# Pyramides Combinatoires & Segmentation

## Journées du GdR IM

Luc Brun

Groupe de Recherche en Informatique, Image, Automatique et Instrumentation de Caen (GREYC)

5 novembre 2007

# Plan

Pyramides Combinatoires Où se situe t'on ?

$$\{\text{ Pyramides Irrégulières}\} \cap \{\text{ Cartes Combinatoires}\}$$

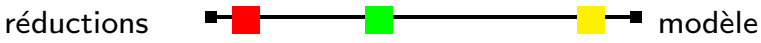
Une pyramide combinatoire est une pyramide irrégulière composée de cartes combinatoires.

## Pyramides irrégulières : Def 1

- Une pyramide irrégulière est constituée d'une
  - pile de graphes successivement réduits.
- avantages :
  - opérations de réduction applicables en parallèle
  - description d'un même objet à plusieurs niveaux de détails (e.g. image)

Pyramides irrégulières : Def 2 Par quoi est défini un modèle de pyramide irrégulière ?

- Le modèle de graphe utilisé (modèle),
- les opérations de réductions (réduction).



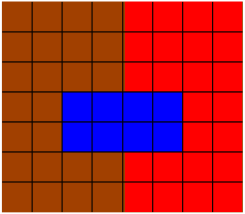

Noyau
Décimation guidée par les données
Décimation par couplage maximal

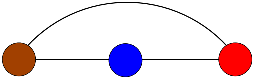
Graphes simples
Graphes duaux
Cartes combinatoires

- Jolion (LIS)
- Kropatsch(PRIP)
- Brun (GREYC)

# Graphe simple

1. pas de boucles, 
2. Pas d'arêtes doubles 





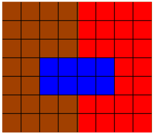
Pas de multi-adjacence entre régions,

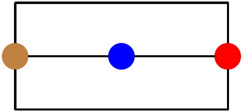
Pas de caractérisation des relations d'inclusions,

Conclusion : Description «pauvre» d'un objet riche : la partition.

# Graphes duaux : Définition

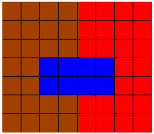
- Modèle des graphes duaux : $(G, \overline{G})$
- $G = (V, E)$ non simple,

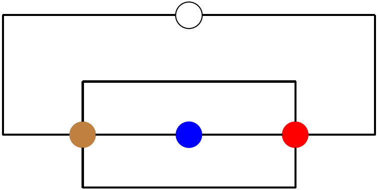




# Graphes duaux : Définition

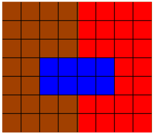
- Modèle des graphes duaux : $(G, \overline{G})$
- $G = (V, E)$ non simple,
  - ◯ code l'extérieur de l'image

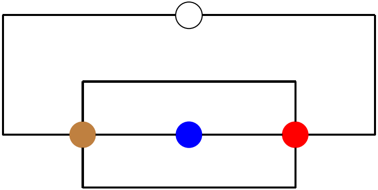




# Graphes duaux : Définition

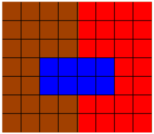
- Modèle des graphes duaux : $(G, \overline{G})$
- $G = (V, E)$ non simple,
- $\overline{G} = (\overline{V}, \overline{E})$

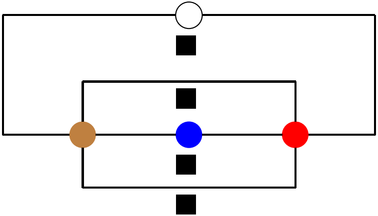




# Graphes duaux : Définition

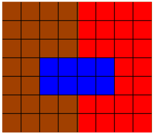
- Modèle des graphes duaux : $(G, \overline{G})$
- $G = (V, E)$ non simple,
- $\overline{G} = (\overline{V}, \overline{E})$
  - $\overline{V}$ : un sommet de $\overline{G}$ par face de $G$.

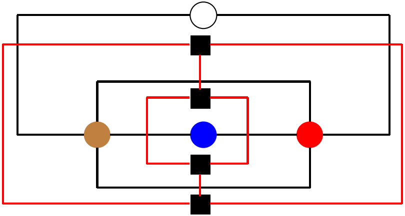




# Graphes duaux : Définition

- Modèle des graphes duaux : $(G, \overline{G})$
- $G = (V, E)$ non simple,
- $\overline{G} = (\overline{V}, \overline{E})$
  - $\overline{V}$ : un sommet de $\overline{G}$ par face de $G$.
  - $\overline{E}$ : chaque $\overline{e} \in \overline{E}$ coupe une et une seule arête de $E$.





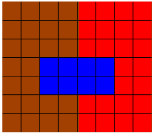
Résumé :

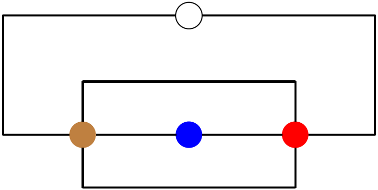
- Pyramides de graphes simples (Meer,Montanvert, Jolion. . .),
  - ☹ Pas d'arêtes multiples.
  - ☹ Pas de relations d'inclusions.
- Pyramides de graphes duaux (W.G. Kropatsch,. . .).
  - ☺ arêtes multiples,

    Mais :
    - ☹ impossibilité de différentier certaines configurations (inclusion),
    - ☹ stockage de deux graphes.

# Cartes Combinatoires : Les arêtes

- $G = (\mathcal{D}, \sigma, \alpha)$

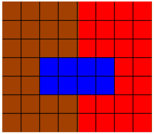


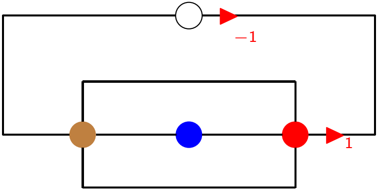


- Chaque arête est découpée en deux demi arêtes appelées brins.

# Cartes Combinatoires : Les arêtes

- $G = (\mathcal{D}, \sigma, \alpha)$

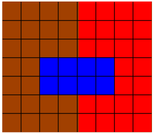


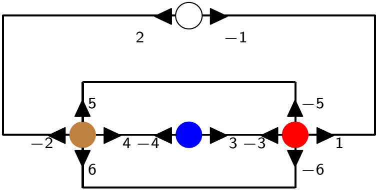



- Les deux brins d'une même arête sont liés par une involution $\alpha : \alpha(1) = -1, \alpha(-1) = 1$

# Cartes Combinatoires : Les arêtes

- $G = (\mathcal{D}, \sigma, \alpha)$

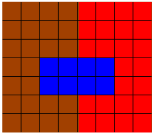


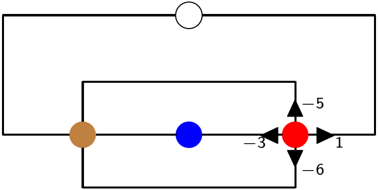



$$\mathcal{D} = \{-6, \ldots, -1, 1, \ldots, 6\}$$
$$\forall b \in \mathcal{D} \ \alpha(b) = -b$$

$$\alpha = (1, -1)(2, -2)(3, -3)(4, -4)(5, -5)(6, -6)$$

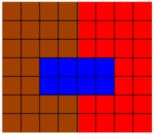
# Les sommets

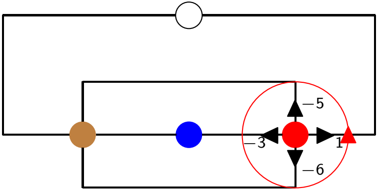





- Les sommets sont codés par les cycles de $\sigma$.

# Les sommets

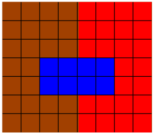


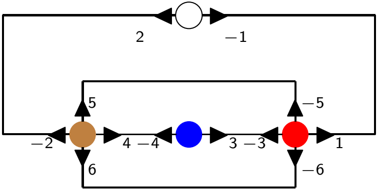



- $\sigma^*(b)$ correspond à la suite de brins rencontrés en tournant dans le sens positif autour du sommet contenant $b$.

$$\sigma^*(1) = (1, -5, -3, -6)$$

# Les sommets



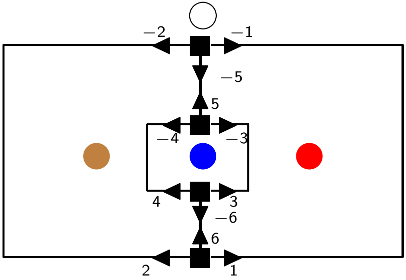



$$\sigma = (1, -5, -3, -6)(6, 4, 5, -2)(2, -1)(3, -4)$$

# Calcul du dual

- Si $G = (\mathcal{D}, \sigma, \alpha)$ alors $\overline{G} = (\mathcal{D}, \varphi = \sigma \circ \alpha, \alpha)$.
- Les cycles de $\varphi$ codent les faces de la carte duale (et donc la carte duale).

$$\varphi = (-2, -1, -5)(-4, 5, -3)(4, 3, -6)(2, 6, 1)$$



# Cartes Combinatoires : Bilan

- Codage implicite du dual : On travaille avec un seul «graphe»

# Cartes Combinatoires : Bilan

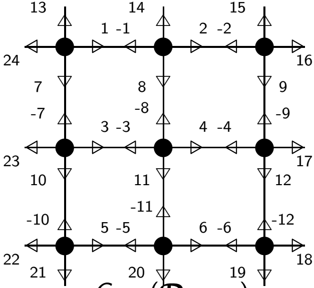
- Codage implicite du dual : On travaille avec un seul «graphe»
- On code explicitement l'orientation
  - On peut efficacement (cardinal d'un sommet) caractériser les relations d'inclusion.

# Cartes Combinatoires : Bilan

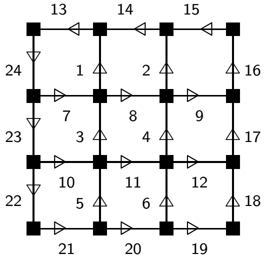
- Codage implicite du dual : On travaille avec un seul «graphe»
- On code explicitement l'orientation
  - On peut efficacement (cardinal d'un sommet) caractériser les relations d'inclusion.
- Elles peuvent de plus être étendues à des dimensions supérieures (3D,4D,. . .nD).

# Construction de la pyramide

- Une grille $3x3$ :

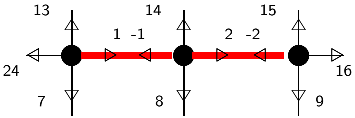


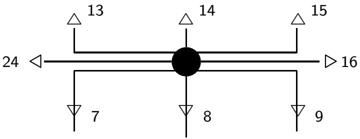

$G = (\mathcal{D}, \sigma, \alpha)$




$\overline{G} = (\mathcal{D}, \varphi = \sigma \circ \alpha, \alpha)$

## Réduction : Contraction

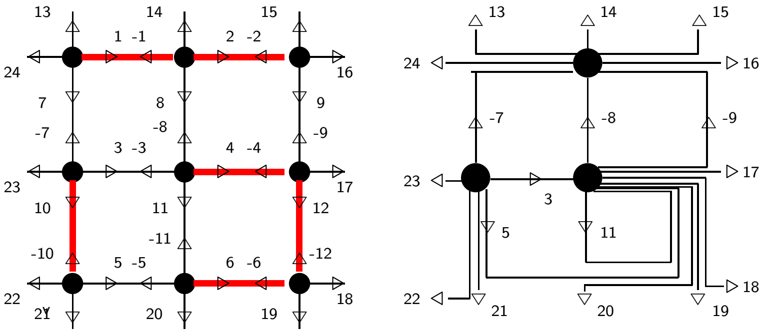


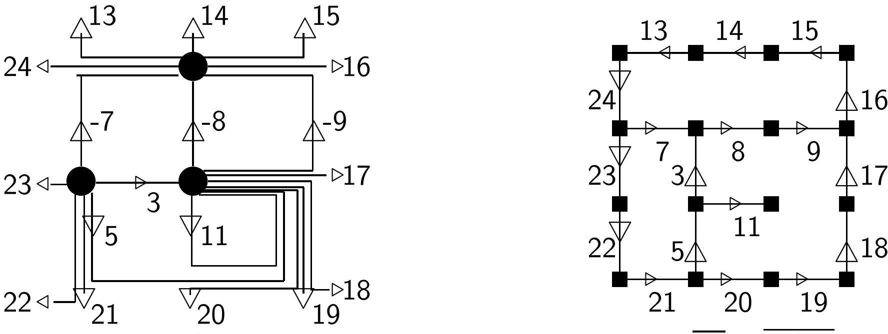

- Contraction d'une arête e=(u,v) :
  - Identification de *u* et *v*,
  - suppression de *e*.

# Réduction : noyau de contraction

- Forêt du graphe initial

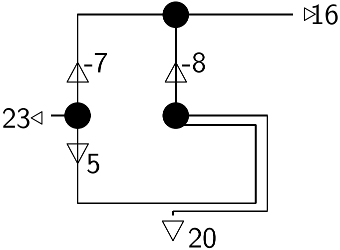


# Simplification : Noyau de suppression

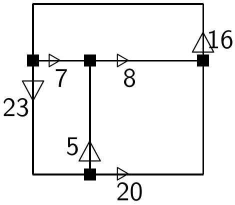



$G_1 = G/K_1$ $\qquad$ $\overline{G_1} = \overline{G/K_1}$

# Simplification : Noyau de suppression




$(G/K_1) \setminus K_2$



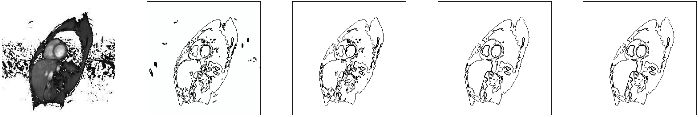

$\overline{(G/K_1) \setminus K_2}$

Construction de la pyramide

$$\begin{array}{ll} G_i & \\ \downarrow & \text{noyau de contraction} \\ G_{i+1} & \\ \downarrow & \text{noyau de suppression} \\ G_{i+2} & \end{array}$$

$\mathcal{P} = (G_0, G_1 \dots, G_n)$.

## Construction explicite



. . .

original niveau 1 niveau 2 niveau 3 niveau 4

Inconvénient :

- Codage de toutes les cartes réduites : $G_0, G_1 \dots, G_n$
  - Pyramides de hauteur limitée ($n < N_0$),
  - $\Rightarrow$Facteur de décimation fixe :

$$|G_{i+1}| \approx q|G_i|, \ \ q < 1 \Rightarrow n = log_{\frac{1}{q}}(|G_0|)$$

    - Noyaux,
    - Décimation guidée par les données,
    - Décimation par couplage maximal.
  - $\Rightarrow$ Contrainte sur la segmentation

Du codage explicite $\mathcal{P} = (G_0, G_1 \dots, G_n)$ avec $G_i = (\boldsymbol{\mathcal{D}}_i, \sigma_i, \alpha)$
On remarque que :

$$\boldsymbol{\mathcal{D}}_n \subset \boldsymbol{\mathcal{D}}_{n-1} \cdots \subset \boldsymbol{\mathcal{D}}_0$$

Deux fonctions :

$$\left\{ \begin{array}{ll} \forall b \in \boldsymbol{\mathcal{D}}_O & b \in \boldsymbol{\mathcal{D}}_{niveau(b)-1}, b \notin \boldsymbol{\mathcal{D}}_{niveau(b)} \\ \forall i \in \{1, \dots, n\} & etat(i) \in \{Contract\acute{e}, Supprim\acute{e}\} \end{array} \right.$$

Au codage Implicite : $\mathcal{P} = (G_0, niveau, etat)$, $|\mathcal{P}| \approx \theta(3|\boldsymbol{\mathcal{D}}_0|)$

Compression maximum : $\mathcal{P} = (niveau)$
$|\mathcal{P}| \approx \theta(|\boldsymbol{\mathcal{D}}_0|)$
Codage Utilisé : $\mathcal{P} = (G_0, G_n, niveau)$
$|\mathcal{P}| = \mathcal{O}(5|\boldsymbol{\mathcal{D}}_0|)$

## Utilisation du codage

- ascendant

| | séquentiel | parallèle |
|---|---|---|
| $G_i$ | $\mathcal{O}(\lvert\partial G_i\rvert)$ | $\mathcal{O}(\log(\lvert\partial G_i\rvert))$ |
| $(G_1,\ldots,G_n)$ | $\mathcal{O}(\lvert\mathcal{D}_0\rvert)$ | $\mathcal{O}(\log(\lvert\mathcal{D}_0\rvert))$ |

- descendant :

| | | séquentiel | parallèle |
|---|---|---|---|
| *région* | $\sigma_i^*(b)$ | $\mathcal{O}(\lvert R_{\sigma_i^*(b)}\rvert)$ | $\mathcal{O}(\log(R_{\sigma_i^*(b)}))$ |
| *frontière* | $b$ | $\mathcal{O}(\lvert\partial RF_i(b)\rvert)$ | $\mathcal{O}(\log(\lvert\partial RF_i(b)\rvert))$ |

# Bilan

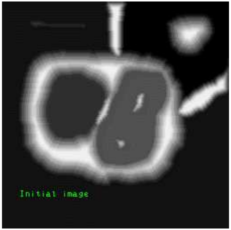
- Taille indépendante (en pratique) du nombre de niveaux.
- Description fine de la partition :

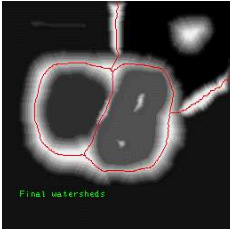
| | adjacence | contours | inclusions | géométrie |
|---|---|---|---|---|
| Pyr. Graphes Simples | Oui | Non | Non | Non |
| Pyr. Graphes Duaux | Oui | Oui | calcul global ? | Oui/Non |
| Pyr. Combinatoires | Oui | Oui | calcul local | Oui |

- Mais :
  - Les pyramides combinatoires ne sont qu'un modèle permettant de coder une hiérarchie de partitions.
  - $\Rightarrow$ La construction des pyramides nécessite un critère de fusion (e.g. watershed).

## Les Watersheds Hiérarchiques

- Définition des Watersheds (par écoulement) :
  - Soit une image en niveaux de gris (e.g. image de gradient), considérer l'image comme un relief (0 : bas, 255 : haut),
  - Bassins : associés à un minima.
    - Toute goutte d'eau tombant dans le bassin converge vers le minima.
  - Lignes de partage des eaux (watersheds)
    - Une goutte d'eau tombant d'une ligne de partage des eaux peut s'écouler vers plus d'un bassin.





## Valeur de passage

- La valeur de passage entre 2 bassins d'un pixel watershed $P$ est l'altitude minimale à laquelle on doit «monter» pour relier les deux bassins en passant par $P$.

| | | | |
|---|---|---|---|
| 2 | 3 | 6 | 2 |
| 3 | 6 | 6 | 6 |
| 255 | 7 | 6 | 6 |
| 2 | 255 | 7 | 6 |
| 1 | 2 | 255 | 5 |

Image Initiale

| | | | |
|---|---|---|---|
| 2 | 3 | 6 | 2 |
| 3 | 6 | 6 | 6 |
| 255 | 7 | 6 | 6 |
| 2 | 255 | 7 | 6 |
| 1 | 2 | 255 | 5 |

LPE

Rem 1 : Si $P$ entre deux bassins : $hauteur(P) = Val(P)$.

Rem 2 : La valeur de passage reste valide même si $P$ appartient à un watershed épais.

# Schéma général

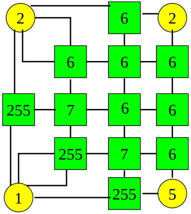
- Principe de l'algorithme :
  1. Obtenir une partition de l'image en bassins
  2. **Agréger** : $Val(P) \rightsquigarrow Val(C)$ : C contour entre deux bassins.
  3. **Fusionner** les Bassins les plus «proches» ($/Val(C)$).

## Problème 1 : $G_0 \xrightarrow{K_1} G_1$

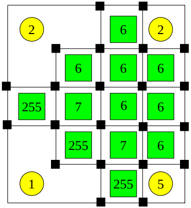
- Les Watersheds ne définissent pas une partition de l'image en régions mais en régions **et** pixels Watersheds.

| | | | |
|---|---|---|---|
| 2 | 3 | 6 | 2 |
| 3 | 6 | 6 | 6 |
| 255 | 7 | 6 | 6 |
| 2 | 255 | 7 | 6 |
| 1 | 2 | 255 | 5 |

Image




$G_1$

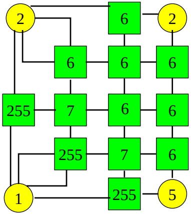



$\overline{G_1}$

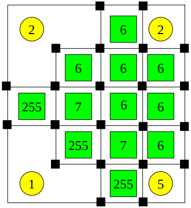
## Problème 1 : $G_0 \xrightarrow{K_1} G_1$

| | | | |
|---|---|---|---|
| 2 | 3 | 6 | 2 |
| 3 | 6 | 6 | 6 |
| 255 | 7 | 6 | 6 |
| 2 | 255 | 7 | 6 |
| 1 | 2 | 255 | 5 |

Image




$G_1$




$\overline{G_1}$

**Objectif :** Agréger les pixels watersheds tout en garantissant un calcul local des valeurs de passage.

Un pixel watershed ne peut pas être un minimum local.

$\Rightarrow$ On peut toujours trouver au moins un chemin décroissant vers un bassin.

# Algorithme d'agrégation

1. Enfiler tous les pixels watersheds adjacents à un bassin
2. Tant que la file de priorité est non vide
   2.1 Sortir un pixel d'altitude minimale de la file et le fusionner à un bassin adjacent.
   2.2 Tous les pixels watersheds adjacents au pixel retiré et n'appartenant pas à lar file sont ajoutés à celle-ci.
3. Fin : $\rightsquigarrow$ Partition en Bassins

| | | | |
|---|---|---|---|
| 2 | 3 | 6 | 2 |
| 3 | 6 | 6 | 6 |
| 255 | 7 | 6 | 6 |
| 2 | 255 | 7 | 6 |
| 1 | 2 | 255 | 5 |

# Algorithme d'agrégation

1. Enfiler tous les pixels watersheds adjacents à un bassin
2. Tant que la file de priorité est non vide
   2.1 Sortir un pixel d'altitude minimale de la file et le fusionner à un bassin adjacent.
   2.2 Tous les pixels watersheds adjacents au pixel retiré et n'appartenant pas à lar file sont ajoutés à celle-ci.
3. Fin : $\rightsquigarrow$ Partition en Bassins

| | | | |
|---|---|---|---|
| 2 | 3 | 6 | 2 |
| 3 | 6 | 6 | 6 |
| 255 | 7 | 6 | 6 |
| 2 | 255 | 7 | 6 |
| 1 | 2 | 255 | 5 |

# Algorithme d'agrégation

1. Enfiler tous les pixels watersheds adjacents à un bassin
2. Tant que la file de priorité est non vide
   2.1 Sortir un pixel d'altitude minimale de la file et le fusionner à un bassin adjacent.
   2.2 Tous les pixels watersheds adjacents au pixel retiré et n'appartenant pas à lar file sont ajoutés à celle-ci.
3. Fin : $\rightsquigarrow$ Partition en Bassins

| | | | |
|---|---|---|---|
| 2 | 3 | 6 | 2 |
| 3 | 6 | 6 | 6 |
| 255 | 7 | 6 | 6 |
| 2 | 255 | 7 | 6 |
| 1 | 2 | 255 | 5 |

# Algorithme d'agrégation

1. Enfiler tous les pixels watersheds adjacents à un bassin
2. Tant que la file de priorité est non vide
   2.1 Sortir un pixel d'altitude minimale de la file et le fusionner à un bassin adjacent.
   2.2 Tous les pixels watersheds adjacents au pixel retiré et n'appartenant pas à lar file sont ajoutés à celle-ci.
3. Fin : $\rightsquigarrow$ Partition en Bassins

| | | | |
|---|---|---|---|
| 2 | 3 | 6 | 2 |
| 3 | 6 | 6 | 6 |
| 255 | 7 | 6 | 6 |
| 2 | 255 | 7 | 6 |
| 1 | 2 | 255 | 5 |

# Algorithme d'agrégation

1. Enfiler tous les pixels watersheds adjacents à un bassin
2. Tant que la file de priorité est non vide
   2.1 Sortir un pixel d'altitude minimale de la file et le fusionner à un bassin adjacent.
   2.2 Tous les pixels watersheds adjacents au pixel retiré et n'appartenant pas à lar file sont ajoutés à celle-ci.
3. Fin : $\rightsquigarrow$ Partition en Bassins

| | | | |
|---|---|---|---|
| 2 | 3 | 6 | 2 |
| 3 | 6 | 6 | 6 |
| 255 | 7 | 6 | 6 |
| 2 | 255 | 7 | 6 |
| 1 | 2 | 255 | 5 |

# Algorithme d'agrégation

1. Enfiler tous les pixels watersheds adjacents à un bassin
2. Tant que la file de priorité est non vide
   2.1 Sortir un pixel d'altitude minimale de la file et le fusionner à un bassin adjacent.
   2.2 Tous les pixels watersheds adjacents au pixel retiré et n'appartenant pas à lar file sont ajoutés à celle-ci.
3. Fin : $\rightsquigarrow$ Partition en Bassins

| | | | |
|---|---|---|---|
| 2 | 3 | 6 | 2 |
| 3 | 6 | 6 | 6 |
| 255 | 7 | 6 | 6 |
| 2 | 255 | 7 | 6 |
| 1 | 2 | 255 | 5 |

# Algorithme d'agrégation

1. Enfiler tous les pixels watersheds adjacents à un bassin
2. Tant que la file de priorité est non vide
   2.1 Sortir un pixel d'altitude minimale de la file et le fusionner à un bassin adjacent.
   2.2 Tous les pixels watersheds adjacents au pixel retiré et n'appartenant pas à lar file sont ajoutés à celle-ci.
3. Fin : $\rightsquigarrow$ Partition en Bassins

| | | | |
|---|---|---|---|
| 2 | 3 | 6 | 2 |
| 3 | 6 | 6 | 6 |
| 255 | 7 | 6 | 6 |
| 2 | 255 | 7 | 6 |
| 1 | 2 | 255 | 5 |

Définition de la partition

$G_0$ : grille $m \times p$

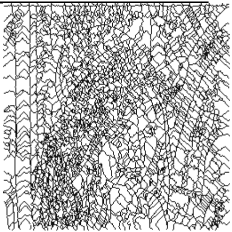
$\downarrow$ $K_0$ : Forêt recouvrant chaque bassin

$G_1$ : bassins et watersheds

$\downarrow$ $K_1$ File de priorité

$G_2$ : partition en bassins (*sur-segmentation*)

Watershed



Définition de la partition

$G_0$ : grille $m \times p$
$\downarrow$ $K_0$ : Forêt recouvrant chaque bassin
$G_1$ : bassins et watersheds
$\downarrow$ $K_1$ File de priorité
$G_2$ : partition en bassins (*sur-segmentation*)

Watershed

---

$\vdots$ fusions de bassins
$G_i$
$\vdots$
$G_n$

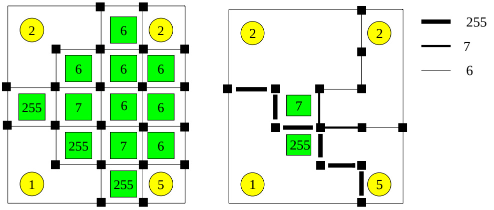
Hiérarchiques

$\Rightarrow$ valuation des contours entre les bassins.

# Valuation des lignels

| | | | |
|---|---|---|---|
| 2 | 3 | 6 | 2 |
| 3 | 6 | 6 | 6 |
| 255 | 7 | 6 | 6 |
| 2 | 255 | 7 | 6 |
| 1 | 2 | 255 | 5 |

Image



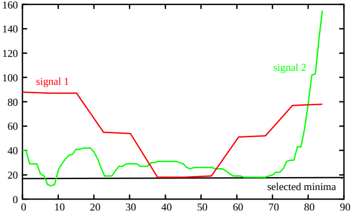

$\overline{G_1}$ $\overline{G_2}$

Valeur de passage d'un lignel l(P,Q) : Altitude minimale à atteindre pour connecter les deux bassins incidents à $l$ en passant par $P$ et $Q$.

$$Val(l) = max\{h(P), h(Q)\}$$

# Valuation des arêtes

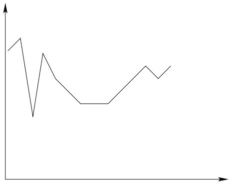
- $b \in G_i \rightsquigarrow \partial_i b = b_1 \ldots, b_p \in G_0 \rightsquigarrow l_1 \ldots, l_p, \ l_i$ lignel.
- $Val(l_1) \ldots, Val(l_p)$ : Profil des valeurs de passage.




- Comment sélectionner la «bonne» valeur de passage d'un contour ?

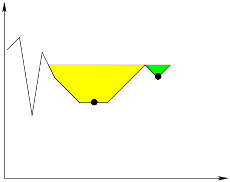
Valuation des arêtes

Valeur de passage d'une arête : *L'attitude minimale de son bassin de volume maximal* ⇔ (Watershed hiérarchique sur le profil 1D+Calcul des volumes de bassins).



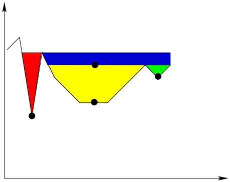
Valuation des arêtes

Valeur de passage d'une arête : *L'attitude minimale de son bassin de volume maximal* $\Leftrightarrow$ (Watershed hiérarchique sur le profil 1D+Calcul des volumes de bassins).



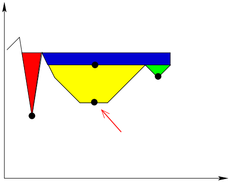
Valuation des arêtes

Valeur de passage d'une arête : *L'attitude minimale de son bassin de volume maximal* $\Leftrightarrow$ (Watershed hiérarchique sur le profil 1D+Calcul des volumes de bassins).



Valuation des arêtes

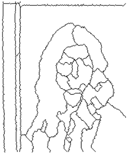
Valeur de passage d'une arête : *L'attitude minimale de son bassin de volume maximal* ⇔ (Watershed hiérarchique sur le profil 1D+Calcul des volumes de bassins).

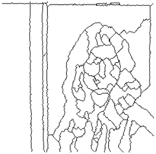


## Exemple

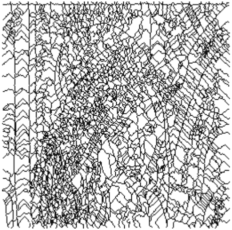
- niveau n+1 : la barre blanche est une seule région.


Original


min value

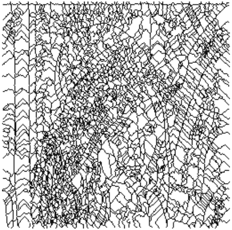

min max value

# Problème de la sur-segmentation

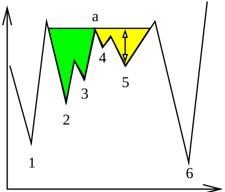


- Les bassins sont petits et faiblement représentatifs,
- Les contours sont petits et faiblement représentatifs,

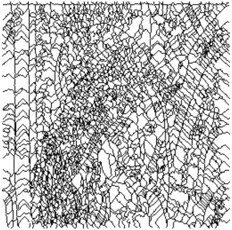
# Problème de la sur-segmentation



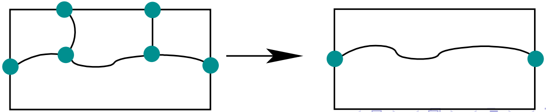
- Les bassins sont petits et faiblement représentatifs,
    - Solution : Utiliser la dynamique des contours de Najman.



# Problème de la sur-segmentation



- Les bassins sont petits et faiblement représentatifs,
- Les contours sont petits et faiblement représentatifs,
  - Solution : Effectuer des fusions progressives, et recalculer le minimum de chaque arête après chaque étape de fusion.



Évolution des poids des arêtes

- Algorithme
  1. Valuer les arêtes de $G_i, i \geq 2$,
  2. Calculer les dynamiques de contours (valeurs d'arêtes et de bassins),
  3. $K_i$ : noyau de contraction $\subset$ arêtes de poids minimal,
  4. Contracter $G_i$ par $K_i \rightarrow G_{i+1}$
  5. retourner au point 1
- Possible uniquement grace au codage implicite $\mathcal{P} = (G_0, G_n, niveau)$.

# Bilan Watershed

1. Calcul d'une tranformée Watershed,

# Bilan Watershed

1. Calcul d'une tranformée Watershed,
2. Agrégation des pixels watersheds aux bassins,

# Bilan Watershed

1. Calcul d'une tranformée Watershed,
2. Agrégation des pixels watersheds aux bassins,
3. Transfert des valeurs de passage aux lignels,

# Bilan Watershed

1. Calcul d'une tranformée Watershed,
2. Agrégation des pixels watersheds aux bassins,
3. Transfert des valeurs de passage aux lignels,
4. Calcul de la valeur de passage de chaque contour (min du bassin 1D de volume max),
   - Parcours du plongement géométrique des contours

# Bilan Watershed

1. Calcul d'une tranformée Watershed,
2. Agrégation des pixels watersheds aux bassins,
3. Transfert des valeurs de passage aux lignels,
4. Calcul de la valeur de passage de chaque contour (min du bassin 1D de volume max),
   - Parcours du plongement géométrique des contours
5. Calcul de la dynamique des contours baseé sur le min
   - Utilisation des adjacences entre bassins

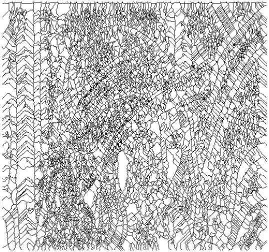
# Bilan Watershed

1. Calcul d'une tranformée Watershed,
2. Agrégation des pixels watersheds aux bassins,
3. Transfert des valeurs de passage aux lignels,
4. Calcul de la valeur de passage de chaque contour (min du bassin 1D de volume max),
   - Parcours du plongement géométrique des contours
5. Calcul de la dynamique des contours baseé sur le min
   - Utilisation des adjacences entre bassins
6. Fusion itérative, en ne fusionnant à chaque étape que les contours de dynamique minimale.
   - Utilisation du codage implicite

# Exemple

Lenna

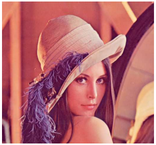
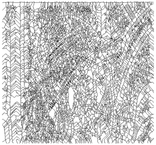
Watershed

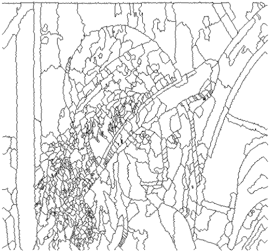


Dynamique

# Exemple

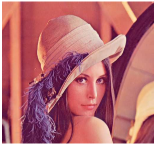
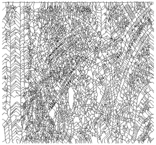
Lenna | Watershed | Dynamique

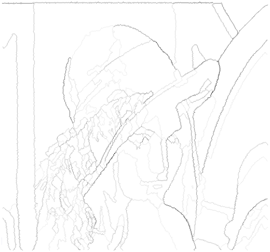




# Exemple

Lenna | Watershed | Dynamique



## Conclusion

- Bonne complémentarité entre les Watershed hiérarchiques et les pyramides combinatoires.

  Watersheds : Formalisme des critères de fusions,

  Pyr. Combi. : Formalisme permettant de manipuler la hiérarchie.

- En revanche : L'information couleur (niveau de gris) devient insipensable au delà de certains niveaux.

  $\Rightarrow$ Thèse de Jean Hugues Pruvot.

  - Hiérarchies et Énergies (cf. L. Guigues).
  - Appariement de Pyramides combinatoires (Plus large domaine commun entre deux images).