . . | 217 |
| Programmation dynamique (1) . . . . .                 | 218 |
| Problème de minimisation . . . . .                    | 219 |
| Formulation récursive . . . . .                       | 220 |
| Programmation dynamique . . . . .                     | 221 |
| Coût de transition . . . . .                          | 222 |
| Intervalle de disparité . . . . .                     | 223 |
| Exemple . . . . .                                     | 224 |
| Exemple 2 . . . . .                                   | 225 |

## Chapitre 1

# Présentation du cours



---

---

## Algorithmes pour le traitement de l'image

*Cours d'introduction*

Mathias Ortner

Avril 2004, Licence d'informatique

---

---

Présentation du cours

Nr. 1

**Coordonnées**

**Émail**     mathias.ortner@sophia.inria.fr

**Tel**        04 92 38 75 68

**Adresse**    Projet ARIANA  
              INRIA Sophia Antipolis  
              2004 Routes des Lucioles  
              06902 Sophia Antipolis B.P 93

---

---

## Plan du cours

1. **Courte Introduction**  
Les enjeux du traitement de l'image.
2. **l'image numérique**  
Les notions de bases : résolution, intensité, stockage.
3. **Histogrammes et contraste**  
Dynamique de l'image, rehaussement de contraste.
4. **Détection des contours**
5. **Bruit et filtrage**  
Courte présentation de notions de traitement du signal.
6. **Morphologie Mathématique**  
Une classe d'algorithmes et de méthodes couramment utilisées.
7. **Segmentation des images**  
Quelques notions sur les approches régions.

## Chapitre 2

### Courte introduction

---

---

## Définition et historique

Une image :

**Dans le dictionnaire** *Représentation de la forme ou de l'aspect d'un être ou d'une chose*

**En informatique** Une quantité d'information visuelle, sous une certaine représentation.

| Date,Lieu      | Exemple          | Support de l'information |
|----------------|------------------|--------------------------|
| Carthage, -300 | Mozaiques        | Petites pierres          |
| 1500           | Joconde          | Peinture                 |
| Cortex         | Image rétinienne | Stimuli organiques       |
| 1820           | Photo argentique | Cristaux                 |
| 1963           | Polaroid couleur | Cristaux                 |
| 1990           | Photo numérique  | Mémoire informatique     |

---

---

Courte introduction

Nr. 1

### Limites

Les limites de la notion d'image sont floues. On parle d'image sonore par exemple.

La notion d' image peut s'éloigner de la notion de vision. Considérer, par exemple, une image issue d'un IRM ou d'un scanner. Il s'agit plus du résultat d'acquisition d'un capteur que d'une image. Pourtant le traitement de telles données entre dans le vaste domaine d'application du traitement d'images.

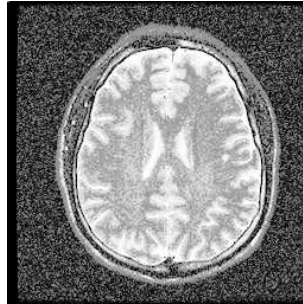
---

---

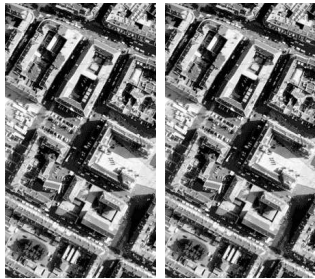
## Exemples d'images



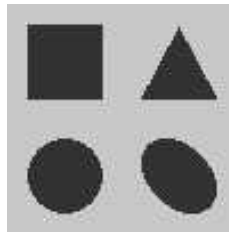
Lena



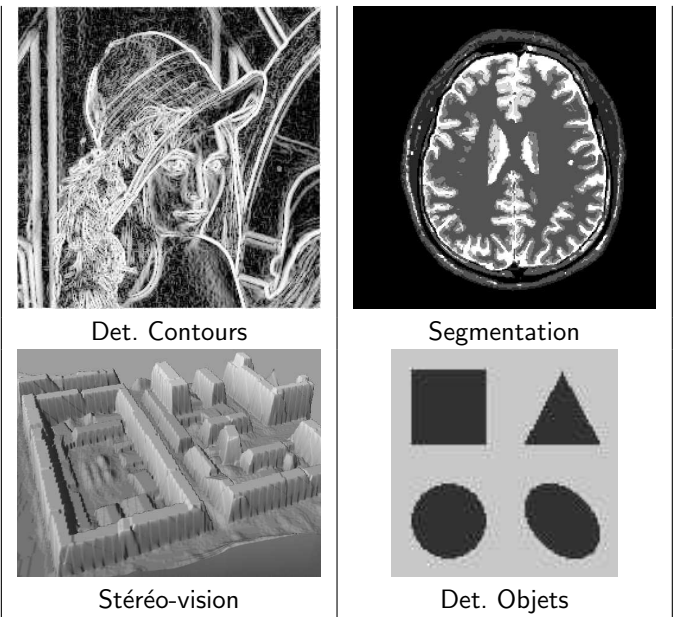
IRM



Amiens (IGN)



Exemples de traitements



Un contour est une frontière séparant deux *zones homogènes*. Nous verrons plusieurs algorithmes d'extraction de contours.

Détection de contour

La segmentation est un processus élémentaire mais difficile du traitement d'image. Elle consiste en la séparation d'une image en plusieurs *zones homogènes*.

Segmentation

Une zone est homogène si on peut trouver un critère permettant de la décrire : couleur, texture, etc...

Zone homogène

La stéréo-vision exploite plusieurs prises de vues d'un objet pour en reconstruire le relief. Ce n'est pas la seule manière de "comprendre" le relief d'un objet, mais c'est la plus couramment utilisée.

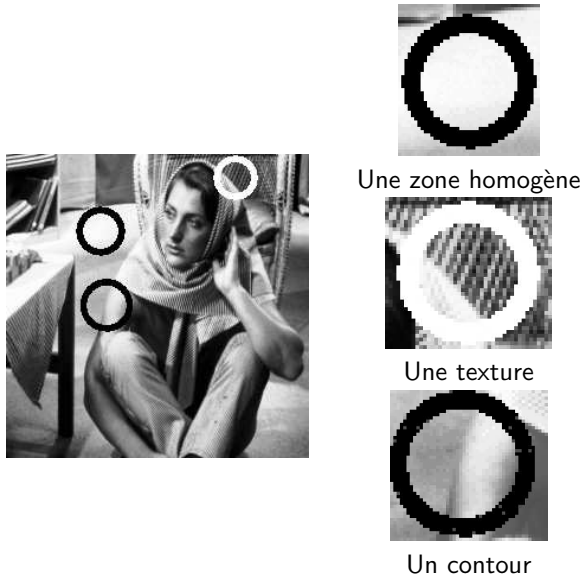
Stéréo-vision

La détection d'objets fait partie des traitements de haut niveau: compréhension sémantique, reconnaissance de formes, etc...

Détection d'objet

---

## Quelques éléments simples d'une image




---

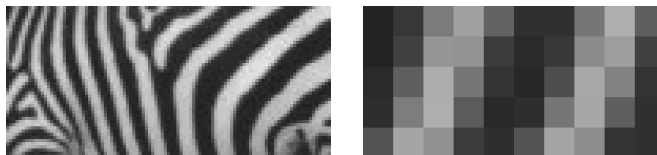
Courte introduction

Nr. 4

La définition de texture en image est floue. On retiendra que le terme qualifie la répétition (exacte ou non) d'un motif visuel (Penser aux pelages d'animaux).

### Texture

La notion de texture est étroitement liée à la notion d'échelle : en dessous d'une certaine échelle d'observation le motif n'est plus observable.



## Chapitre 3

# Image Numérique



---

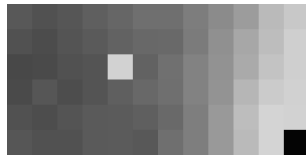
---

## Image Numérique

Une image numérique est un tableau de **pixels**

### Notion de Pixel

**Définition :** Un pixel est l'unité indivisible permettant de coder l'information relative à la luminosité en une certaine position.



Remarques :

- Les pixels sont en général carrés.
- Pixel vient de “picture element”
- Un ensemble de pixels donne une image.

---

---

Image Numérique

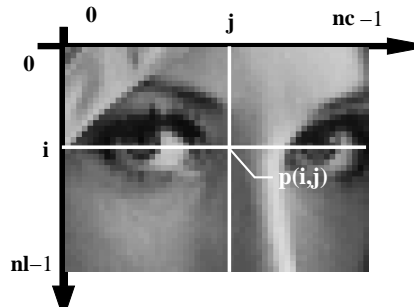
Nr. 1

La représentation sous cette forme d'une image provient directement des capteurs numériques et en particulier des capteurs CCD (*Charge Coupled Device*, Dispositif à Transfert de Charge). C'est la manière la plus usuelle de représenter une image en informatique et c'est celle à laquelle nous serons confrontés en pratique. Ce n'est pas l'unique représentation d'une image en informatique. Nous verrons plus loin d'autres types de représentation d'image (par exemple : image vectorielle, l'image est alors un ensemble d'objets ; c'est le cas du standard Photoshop, par exemple).

**Remarque**

## L'image Numérique

Une image est un tableau de pixels: si le nombre de lignes vaut  $nl$  et le nombre de colonnes  $nc$ :



Un pixel est donc composé :

- de coordonnées  $(i, j)$  permettant de le situer,
- d'une valeur  $v = p(i, j)$  représentant sa couleur .

Image Numérique

Nr. 2

### En pratique

Pour parcourir les pixels d'une image, deux boucles for :

```
int i, int j;
```

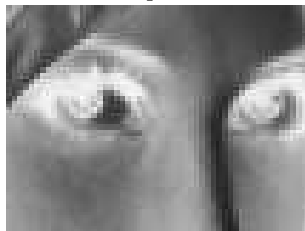
```
for(i=0; i<nl; i++)
  for(j=0; j<nc; j++)
    p(i, j)=255-p(i, j);
```

Cette séquence permet de trouver l'inverse vidéo d'une image en niveau de gris codée sur 8 bits (ce qui veut dire  $p(i, j) \in [0, 255]$ ) :

Avant



Après



---

---

## Notion de résolution

La résolution est donnée par le nombre de pixel  $n_c \times n_l$ .



La résolution correspond à la finesse de la description spatiale de l'image (taille du pixel).

On peut distinguer différentes mesures de résolution spatiale :

- Le nombre total de pixels (photographie numérique)
- La taille d'un pixel (photographie, sciences des capteurs) : par exemple dans le domaine de la **télédétection** qui considère des images satellitaires ou aériennes, la résolution correspond à la taille au sol d'un pixel et est donc décrite en *m* ou en *cm*.
- Le nombre de pixels par *inchs* au carrés (impression. scannage). L'unité est le *dpi*.

En fonction du domaine d'application, la résolution est considérée sous des angles différents.

---

---

## Le codage des couleurs

**Rappel :** La valeur  $v$  d'un pixel  $p$  = intensité lumineuse

**En niveaux de gris**

**binaire:**  $v = 0$  (noir) ou  $v = 1$  (blanc),

**codage 8 bit:**  $v = 0, \dots, 255$  (du plus foncé au plus clair)

**codage 4 bit:**  $v = 0, \dots, 15$ .

**En couleur**

Le cas le plus courant : trois intensités lumineuses,  $vR, vG, vB$  (rouge, vert, bleu).

**codage 24 bit**  $vR = 0, \dots, 255$ ,  $vG = 0, \dots, 255$  et  $vB = 0, \dots, 255$ .

Une image couleur codée sur 24 bits est dite en **vraie couleur**. 24 bits permettent en effet 16.5 millions de couleurs différentes (toutes les combinaisons possibles de rouge, vert et bleu).




**Vraie couleur**

On rappelle que le **bit** est l'unité de stockage informatique. Cela correspond à une unité en binaire. Par exemple si on se donne 8 bits, c'est à dire 8 cases où l'on peut mettre 0 ou 1, on a 255 cas possibles :

**Un bit ?**

| binaire   | décimal | image |
|-----------|---------|-------|
| 0000 0000 | 0       | noir  |
| 0000 0001 | 1       |       |
| 0000 0010 | 2       |       |
| ⋮         | ⋮       | ⋮     |
| 0001 0000 | 16      | gris  |
| ⋮         | ⋮       | ⋮     |
| 1111 1111 | 255     | blanc |

Codage des couleurs, illustration

|             |   |   |  |
|-------------|---|---|--|
| Nb bits     | 8   | 1   | 4  |
| Nb couleurs | 255   | 2   | 16   |
|             |  |  |  |

**Conclusion :** La qualité et la taille mémoire d'une image dépendent directement de la **résolution** et de la **quantification** (nb de bits par pixel).

Image Numérique

Nr. 5

En couleur, on a trois valeurs :  $vR, vG$  et  $vB$ . On décrit donc la valeur d'un pixel par un vecteur :  $(.,.,.)$ . Par exemple, en 24 bits:

Et en couleur?

|        |               |
|--------|---------------|
| Rouge  | (255,0,0)     |
| Vert   | (0,255,0)     |
| Bleu   | (0,0,255)     |
| Jaune  | (255,255,0)   |
| Violet | (255,0,255)   |
| Cyan   | (0,255,255)   |
| Noir   | (0,0,0)       |
| Blanc  | (255,255,255) |

Tous les vecteurs où les trois valeurs sont égales correspondent à l'un des 256 niveaux de gris d'une image 8 bit.

Du gris dans la couleur

$$\forall c \in \{0, \dots, 255\} \quad (c,c,c)_{couleur} \iff c_{gris}$$

---

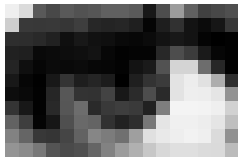
---

## Les formats de stokages

Les formats ont différentes propriétés. Les critères de comparaison élémentaires:

- Type de Format (matriciel ou vectoriel?)
- Nature du codage (binaire ou ASCII?)
- Possesseur du brevet (format libre ou non)
- Nombre de couleurs supportées
- Compression :
  - type de compression
  - avec ou sans perte
- Fonctions bonus: transparence, animation entrelacage, etc...

Exemple d'un format de codage : PGM



- C'est un format ASCII
- C'est un format matriciel.
- Deux parties : une entête et le corps de l'image.

```
P2
# CREATOR: XV Version 3.10a  Rev: 12/29/94
17 11
255
239 219 89 72 89 82 86 96 96 90 24 77 156 97 65 72 79
206 57 56 57 54 62 39 36 38 31 4 30 49 19 11 22 55
46 45 41 33 15 24 17 15 9 7 11 19 24 19 4 0 2
33 31 9 11 13 15 19 7 2 4 4 40 77 60 26 6 2
36 15 6 11 9 11 7 6 3 19 26 82 212 212 109 39 2
24 6 1 24 52 30 37 28 0 31 51 59 227 237 225 174 46 ...
```

|                 |       |
|-----------------|-------|
| Image Numérique | Nr. 7 |
|-----------------|-------|

P2        veut dire : codage en niveau de gris (“grayscale”)

17 11    corresponds à *nc* et *nl*

255      Niveau maximum d’intensité

#        Marque de commentaires

Si on a un format P3 au lieu de P2 on code alors les couleurs : chaque pixel à alors trois valeurs. Par exemple, la même image commence par :

```
P3
#
17 11
255
239 239 239 219 219 219 89 89 89 ....
```

## Principaux formats

On ne détaille que quelques formats usuels, tous matriciels.

| Format   | Nature  | Brevet ? | Nb couleurs | Comp               | Usage   |
|----------|---------|----------|-------------|--------------------|---|
| PGM      | ASCII   | Non      | Variable    | NON !              | Format très simple d'utilisation                          |
| PGMraw   | binaire | Non      | 256         | Non                | idem  |
| BMP      | Binaire | -        | -           | Non                | simple d'utilisation                                      |
| Gif      | Binaire | Unisys   | 256         | sans perte         | Logos, images avec peu de couleurs et des zones homogènes |
| JPEG2000 | Binaire | Non      | 16 M        | avec ou sans perte | Images naturelles   |



---

---

## L'image comme une fonction

On peut voir l'image comme une fonction  $u$  et donc une surface.

$$\begin{array}{rcl} u : I \times J & \rightarrow & V \\ (i,j) & \rightarrow & p(i,j) \end{array}$$

"A un point on associe une valeur d'intensité"

### En discret

$$\begin{array}{ll} I = \{0, \dots, nl - 1\} & \text{et, par exemple :} \\ J = \{0, \dots, nc - 1\} & V = \{0, \dots, 255\} \end{array}$$

### En continu

$$\begin{array}{ll} I = [0, l] & \text{et :} \\ J = [0, L] & V = [0, 1] \end{array}$$

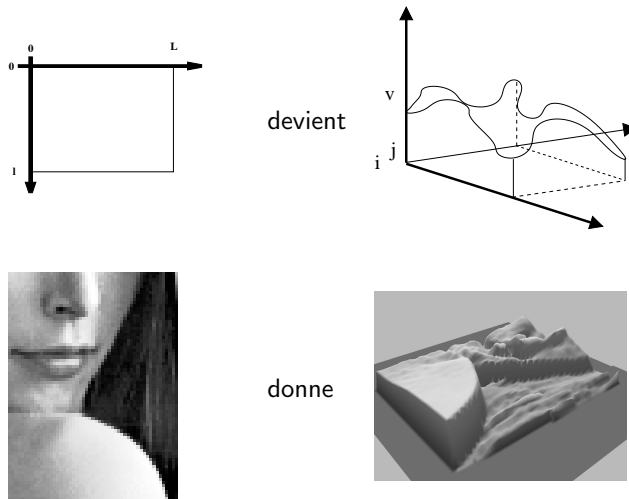
L'avantage d'une telle représentation continue vient de la possibilité de dériver...

---

---

## L'image comme une surface

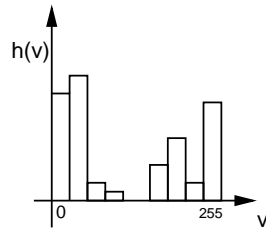
En utilisant la représentation précédente, on peut voir l'image comme une surface.



## Chapitre 4

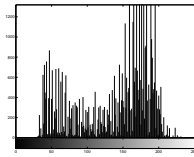
# Histogrammes et contraste

## Histogramme d'une image



On se donne une image 8 bits  $u$ . On peut calculer la présence d'un niveau de gris  $v$  dans l'image :

$$\forall v \in \{0, \dots, 255\} \quad h_u(v) = \text{Nb de pixels d'intensité } v$$



Le tracé de la fonction  $h$  donne donc l'histogramme de l'image.

### En pratique

Pour calculer l'histogramme d'une image 8 bit, on peut utiliser la routine suivante :

```
int h[256];
int i,j,v;

for(v=0;v<256;v++)
    h[v]=0;

for(i=0;i<nl;i++)
    for(j=0;j<nc;j++)
    {
        v=p[i][j];
        h[v]++;
    }
```

---

---

## Densité de probabilité

On peut aussi calculer le **taux de présence** de chaque niveau de gris :

$$P_u(v) = \frac{h_u(v)}{nl.nc}$$

On peut vérifier que :

$$\sum_{v=0}^{v=255} P_u(v) = 1$$

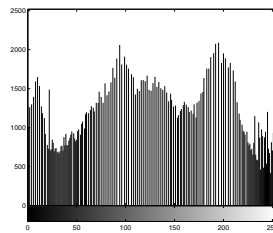
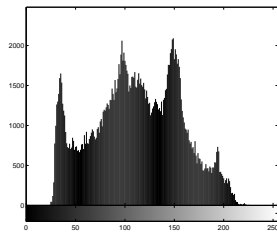
donc  $P_u(\cdot)$  est une densité de probabilité :  $P_u(v)$  est la probabilité de trouver un niveau de gris  $v$  en tirant un pixel au hasard dans l'image  $u$ .

---

---

## Utilité de l'histogramme

Comparons les deux images suivantes et leurs histogrammes :



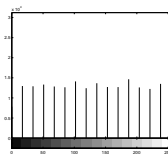
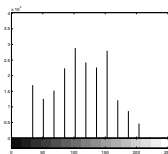
**Conclusion :** en jouant sur l'histogramme, on peut jouer sur le contraste.

## Histogrammes à $n$ niveaux

On peut construire des histogrammes en utilisant moins d'intervalles. Par exemple si on considère 16 niveaux :

$$V_0 = \{0, \dots, 15\} \quad V_i = \{16 * i, \dots, 16(i + 1) - 1\}$$

$$V_{15} = \{240, \dots, 255\}$$



On constate que pour rehausser le contraste on a opéré une **égalisation d'histogramme à 16 niveaux**.

### L'algorithme

1. Initialiser un tableau  $h$  à  $n$  valeurs toutes nulles.
2. Itérer sur les pixels de l'image :
  - (a) lire la valeur  $v$  du pixel courant  $p$ ,
  - (b) déterminer le niveau  $i$  tel que  $v \in V_i$
  - (c) Incrémenter la valeur de l'histogramme correspondant au niveau  $i$  :  $h[i]++$ .
3. Retourner  $h$ .

### Extension

Cette notion d'histogramme peut facilement s'étendre aux images qui ne sont pas entre 0 et 255.

---



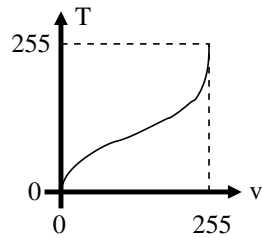
---

## Transformation d'histogrammes

On imagine l'algorithme suivant de transformation des intensités :

- Parcourir les pixels de l'image
  - Pour le pixel courant  $p$  :
    1. lire la valeur de l'intensité  $v$
    2. remplacer  $v$  par  $T(v)$ .

La fonction  $T(\cdot)$  remplace un niveau de gris par un autre. Cette fonction est appelée **transformation d'histogramme**. On la représente par son tracé :



En fonction de l'espace d'arrivée de la fonction  $T(\cdot)$  on obtient des images plus ou moins détaillée. Si  $T$  est à valeur dans :

- $\{0,1\}$ , l'image obtenue est binaire. Par exemple :

$$T : v \rightarrow \begin{cases} 0 & \text{si } v < m \\ 1 & \text{si } v \geq m \end{cases}$$

On peut prendre  $m = 129$ . On peut aussi prendre comme valeur de  $m$  la valeur médiane de l'image  $u$ . Quel est l'intérêt ? (On rappelle que la médiane d'un ensemble de valeurs est obtenue en triant les valeurs dans un ordre croissant et en prenant la valeur du milieu de l'ensemble trié obtenu.)

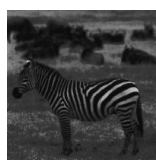
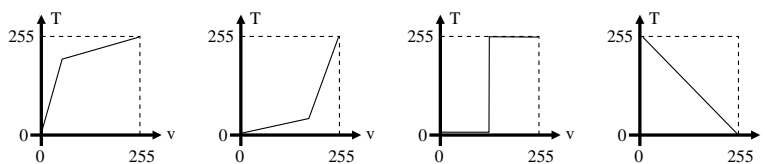
- Si on pose  $V = \{0, \dots, 255\}$ ,  $V' = \{0, \dots, 15\}$ , et  $T : V \times V \times V \rightarrow V' \times V' \times V'$ , alors  $T$  permet de passer d'un codage des couleurs 24 bits à un codage 12 bits. Une telle opération est appelée **quantification des couleurs**.

### Valeurs d'arrivée

### Exercice



## Exemples



Histogrammes et contraste

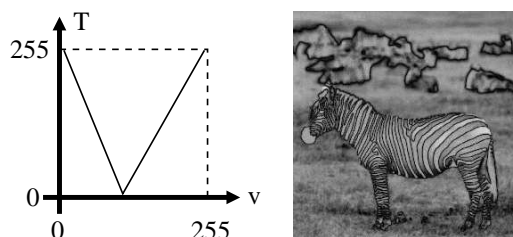
Nr. 6

### Exercice

A quoi ressemblerait le graphique de la transformation d'histogramme la plus simple permettant de passer d'un codage 8 bit à un codage 4 bit ?

### Croissance

On peut appliquer n'importe quelle transformation d'histogramme sur une image. Malheureusement cela peut perturber le sens "visuel" comme par exemple dans le cas de l'inverse vidéo, ou dans le cas suivant où les zébrures du zèbre disparaissent (presque!):

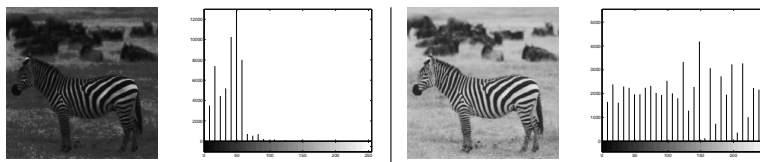


Dans certains cas, on cherche des transformations d'histogrammes. Dans le cas du rehaussement ou de la réduction de **contraste**, il est impératif de ne pas perdre le sens "visuel" de l'image. On adopte alors la contrainte suivante : **La transformation d'histogramme doit être croissante.**

Cette contrainte est primordiale.

## Dynamique de l'image

On appelle **dynamique** d'une image la qualité de l'utilisation des niveaux de gris possibles.

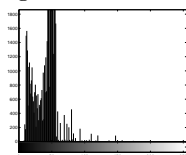


Le rehaussement de contraste revient souvent à une amélioration de la dynamique de l'image. Il s'agit de trouver une transformation d'histogramme qui permette de passer de l'image de gauche à l'image de droite.

Histogrammes et contraste

Nr. 7

L'image de droite ci-dessus a une dynamique qui n'utilise pas toutes les couleurs possibles entre 0 et 255. L'histogramme à 256 valeurs ci-dessous le prouve :

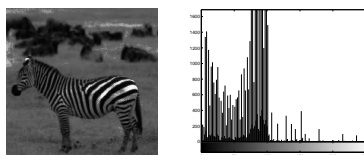


### Normalisation

Proposer une routine de **normalisation**. Il s'agit de trouver une transformation d'histogramme simple et croissante qui réétale les couleurs de l'histogramme entre 0 et 255.

### Exercice

Cette transformation d'histogramme améliore la dynamique de l'image puisque l'image obtenue utilise tous les niveaux de gris. Si l'on trace le graphique de la transformation obtenue, quel est son aspect? (Distinguer trois parties.) Voici le résultat obtenu :



Quelle est la différence entre cet histogramme et celui du transparent, en haut à droite?

---



---

## Égalisation d'histogrammes

Soit une image  $u$ . On rappelle que:  $P_u(v) = \frac{h_u(v)}{nl.nc}$   
(histogramme normalisé.)

On considère la **fonction de cumul** :

$$\forall k \in \{1, \dots, 255\} \quad \phi_u(k) = \sum_{v=0}^{v=k} P_u(v)$$

$\phi_u(k)$  représente le taux de pixels dont le niveau de gris  $v$  est inférieur à  $k$  dans l'image  $u$ .

Si on applique la transformation d'histogramme  $T(v) = 255 * \phi_u(v)$  à  $u$  on trouve l'**image normalisée**  $u'$  :

$$\forall v \in \{0, \dots, 255\} \quad P_{u'}(v) \simeq 1/256$$

C'est à dire que dans l'image obtenue le nombre d'occurrences de chaque niveau de gris est à peu près égal aux autres.

Démontrer que  $T$  est croissante. (Calculer  $T(k+1) - T(k)$ ).

**Exercice 1**

Vérifier  $T(255) = 255$ . On note  $v_{min}$  le niveau minimal de  $u$  pour lequel il y a au moins un pixel:  $v_{min} = \min\{v \in u \text{ tel que } h_u(v) > 0\}$ . Vérifier que  $T(0) = T(1) = \dots = T(v_{min}) = 0$ .

**Exercice 2**

Donner l'expression en fonction de  $\phi$  du nombre de pixels d'une image dont la valeur est comprise entre  $v_m$  et  $v_M$ :  $v \in [v_m, v_M]$ .

**Exercice 3**

Soit une image  $u$ . On cherche une transformation d'histogramme  $T$

- croissante
- qui donne comme résultat un histogramme "uniforme", c'est à dire que tous les niveaux de gris apparaissent avec la même fréquence, à peu près.

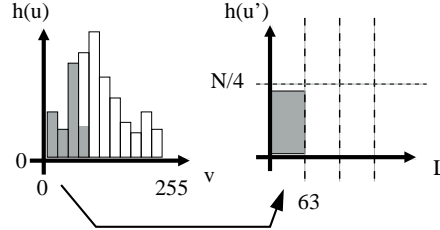
**Démonstration  
grossière**

Nous allons vérifier que la transformation  $T(.) = 255 * \phi_u(.)$  vérifie les deux propriétés. (Croissance en exercice.)

Imaginons que l'on regarde l'histogramme a  $L$  niveaux de l'image d'arrivée

$u'$ .  $L$  pourrait valoir 256, mais par simplicité, posons  $L = 4$ . On cherche à obtenir une image  $u'$  telle que l'histogramme a  $L$  niveaux soit uniforme, c'est à dire qu'il y ait pour tous les niveaux le même nombre de pixels. Si il y a  $N$  pixels dans l'image  $u$ , cela revient à dire que l'on veut pour le premier niveau  $([0,63])$   $N/4$  pixels.

Il va donc falloir que les pixels dans l'image obtenue  $u'$  dont la valeur est plus petite que 63 proviennent des  $N/4$  premiers pixels de l'histogramme de  $u$ . C'est ce qu'illustre la figure suivante:



Pour trouver le premier quart des pixels de  $u$ , on note  $l$  le niveau tel que  $\phi_u(l) = 1/4$ . Si on applique alors la transformations proposée :

$$\begin{aligned} T : \{0, \dots, 255\} &\rightarrow \{0, \dots, 255\} \\ v &\rightarrow 255 * \phi_u(v) \end{aligned}$$

on applique alors bien une fonction croissante, qui vérifie :

$$\forall v \leq l \quad T(v) \leq T(l) = 255 * \phi(l) \simeq 63$$

C'est à dire qu'elle place bien le premier quart des pixels de  $u$  dans le premier niveau de  $u'$ .

D'autres transformations vérifiant la même propriété auraient pu être utilisées : par exemple, en décomposant  $T$  en quatre parties linéaires correspondant aux quatre niveaux on obtiendrait pour le premier niveau :

**Pourtant ?**

$$\begin{aligned} T_1 : \{0, \dots, l\} &\rightarrow \{0, \dots, 63\} \\ v &\rightarrow 63 * \frac{v}{l} \end{aligned}$$

Cette transformation est linéaire entre 0 et  $l$ , croissante et place bien un quart des pixels de  $u$  dans le premier niveau de  $u'$ . Elle convient donc pour égaliser l'histogramme à 4 niveaux.

Mais si l'on change  $L$ , par exemple  $L = 16$ , la transformation linéaire ci-dessus ne remplira plus les conditions nécessaires, alors que la transformation  $T$  proposée les vérifie.  $T$  est donc la transformation qui permet d'égaliser tous les histogrammes à  $L$  niveaux !

## Chapitre 5

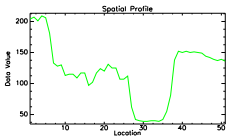
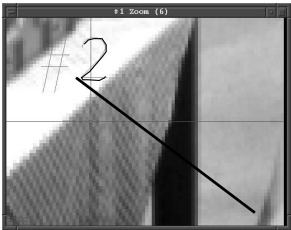
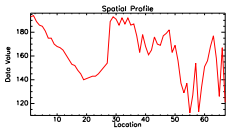
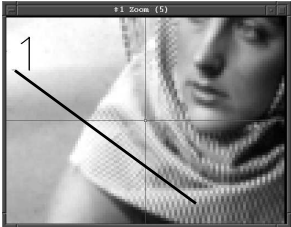
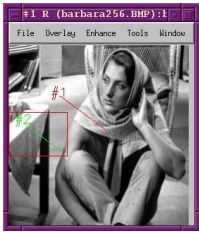
# Détection de contours

Introduction

Dans ce chapitre, l'image est considérée comme une fonction :

$$\begin{aligned} u : I \times J &\rightarrow V \\ (x,y) &\rightarrow u(x,y) \end{aligned}$$

A un point de l'image, cette fonction associe une valeur.

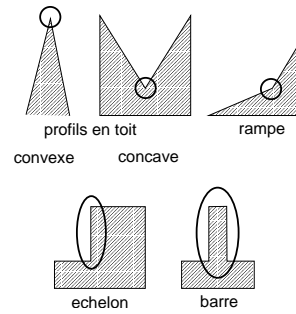


---

---

## Discontinuités

Les différents types de discontinuités intéressantes dans une image :



Ces différentes discontinuités correspondent à des transitions entre deux zones différentes ou la coupure d'une zone par une ligne.

Pour détecter ces discontinuités, il va falloir utiliser la dérivée d'un profil.

Les discontinuités sont présentes quand la dérivée change rapidement autour du point d'intérêt.

---

---

## Différents types de contours

- **l'échelon** correspond au contour idéal et à une discontinuité franche.
- **la rampe** correspond au dégradé.
- **les profils en toit** correspondent à des lignes blanches (convexe) ou noires (concave).
- **la barre** correspond à une fine bande séparant deux zones.

Retrouver dans le premier transparent : une rampe, trois échelons, une barre inversée.

**Exercice**

Constater la différence entre les profils théoriques et la réalité des images.



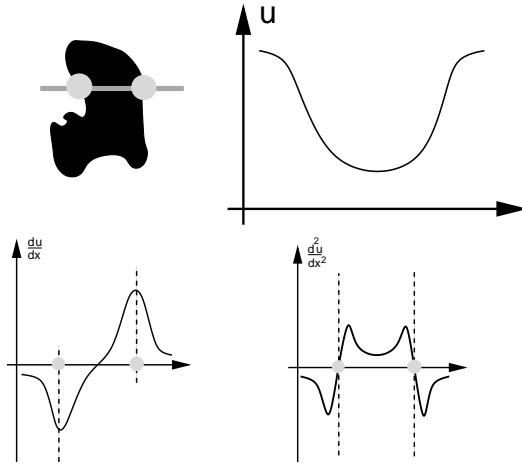
---

---

## Détection d'une discontinuité

Repose sur l'utilisation de la notion de dérivée : il ya deux possibilités :

- Recherche des maxima de la dérivée
- Recherche des passages par 0 de la dérivée seconde.



---

---

## Gradient

Le gradient en un point est un vecteur (de  $\mathbb{R}^2$ ) dont les composantes  $(\delta_x, \delta_y)$  mesurent à quelle vitesse la valeur des pixels évolue dans chacune des directions  $x$  et  $y$ , autour du point d'intérêt.

$$\begin{aligned}\delta_x &= \frac{\partial u}{\partial x} \simeq \frac{u(x + d_x, y) - u(x, y)}{d_x} \\ \delta_y &= \frac{\partial u}{\partial y} \simeq \frac{u(x, y + d_y) - u(x, y)}{d_y}\end{aligned}$$

avec  $d_x$  et  $d_y$  qui mesurent des distances dans les directions  $x$  et  $y$  respectivement.

**Le gradient permet d'obtenir la dérivée dans n'importe quelle direction.**

Si l'on cherche à exprimer en fonction de  $\delta_x$  et de  $\delta_y$  la valeur du gradient dans la direction donnée par  $\theta$ , il faut calculer le produit scalaire :

$$\delta_\theta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} \cdot \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix}$$

**Rappel**

---



---

## Dans le cas discret

$d_x$  et  $d_y$  mesurent des distances en pixels. On peut prendre  $d_x = d_y = 1$ .

$$\begin{aligned}\delta_x &= u(i+1, j) - u(i, j) \\ \delta_y &= u(i, j+1) - u(i, j)\end{aligned}$$

### Discontinuités

Pour détecter la présence ou non d'une discontinuité de gradient, on peut calculer :

- **La norme du gradient**  $G = \sqrt{\delta_x^2 + \delta_y^2}$
- **Sa direction**  $\theta = \arctan(\delta_y/\delta_x)$ . La direction du gradient est la direction dans laquelle la dérivée est la plus grande.

---



---

Détection de contours

Nr. 6

La présence d'une discontinuité est assurée si le gradient au point considéré est suffisamment fort.:

$$\|G\| \geq \text{seuil}$$

La direction du gradient est la direction orthogonale à la ligne de niveau. La ligne de niveau est la ligne telle que la variation de l'intensité est nulle dans sa direction. La direction est donc donnée par :

$$\begin{aligned}\delta_{\theta+\frac{\pi}{2}} &= 0 \\ \delta_x \cos(\theta + \frac{\pi}{2}) + \delta_y \sin(\theta + \frac{\pi}{2}) &= 0 \\ -\delta_x \sin(\theta) + \delta_y \cos(\theta) &= 0 \\ \frac{\delta_y}{\delta_x} - \frac{\sin(\theta)}{\cos(\theta)} &= 0 \\ \tan(\theta) &= \frac{\delta_y}{\delta_x}\end{aligned}$$

**Présence de la  
discontinuité**

**Direction**

---

---

## Commentaires

- La détection des contours dépend du type d'image considéré. **Il n'y a pas de détecteur de contours universel.**
- En fonction de l'objectif, on optera pour telle ou telle méthode de détection des contours.
- Lors de la conception d'un détecteur de contours, il convient de toujours percevoir l'image comme un paysage dont la topographie constitue l'information importante.

---



---

## Masques de convolutions

A droite, un masque  $m$ , à gauche un morceau de l'image  $u$  :

|   |   |   |
|---|---|---|
| a | b | c |
| d | e | f |
| g | h | i |

|  |           |         |           |  |
|--|-----------|---------|-----------|--|
|  | (i-1,j-1) | (i-1,j) | (i-1,i+1) |  |
|  | (i,j-1)   | (i,j)   | (i,i+1)   |  |
|  | (i+1,j-1) | (i+1,j) | (i+1,i+1) |  |

**Convolver** l'image  $u$  par le masque  $m$  veut dire que l'on calcule l'image  $u'$  en appliquant la procédure suivante :

Pour tout pixel de l'image :

$$\begin{aligned}
 u'(i,j) &= l.u(i-1,j-1) + h.u(i-1,j) + g.u(i-1,j+1) \\
 &+ f.u(i,j-1) + e.u(i,j) + d.u(i,j+1) \\
 &+ c.u(i+1,j-1) + b.u(i+1,j) + a.u(i+1,j+1)
 \end{aligned}$$

---



---

Détection de contours

Nr. 8

### Exercices

Écrire un masque qui permet de calculer  $\delta_x$  et  $\delta_y$  par convolution. Écrire un autre masque qui permet de calculer la valeur moyenne des niveaux de gris dans un voisinage  $3 \times 3$ .

---



---

## Masques de gradients

Par exemple pour calculer les gradients on peut prendre les masques suivants (connus sous le nom d'opérateurs de Roberts)

| $\delta_x$ |   |    | $\delta_y$ |    |   |
|------------|---|----|------------|----|---|
| 0          | 0 | 0  | 0          | 0  | 0 |
| 0          | 1 | -1 | 0          | 1  | 0 |
| 0          | 0 | 0  | 0          | -1 | 0 |

Il existe une variante :

| $\delta_1$ |   |    | $\delta_2$ |    |   |
|------------|---|----|------------|----|---|
| 0          | 0 | 0  | 0          | 0  | 0 |
| 0          | 1 | 0  | 0          | 0  | 1 |
| 0          | 0 | -1 | 0          | -1 | 0 |

Ces opérateurs calculent les gradients dans les directions +45 degrés et -45 degrés.

Cependant : ces opérateurs sont très simples. Il existe d'autres opérateurs qui utilisent totalement le masque 3\*3.

Le traitement des pixels du bord est toujours en problème lorsque l'on utilise des masques de convolution. Dans le cas du calcul des gradients, il suffit de mettre les gradients dont le calcul nécessite des pixels en dehors de l'image à zéro.

---

---

## Les opérateurs de Sobel

Les avantages provenant de l'utilisation d'un masque 3\*3 :

- Les erreurs sont moins grandes parce que les comportements aberrants sont moyennés sur une plus grande fenêtre,
- l'opérateur est centré, il n'y a pas de problème de parité.

| $\delta_x$ |   |   | $\delta_y$ |    |    |
|------------|---|---|------------|----|----|
| -1         | 0 | 1 | 1          | 2  | 1  |
| -2         | 0 | 2 | 0          | 0  | 0  |
| -1         | 0 | 1 | -1         | -2 | -1 |

---

---

## Maxima du gradient

Nous savons calculer :

- les composantes du gradients
- la norme du gradient
- la direction du gradient

en un point de l'image.

Nous devons trouver les points tels que **le gradient soit maximal localement** : pour cela, deux méthodes :

- seuillage du gradient
- recherche du maximum local



---

---

## Seuillage du gradient

Une fois la norme du gradient calculée en tous les points de l'image, on sélectionne les pixels tels que :

$$G = \sqrt{\delta_x^2 + \delta_y^2} > G_{\min}$$

Par exemple :



Gradient



Gradient seuillé ( $|G| > G_{\min}$ )

Cette méthode repose sur l'hypothèse que les maxima du gradients correspondent a peu près au gradients plus grands que le seuil.

---

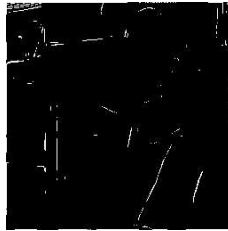
---

## Problème du seuillage

Le problème des méthodes par seuillage vient du seuil à fixer. Par exemple :



Seuil faible



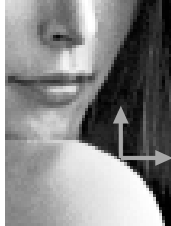
Seuil grand

---

---

## Recherche d'un maximum local

Pour éviter ce problème, il existe une autre méthode qui repose sur la détection d'un maximum local dans la direction du gradient.



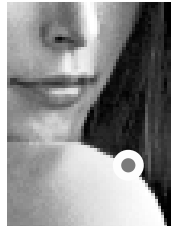
Calcul de gradient  $\delta_x, \delta_y$



Recherche de la direction du gradient :  $\arctan(\delta_y/\delta_x)$



Dans la direction : recherche du maximum du gradient



Si le maximum coïncide avec le point considéré : on a une discontinuité.

---

---

## Contours et Laplacien

Une autre méthode de détection des discontinuités repose sur l'utilisation du Laplacien :

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

On cherche ensuite les endroits où le Laplacien passe par 0.

---



---

## Calcul du Laplacien

On peut utiliser l'un des deux masques suivants :

$$M_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad M_2 = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$M_1$  est obtenu par composition des masques de dérivation,  $M_2$  est une estimation directe d'un masque de Laplacien.

L'origine de ces masques : le développement de Taylor de la fonction  $u$  :

$$u(x+l, y+h) = u(x, y) + lu_x(x, y) + hu_y(x, y) + \frac{l^2}{2}u_{xx} + \frac{h^2}{2}u_{yy} + hlu_{xy} + o(h^2, l^2)$$

$$\text{où } u_x = \frac{\partial u}{\partial x}, u_{xx} = \frac{\partial^2 u}{\partial x^2}.$$

En écrivant les développements pour les voisins de  $u(i, j)$  vérifier que les différents masques (Sobel, Robert, Laplacien) donnent bien les valeurs attendues.

**Exercice**

---

---

## Détection du passage par 0

On commence par calculer la valeur du Laplacien sur toute l'image. On applique ensuite la règle suivante :

|            |            |            |
|------------|------------|------------|
|            | $\Delta_1$ |            |
| $\Delta_2$ | $\Delta$   | $\Delta_3$ |
|            | $\Delta_4$ |            |

Si  $\Delta < 0$  et l'un des autres  $\Delta_i \geq 0$  ou  $\Delta > 0$  et l'un des autres  $\Delta_i \leq 0$  on considère qu'il y a un changement de signe.

Souvent on ne considère que les points où le Laplacien  $\Delta$  est suffisamment grand.

---

---

## Problème du bruit

Si l'image est trop bruitée, la détection par le Laplacien n'est pas très bonne. Il faut appliquer un filtre moyen auparavant (cf chapitre sur le filtrage)



---

---

## Gradients et contours

Nous avons vu comment détecter la présence ou non d'une discontinuité en un pixel. Si l'on veut numérotter les pixels :

On peut utiliser :

- le **détecteur** (par exemple seuil sur la norme du gradient, maximum local ou passage par 0 du Laplacien)
- et sa **direction** si on cherche à **numéroter les contours**. Deux pixels voisins où l'on a détecté la présence de forts gradients appartiennent au même contour si les gradients ont même direction.

Par exemple :



Gradient



Gradient seuillé ( $|G| > G_{min}$ )



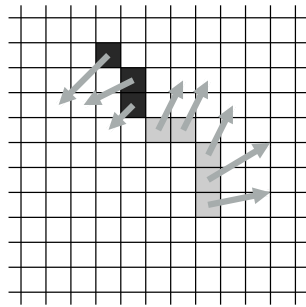
---

## Un algorithme de numérotation des contours (1)

**Test de présence :** en un pixel  $G > G_{min}$

**Contrainte de similarité entre deux pixels**  
 $p$  et  $p'$ : Les gradients sont orientés de la même  
 manière :

$$|\theta(p) - \theta(p')| \leq \Delta\theta$$



Par exemple sur l'image : 2 contours

---

---

## Un algorithme de numérotation des contours (2)

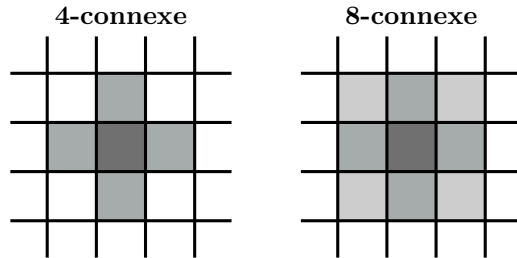
1.  $C = 0$
2. Choisir un pixel  $p$  n'ayant pas encore été attribué à un contour tel que ( $G(p) > G_{min}$ )
3. Attribuer au pixel  $p$  l'étiquette  $C$ .
4. Regarder les pixels voisins de  $p$ . Si un voisin  $p'$  de  $p$  vérifie :
  - le gradient  $G(p')$  est plus grand que  $G_{min}$ ,
  - ainsi que la contrainte de similarité entre  $p$  et  $p'$alors  $p'$  appartient au même contour :
  - (a) on lui attribue l'étiquette  $C$
  - (b) on explore ses voisins (*voisinage + diffusion*).
5. Lorsque tout le voisinage pertinent a été exploré :  $C = C + 1$ , retour en 2 sauf si tous les pixels tels que  $G > G_{min}$  ont été attribués à un contour.

---

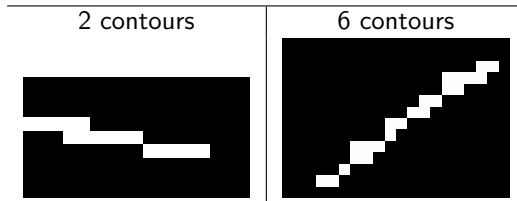
---

## Notion de voisinage d'un pixel

Il y a deux types de voisinage simple:



Deux exemples : un seul contour en 8-connexité mais en 4-connexité :

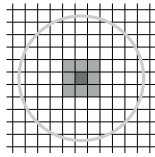


---

---

## Mécanisme de diffusion(1)

**But :** On veut explorer un objet (par exemple un contour) en connaissant : un point d'origine et une règle d'appartenance.



Par exemple : on veut explorer l'intérieur d'un cercle dont on connaît t :

- le centre  $p_0 = \Omega$ ,
- la règle  $R : d(p, \Omega) \leq \rho$

---



---

## Mécanisme de diffusion(2)

Structures de donnée annexe :

- une image d'étiquette,
- une pile de pixels  $P$  :

1. Initialiser les étiquettes de tous les pixels à 0.  
empiler  $p_0$  dans  $P$  :
2. TANT QUE ( $P$  non vide)
  - (a) dépiler le pixel  $p$  de la pile  $P$ ,
  - (b) mettre l'étiquette de  $p$  à 1,
  - (c) (\*)
  - (d) empiler dans  $P$  les voisins de  $p$ :
    - dont l'étiquette vaut 0,
    - qui vérifient la règle d'appartenance.

Le traitement de  $p$  à lieu dans (\*)...

---



---

Détection de contours

Nr. 24

On appelle **composante connexe** un ensemble de pixels tel qu'entre tout couple de pixels de cet ensemble il existe un chemin de pixels voisins les reliant.

**Définition**

Montrer que l'algorithme termine.

**Exercice 1**

Montrer que l'algorithme explore tous les pixels de la composante connexe contenant  $p_0$ . (On appelle "explorer  $p$ " l'étape (\*))

**Exercice 2**

Montrer qu'il n'explore qu'une composante connexe.

**Exercice 3**

Vérifier que chaque pixel exploré ne l'est qu'une seule fois. Cela est très important, puisque cela permet par exemple de compter le nombre de pixels d'une composante connexe.

**Exercice 4**

La condition d'appartenance additionnée à la notion de voisinage permet de définir une notion de voisinage plus complexe.

**Remarque**

---

---

### Un algorithme de numérotation des contours (3)

1. Mettre toutes les étiquettes à  $-1$
2.  $C = 0$
3. Choisir un pixel  $p_0$  n'ayant pas encore été attribué à un contour tel que  $(G(p) > G_{min})$ .
4. Empiler  $p_0$  dans  $P$ .
5. TANT QUE (P non vide)
  - (a) dépiler le pixel  $p$  de la pile  $P$ ,
  - (b) mettre l'étiquette de  $p$  à  $C$ ,
  - (c) empiler dans  $P$  les voisins  $p'$  de  $p$ :
    - dont l'étiquette vaut  $-1$ ,
    - tel que le gradient  $G(p') > G_{min}$ ,
    - qui vérifient la condition de similarité avec  $p$  sur les angles.
6.  $C = C + 1$ ,
7. Retour en 3 sauf si tous les pixels tels que  $G > G_{min}$  ont été attribués à un contour.

## Chapitre 6

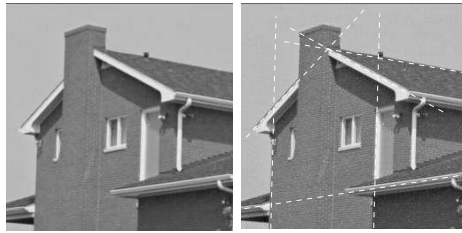
# Détection de Droites (Hough)

---

---

## Détection de droites

Le but est de trouver des droites dans une image. Par exemple :



Cette partie s'inscrit donc dans un contexte de détection d'objet.

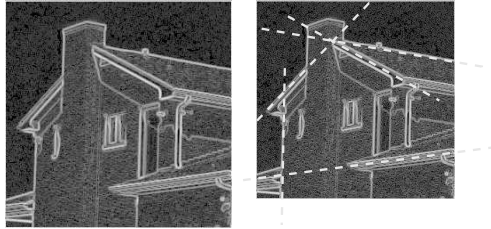


---

---

## Méthode :

On va utiliser une détection de gradient :



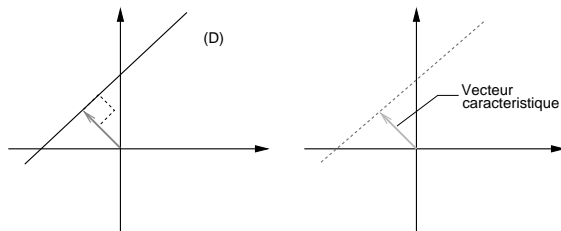
**Le problème :** comment identifier les droites dans l'image du gradient ?

---

---

## Comment caractériser une droite?

Le plus simple:  $y = ax + b$ . Ce n'est pas le plus utile pour nous :



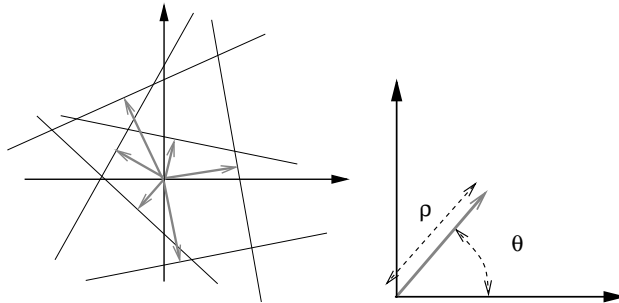
On va utiliser le vecteur caractéristique ci-dessus: c'est le vecteur normal à la droite mesurant la distance entre la droite et l'origine.

---

---

## Paramétrisation de la droite

On considère la paramétrisation suivante, qui permet de considérer tous les cas de droites du plan.



On peut faire les remarques suivantes :

Une droite est caractérisée par :

- la norme  $\rho$ .
- l'orientation  $\theta$

du vecteur caractéristique.

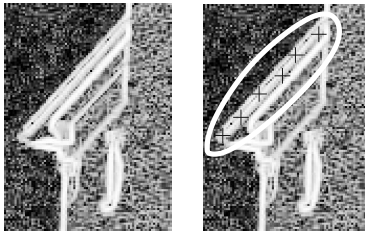
toutes les droites sont caractérisable par un couple.

---

---

## Norme du gradient et droite

**Question :** qu'est ce qui donne l'impression d'une droite sur l'image des contours ?



Une droite de l'image semble être caractérisée par l'alignement des discontinuités.

**Conclusion :** Une solution pour détecter les droites est de détecter les alignements de points de discontinuités.

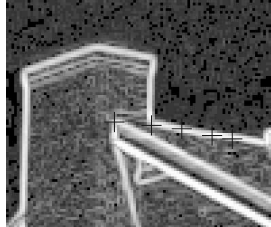
Mais ce n'est pas suffisant...

---

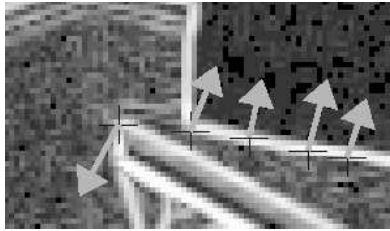
---

## Direction du gradient et droite

Sur l'exemple suivant on remarque qu'il n'est pas suffisant de détecter les alignements de discontinuités fortes :



Il faut considérer les directions des contours :



---

---

## Rappel Contours et gradient

Le gradient d'une image  $\nabla u = (\delta_x, \delta_y)^T$  en un point donne la variation de l'image dans toutes les directions au travers d'un produit scalaire :

$$\Delta_\theta u \simeq \nabla u \cdot v_\theta$$

où  $v = (\cos(\theta), \sin(\theta))^T$  est le vecteur unitaire de la direction  $\theta$ .

La **direction du gradient** est la direction de variation maximale de l'image :

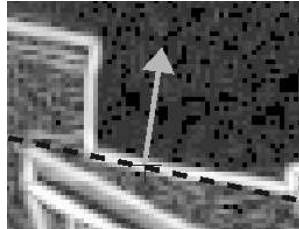
$$\theta = \arctan \frac{\delta_y}{\delta_x} \quad (6.1)$$

---

---

## Gradient et droites

Lorsque l'on calcule la direction du gradient, cela donne la normale à la droite plausible.



**Conclusion :** chaque point tel que le norme du gradient  $G = \sqrt{\delta_x^2 + \delta_y^2}$  soit plus grande qu'un certain seuil  $G_{\min}$  peut être vu comme une hypothèse de droite de direction normale  $\theta$ , la direction du gradient.

Et  $\rho$ ? On vérifie que :

$$\rho = x \cos(\theta) + y \sin(\theta)$$

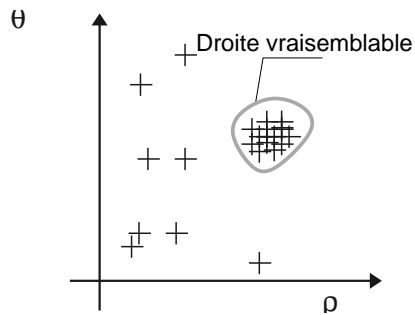
---

---

## Mécanisme de vote

La transformée de Hough repose sur le mécanisme de vote suivant :

1. On parcourt les pixels de l'image,
2. On calcule en chaque point le gradient, sa norme et sa direction,
3. On en déduit l'hypothèse de droite  $(\rho, \theta)$ ,
4. Si le gradient est suffisamment grand ( $G \geq G_{\min}$ ), on vote, c'est à dire que l'on rajoute un point dans l'espace des droites :





---

---

## Trouver les droites

Il faut ensuite trouver les amas de points dans l'espace  $(\rho, \theta)$ .

Cela commence par un lissage. Ensuite : extraction des maxima locaux ou seuillage.

---

---

## Identification des droites

Une fois que l'on a identifié les maximaux locaux, on en déduit les droites présentes dans l'image.

---

---

## En pratique

On applique les étapes suivantes :

1. Discrétiser l'espace des paramètres  $\rho, \theta$  en  $\rho_1, \dots, \rho_N$  et  $\theta_1, \dots, \theta_M$ .
2. Parcourir les pixels, pour chacun calculer la norme et la direction du gradient. Si le gradient est suffisamment grand, calculer le  $\rho$  et le  $\theta$  correspondant. Dédire le couple  $\rho_i$  et  $\theta_j$  correspondant dans l'espace des paramètres discrétisés,
3. Lisser et optimiser : c'est à dire trouver les couples  $(\rho_{i_k}, \theta_{j_k})$  pour les quels le nombre de votes a été suffisant.

---

---

## Conclusion

La transformée de Hough est un outil efficace pour trouver les droites dans une image.

Il existe d'autres transformées de Hough, dites transformées de Hough généralisées pour extraire d'autres formes comme les cercles ou les ellipses.

L'extraction des droites est souvent une première étape de la détection d'objets.

---

---

## Exercice

Proposer un accumulateur qui permet de trouver des cercles de tailles fixées dans des images en niveau de gris.

## Chapitre 7

# Bruit et Filtrage

---

---

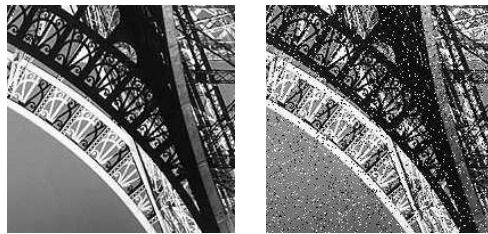
## Le bruit

Le bruit est une altération de l'image : toute l'information pertinente dans l'image n'est pas simplement accessible.

### Origine :

- Acquisition
  - illumination
  - perturbations
- Transmission
- Stockage
  - échantillonnage,
  - quantification

### Exemple :



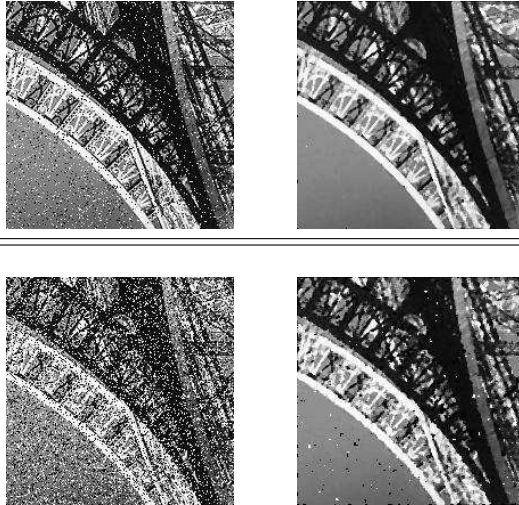
---

---

## Le filtrage

Consiste à retrouver le maximum d'information sous l'image bruitée.

**Exemple :**





---

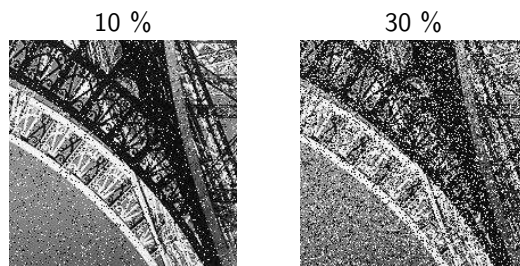


---

## Bruit Sel et poivre

**Définition :** Le bruit “Sel et poivre à  $x\%$ ” est obtenu en remplaçant

- $x/2\%$  des pixels de l'image par des pixels blancs
- et  $x/2\%$  des pixels de l'image par des pixels noirs.




---



---

Bruit et Filtrage

Nr. 3

En C, la fonction `int rand()` permet de tirer des entiers au hasard entre 0 et `RAND_MAX` inclus (`RAND_MAX` est une variable globale). Pour générer un entier aléatoire dans  $\{0, \dots, M-1\}$  on utilisera la routine suivante :

```
int r, int res;
int M=nc;

r=rand();
res=(int) (r/(RAND_MAX+1))*M;
```

**Aléa**

Ne JAMAIS utiliser le générateur pseudo aléatoire `rand()` et l'opérateur modulo `%` : les bits de poids faible sont les moins aléatoires...

**Avertissement**

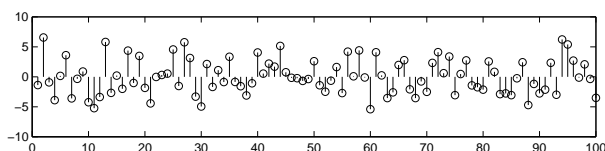
Proposer une routine qui applique le bruit de sel et poivre à une image (par exemple  $nc = nl = 256$  et  $x = 25\%$ ). Tenir compte du fait que l'on veut *exactement*  $x\%$  de pixels remplacés (utiliser une image d'étiquette).

**Exercice**

Vérifier que la routine est en temps linéaire avec  $x$  (examiner le cas limite  $x = 99\%$ )...

## Bruit Gaussien (1)

On génère au hasard  
une suite de valeurs  
réelles  $v_0, \dots, v_N$  :



On ajoute successivement à chaque pixel de l'image une valeur  $v_i$  :

$$u(0,0) = u(0,0) + v_0 \quad (7.1)$$

$$u(0,1) = u(0,1) + v_1 \quad (7.2)$$

$$\vdots \quad (7.3)$$

$$u(nl,nc) = u(nl,nc) + v_N \quad (7.4)$$

Le bruit est dit Gaussien parce que la génération aléatoire est faite en suivant une **Loi gaussienne** (de moyenne  $m = 0$ ).

### Loi Gaussienne

La densité de probabilité d'une loi gaussienne est donnée par :

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-m)^2}{\sigma^2}\right)$$

$m$  et  $\sigma$  sont les deux paramètres de la loi.

- $m$  correspond à la **moyenne** théorique  $m = \mathbb{E}(X)$ . Dans le cadre du transparent on a  $m = 0$ . On trouverait par exemple :

$$0 = m \simeq \frac{1}{N+1} \sum_{i=0}^N v_i$$

- $\sigma^2$  est la **variance**, et  $\sigma$  l'écart type donné par :  $\sigma^2 = \mathbb{E}(X - m)^2 = \mathbb{E}(X - \mathbb{E}(X))^2$ .

Dans le cas pratique cela donne :

$$\sigma^2 \simeq \frac{1}{N+1} \sum_{i=0}^N (v_i - m)^2 = \frac{1}{N+1} \sum_{i=0}^N v_i^2$$

### Rappel

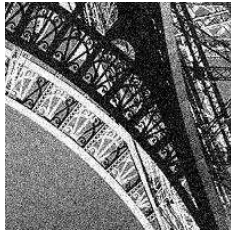
L'écart type permet de faire la différence entre les deux cas de figures suivants : les moyennes des résultats à l'examen de deux classes valent 10/20, mais dans une des classes presque toutes les notes sont entre 9 et 11 ( $\sigma$  faible), tandis que dans l'autre il y a beaucoup de 3 et beaucoup de 17 ( $\sigma$  fort). La variance est en effet **la moyenne du carré de l'écart à la moyenne**.

---

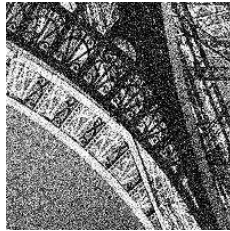
---

## Bruit Gaussien (2)

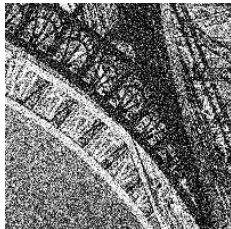
Le bruit Gaussien est caractérisé par sa variance (ou de manière équivalente par l'écart type).



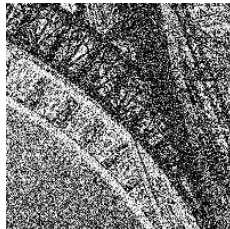
$$\sigma = 0.01 * 256$$



$$\sigma = 0.05 * 256$$



$$\sigma = 0.1 * 256$$



$$\sigma = 0.20 * 256$$

---

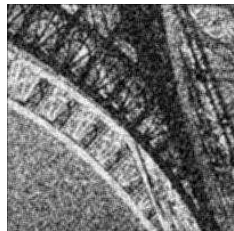
---

## Filtrage Linéaire

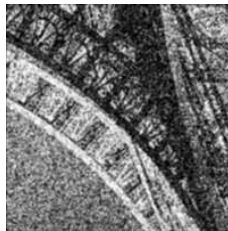
Le filtrage linéaire repose sur l'utilisation de masques de convolution (cf cours sur les contours), comme par exemple :

$$M_1 = \frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \quad M_2 = \frac{1}{16} \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

En appliquant  $M_1$  et  $M_2$  sur l'image  $\sigma = 0.1 * 256$  de la page précédente :



Avec  $M_1$



Avec  $M_2$

---

---

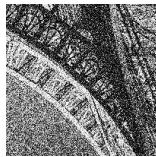
## Filtrage linéaire itéré

Pour améliorer les filtres linéaires on peut :

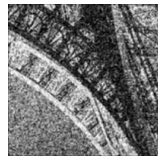
- chercher les meilleures valeurs du masque,
- augmenter la taille du masque,
- Appliquer plusieurs fois de suite le même masque.

**Remarque :** On peut trouver un lien entre ces trois points.

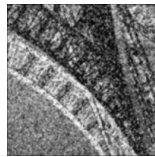
**Exemple :** Si on applique  $M_2$  :



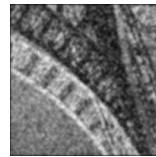
0 fois



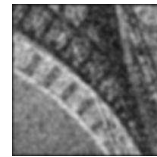
1 fois



2 fois



5 fois



10 fois

---

---

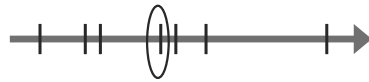
## Filtrage Non linéaire

Le filtre non linéaire le plus utilisé est le **filtre médian**. Ce filtre remplace la valeur d'un pixel par la médiane des valeurs de ses voisins.

**Rappel:** La médiane d'un ensemble de valeur, est la valeur " du milieu".

**Algo :**

1. Chercher les valeurs des voisins du pixel courant,
2. Ordonner ces valeurs (**tri**),
3. Sélectionner celle du milieu.



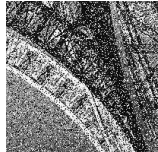
---

---

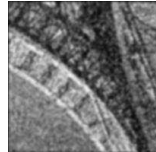
## Filtrage Non linéaire (2)

Les avantages du filtre médian :

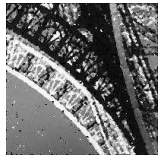
- il préserve les contours,
- il est plus robuste à certain bruits (par exemple, le bruit de sel et poivre.)



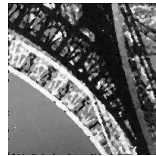
Sel et Poivre 30 %



$M_2$  5 fois



Médian



Médian 2 fois

## Chapitre 8

# Morphologie Mathématique



---

---

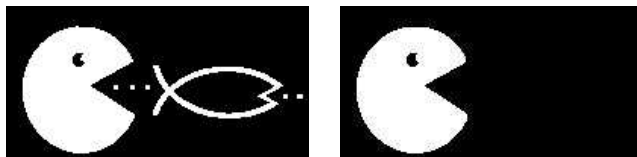
## Introduction

La morphologie mathématique a été développée à partir des années 70.

Elle s'énonce et se comprend plus aisément sur des **images binaires**.

Voici deux exemples de ce que l'on peut faire avec cet outil :

Suppression des structures fines :



Comblement des trous :

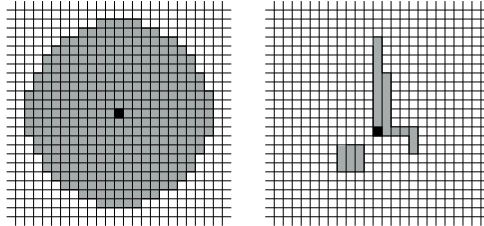


---

---

## Élément structurant

La morphologie mathématique repose sur l'utilisation d'un élément structurant.



Un élément structurant est composé :

- d'un pixel central (en noir)
- d'un ensemble de pixels (en gris)

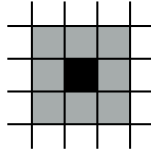
### Parcours de l'image

Les algorithmes de morphologie mathématique parcourent l'image :

- en chaque pixel de départ  $(u(i,j))$  on place l'élément structurant centré sur le pixel noir,
- un test est réalisé pour déterminer la couleur du pixel d'arrivée  $u'(i,j)$ .

## Dilatation

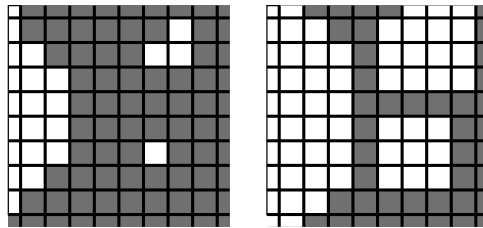
On se donne une image **binaire**  $u$   
et un élément structurant :



**Algo :** Parcourir les pixels de l'image  $u$ ,

pour chaque pixel  $u(i,j)$ :

1. Centrer l'élément structurant sur ce pixel,
2. Considérer les voisins du pixel dans l'élément structurant (gris + noir)
3. Si l'un de ces pixels est en blanc, mettre  $u'(i,j)$  en blanc.



Dans le cas binaire, on peut assimiler  $u$  et l'ensemble des pixels blancs de l'image.

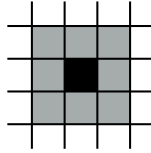
Montrer que l'image  $u'$  contient l'image  $u$ . Commencer par vérifier l'équivalence des trois assertions :

**Exercice**

- “ l'image  $u'$  contient l'image  $u$ ”,
- “ l'ensemble des pixels blancs de  $u'$  contient l'ensemble des pixels blancs de  $u$ ”,
- “si un pixel est blanc dans  $u$  alors il est blanc dans  $u'$ ”.

## Érosion

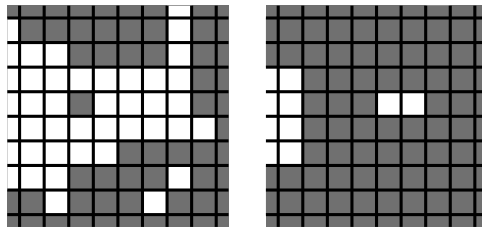
On se donne un élément structurant :



**Algo :** Parcourir les pixels de l'image  $u$ ,

pour chaque pixel  $u(i,j)$ :

1. Centrer l'élément structurant sur ce pixel,
2. Considérer les voisins du pixel (gris + noir de l'élément structurant)
3. Si l'un de ces pixels est en noir, mettre  $u'(i,j)$  en noir.





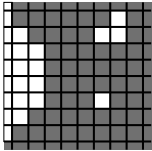
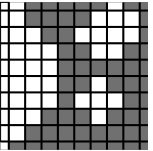
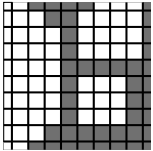
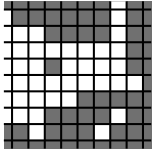
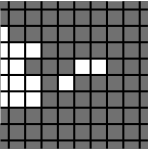
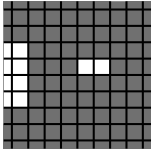
De la même manière que pour la dilatation, montrer que  $u'$  est incluse dans  $u$ .

**Exercice 1**

Montrer que l'érosion peut s'exprimer comme une dilatation.

**Exercice 2**

### Rapport à l'élément structurant

|            |   |   |   |
|------------|---|---|---|
|            | Image originale   |  |  |
| Dilatation |  |  |   |
| Érosion    |  |  |   |

Morphologie Mathématique

Nr. 5

#### Propriétés (Exercices)

On note  $D(X,b)$  et  $E(X,b)$  l'ensemble des pixels blancs issus de la dilatation et de l'érosion de l'ensemble des pixels blanc  $X$  par un élément structurant  $b$  :

- **Par rapport à  $X$** , L'érosion et la dilatation sont croissantes :

$$X_1 \subseteq X_2 \quad \begin{array}{l} D(X_1,b) \subseteq D(X_2,b) \\ E(X_1,b) \subseteq E(X_2,b) \end{array}$$

- **Par rapport à  $b$** , la dilatation est croissante et l'érosion décroissante :

$$b_1 \subseteq b_2 \quad \begin{array}{l} D(X,b_1) \subseteq D(X,b_2) \\ E(X,b_2) \subseteq E(X,b_1) \end{array}$$

La dilatation et l'érosion sont deux opérations **duales**. Notons  $X^c$  le complémentaire d'un ensemble  $X$  dans l'image, c'est à dire dans notre cas puisque  $X$  est l'ensemble des pixels blancs,  $X^c$  est l'ensemble des pixels noirs. On a alors :

$$D(X,b) = (E(X^c,b))^c \quad E = (D(X^c,b))^c$$

---

---

## Ouverture et Fermeture

La dilatation et l'érosion ne sont **pas des transformations inverses**. si l'on applique successivement une érosion puis une dilatation on ne retrouve pas l'image d'origine.

La question qui se pose donc : dans quel ordre appliquer ces transformations pour obtenir quoi ?

Pour de gros ensembles blancs : l'érosion diminue la taille de l'ensemble, tandis que la dilatation l'augmente. En appliquant l'une puis l'autre on obtient des ensembles dont la taille est la taille d'origine.

On appelle :

**Ouverture** : Érosion puis dilatation.

**Fermeture** : Dilatation puis Érosion.

On note  $O(u,b)$  l'ensemble des pixels blancs issus de l'ouverture de l'ensemble des pixels blancs de  $u$  par l'élément structurant  $b$ .

Montrer que :

$$O(u,b) \subseteq u$$

**Exercice**

De la même manière, montrer pour la fermeture que :

$$u \subseteq F(u,b)$$

---



---

## Ouverture

Avec un élément structurant :



On obtient :



De gauche à droite : original , érodé, dilaté (ouverture).



De gauche à droite : original, ouverture et différence.

NB: sur ce transparent les couleurs sont inversées  
(ensemble d'intérêt en noir.)

---



---

## Fermeture

Avec un élément structurant :



On obtient :



De gauche à droite: original , dilaté, érodé (fermeture).



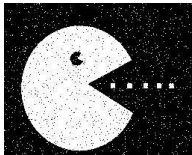
De gauche à droite: original, fermeture et différence.

NB: sur ce transparent les couleurs sont inversées (ensemble d'intérêt en noir.)



Exemple

Avec un élément structurant 3\*3 :



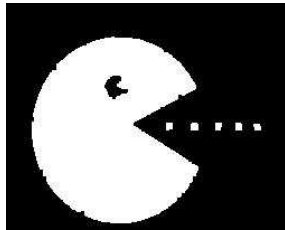
| Opérateur | 1              | 2              |
|-----------|----------------|----------------|
| Fermeture | Dilatation<br> | Érosion<br>    |
|           | Érosion<br>    | Dilatation<br> |
| Ouverture |                |                |

---

---

## Filtre Alterné séquentiel

En appliquant successivement ouvertures et fermetures, on obtient le filtre alterné séquentiel :



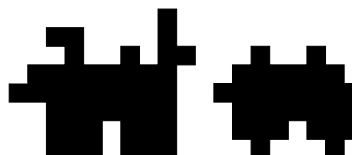
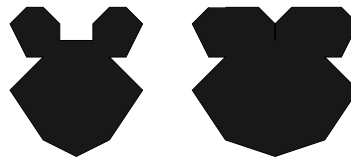
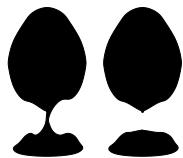
Ce filtre permet de filtrer aisément et efficacement le **bruit de canal**, obtenu en inversant la valeur de  $x\%$  des pixels tirés au hasard.

---

---

## Exercices

Pour chacune des figures suivantes, trouver la transformation et l'élément structurant qui a permis de passer de l'original (gauche) au résultat (droite).



---

---

## Autre formulation

Il faut remarquer que les opérateurs de dilatation et d'érosion peuvent être écrites sous une formulation qui rappelle le filtre médian.

Si on appelle 'voisins'  $v \in V(i,j)$  d'un pixel  $(i,j)$  les pixels grisé et noirs de l'élément structurant lorsque celui ci est centré sur le pixel :

- **Érosion** :  $u'(i,j) = \min\{v \in V(i,j)\}$
- **Dilatation** :  $u'(i,j) = \max\{v \in V(i,j)\}$

Cette formulation permet d'étendre les outils de la morphologie mathématique aux images en niveaux de gris.

---

---

## Passage aux niveaux de gris

On se donne un élément structurant du même type que précédemment (type voisinage).

On applique alors les algorithmes suivants pour obtenir la **dilatation** et l'**érosion** en niveaux de gris :

- **Érosion** :  $u'(i,j) = \min\{v \in V(i,j)\}$
- **Dilatation** :  $u'(i,j) = \max\{v \in V(i,j)\}$

On note  $D(u,b)$  l'image dilatée de  $u$  par l'élément structurant  $b$  et  $E(u,b)$  l'érodée.

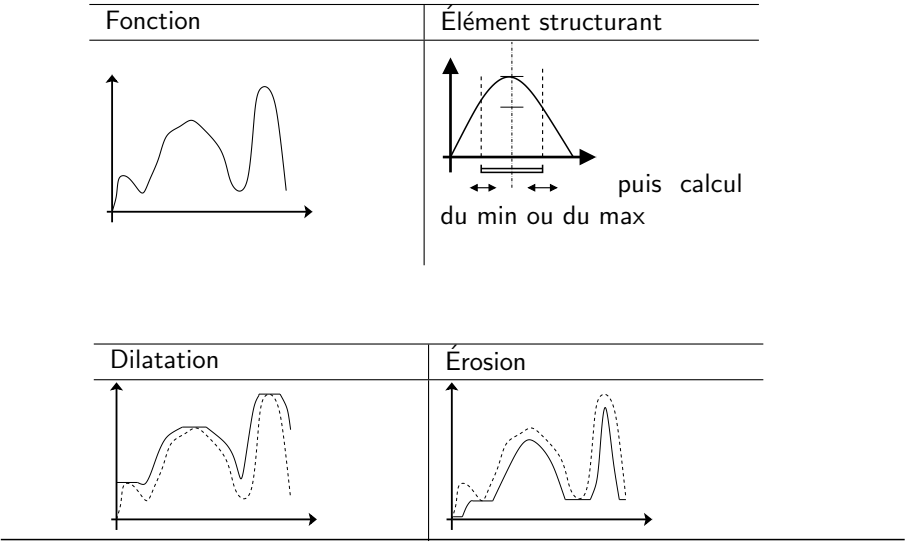
Le passage du traitement des images binaires aux traitement des images en niveaux de gris présenté ici est totalement artificiel.

**En vrai**

On définit normalement un élément structurant en niveaux de gris comme une fonction, de  $\mathbb{R}^2 \rightarrow \mathbb{R}$ . Dans notre cas, seul le support de cette fonction est défini (voisinage). Le cas présenté ici, correspond au cas où élément structurant est dit **plan**.

La morphologie mathématique présentée dans le cadre de ce cours est un petit sous ensemble de la morphologie mathématique telle qu'entendue usuellement.

Illustration : cas 1D



---



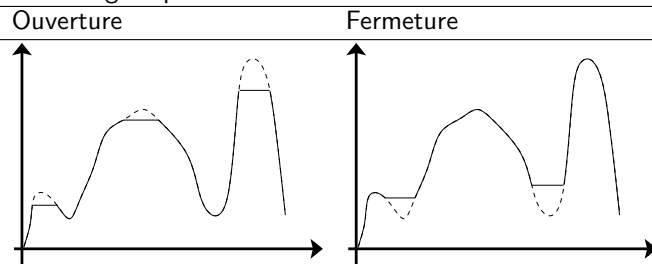
---

## Ouverture et Fermeture

On définit ces deux opérateurs de la même manière que pour les images binaires :

- **L'ouverture** est une érosion suivie d'une dilatation,
- **La fermeture** est une dilatation suivie d'une érosion.

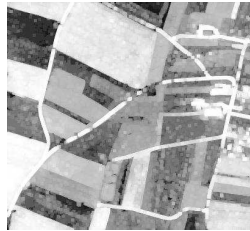
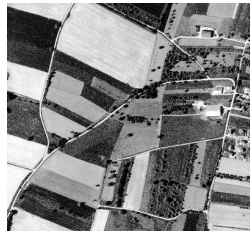
On note  $O(u,b)$  l'ouverture et  $F(u,b)$  la fermeture d'une image  $u$  par un élément structurant  $b$ .



---

---

## Exemples



Dilatation



Érosion



Ouverture



Fermeture



---

---

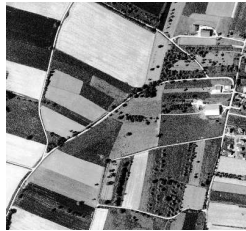
## Autres Opérateurs

### Gradient morphologique

Le gradient morphologique est obtenu en soustrayant l'image érodée à l'image dilatée (avec un élément structurant carré):

$$G(u,b) = \frac{1}{2}(D(u,b) - E(u,b))$$

Par exemple :



Originale



Gradient morphologique

### Exercice

Écrire un algorithme calculant le gradient morphologique sans passer par le calcul de l'érodée et de la dilatée de l'image originale.

Que donne le gradient morphologique sur des images binaires?

---

---

## Autres Opérateurs

On peut combiner de différentes manières les filtres morphologiques. Par exemple, pour extraire les “bosses” de l'image :

**Filtre “top hat”**

$$TH(u,b) = u - O(u,b)$$

L'opérateur dual permet d'extraire les “creux” :

$$TH^*(u,b) = F(u,b) - u$$

 $TH(u,b)$  $TH^*(u,b)$

---

---

## Exercice

Proposer un filtre utilisant la dilatée et l'érodée d'une image pour améliorer la netteté des contours.

## Chapitre 9

# TP Watersheds

---

---

## Algorithme de watershed

Dans cette partie on s'intéresse en détail à un algorithme : l'algorithme de "ligne de partage des eaux".

Ce chapitre est tiré d'un article de **Luc Vincent et Pierre Soille** dans "IEEE Transactions on Pattern Analysis And Machine Intelligence" de Juin 1991.

L'article s'intitule "Watersheds in Digital Spaces : An efficient Algorithm Based on Immersion Simulations"

Ce chapitre a pour but de définir la notion de ligne de partage des eaux.

---

---

## Watershed ?

On regarde l'image comme un relief. Imaginons deux creux (deux cratères) côte à côte. Ces deux creux sont appelés bassins d'attractions.

Si on laisse rouler une balle de n'importe quel point d'un bassin d'attraction, la balle va au fond de ce bassin.

La ligne séparant deux bassins d'attractions s'appelle "la ligne de partage des eaux".

Ce chapitre présente un algorithme extrayant de telles lignes de partage des eaux. Pour cela il faut d'abord définir mathématiquement ce que sont de telles lignes.

---



---

## Définitions : les bases

L'image  $I$  est numérique,  $D_I$  est son domaine.

$$\begin{array}{ccc} I : D_I \subseteq \mathbb{Z}^2 & \rightarrow & \{0,1,\dots,N\} \\ p & \rightarrow & I(p) \end{array} \quad (9.1)$$

On note  $G$  la grille de voisinage des pixels (Grid).  $G$  peut correspondre à la 4 ou à la 8 connectivité si la trame est carrée ou à la 6 connectivité si la trame est hexagonale.

L'ensemble des voisins d'un pixel  $p$  est noté  $N_G$  :

$$N_G(p) = \{p' \in \mathbb{Z}^2 \mid (p,p') \in G\} \quad (9.2)$$

**Définition 1 :** Un chemin  $P$  de longueur  $l$  entre deux pixels  $p$  et  $q$  dans l'image  $I$  est un  $(l+1)$  uplet de pixels  $(p_0, p_1, \dots, p_{l-1}, p_l)$  tel que  $p_0 = p$ ,  $p_l = q$  et  $\forall i \in [1, l] \ (p_{i-1}, p_i) \in G$ .

---



---

## Définitions : un minimum

**Définition 2 :** Un **minimum**  $M$  de  $I$  d'altitude  $h$  est un plateau de pixels connectés dont la valeur vaut  $h$  tel qu'il soit impossible d'atteindre un point d'altitude inférieure sans devoir grimper.

$$\begin{aligned} \forall p \in M \quad \forall q \notin M \quad \text{t.q.} \quad I(q) &\leq I(p) \\ \forall P = (p_0, p_1, \dots, p_l) \quad \text{t.q.} \quad p_0 = p \text{ et } p_l = q \\ \exists i \in [1, l] \quad \text{t.q.} \quad I(p_i) &> I(p_0) \end{aligned}$$

Un minimum est donc un ensemble de pixels de même altitude, où le niveau de gris est strictement plus foncé que dans les pixels voisins. On appelle ces extrema des minima régionaux.



---

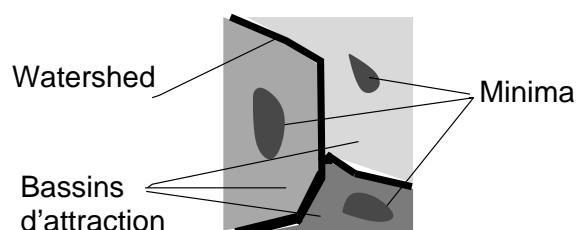


---

## Définitions : watershed

**Définition 3:** Un bassin d'attraction  $C(M)$  associé à un minimum  $M$  est l'ensemble des pixels  $p$  de l'image tel qu'un ruisseau d'eau partant de  $p$  coulerait le long du relief jusqu'à atteindre  $M$ .

**Définition 4:** Les lignes séparant différents bassins d'attraction forment ce que l'on appelle les **lignes de partage des eaux** ou **watersheds**.



Une définition plus algorithmique peut être donnée : supposons qu'un trou ait été percé dans chaque minimum régional de l'image. On immerge alors progressivement l'image dans un lac. En partant du minimum d'altitude la plus basse, l'eau va progressivement remplir les différents bassins d'attraction de l'image. Maintenant, à chaque pixel de l'image où l'eau arrivant de deux minima différents se rencontre on place un marqueur.

A la fin de la procédure, chaque minimum est totalement entouré par des marqueurs qui délimitent le bassin d'attraction.

Ces marqueurs correspondent aux watersheds de  $I$ .

---

---

## Bassin d'attraction

On note  $h_{\min}$  la plus petite valeur de l'image,  $h_{\max}$  la plus grande valeur et  $T_h(I)$  l'ensemble des pixels dont l'altitude est plus basse que  $h$  :

$$T_h(I) = \{p \in D_I, I(p) \leq h\}$$

On note  $C(M)$  le bassin d'attraction associé a un minimum  $M$  et  $C_h(M)$  le sous ensemble de ce bassin dont les points ont une altitude plus petite ou égale à  $h$  :

$$C_h(M) = \{p \in C(M), I(p) \leq h\} = C(M) \cap T_h(I)$$

**Notation** : pour finir nous notons  $\min_h(I)$  l'ensemble des pixels appartenant aux minima d'altitude  $h$ .

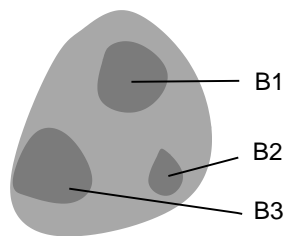
---

---

## Rappel : connexité

**Définition 4 :**  $A$  est un ensemble **connexe** de pixels si pour tout couple de pixels  $(p, q) \in A$  il existe un chemin  $P$  reliant les deux, qui soit entièrement dans  $A$ .

**Définition 5 :** Les composantes connexes  $B_1, \dots, B_k$  d'un ensemble  $B$  sont les plus grands ensembles  $B_i$  tels que chaque  $B_i$  soit connexe et que l'union des  $B_i$  donne  $B$ .



En gris foncé :  $B$  et ses composantes connexes. En gris clair, on a un ensemble  $A$  contenant  $B$ .

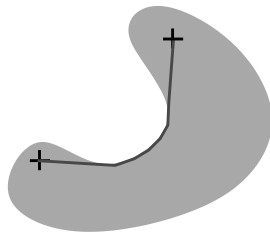
---

---

## Distance géodésique

Soit  $A$  un ensemble connexe.

**Définition 6:** La distance géodésique  $d_A(p,q)$  entre deux pixels  $p$  et  $q$  de  $A$  est la longueur du plus petit chemin reliant  $p$  et  $q$  dans  $A$ .



La distance géodésique entre deux points d'un ensemble.

---



---

## Zone d'influence

Soit  $A$  un ensemble connexe,  $B$  un autre ensemble  $B \subseteq A$  et soient  $B_1, \dots, B_k$  les composantes connexes de  $B$ .

**Définition 7:** La **zone d'influence géodésique**  $iz_A(B_i)$  d'une composante de  $B$  est l'ensemble des points tels que leur distance géodésique à  $B_i$  soit **strictement** plus petite que leur distance géodésique aux autres composantes de  $B$ .

$$iz_A(B_i) = \{p \in A, \forall j \in [1, k] \setminus i, d_A(p, B_i) < d_A(p, B_j)\}$$

**Définition 8 :** Les points de  $A$  qui n'appartiennent à aucune zone d'influence (ils sont à **égale distance** entre deux composantes) forment le squelette géodésique de  $B$  dans  $A$ , noté  $SKIZ_A(B)$ :

$$SKIZ_A(B) = A \setminus IZ_A(B) \quad IZ_A(B) = \bigcup_{i=\{1, dots, k\}} iz_A(B_i)$$

On suppose que ce squelette fait toujours **Un pixel d'épaisseur**.

---



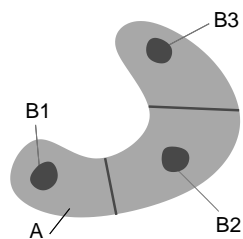
---

---

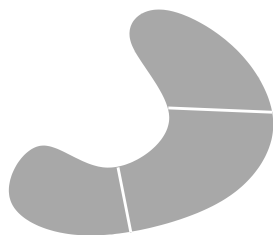


---

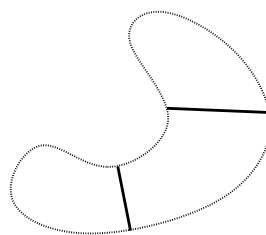
## Illustration Zones d'influences



Un ensemble  $B$  et les 3 zones d'influence de ses composantes (en gris clair).



$IZ_A(B)$



$SKIZ_A(B)$  en noir épais (deux lignes).

---

---

## Immersion

1 ) On part de l'ensemble  $T_{h_{\min}}$  des pixels qui sont les premiers touchés par l'eau. Ces pixels constituent l'état initial de l'immersion :

$$X_{h_{\min}} = T_{h_{\min}}(I) = \{p \mid I(p) = h_{\min}\}$$

2) On considère le niveau  $h_{\min} + 1$  et les pixels qui le composent :  $Y = T_{h_{\min}+1}(I)$ . Evidemment :

$$X_{h_{\min}} \subseteq T_{h_{\min}+1}(I)$$

On décompose  $Y$  en composantes connexes  $Y_1, \dots, Y_k$ , et on examine pour chaque  $Y_i$  son intersection avec  $X_{h_{\min}}$  :

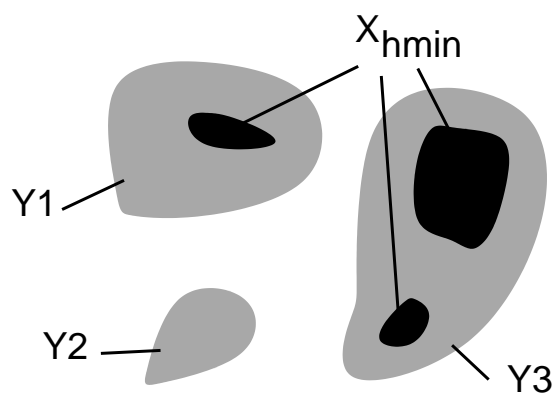
$$Y_i \cap X_{h_{\min}}$$

Il y a trois cas possibles.

---

---

### Illustration



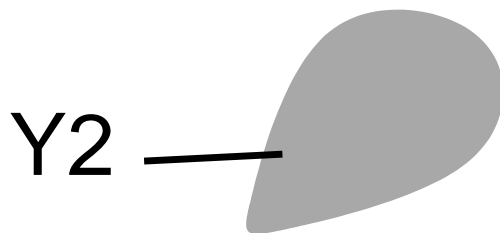


---

---

### Premier cas

$$Y_i \cap X_{h_{\min}} = \emptyset$$



Dans ce cas  $Y_i$  est un nouveau minimum.

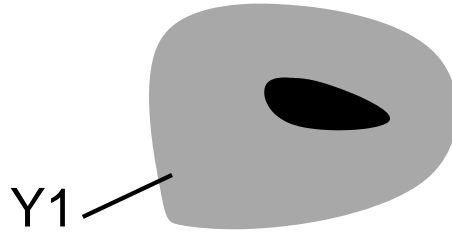
---



---

## Deuxième cas

$$Y_i \cap X_{h_{\min}} \neq \emptyset \text{ et est connexe}$$



Dans ce cas  $Y_i$  est composé des pixels appartenant au bassin d'attraction du minimum  $Y_i \cap X_{h_{\min}}$  qui ont un niveau inférieur ou égal à  $h_{\min} + 1$ .

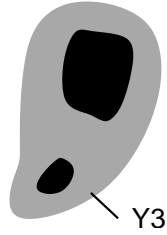
$$Y = C_{h_{\min}+1}(Y_i \cap X_{h_{\min}})$$

---

---

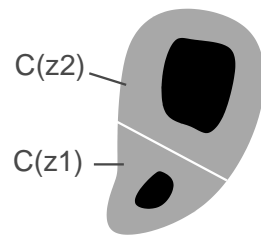
### Troisième cas

$Y_i \cap X_{h_{\min}} \neq \emptyset$  et n'est pas connexe



$Y_i$  contient donc des minima de  $I$  différents. On les notes  $Z_1, \dots, Z_l$ . Pour chaque  $Z_j$  le meilleur bassin d'attraction  $C_{h_{\min}+1}(Z_j)$  que l'on puisse trouver est alors sa zone d'influence géodésique dans  $Y_i$  :

$$C_{h_{\min}+1}(Z_j) = iz_{Y_i}(Z_j)$$



---



---

## Itérations et résultat

Une fois tous les  $Y_i$  examinés on peut passer à l'itération suivante :

$$X_{h_{\min}+1} = \min_{h_{\min}+1} \cup IZ_{T_{h_{\min}+1}}(I)(X_{h_{\min}})$$

**Dernière définition : bassins d'attraction et watersheds par immersion :** L'ensemble des bassins d'attractions d'une image en niveau de gris  $I$  est donnée par l'ensemble  $X_{h_{\max}}$  obtenu après la récursion suivante :

- a)  $X_{h_{\min}} = T_{h_{\min}}(I)$
- b)

$$\forall h \in [h_{\min}, h_{\max} - 1], X_{h+1} = \min_{h+1} \cup IZ_{T_{h+1}}(I)$$

Les lignes de partage des eaux sont le complémentaire de  $X_{h_{\max}}$  dans l'image, c'est à dire l'ensemble des points qui n'appartiennent à aucun bassin d'attraction.

### Description générale

Le but est de trouver les bassins d'attraction d'une image. Pour cela il faut explorer des ensembles de niveaux successifs et savoir calculer des zones d'influences.

L'algorithme se décompose en deux parties : la première consiste en un tri des pixels par ordre croissant des niveaux de gris. La seconde consiste en le calcul des zones d'influence géodésiques. Cela est fait en utilisant une *file de pixels*, c'est à dire une structure *First In First Out* (**Fifo\_add(p)** enfile un pixel, **Fifo\_first()** retire un pixel et **Fifo\_vide()** teste si la file est vide.

### L'étape de tri

La méthode de tri choisie est un algorithme distributif. Pour cela il faut d'abord calculer l'histogramme de l'image puis la fonction  $\phi$  de cumul et enfin parcourir l'image pour placer correctement chaque pixel dans un tableau.

### L'étape d'immersion

Une fois que les pixels ont été triés, l'immersion est simulée. Supposons que l'immersion aie été simulée jusqu'au niveau  $h$ . Tous les bassins d'attraction dont le minimum correspondant est d'altitude inférieure ou égale à  $h - 1$  déjà découverts sont supposé avoir une étiquette unique. Grace au tri initial on a alors accès aux pixels d'altitude  $h + 1$ . On leur donne l'étiquette **MASK**. Parmi ces pixels, ceux qui ont un voisin appartenant à un bassin déjà découvert sont mis dans la file. Grace à eux, la structure de queue permet d'étendre les bassins d'attractions déjà découverts dans l'ensemble des pixels masqués (mécanisme de diffusion).

Après cette étape, seuls les *minima* de niveau  $h + 1$  n'ont pas encore été découverts. Ils ne sont en effet connectés à aucun des bassins d'attraction déjà découverts. Un second parcours des pixels de niveau  $h + 1$  est donc nécessaire pour détecter les pixels qui ont toujours une valeur **MASK** et pour leur donner une nouvelle étiquette.

A la fin on obtient un étiquetage de tous les bassins d'attraction. Les pixels non étiquetés sont ceux qui sont à égale distance entre deux bassins d'attraction : ils forment les lignes de partage des eaux.

#### L'algorithme : Fast Watershed

En entrée: `im_in` l'image initiale en niveau de gris.

En sortie: `im_out` le résultat : bassin d'attractions et watersheds.

```
#define MASK -2      /*valeur initiale d'un niveau en cours d'exploration */
#define WSHED 0      /*marqueur des lignes de partages des eaux*/
#define INIT -1 /*Valeur initiale de im_out */
```

Initialisations :

- Tous les pixels de `im_out` sont mis à `INIT`.
- `int current_label=0`
- `int current_dist=0`
- `im_dist` est une image des distances initialisée à 0.
- `p_fictif` est un pixel fictif ((-1,-1) par exemple).
- `voisins(p)` retourne les voisins de `p`.

**Première étape :** trier les pixels de `im_in` par ordre croissant des niveaux de gris. `h_min` et `h_max` sont les niveaux les plus bas et les plus hauts.

#### Deuxieme étape :

Pour `h=h_min` à `h=h_max` /\* Recherche SKIZ du niveau `h-1` dans niveau `h` \*/

```
{
  Pour tout pixel p tel que im_in(p)=h
  {
    im_out(p)=MASK
    Ng=voisins(p);

    Si ( il existe p' dans Ng tel que im_out(p')>0 ou im_out(p')=WSHED)
    {
      im_dist(p)=1;
      Fifo_add(p);
    }
  }
}
```

```

    }
}
current_dist=1;
Fifo_add(p_fictif);

Repeter Indefiniment
{
    p=Fifo_first();
    si(p=p_fictif)
    {
        si(Fifo_vide()==vrai)
            BREAK;
        sinon
        {
            Fifo_add(p_fictif);
            current_dist=current_dist+1;
            p=Fifo_first();
        }
    }
}

Ng=voisins(p)
Pour tout pixel p' dans Ng
{
    si (im_dist(p')<current_dist et
        (im_out(p')>0 ou im_out(p')=WSHED))
        /*p' est dans un bassin deja decouvert */
    {
        si(im_out(p')>0)
        {
            si( im_out(p)=MASK ou im_out(p) =WSHED)
                im_out(p)=im_out(p');
            sinon
                si(im_out(p) != im_out(p'))
                    im_out(p)=WSHED;
        }
        sinon
            si(im_out(p)=MASK)
                im_out(p)=WSHED;
    }
    sinon
        si(im_out(p')=MASK et im_dist(p')=0)
        {
            im_dist(p')=current_dist+1;
            Fifo_add(p');
        }
    }
}

/* On verifie si de nouveaux minimums ont ete decouverts */
Pour tout pixel p tel que im_in(p)=h
{
    im_dist(p)=0;
    si(im_out(p)=MASK)
    {
        current_label=current_label+1;
        Fifo_add(p);
    }
}

```

```

im_out(p)=current_label;

Tant que(Fifo_vide())=faux)
{
    p'=Fifo_first();
    Ng=voisins(p');
    Pour tout pixel p'' dans Ng
    {
        si(im_out(p'')=MASK)
        {
            Fifo_add(p'');
            im_out(p'')=current_label;
        }
    }
}
}

```

**Le résultat** Si on veut obtenir une classification des pixels en bassins d'attraction, il suffit de leur donner l'étiquette de l'un de leur voisin (attention, il se peut que plusieurs voisins soient eux même des pixels d'une ligne de partage). A l'inverse, si l'on veut n'obtenir que les lignes, il suffit de donner la valeur *WSHED* a tous les pixels ayant dans leur voisinage au moins un pixel avec une étiquette plus petite (mais plus grande que 0).

## Chapitre 10

# Segmentation des images

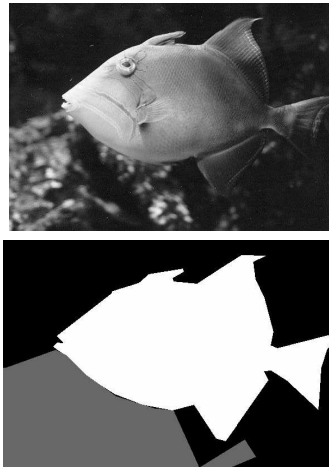


---

---

## Introduction

La segmentation s'intéresse aux régions de l'image. Le but de la segmentation est de **décomposer** une image en un ensemble de régions disjointes, chacune de ces régions formant **un ensemble homogène** de pixels au sens d'un certain critère.



---

---

## Définition

Une **région** de l'image est un ensemble  $R$  de pixels.

Une segmentation d'une image  $u$  est une partition  $R_0, \dots, R_{p-1}$  de l'image :

- L'union des régions donne l'image entière :

$$u = R_1 \cup \dots \cup R_p$$

- Les régions sont disjointes :

$$\forall i, j \quad R_i \cap R_j = \emptyset$$

- Les régions doivent être homogènes vis à vis d'un certain critère,
- On demande parfois aux régions  $R_i$  d'être connexes.

## But

- Extraire une information pertinente : zones homogènes
- Faciliter la compression des images
- Permettre la compréhension sémantique des images

---

---

## Dans ce chapitre

Ne sont considérées que les approches duales à la détection de contours : on les appelle les **approches par régions**.

En particulier : deux approches différentes pour réaliser la segmentation :

- **Division de région** : on part d'une très grande région que l'on divise en sous région, et on itère...
- **Croissance de région** : on part d'une toute petite région que l'on agrandit au fur et à mesure.

---

---

## Définition

### Région

Une région est un ensemble de pixels.

### Homogénéité

Chaque région de la segmentation finale doit être homogène :

- On se donne un **prédicat d'homogénéité** :

$$\text{Pred}(R) = \text{Vrai} \quad \text{ssi } R \text{ homogène}$$

- Toute région de la segmentation finale doit vérifier ce prédicat.

### Contrainte

Une région composée d'un seul pixel est homogène.

---

---

## Division de région (1)

**État initial :**  $R_0 =$  toute l'image

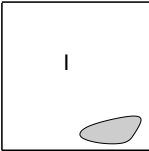
**Algorithme :** A chaque instant  $t$ , on parcourt toutes les régions  $R'_i$ .

- Pour une région d'intérêt, on regarde si les pixels à l'intérieur de la région satisfont le **prédicat d'homogénéité**.
  - Si **oui** alors la région est dite homogène et est conservée comme telle jusqu'à la fin.
  - Si **non**, on divise la région en  $p_i^t$  sous régions (par exemple 4) et on remplace la région par ses filles :  $R'_{i,1}, \dots, R'_{i,4}$
- On continue jusqu'à ce qu'il n'y ait plus de région divisible.

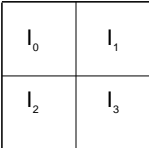
**Remarque :** Parfois une division produit une région qui est homogène avec l'une de ses voisines. Il est possible d'ajouter une opération de **Fusion**.

Division de région (2)

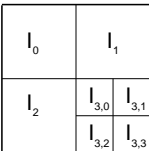
Par exemple: au début il n'y a qu'une région  $I_0$ . Imaginons que celle ci ne soit pas homogène.



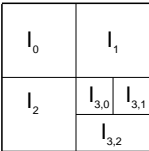
On la divise donc en quatre sous régions



Supposons que  $I_0, I_1$  et  $I_2$  sont homogènes alors que  $I_3$  ne l'est pas :



Si  $I_{3,2}$  et  $I_{3,3}$  sont semblables : on peut éventuellement les fusionner. On obtient alors :



Montrer que si l'on considère qu'une région composée d'un pixel vérifie le prédicat d'homogénéité l'algorithme termine et que l'on obtient bien une segmentation de l'image.

Exercice

Si une région composée d'un pixel n'est pas considérée comme homogène, on obtient une région dite "de fond" qui contient tous les résidus et un ensemble de régions homogènes.

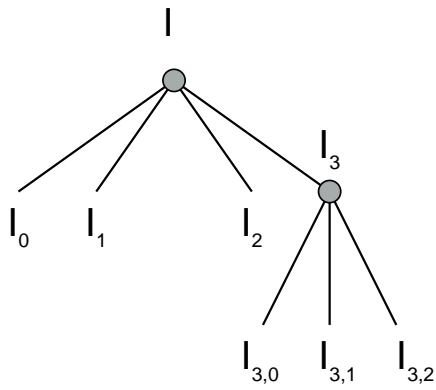
Remarque

---

---

## Arbre de division

On peut décrire la procédure précédente de division en utilisant un arbre. Nous verrons plus tard la structure particulière appelée **quadtrees**.



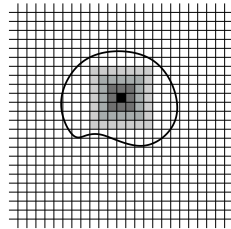
---

---

## Croissance de région

Utilise par exemple le processus de diffusion (cf cours sur les contours).

- On commence en choisissant un **pixel graine** dans l'image.
- La région grandit à partir de cette graine en ajoutant les voisins qui vérifient un critère de similarité.
- Quand aucun des voisins d'une région ne vérifie le critère de similarité, la croissance de la région est terminée.
- On choisit alors une autre graine et on recommence jusqu'à ce que tous les pixels aient été attribués.



Écrire en détail l'algorithme de croissance de région.

**Exercice**



---

---

## Commentaires

### Avantages

Cet algorithme suit bien les contours.

### Quelques problèmes

Le fait de faire grandir entièrement une région avant de passer à une autre biaise la segmentation :

- La région couramment considérée domine le processus : s'il y a une ambiguïté elle sera arbitrairement levée à son profit,
- Des choix différents de pixels graines donnent des segmentations différentes,
- Si un pixel graine est situé sur un contour, la segmentation obtenue peut être problématique.

---

---

## Variantes

On peut donc faire grandir les régions **simultanément**.

- Les similarités entre différentes régions peuvent être prises en compte pendant le processus
- Aucune région ne domine le processus à un instant donné,
- Le contrôle des différentes croissances peut être assez délicat
- Il existe une possibilité de paralléliser l'algorithme.

---

---

## Critères d'homogénéité

### Rappels

On considère une région  $R$  de l'image  $u$ . On définit :

- le **cardinal** : de la région :

$$\text{card } R = \text{Nb de pixels de } R$$

- la **moyenne** :

$$m(R) = \frac{1}{\text{card } R} \sum_{p \in R} u(p)$$

- la **variance** :

$$\sigma^2(R) = \frac{1}{\text{card } R} \sum_{p \in R} (u(p) - m(R))^2$$

---



---

## Critères d'homogénéité

### Deux prédicats simples

Fondés sur la même philosophie :

1. La variance des niveaux de gris d'une région doit être faible :

$$\sigma^2(R') \leq \text{seuil}$$

Où  $R'$  correspond à  $R$  + un nouveau pixel,

2. Autre formulation de la même idée :  
On considère l'ensemble suivant :

$$\begin{aligned} M(R) &= \{p \in R \text{ tels que } v(p) \\ &\quad \notin [m(R) - \sigma(R), m(R) + \sigma(R)]\} \\ &\subseteq R \end{aligned}$$

et on vérifie que sa taille est suffisamment petite :

$$\frac{\text{card } M(R')}{\text{card } R'} < \text{seuil}$$

---

---

## **Autres possibilités**

Utilisation de caractéristiques de texture, de couleurs etc... et d'un seuil.

Utilisation de mesure locales: cf. exemple les histogrammes locaux.

---

---

## Fusion de régions dans un graphe

L'algorithme de division de région produit un ensemble de régions homogènes.

Des régions voisines peuvent être fusionables (au sens du critère d'homogénéité).

Le but est d'obtenir :

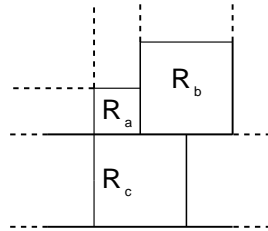
- Ou bien une représentation plus simple,
- Ou bien un ensemble de représentations correspondant à différentes échelles.

---

---

## Problématique

l'ordre de fusion importe sur le résultat final. Par exemple, on a obtenu trois régions adjacentes  $R_a$ ,  $R_b$  et  $R_c$  :



On peut avoir :

- les régions sont fusionables deux à deux :

Critère  $(R_a \cup R_b)$       Vrai

Critère  $(R_b \cup R_c)$       Vrai

Critère  $(R_c \cup R_a)$       Vrai

- Mais pas les trois ensembles :

Critère  $(R_a \cup R_b \cup R_c)$       Faux

Il faut donc une méthode qui permette de distinguer les trois cas de fusion possibles, et d'en choisir un.

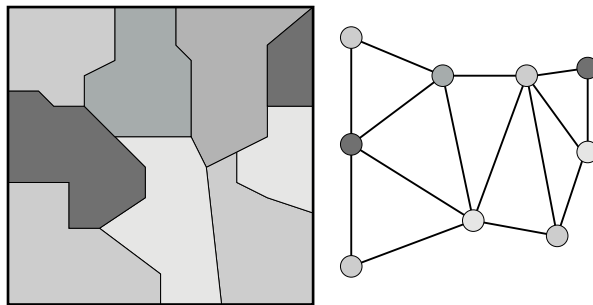
---

---

## Graphe d'adjacence

Pour une segmentation donnée de l'image :  $R_1, \dots, R_p$ , on définit  $G = (V, E)$ , un graphe dont :

1. les **noeuds** sont les régions  $R_i$ ,
2. les **arêtes** représentent l'adjacence des régions (il y a une arête entre deux noeuds si les régions sont adjacentes).







---



---

## Stratégies de fusion

### La plus simple

1. Chercher dans le graphe  $G(V,E)$  l'arête  $e$  de moindre coût.
2. Fusionner les deux régions correspondantes.
3. Itérer jusqu'à ce qu'aucune fusion ne soit plus possible.

### Alternative

Fusionner en parallèle plusieurs régions.

- Sélectionner  $(E', V')$ , un sous graphe de  $(E, V)$  tel que :

$$\forall v_k \in V' \quad \exists ! e = (v_i, v_j) \in E'$$

tel que  $v_k = v_i$  ou  $v_k = v_j$

- Fusionner 2 à 2 les régions associées à  $E'$ .
- Mettre à jour  $V$  et  $E$ .
- Itérer.

---

---

## Stratégies de fusion 1

### Première stratégie

Un exemple de construction de  $E'$  : On peut prendre :

$$E' = \{ \text{Arêtes minimum locaux stricts du coût de fusion} \leq \textit{seuil} \}$$

### Seconde stratégie

Un autre exemple :

$$E' = \{ (v_i, v_k) \text{ t.q } v_i \text{ définit l'arc de coût} \\ \text{de fusion minimal pour } v_k \text{ et inversement.} \}$$

---

---

## Stratégie de fusion 2

On peut construire  $(E', V')$  de façon itérative, en utilisant un seuil :

- Sélection du sous graphe  $(E_T, V_T)$  :

$$E_T = \{e \in E \text{ t.q. } c(e) \leq \text{seuil} \}$$

- Itération des étapes suivantes :

1. Sélection de l'arête  $\tilde{e}$  de coût minimum dans  $E_T$ , ajout de  $\tilde{e}$  à  $E'$ ,
2. Suppression de  $\tilde{e}$  dans  $E_T$  et des 2 sommets concernés (cela enlève d'autres arêtes)
3. Retour en 1, sauf si  $E_T$  est vide.

- Fusionner deux à deux les régions concernées dans  $(E', V')$ .

Appliquer les différents processus de fusion au graphe du transparent 17.

**Exercice**

---

---

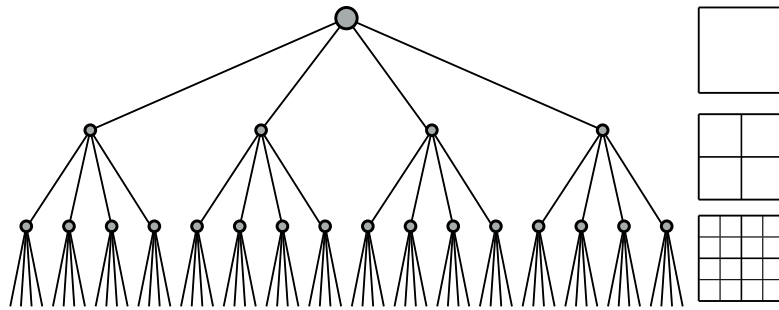
## Quadtree

- Le quadtree est une structure de donnée, utile pour :
  - la fusion de régions (prédicat de fusion)
  - la division de régions (prédicat de division)
- Le quadtree est un arbre, représentant les dichotomies successives de l'image.
- Seulement pour une image de dimension  $N,N$  avec  $N = 2^L$ . (256\*256, 512\*512.)
- A une profondeur  $L = \log_2(N)$ , les feuilles sont de taille 1 pixel.
- La racine est de profondeur 0, et sa taille  $N*N$  (image entière).
- Chaque noeud a quatre enfants de même taille.

---

---

## Quadtree (2)



---

---

## Arbre et coupure

### Définition :

une **coupure** d'une arborescence est un ensemble de noeuds tels que :

- Deux noeuds ne peuvent appartenir à un même chemin allant de la racine vers une feuille,
- Tout noeud supplémentaire à cet ensemble fait perdre sa qualité de coupure.

### Propriété :

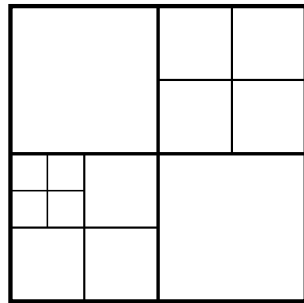
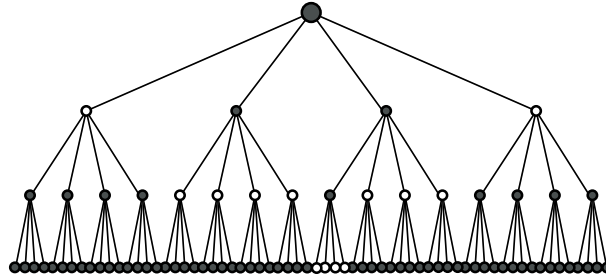
Une coupure dans un quadtree définit une segmentation.

---



---

## Coupure et segmentation




---



---

Segmentation des images

Nr. 24

Le quadtree permet de faire de la fusion de région pour segmenter une image.

Vérifier que le critère de fusion du transparent 17 est facilement calculable pour un noeud du quadtree à partir de celui de ses enfants. Expliquer pourquoi un processus de fusion est alors largement préférable à un processus de division. Écrire l'algorithme correspondant.

### Quadtree et Fusion

#### Exercice



---

---

## Approches pyramidales

Le but est d'obtenir non pas une mais **plusieurs segmentations de l'image**.

Les fusions ne se font pas seulement deux à deux, mais plusieurs régions peuvent être fusionnées d'un seul coup.

A chaque étape les fusions par groupe sont effectuées sur le graphe d'adjacence ce qui définit la **pyramide de graphes**.

---

---

## Pyramide de graphes par Fusion

La pyramide au niveau  $\lambda$  est un graphe non orienté  $G[\lambda] = (V[\lambda], E[\lambda])$ , où  $V$  sont les sommets et  $E$  les arêtes.

$G[0]$  est l'image originale en 8 connexité.

Pour **Définir**  $G[\lambda + 1]$  à partir de  $G[\lambda]$ : problème de contraction de graphe. Il faut

- choisir les sommets de  $V[\lambda + 1]$  parmi ceux de  $V[\lambda]$ . On les appelle les **survivants**;
- affecter tout non survivant du niveau  $\lambda$  à un survivant du niveau  $\lambda + 1$ . C'est la génération des liens fils-père,
- créer les arêtes de  $E[\lambda + 1]$  pour conserver l'adjacence entre les régions;

---

---

## Notion de stable

Une solution consiste à rechercher un stable dans  $G[\lambda]$ . L'ensemble des sommets survivants est calculé pour être un **stable** :

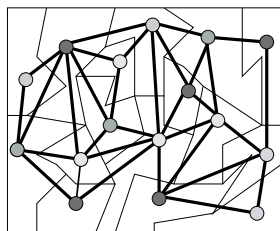
- Tout sommet non survivant au niveau  $\lambda$  possède au moins un sommet adjacent qui va survivre au niveau  $\lambda + 1$ , (ce qui permet de créer des liens fils-père qui garantissent la constructions de régions connexes.)
- Deux sommets adjacents ne peuvent survivre tous les deux (le nombre de sommets décroît significativement.)
- Une arête n'est présente dans  $E[\lambda + 1]$  que s'il existe au moins deux sommets adjacents parmi les deux sous ensembles de sommets fils  $G[\lambda]$ .

---

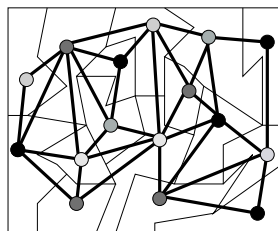
---

## Illustration

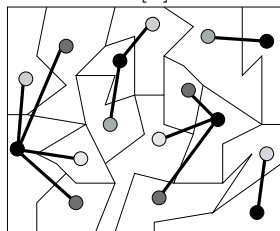
Exemples :



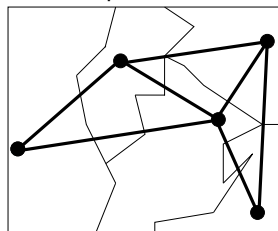
$G[\lambda]$



Exemple de stable



Liens fils père



$G[\lambda + 1]$

---



---

## Stable en pratique

**But : Construire un stable de  $G[\lambda]$**

On note  $(c_i)_{i=1,\dots,r}$  les sommets de  $G[\lambda]$  (les régions). Pour chaque arête on a un coût de fusion :

par exemple  $C(i,j) = |moy(c_i) - moy(c_j)|$ .

On construit un graphe orienté  $G'[\lambda] = (V[\lambda], A[\lambda])$  :

- Pour chaque  $c_i$  on regarde les voisins et les coûts de fusion et en déduit un seuil :  $S[c_i]$
- Une flèche de  $c_i$  à  $c_j$  est ajoutée dans  $G'$  si  $C(i,j) < S[c_i]$  : cela veut dire que  $i$  est susceptible d'être le fils de  $j$ .
- Une fois que le **graphe de similarité**  $G'$  est calculé : on choisit les survivants et les liens père fils.

---



---

Segmentation des images

Nr. 29

Quel seuil  $S[c_i]$  choisir ? On peut prendre le seuil le plus discriminant. On regarde les  $k$  voisins de  $i$  et les différents coûts de fusion :  $C_1, \dots, C_k$ .

On ordonne ces coûts :  $C_{i_1} \leq \dots \leq C_{i_k}$ .

On cherche le seuil  $S = C_{i_p}$  parmi ces coûts tel que la différence entre la moyenne des coûts inférieur à  $S$  et la moyenne des coûts supérieurs à  $S$  soit maximale.

Pour chaque  $p = i_k$ , Notons :

$$A(p) = \frac{\sum_{j=1}^p C_{i_j}}{p} \quad B(p) = \frac{\sum_{j=p+1}^{i_k} C_{i_j}}{i_k - p}$$

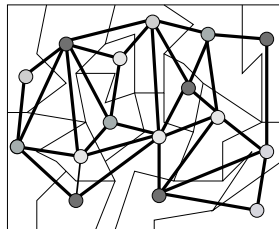
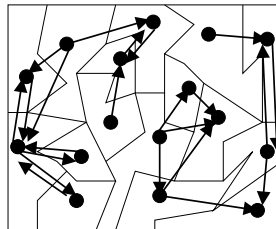
on prend  $S = C_{i_{\tilde{p}}}$  avec :

$$\tilde{p} = \text{Argmax}(B(p) - A(p))$$

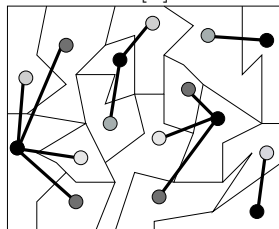
---

---

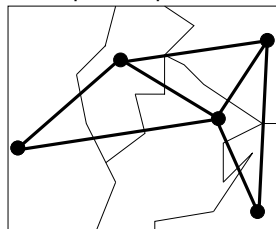
## Illustration

 $G[\lambda]$ 

Graphe de proximité



Liens fils père retenus

 $G[\lambda + 1]$

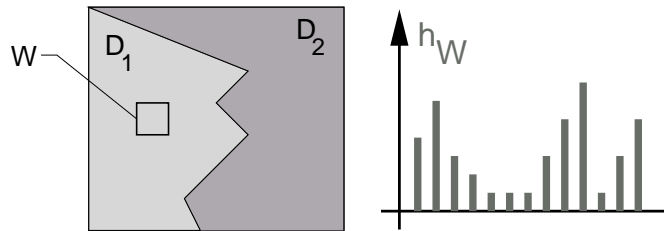
---

---

## Segmentation par corrélation d'histogrammes locaux

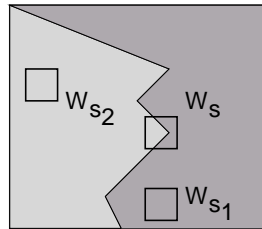
Autre critère d'homogénéité : les histogrammes locaux. On présente ici deux algorithmes de segmentation par croissance de régions utilisant les histogrammes locaux.

On note  $h_W$  l'histogramme (c'est un vecteur) d'une image calculé sur une fenêtre :



## Approximations

Deux régions homogènes :  $D_1$  et  $D_2$  et une fenêtre  $W$ .



- Si  $W \subset D$ , on suppose : L'histogramme sur une fenêtre est caractéristique d'une région :  
 $h_{W_{s1}} \approx h_{R1}$  et  $h_{W_{s2}} \approx h_{R2}$

- Si  $W$  à cheval sur deux régions : on a :

$$h_{W_s} \approx \alpha h_{R1} + (1 - \alpha) h_{R2}$$

avec

$$\alpha = \frac{|R1 \cap W_s|}{|W_s|}$$

- $h_R$  est un vecteur, on peut calculer sa norme :  
 $\|h_R\|$



---

---

## Propriétés de l'histogramme local

- Les **maxima** de la norme sont significatifs de régions **très homogènes**
- Les **minima** de la norme sont significatifs de régions **bruitées ou de frontières** entre deux régions.
- On appelle coefficient de corrélation entre deux vecteurs histogramme  $h_1$  et  $h_2$  le produit scalaire normalisé :

$$r(h_1, h_2) = \frac{h_1 \cdot h_2}{\|h_1\| \|h_2\|}$$

Quand  $r$  est proche de 1,  $h_1 \approx h_2$ , tandis que plus  $r$  est faible, plus l'écart entre  $h_1$  et  $h_2$  est grand.

**Critère d'homogénéité :**  $r \geq T$ .

---



---

## H. locaux et relaxation

L'algorithme utilise trois seuils:  $1 > T_2 > T_3 > T_1 > 0$

1. Pour tout pixel  $t$ , calcul de l'histogramme  $h_{Wt}$  sur une fenêtre centrée en  $t$ .
2. Recherche du point  $s = \text{Argmax}\|h_{Wt}\|$  parmi les points non encore attribués
3. Extraction de l'ensemble  $C_s$ , **connexe** tel que :

$$\forall q \in C_s \quad r(h_{Ws}, h_{Wq}) > T_1$$

4. Extraction de  $Z_{ref} \subset C_s$ :

$$Z_{ref} = \{q \in C_s \quad r(h_{Ws}, h_{Wq}) > T_2\}$$

5. Calcul de  $h_{ref}$  : histogramme sur  $Z_{ref}$
6. Extraction de  $R_s$  contenant  $Z_{ref}$  **connexe** à  $s$  :

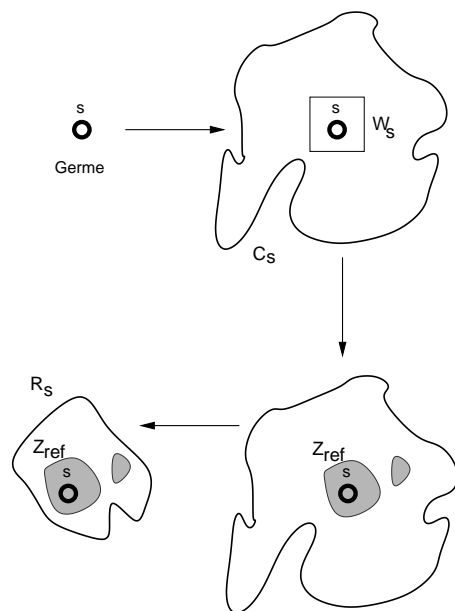
$$R_s = \{q \quad r(h_{Wq}, h_{ref}) > T_3\}$$

7. Dilatation morphologique de  $R_s$
  8. Étiquetage de  $R_s$
  9. Retour en 2
- 
-

---

---

### Relaxation : illustration



---



---

## H. Locaux et propagation

La technique par propagation repose sur une idée différente. On commence par rechercher une composante connexe  $C_s$  autour du germe  $s$ .

On regarde l'ensemble  $F$  des points situés sur la frontière de  $C_s$  tel que :

$$\forall f \in F \quad c(h_{Wf}, h_{Ws}) \leq T_0$$

On calcule ensuite  $V(f)$  : points voisins d'un point  $f \in F$ . Il y a deux cas :

1.  $\forall q \in V(f), r(h_{Wq}, h_{Ws}) > T_1$  avec  $T_1 < T_0$ .  
Les points autour de  $f$  sont assez corrélés avec le germe  $s$  : on va se servir de  $f$  comme d'un nouveau **point de propagation** éventuel pour étendre  $C_s$
2.  $\exists q \in V(f), r(h_{Wq}, h_{Ws}) \leq T_1$ . Il existe dans le voisinage de  $f$  des points décorrélés de  $s$ . Dans ce cas,  $f$  est un **point de contour** de  $C_s$ .

---



---

## H. Locaux et propagation(2)

Avec deux seuils  $T_1 < T_0$  :

1. Pour tout pixel  $t$ , calcul de l'histogramme  $h_{W_t}$  sur une fenêtre centrée en  $t$ .
2. Recherche du point  $s = \text{Argmax} \|h_{W_t}\|$  parmi les points non encore attribués
3. Extraction de l'ensemble  $C_s$ , **connexe** autour de  $s$  tel que :

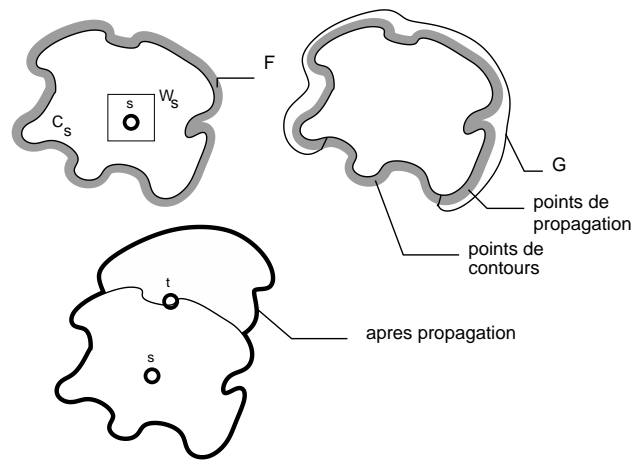
$$\forall q \in C_s \quad r(h_{W_s}, h_{W_q}) > T_0$$

4. Détermination de la frontière  $F$  de  $C_s$
5. Détermination des **points de contours**
6. Détermination des **points de propagation**
7. Dilatation morphologique des points de propagation : ensemble  $G$
8. Recherche du prochain point  $t$  de norme maximum dans la partie non segmentée :
  - Si  $t \in G \cup F$  alors retour en 3 pour calculer  $R_t$  qui sera une extension de  $R_s$ ,
  - Si  $t$  n'a jamais été exploré : alors retour en 2 pour calculer une nouvelle région  $R_t$ .
  - Étiquetage de toutes les régions  $R_s$ .

---

---

### Propagation : illustration



## Chapitre 11

# Champs de Markov

---

---

## Modèle probabiliste

**Le cadre :** On considère des images avec 256 niveaux de gris. On note  $S$  l'ensemble des pixels  $s$ . On note  $\Omega$  l'ensemble des images possibles.

**Question :** qu'est ce qu'un modèle probabiliste?

Un modèle probabiliste attribue une probabilité à chaque image possible de  $\Omega$ . Soit  $X$  une image, on note :  $P(X)$  la probabilité de cette image.

On appelle **Champ aléatoire** tout modèle aléatoire de l'image.



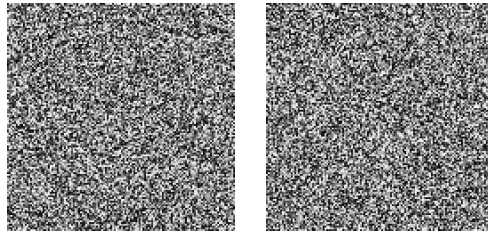
---

---

## Le modèle aléatoire le plus simple

C'est le **modèle uniforme** : toutes les images ont la même probabilité d'apparaître.  $P(X) = 1 / \text{card}(\Omega)$

Pour tirer une image au hasard suivant ce modèle : chaque pixel est tiré au hasard entre 0 et 255.

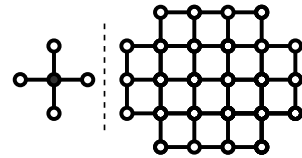


La valeur en chacun des pixels ne dépend pas des voisins: indépendance.

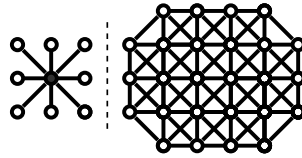
En chaque pixel, toutes les valeurs ont la même probabilités: uniformité.

## Notion de voisinage

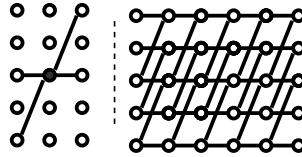
A gauche, les voisins  $\partial(s)$  d'un pixel  $s$ , a droite le graphe  $G$  de voisinage induit.



4 connexité



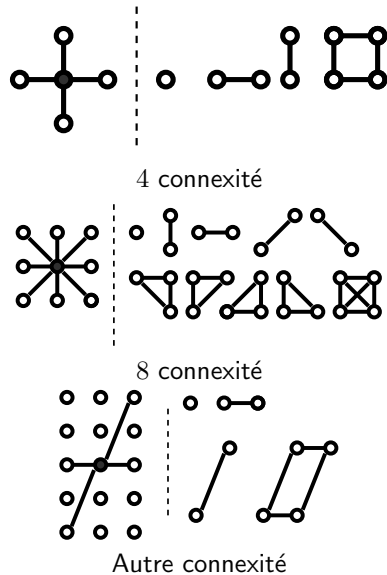
8 connexité



Autre connexité

## Cliques

On considère le graphe  $G$  de voisinage sur l'image induit par la structure de voisinage considérée: les **cliques** sont les **ensembles complets**. L'ordre d'une clique donne le nombre de pixels impliqués.



---

---

## Dépendance

On a vu : le modèle **global** le plus simple :  $\mathbf{P}(\mathbf{X}) = 1 / \text{card}(\Omega)$  implique le comportement **local** suivant :

La valeur en un pixel ne dépend pas des voisins, et ce pixel peut prendre n'importe quelle valeur avec la même probabilité.

On va faire le raisonnement inverse : on veut regarder ce qui se passe quand la valeur en un pixel dépend des voisins (**local**) et en déduire la loi de comportement **globale**.

---

---

## Champs de Markov :local

On se donne une structure de voisinage. Un modèle aléatoire est appelé champs de Markov si la propriété **locale** est vérifiée :

**La valeur d'un pixel  $s$  ne dépend que de ses voisins  $\partial(s)$ .**

Cela veut dire que pour connaître la loi de probabilité d'un pixel, si l'on connaît les valeurs partout ailleurs dans l'image, il suffit de prendre en compte ses voisins (propriété de Markov).

---



---

## Champs de Markov : global

**Théorème d'Hammersley-Clifford:**

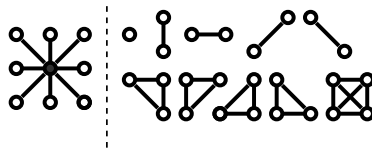
$X$  est un champ de Markov  $\Leftrightarrow$  on peut écrire la probabilité de  $X$  sous la forme :

$$\mathbf{P}(X) = \frac{1}{Z} \exp(-U(X))$$

avec

$$U(X) = \sum_{c \in C} V_c(x_s, s \in c)$$

Où  $x_s$  représente un pixels,  $C$  l'ensemble des cliques et  $(x_s, s \in c)$  l'ensemble des pixels intervenant dans la clique  $c$ . Les cliques du voisinage 8 connexe:



---

---

## Le modèle d'Ising

On considère une image binaire, en 4 connexité.

On ne considère que les cliques d'ordre 2  $c = s, t$ . On prend :

$$V = -\beta \mathbf{1}(x_s = x_t)$$

Ce qui donne la loi suivante :

$$U(X) = - \sum_{s \sim t} \beta \mathbf{1}(x_s = x_t)$$

Où  $t \sim s$  signifie que les pixels  $s$  et  $t$  sont voisins.

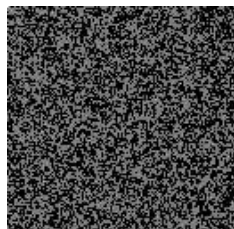
Quand  $\beta > 0$  le modèle favorise l'homogénéité, tandis que pour  $\beta < 0$  c'est l'hétérogénéité qui est favorisée.

---

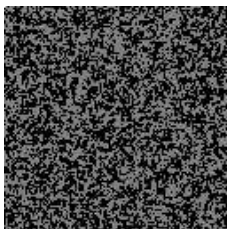
---

## Le modèle d'Ising (2)

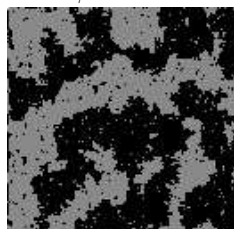
Quelques exemples :



$\beta = 0.2$



$\beta = 0.5$



$\beta = 1$



$\beta = 10$



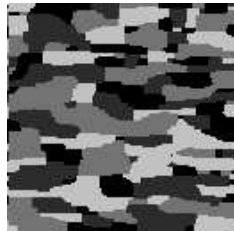
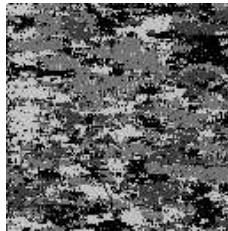
---

---

## Le modèle de Potts

C'est le même modèle que le modèle d'Ising, excepté que l'on utilise plus de niveaux de gris.

Par exemple : avec 4 niveaux de gris, et une structure de voisinage différente en considérant les cliques à deux éléments parmi les cliques suivantes :

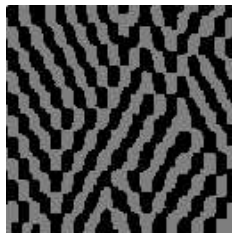
 $\beta = 10$  $\beta = 0.66$

## Exemple de modèle de texture

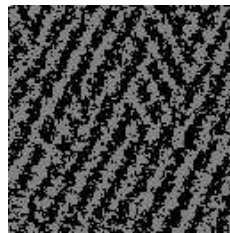
On favorise les interactions à courte distance et pénalise les interactions à longue distance :

$$\begin{array}{ll}
 \begin{array}{c} \circ - \circ \\ | \\ \circ - \circ \end{array} & V = -\frac{3}{T} * \mathbf{1}(x_s = x_t) \\
 \begin{array}{c} | \\ \circ - \circ \end{array} & V = +\frac{1}{T} * \mathbf{1}(x_s = x_t)
 \end{array}$$

$T$  est un paramètre de **température** qui contrôle la force des interactions.



$T = 1$



$T = 5$

---

---

## Le triangle d'or

Pour construire une texture : trois réglages.

- La structure de voisinage : définition des cliques considérées.
- Les potentiels  $\beta$  associés à chaque cliques ( $\beta > 0$  pour un comportement attractif et  $\beta < 0$  pour un comportement répulsif).
- La température  $T$  qui régule la force des interactions.

---

---

## Champs de Markov Gaussiens

Dans le cas des images avec 255 niveaux de gris, le modèle de Potts n'est pas intéressant : le couple (45,46) est considéré de la même manière que (45,220).

- Pour les cliques d'ordre 2 (texture) :

$$V_c(x_s, x_t) = \beta(x_s - x_t)^2$$

- Pour les cliques d'ordre 1 (restauration)

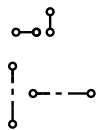
$$V_c(x_s) = \alpha(x_s - \mu)^2$$

$\mu$  est un paramètre donné par une image d'attache aux données.

---

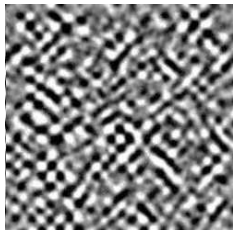
---

Exemple

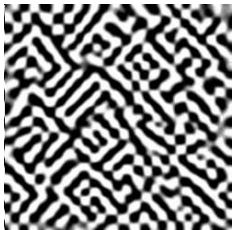


$$V = -\frac{1}{T} * (x_s - x_t)^2$$

$$V = +\frac{0.08}{T} * (x_s - x_t)^2$$



T=1



T=0.1

---

---

## Ajouter une texture à une image

Si l'on ajoute une clique d'ordre 1 :

○ 
$$V = 0.3 * (x_s - \mu)^2$$

Où  $\mu$  représente une valeur dépendant du pixel  $s$  : par exemple, la valeur d'une autre image (lena) en ce pixel.



T=1



T=0.1

---

---

## Simuler un champ de Markov

L'algorithme le plus simple est l'algorithme de Metropolis-Hasting :

1. Pour un pixel de  $s$  de l'image, on a une valeur courante  $x_s$ , on propose aléatoirement une nouvelle valeur  $x'_s$ ,
2. On calcule la variation d'énergie induite par cette transformation :

$$\Delta U(x'_s, x_s) = U(x'_s) - U(x_s)$$

3. on calcule la probabilité  $p$  :

$$p = \min(1, \exp(-\Delta U(x'_s, x_s)))$$

4. Avec une probabilité
  - $p$  on accepte la perturbation :  $x_s \leftarrow x'_s$ ,
  - $1 - p$  on refuse la perturbation,  $x_s$  reste constant.
5. Retour en 1 : on passe au pixel suivant, sauf si l'on a parcouru l'image entièrement un nombre suffisant de fois (plusieurs centaines).

---

---

## Metropolis-Hasting et Propriété de Markov

Il est essentiel de remarquer que la variation d'énergie est facile à calculer grâce à la formulation **locale** des Champs de Markov.

On peut ne considérer que les voisins du pixel perturbé.



## Chapitre 12

# Stéréovision

---

---

## Stéréovision

La stéréovision décrit les techniques permettant de reconstruire une structure tridimensionnelle à partir de deux images (ou plus).



Le cerveau humain réalise cette opération en permanence.

---

---

## Les étapes de la stéréovision

On distingue en général les étapes suivantes :

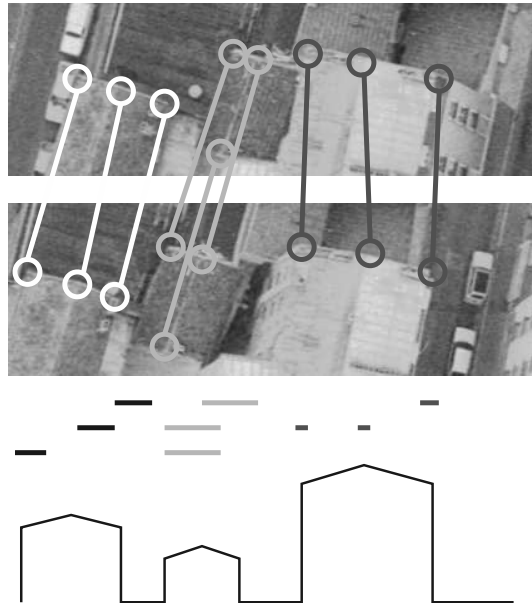
- **La Rectification:** Simplification du problème, correction des images
- **Mise en correspondance:** Étape essentielle
- **Régularisation:** Amélioration du résultat
- **Triangulation:** Déduction de la structure 3D à proprement parler.

---

---

## Comment ça marche?

Le décalage est proportionnel à la hauteur de l'objet.



Cette technique est totalement géométrique.

---

---

## Commentaires

### Mise en correspondance

S'il est facile pour un être humain de savoir quels points sont en correspondance dans les deux images, c'est une opération difficile à faire automatiquement.

La **mise en correspondance** consiste à trouver les couples de points en correspondance.

### Autres techniques que la stéréovision

Il est possible d'utiliser une seule image :

- en utilisant les ombres portées **Shape from shadow**
- en utilisant l'ombrage de l'objet **Shape from shading**

---

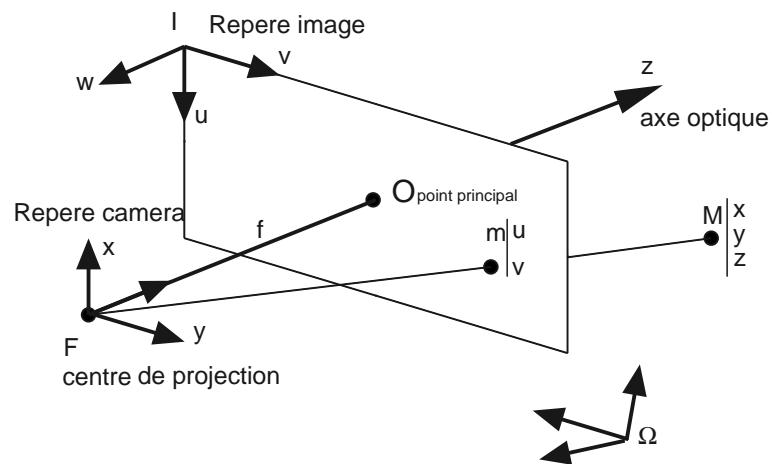
---

## Caméra et Géométrie

On a **3** repères :

- $R_m$  **Repère monde** :  $\Omega, X, Y, Z$ . Un point  $M$  a les coordonnées  $(X, Y, Z)$
- $R_c$  **Repère caméra** :  $F, x, y, z$ . Le même point  $M$  a les coordonnées  $(x, y, z)$ .
- $I$  **Repère image** : le point  $m$ , projeté de  $M$  dans l'image a les coordonnées  $(u, v)$ .

## La projection



---



---

## Du repère Monde au repère Caméra

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}_{Camera} = R * \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{Monde} + T$$

$T$  est le vecteur de **translation** :

$$T = \overrightarrow{F\Omega}$$

$R$  est la matrice de **rotation** :

$$R = [X, Y, Z]_{Camera} \quad R = R_z R_y R_x$$

$R_z, R_y, R_x$  représentent les rotations de la caméra autour des axes  $(\Omega Z), (\Omega Y), (\Omega X)$



---



---

## Du repère Caméra a l' Image

**La projection.** Un point de l'image a une ordonnée qui vaut  $f$  ( focale de l'image dans le repère caméra.

$$\begin{pmatrix} x \\ y \\ f \end{pmatrix}_{Camera} = \frac{f}{Z} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{Camera}$$

**Du point au pixel :**  $p_u$  et  $p_v$  sont deux paramètres de numérisation

$$\begin{cases} u &= p_u.x + u_0 \\ v &= p_v.y + v_0 \end{cases}$$

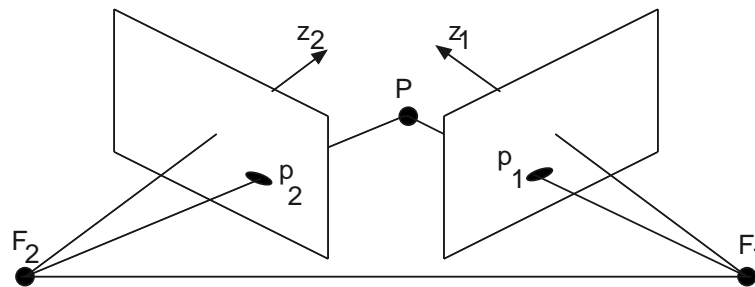
Sur le schéma précédent :  $p_u < 0$  et  $p_v > 0$ .

---

---

## Triangulation

Un point  $P$  du monde est observé dans deux images : les projetés sont appelés  $p$  et  $p'$ .



---



---

## Entre deux images

Deux repères caméra  $F_1, x_1, y_1, z_1$  et  $F_2, x_2, y_2, z_2$ . On considère un point  $M$  :

$$\begin{aligned}
 \overrightarrow{F_2 M}_{Camera2} &= R_2 \overrightarrow{\Omega M}_{Monde} + T_2 \\
 &= R_2 (R_1^t (\overrightarrow{F_1 M}_{Camera1} - T_1)) + T_2 \\
 &= R_2 R_1^t \overrightarrow{F_1 M}_{Camera1} + (T_2 - R_2 R_1^t T_1) \\
 &= R^{inter} \overrightarrow{F_1 M}_{Camera1} + T^{inter}
 \end{aligned}$$

Où l'on a la matrice de rotation :

$$R^{inter} = R_2 R_1^t = \begin{bmatrix} r_1^t \\ r_2^t \\ r_3^t \end{bmatrix}$$

et le vecteur de translation :

$$T^{inter} = T_2 - R^{inter} T_1$$

## Des Images au Monde

On a 2 caméra et leurs différents paramètres :  
 $R_1, R_2, T_1, T_2, f_1, f_2$ .

Un point  $P$  du monde a les deux projetés  $p_1$  et  $p_2$  :

$$p_1 \begin{pmatrix} x_1 \\ y_1 \\ f_1 \end{pmatrix} \quad p_2 \begin{pmatrix} x_2 \\ y_2 \\ f_2 \end{pmatrix}$$

Les coordonnées monde  $X, Y, Z$  vérifient les quatre équations:

$$\begin{cases} x_1 Z - f_1 X = 0 \\ y_1 Z - f_1 Y = 0 \\ x_2(r_3^t X + T_3^{inter}) - f_2(r_1^t X + T_1^{inter}) = 0 \\ y_2(r_3^t X + T_3^{inter}) - f_2(r_2^t X + T_2^{inter}) = 0 \end{cases}$$

On obtient donc un système du type  $H\mathbf{X} = \mathbf{c}$ , que l'on résout par les moindres carrés :

$$\tilde{\mathbf{X}} = (H^t H)^{-1} H^t \mathbf{c}$$

ce qui correspond à l'optimisation suivante :

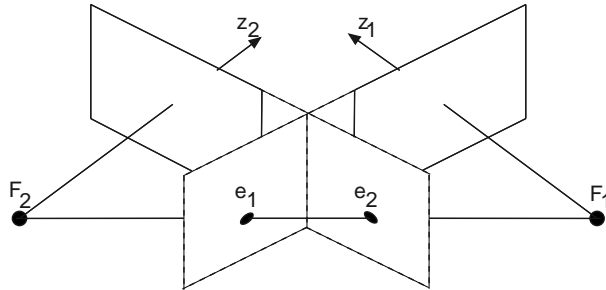
$$\tilde{\mathbf{X}} = \text{Argmin} \|H\mathbf{X} - \mathbf{c}\|^2$$

---

---

## Les épipoles

On appelle **épipoles** les projetés des centres optiques  $F_1$  et  $F_2$  dans les autres images.



Quand les caméras sont parallèle, les épipoles sont à l'infini.

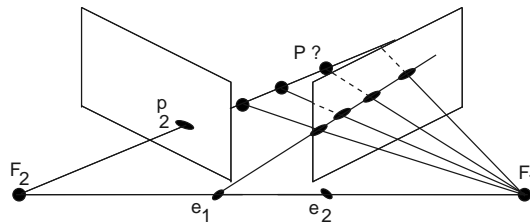
---

## La contrainte épipolaire

Les points  $p_1, p_2, F_1, F_2$  sont coplanaires. Cela s'écrit :

$$\overrightarrow{F_2 p_2} \cdot (\overrightarrow{F_1 F_2} \wedge \overrightarrow{F_1 p_1}) = 0$$

Cette équation caractérise la géométrie épipolaire



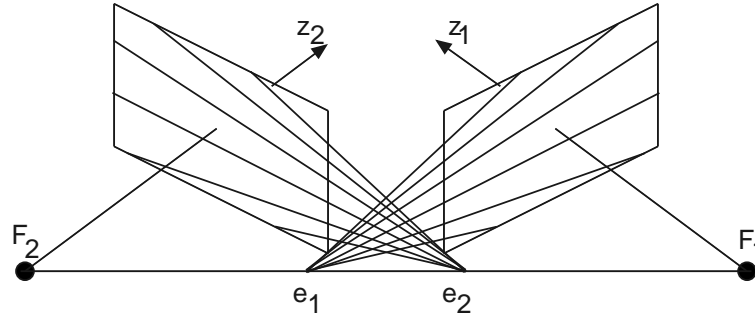
**Lien avec la mise en correspondance :** Lorsque l'on dispose d'un point  $p_2$  les hypothèses de points  $p_1$  possibles décrivent une droite dans l'image 1 qui passe par l'épipole  $e_1$ .

---

---

## Les lignes épipolaires

Si l'on considère l'ensemble des lignes épipolaires :



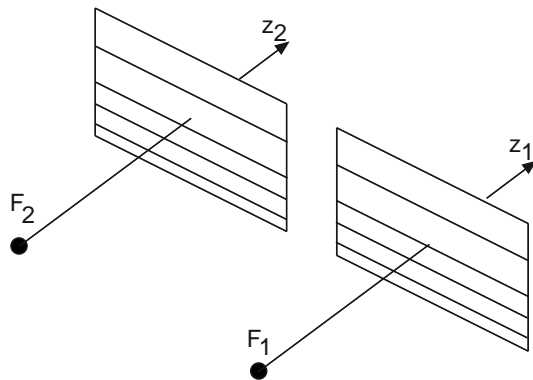
La connaissance des lignes épipolaires permet de simplifier la mise en correspondance.

---

---

## La configuration parallèle

Il existe une configuration intéressante des capteurs, c'est la configuration parallèle :



Les épiholes sont à l'infini et les lignes épipolaires sont parallèles.



---

---

## La rectification

On peut simuler une configuration parallèle des capteurs : le but est donc de prendre les 2 images et d'en construire 2 nouvelles qui sont des simulations de ce que l'on aurait observé si les caméras avaient été parallèle.

Pour cela il faut que les lignes épipolaires dans les images soient parallèles. On considère donc le repère suivant exprimé dans le repère monde :

$$\overrightarrow{i_{rect}} = \frac{\overrightarrow{C_1 C_2}}{\|C_1 C_2\|}$$

On prend  $M$  un point de la scène :

$$\overrightarrow{k_{rect}} = (\overrightarrow{MC_1} \wedge \overrightarrow{MC_2}) \wedge \overrightarrow{i_{rect}}$$

$\overrightarrow{k_{rect}}$  vise donc vers la scène

$$\overrightarrow{j_{rect}} = \overrightarrow{k_{rect}} \wedge \overrightarrow{i_{rect}}$$

Cela donne la matrice de Rotation pour passer du repère monde au repère rectifié :

$$R_{rect} = [\overrightarrow{i_{rect}}, \overrightarrow{j_{rect}}, \overrightarrow{k_{rect}}]^t$$


---

---

---

---

## Rectifier une image

On a un point dans une image de coordonnées  $(u, v)$ , le but est d'obtenir ses coordonnées  $(u', v')$  dans l'image rectifiée.

En fait on fait l'inverse : on va remplir l'image rectifiée.

On parcourt l'image rectifiée, pour un pixel  $(u', v')$  on recherche le point correspondant dans l'image d'origine et le niveau de gris associés :

1. On passe des coordonnées image rectifiées  $(u', v')$  aux coordonnées caméra rectifiée,
2. des coordonnées caméra rectifiée  $(x', y', f)$  aux coordonnées caméra d'origine,
3. des coordonnées caméra d'origine  $(x, y, f)$  aux coordonnées image  $(u, v)$

---



---

## Rectification de l'image 1

On note  $A$  la matrice :

$$A = \begin{pmatrix} f.p_u & 0 & u_0 \\ 0 & f.p_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

On note  $|\cdot|_i$  la  $i$ ème coordonnée d'un vecteur :  
 $|a,b,c|_3 = c$ ,

$$m = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \quad m' = \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix}$$

et

$$R_{1rect} = R_{rect}R_1^t \quad \text{donc} \quad R_{1rect}^t = R_1R_{rect}$$

Le passage de l'image rectifiée à l'image d'origine par la relation :

$$m = \frac{1}{|A_1R_{1rect}^tA_1^{-1}m|_3} A_1R_{1rect}^tA_1^{-1}m'$$

---



---

## Rectification de l'image 2

L'image 2 peut éventuellement avoir des paramètres différents. Cependant on veut que les paramètres des deux images soient les mêmes.

Pour passer de l'image 2 rectifiée à l'image 2 on peut appliquer la transformation suivante :

$$m = \frac{1}{|A_2 R_{2rect}^t A_1^{-1} m|_3} A_2 R_{2rect}^t A_1^{-1} m'$$

La matrice de rotation  $R_{2rect}$  est obtenue de la même façon que  $R_{1rect}$  en remplaçant  $R_1$  par  $R_2$ .

En pratique, on a souvent :  $A_1 = A_2 = A$ .

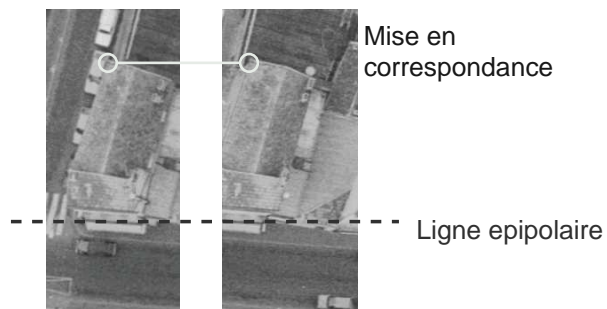
---

---

## Mise en correspondance

Pour calculer la troisième dimension (triangulation), il faut mettre les pixels en correspondance.

grâce à la rectification, la recherche d'un point homologue est simplifiée: les deux points appartiennent à la même ligne.



---

---

## Image de disparité

C'est l'image des décalages de l'image 2 par rapport à l'image 1.

On appelle **appariement** la mise en correspondance de 2 pixel :

$$(i,j) \sim (i,k)$$

On appelle **disparité** le décalage :

$$D(j) = k - j$$

Si l'on prend pour référence l'image 2 :

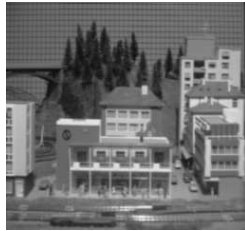
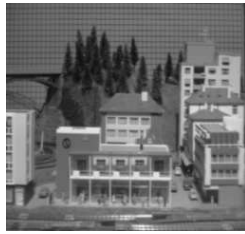
$$D(k) = j - k$$

La mise en correspondance consiste en le calcul des disparité, en partant de l'une ou l'autre des images, ou des deux.

---

---

### Exemple



Couple stéréo

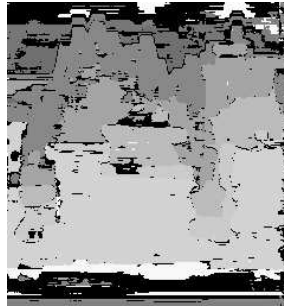


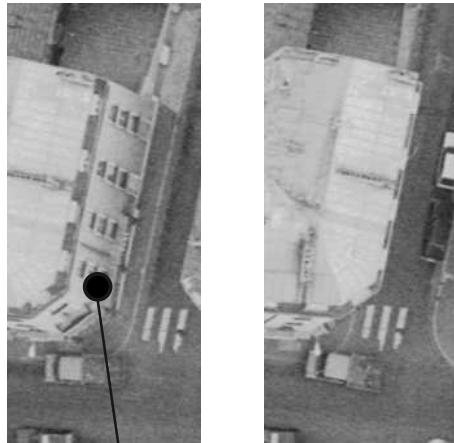
Image de disparité obtenue

---

---

## Occlusions

On appelle “zones d’occlusions” les zones de pixels visibles sur une image mais absents sur l’autre :



Zone d'occlusion



---



---

## Corrélation

Mesure de ressemblance entre deux pixels par comparaison du voisinage. On note  $g_{i,j}$  et  $d_{k,l}$  les pixels  $(i,j)$  et  $(k,l)$  de l'image gauche  $g$  et droite  $d$ . La corrélation entre les deux pixels avec une fenêtre de taille  $r$ :

$$C(g_{i,j}, d_{k,l}) = \frac{1}{(2r+1)(2r+1)} \sum_{m=-r, n=-r}^{r,r} \frac{(g_{i+m, j+n} - \bar{g}_{i,j})}{\sigma g_{i,j}} \frac{(d_{k+m, l+n} - \bar{d}_{k,l})}{\sigma d_{k,l}}$$

Avec la moyenne:

$$\bar{g}_{i,j} = \frac{1}{(2r+1)(2r+1)} \sum_{m=-r, n=-r}^{r,r} g_{i+m, j+n}$$

Et la variance :

$$\sigma g_{i,j}^2 = \frac{1}{(2r+1)(2r+1)} \sum_{m=-r, n=-r}^{r,r} (g_{i+m, j+n} - \bar{g}_{i,j})^2$$


---



---

---

---

## Commentaires

Il existe d'autres critères de corrélation. Par exemple :

$$C(g_{i,j}, d_{k,l}) = \frac{1}{(2r+1)(2r+1)} \sum_{m=-r, n=-r}^{r,r} (g_{i+m, j+n} - d_{k+m, l+n})^2$$

La taille  $r$  de la fenêtre est un paramètre important (plus il est grand, meilleur est le résultat).

Plus la corrélation est grande, plus les pixels se ressemblent au sens du voisinage.

---

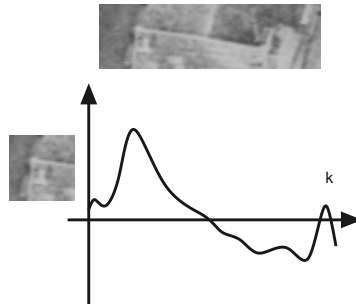


---

## Premier algorithme

1. on choisit une image, par exemple 1,
2. on parcourt les pixels de  $(i,j)$  de l'image 1,
  - pour chaque pixel  $(i,j)$ , on parcourt la ligne  $i$  de l'image 2,
  - pour chaque pixel  $(i,k)$  de l'image 2: on calcule la corrélation  $C(g_{i,j}, d_{i,k})$
- on prend le maximum de corrélation :

$$D(i,j) = \text{Argmax}_k C(g_{i,j}, d_{i,k}) - j$$

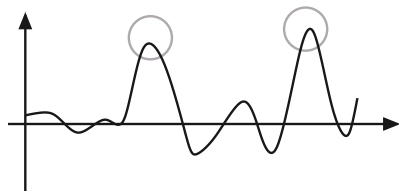


---

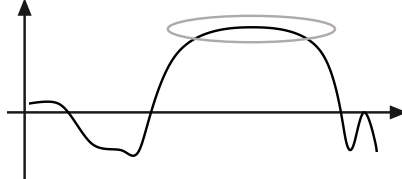
---

## Les problèmes

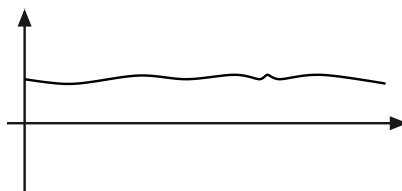
- Plusieurs pics



- Pic large



- Corrélation plate



---

---

## Les solutions

1. Ne garder que les pics supérieurs à un seuil,
2. Ne garder que les pics “pas trop large”
3. Utiliser un algorithme Aller/Retour,
4. Utiliser une méthode de programmation dynamique

---

---

## Aller retour

On ajoute l'étape suivante :

Une fois le pixel optimal  $(i,k)$  de la ligne correspondante dans l'image 2 trouvé, on vérifie que le pixel d'origine  $(i,j)$  de l'image 1 est lui-même le pixel optimal pour le pixel sélectionné  $(i,k)$ .

Il faut vérifier que le système suivant est valide :

$$k = \operatorname{Argmax}_l C(g_{i,j}, d_{i,l})$$

$$j = \operatorname{Argmax}_l C(d_{i,k}, g_{i,l})$$

On rajoute souvent la condition de seuil sur la corrélation :

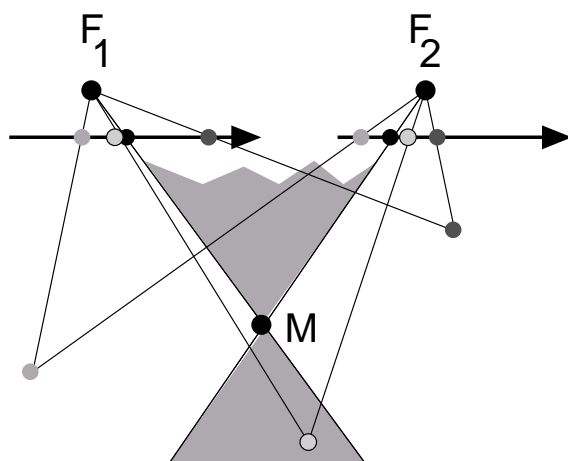
$$C(g_{i,j}, d_{i,k}) \geq T$$

---



---

## Ordre de projection



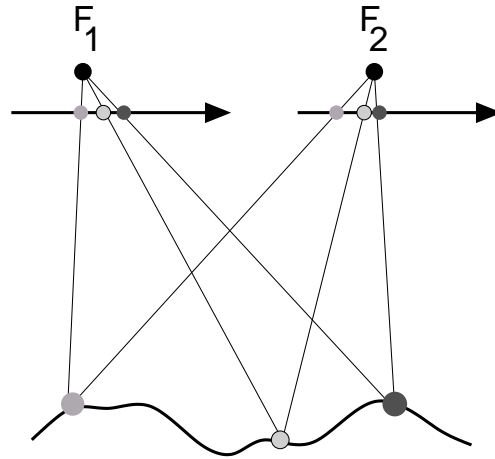
Excepté pour les points situés dans le cône ombré, les projections des points gardent le même ordre par rapport au point  $M$ .

---



---

### Contrainte d'ordre



Si le relief observé n'est pas très chahuté par rapport à la distance d'observation, la **contrainte d'ordre** est respecté :

**Différents points ont le même ordre de projection dans les deux images**

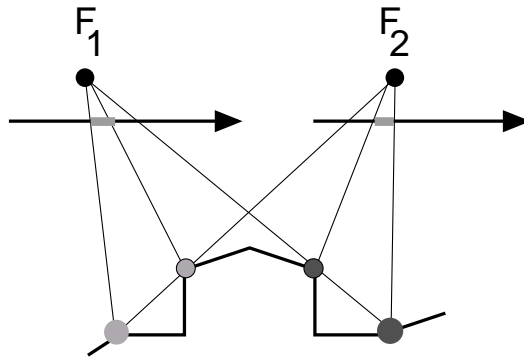


---

---

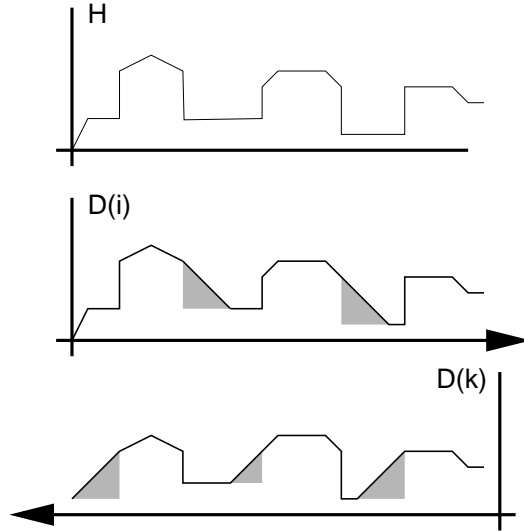
## Occlusions

Les zones d'occlusions sont des zones où l'on perd l'unicité de l'appariement.



## Occlusions et ordre de parcours

En fonction de l'ordre de parcours pour calculer la disparité, les zones d'occlusions provoquent des profils de disparité différent (à cause de la contrainte d'ordre):



**Remarque :** La contrainte d'ordre implique que dans le sens de parcours, la courbe de disparité ne peut pas avoir une vitesse de décroissance inférieure à  $-1$ .

Ce transparent repose sur l'hypothèse que la hauteur et la disparité ont un comportement proportionnel, ce qui n'est vrai que si la scène est observée à une distance suffisante du couple de caméra. En vrai la relation s'écrit :

$$d = \frac{fB}{Z}$$

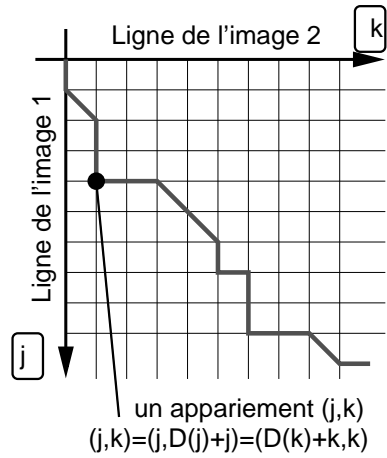
où  $Z$  est la distance entre le point et les caméra,  $f$  la focale,  $B$  la distance entre les deux foyers et  $d$  la disparité. En effectuant un développement de Taylor sur  $Z = H_{camera} - h$ , on retrouve une relation de proportionnalité :

$$\delta h \approx -\left(\frac{H_{camera}}{B}\right) \frac{H_{camera} \delta d}{f}$$

On voit que la précision de la reconstruction est donnée par le rapport  $B/H$ .

## Appariement simultané

Si l'on cherche l'appariement en même temps dans les deux sens :

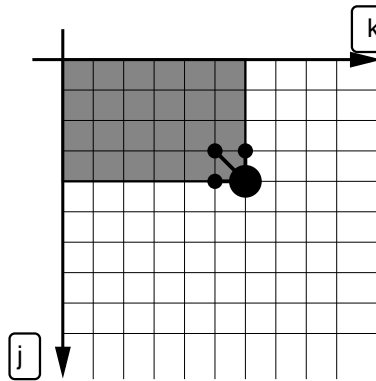


On peut voir un appariement comme un chemin : il s'agit de deux calculs de disparité simultanés, l'un en prenant comme référence l'image 1 ( $D(j)$ ) et l'autre l'image 2 ( $D(k)$ ).

## Appariement simultané et contrainte d'ordre

Les deux contraintes d'ordres (vitesse  $\geq -1$ ) permettent d'affirmer qu'un appariement  $(j, k)$  n'a que trois prédécesseurs possibles :

$$(j, k-1) \quad (j-1, k) \quad (j-1, k-1)$$



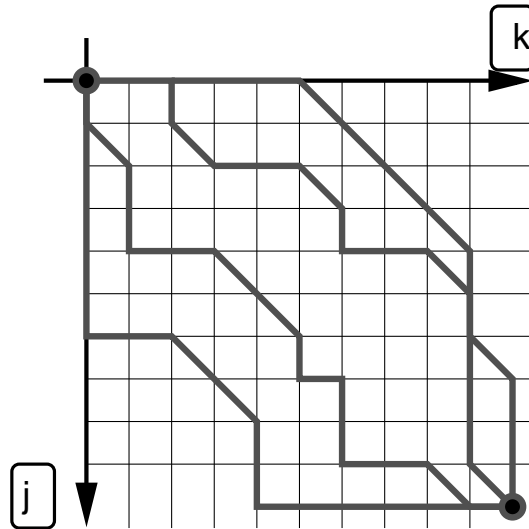
Cette contrainte d'ordre permet un schéma de programmation dynamique.

---

---

## Programmation dynamique (1)

On transforme le problème en un problème de minimisation sur les chemins de mise en correspondance qui appartiennent  $(0,0)$  et  $(nc,nc)$  :



---



---

## Problème de minimisation

Formulation de  $U$ , énergie à minimiser :

Pour un chemin d'appariement

$$L = ((j_1, k_1), \dots, (j_p, k_p), \dots, (j_n, k_n))$$

Avec la contrainte :

$$(j_1 = 1, k_1 = 1) \quad (j_n = nc, k_n = nc)$$

On pose comme énergie , avec  $\tilde{C}(j_p, k_p) = C(g(i, j_p), d(i, k_p))$

$$U(L) = - \sum_{p=1}^n \tilde{C}(j_p, k_p)$$

Une mise en correspondance recherche donc  $L^*$  :

$$L^* = \text{Argmin } U(L)$$

---

---

## Formulation récursive

La solution de la mise en correspondance peut décrire de manière récursive grâce à la contrainte d'ordre.

Si l'on regarde un chemin partiel de longueur  $m$  :

$$L_m = ((j_1, k_1), \dots, (j_p, k_p), \dots, (j_m, k_m))$$

L'énergie minimale pour atteindre  $(j_m, k_m)$  est donnée par  $u^*(j_m, k_m)$  où :

$$u^*(j, k) = -\tilde{C}(j, k) + \min\{u^*(j-1, k), u^*(j, k-1), u^*(j-1, k-1)\}$$

---

---

## Programmation dynamique

1. calcul de tous les  $\tilde{C}(j,k)$
2. **Première passe** : calcul des  $u^*(j,k)$  :
  - on part de  $u^*(1,1)$ ,
  - on itère sur  $p$  pour calculer  $u^*(j \leq p, k \leq p)$
3. **Seconde passe** : détermination de  $L^*$  :
  - on part de  $u^*(nc,nc)$  : la fin de  $L^*$  est donc le couple  $(nc,nc)$
  - on cherche dans les trois prédécesseurs celui d'énergie  $u^*(.,.)$  minimale, et on l'ajoute au début de  $L^*$ ,
  - on itère jusqu'à  $(1,1)$  que l'on ajoute au début de  $L^*$ .



---

---

## Coût de transition

Une transition verticale ou horizontale sur le graphe de mise en correspondance correspond à une zone d'occlusion.

On peut, pour pénaliser ou favoriser la détection d'occlusions ajouter un coût de transition  $t$ .

La formulation récursive devient alors :

$$u^*(j,k) = -\tilde{C}(j,k) + \min\{u^*(j-1,k) + t, u^*(j,k-1) + t, u^*(j-1,k-1) + t\}$$

Pour  $t > 0$  la détection d'occlusion est pénalisée.

---

---

## Intervalle de disparité

Pour toutes les méthodes, on rajoute en général une condition sur la disparité qui permet de limiter la taille du problème à résoudre :

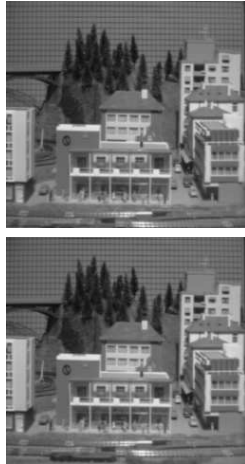
$$D(.) \leq D_{\max}$$

---

---

## Exemple

Voici les deux images tests : (les lignes épipolaires sont verticales)



Couple stéréo

---

---

## Exemple 2

**AL** : Aller-retour, **PD** : Programmation dynamique.

