Introduction aux Cartes Combinatoires : Applications en Traitement d'Images et Simulation Physique

Guillaume Damiand

Laboratoire d'InfoRmatique en Image et Systèmes d'information
LIRIS UMR 5205 CNRS
Université Claude Bernard, Bâtiment Nautibus (710),
43, Boulevard du 11 Novembre 1918, 69622 Villeurbanne Cedex
http://liris.cnrs.fr

30 Avril 2014

Motivations

Cadre

Manipulation d'objets graphiques :

- traitement d'images : ensembles de pixels ou voxels
- modélisation géométrique : objets surfaciques ou volumiques

Besoins

Subdvision en cellules

sommets, arêtes, faces, volumes . . .

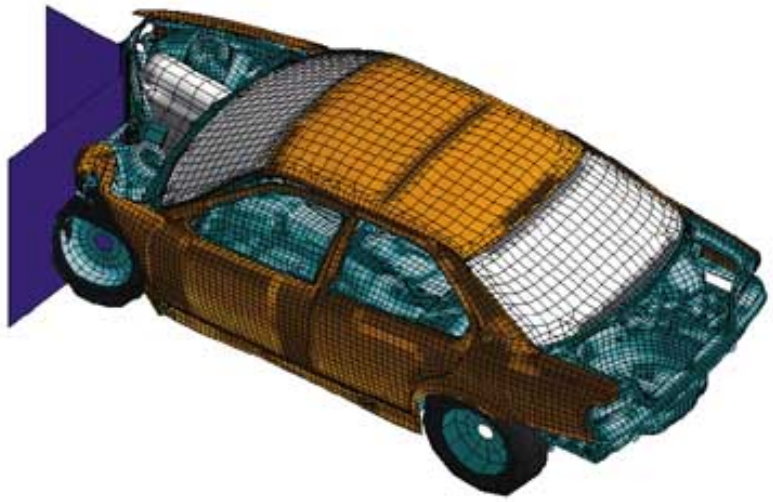
face : couleur, taille, forme
arête : longueur, gradient

sommet : point 3D
face : couleur
volume : résistance

Besoins

Relations entre les cellules

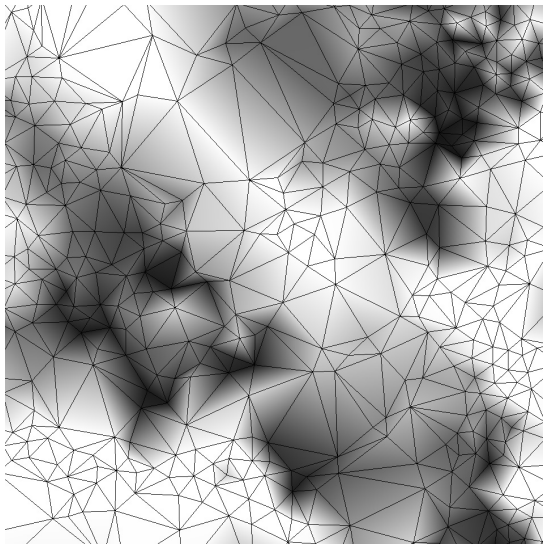
adjacence, incidence, imbrication

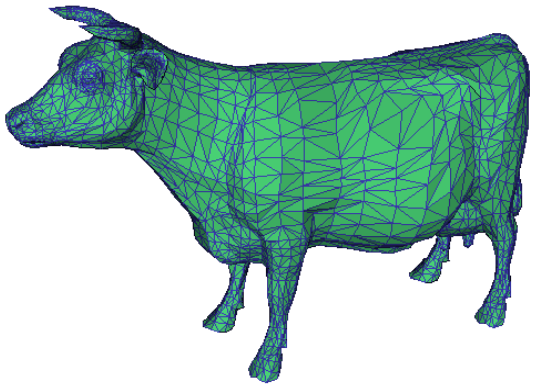


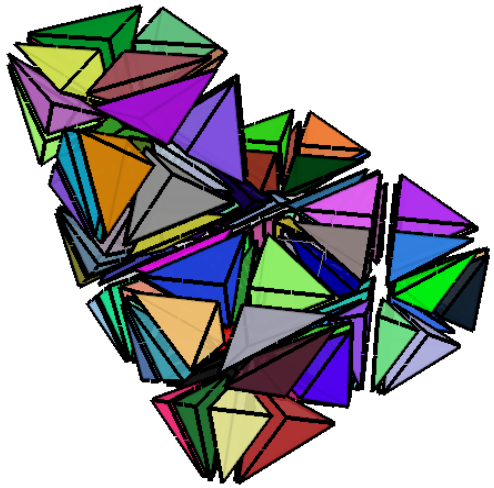
Principaux modèles

Modélisation géométrique

- triangles, tetraèdres, ... : modèles simpliciaux [Spanier 66 ;May 67 ; Agoston 76 ;Lang 95]
  - + : simples, *n*D, boules topologiques
  - - : nombre d'éléments, besoin de trianguler
- windged-edge, half-edge, DCEL, quad-edge... [Weiler 85 ;Baumgart 75 ; Guibas Stolfi 85 ;Mantyla 88]
  - + : formes générales, opérations locales
  - - : 2D, quelques travaux 3D, cellules $\neq$ boules
- puis des modèles à base de cartes...



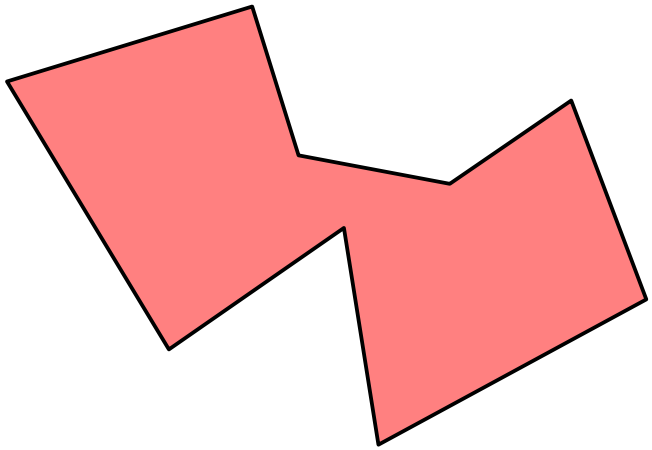




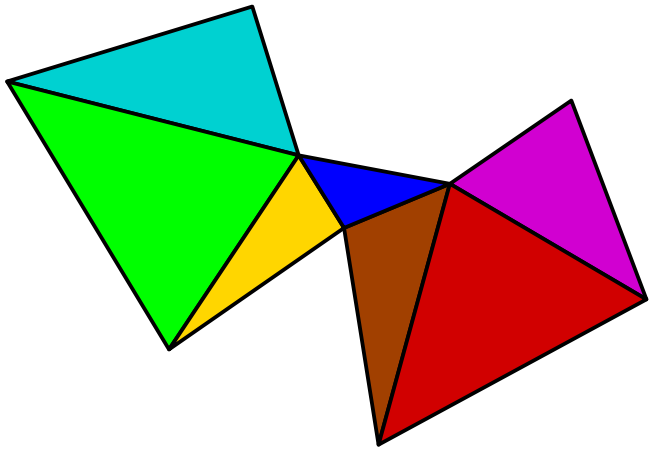
Principaux modèles

Modélisation géométrique

- triangles, tetraèdres, ... : modèles simpliciaux [Spanier 66 ;May 67 ; Agoston 76 ;Lang 95]
  - + : simples, *n*D, boules topologiques
  - - : nombre d'éléments, besoin de trianguler
- windged-edge, half-edge, DCEL, quad-edge... [Weiler 85 ;Baumgart 75 ; Guibas Stolfi 85 ;Mantyla 88]
  - + : formes générales, opérations locales
  - - : 2D, quelques travaux 3D, cellules $\neq$ boules
- puis des modèles à base de cartes...



9 sommets, 9 arêtes, 1 face

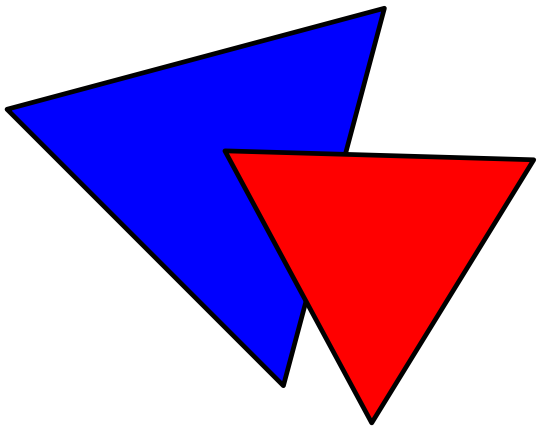
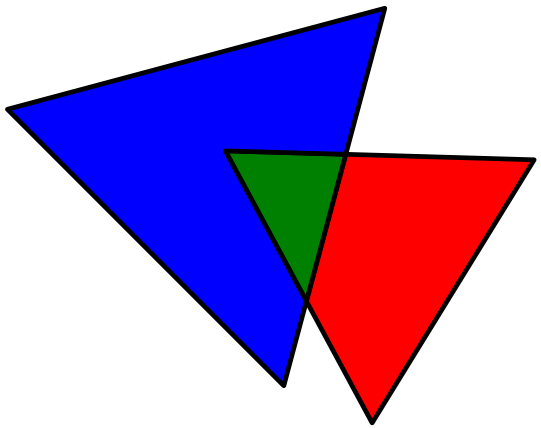
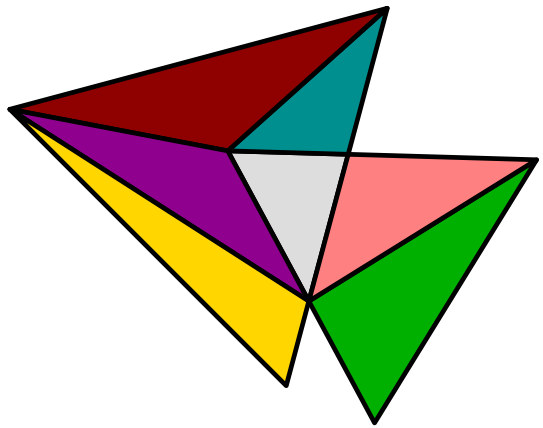


9 sommets, 15 arêtes, 7 faces

Principaux modèles

Modélisation géométrique

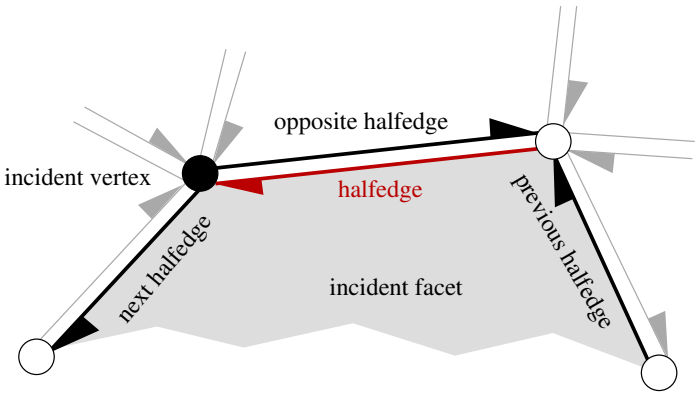
- triangles, tetraèdres, ... : modèles simpliciaux [Spanier 66 ;May 67 ; Agoston 76 ;Lang 95]
  - + : simples, *n*D, boules topologiques
  - - : nombre d'éléments, besoin de trianguler
- windged-edge, half-edge, DCEL, quad-edge... [Weiler 85 ;Baumgart 75 ; Guibas Stolfi 85 ;Mantyla 88]
  - + : formes générales, opérations locales
  - - : 2D, quelques travaux 3D, cellules $\neq$ boules
- puis des modèles à base de cartes...

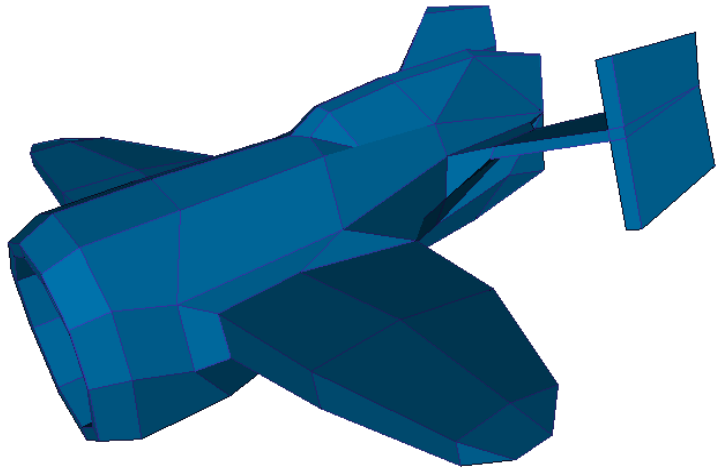




Principaux modèles

Modélisation géométrique

- triangles, tetraèdres, ... : modèles simpliciaux [SPANIER 66 ;MAY 67 ; AGOSTON 76 ;LANG 95]
  - + : simples, *n*D, boules topologiques
  - - : nombre d'éléments, besoin de trianguler
- windged-edge, half-edge, DCEL, quad-edge... [WEILER 85 ;BAUMGART 75 ; GUIBAS STOLFI 85 ;MANTYLA 88]
  - + : formes générales, opérations locales
  - - : 2D, quelques travaux 3D, cellules $\neq$ boules
- puis des modèles à base de cartes...

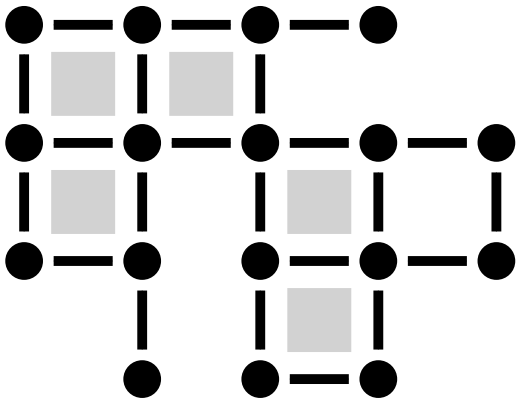




Principaux modèles

Traitement d'images

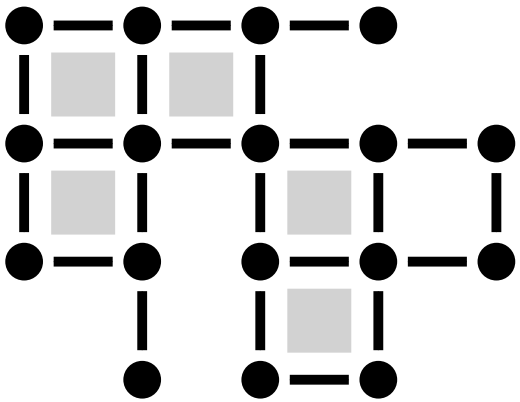
- carrés, cubes, ... : modèles cubiques [KOVALEVSKY 89]
  - + : simples, *n*D, boules topologiques
  - - : nombre d'éléments, besoin de quadranguler
- graphe d'adjacence des régions (RAG), multi-RAG [ROSENFELD 74 ; WILLERSINN KROPATSCH 94]
  graphes duaux
  - + : basés graphes, *n*D pour RAG
  - - : ne représente pas toutes les relations
- puis des modèles à base de cartes...

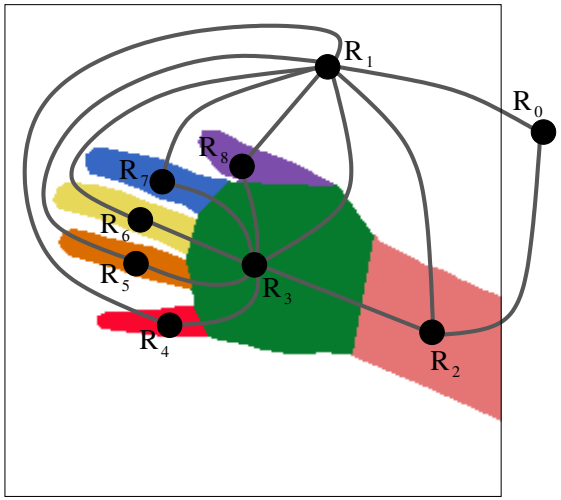


Principaux modèles

Traitement d'images

- carrés, cubes, ... : modèles cubiques [KOVALEVSKY 89]
  - + : simples, *n*D, boules topologiques
  - - : nombre d'éléments, besoin de quadranguler
- graphe d'adjacence des régions (RAG), multi-RAG graphes duaux [ROSENFELD 74 ; WILLERSINN KROPATSCH 94]
  - + : basés graphes, *n*D pour RAG
  - - : ne représente pas toutes les relations
- puis des modèles à base de cartes...

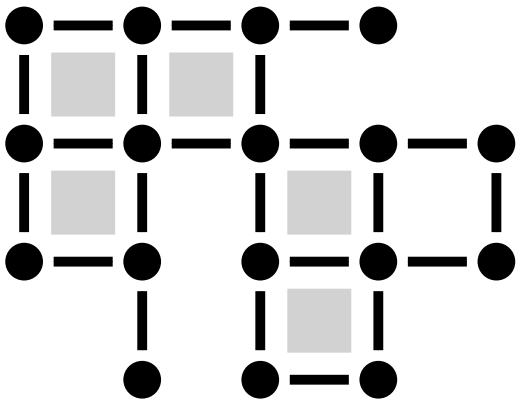


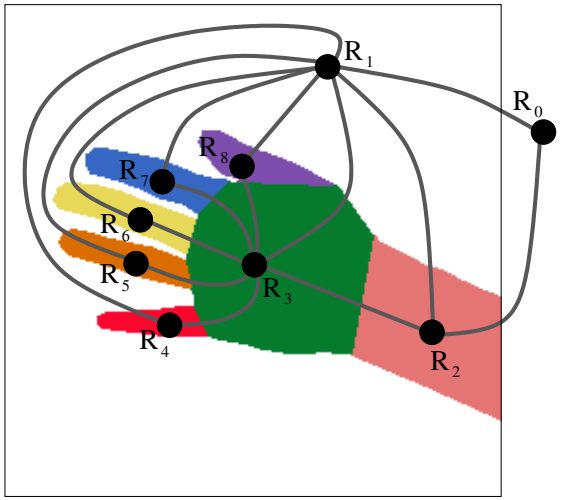


Principaux modèles

Traitement d'images

- carrés, cubes, ... : modèles cubiques [KOVALEVSKY 89]
  - + : simples, *n*D, boules topologiques
  - - : nombre d'éléments, besoin de quadranguler
- graphe d'adjacence des régions (RAG), multi-RAG graphes duaux [ROSENFELD 74 ; WILLERSINN KROPATSCH 94]
  - + : basés graphes, *n*D pour RAG
  - - : ne représente pas toutes les relations
- puis des modèles à base de cartes...





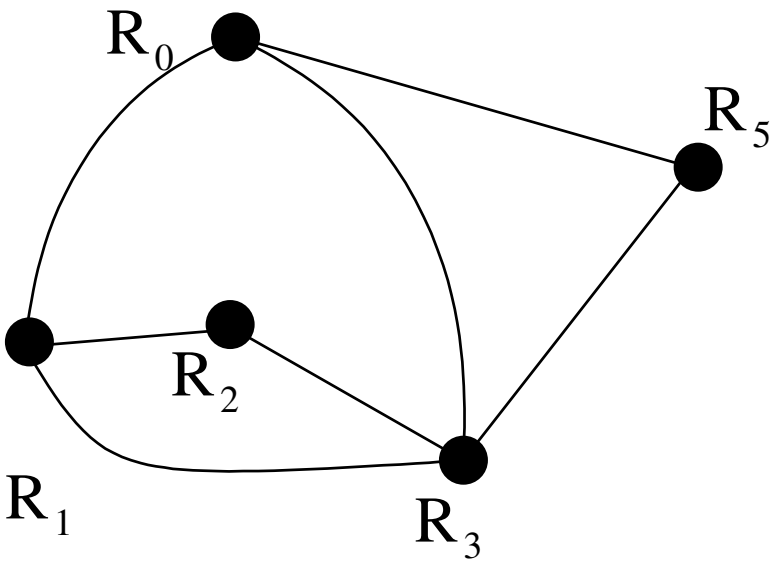
Part I : Cartes Combinatoires et Invariants Topologiques

1. Cartes combinatoires/généralisées
2. Invariants Topologiques
3. Opérations de base

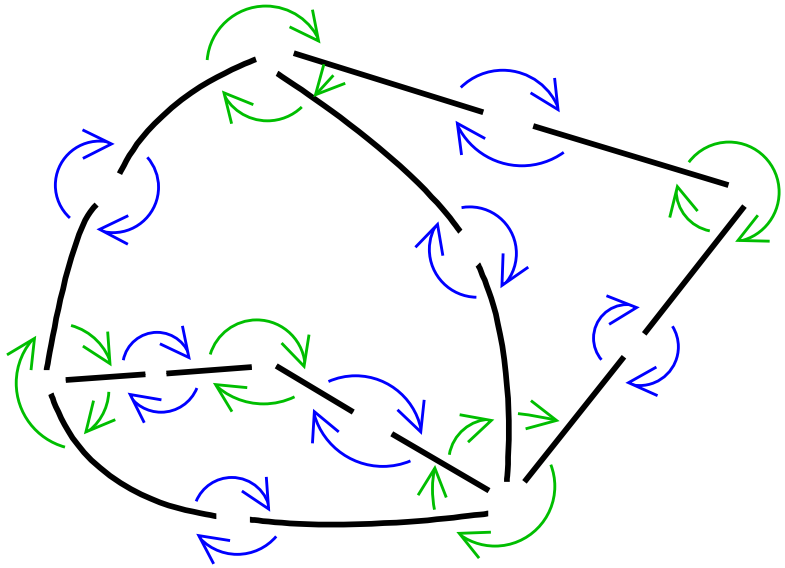
1. Cartes combinatoires/généralisées

Les cartes combinatoires

- Introduites en 2D $\Rightarrow$ graphes planaires [EDMONDS 60 ; TUTTE 63 ; JACQUES 70 ; CORI 73]
- Étendues en 3D [ARQUESKOCH 88 ; LIENHARDT 88 ; SPEHNER 91]
- Définies en nD [LIENHARDT 89]



Graphe planaire



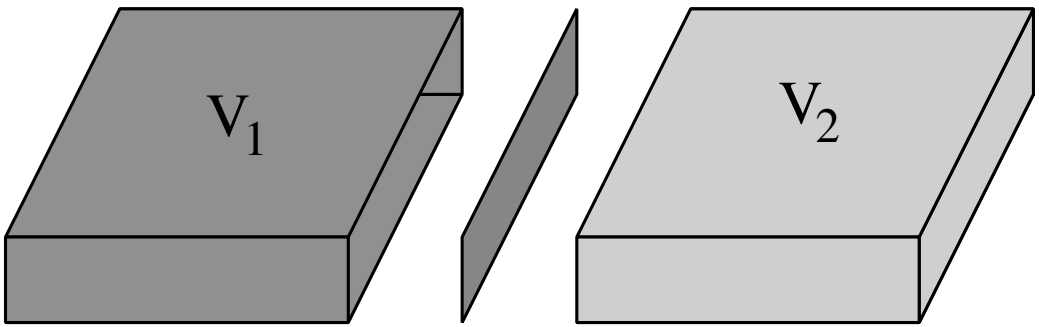
Carte combinatoire

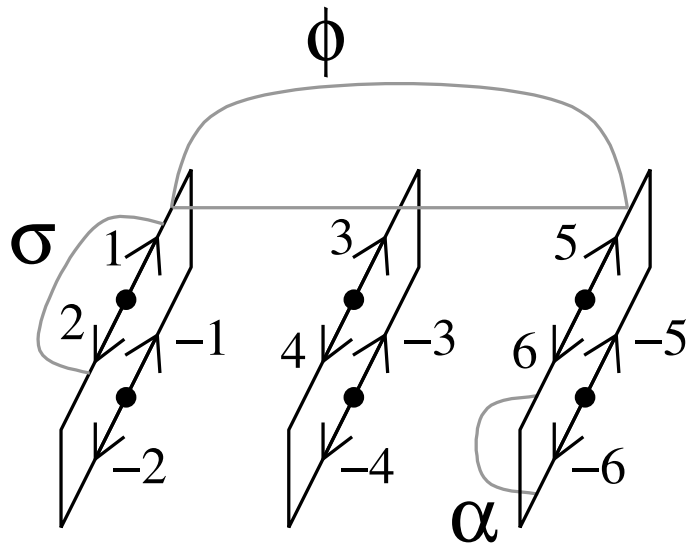
σ (sommet) : **permutation** ;

α (arête) : **involution**

Les cartes combinatoires

- Introduites en 2D $\Rightarrow$ graphes planaires [EDMONDS 60 ; TUTTE 63 ; JACQUES 70 ; CORI 73]
- Étendues en 3D [ARQUESKOCH 88 ; LIENHARDT 88 ; SPEHNER 91]
- Définies en nD [LIENHARDT 89]



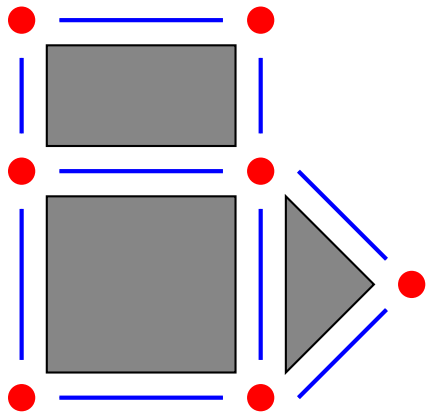


Les cartes combinatoires

- Introduites en 2D $\Rightarrow$ graphes planaires [EDMONDS 60 ; TUTTE 63 ; JACQUES 70 ; CORI 73]
- Étendues en 3D [ARQUESKOCH 88 ; LIENHARDT 88 ; SPEHNER 91]
- Définies en nD [LIENHARDT 89]

Intérêts :

- Représente les subdivisions de l'espace
- Définition simple en nD :
  un seul élément de base
- Représentation de la topologie :
  relations d'adjacence
- Implantation efficace d'algorithmes



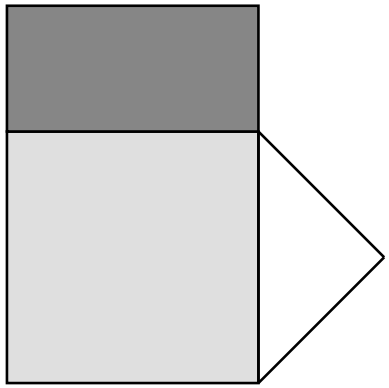
Notions préalables

- **permutation** : bijection f de $E \to E$
- **involution** : bijection f de $E \to E$, tel que $f = f^{-1}$
- $\Phi = \{f_1, \ldots, f_k\}$ des permutations sur E. $\langle\Phi\rangle$ est le **groupe de permutations** engendré par Φ (toutes les permutations qu'il est possible d'obtenir par composition des éléments de Φ et leurs contraires)
- **orbite** : soit $e \in E$. $\langle\Phi\rangle(e) = \{\phi(e) | \phi \in \langle\Phi\rangle\}$

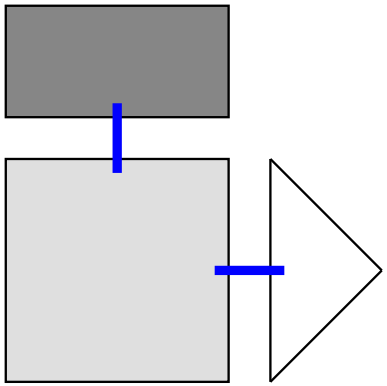
Obtention intuitive d'une carte combinatoire

Décompositions successives des cellules d'un objet

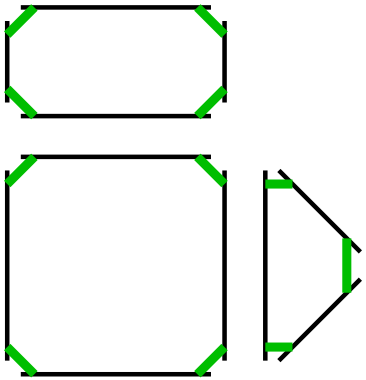
Exemple en 2D :



Objet 2D



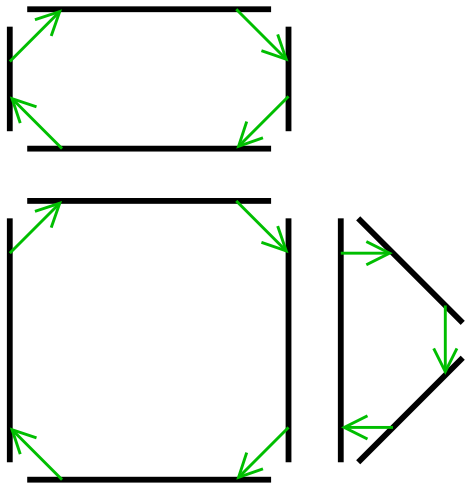
Faces



Arêtes
$\Rightarrow$ Les brins

Obtention intuitive d'une carte combinatoire

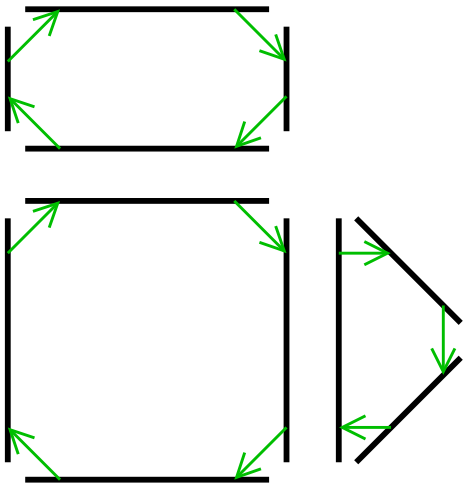
Report des relations d'adjacence sur les brins

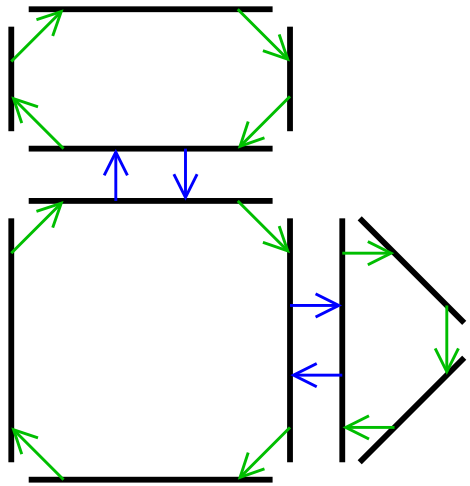


β_1 : brin $\rightarrow$ brin suivant de la même face

Obtention intuitive d'une carte combinatoire

Report des relations d'adjacence sur les brins





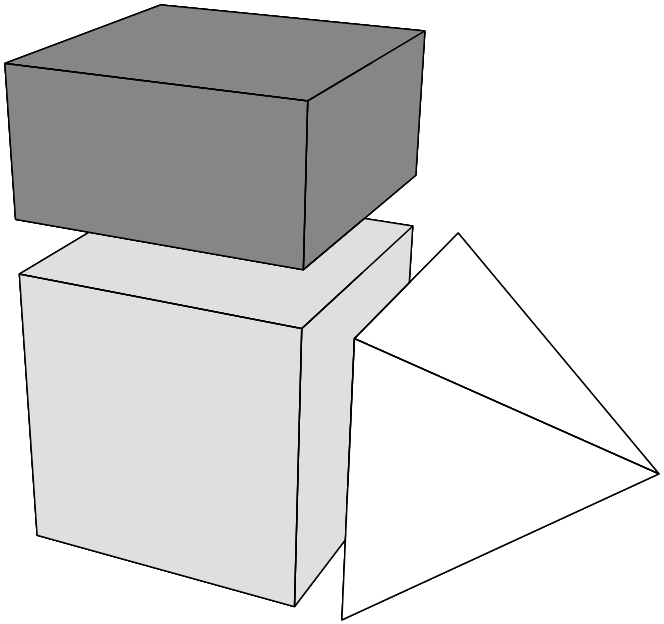
β_1 : brin $\rightarrow$ brin suivant de la même face

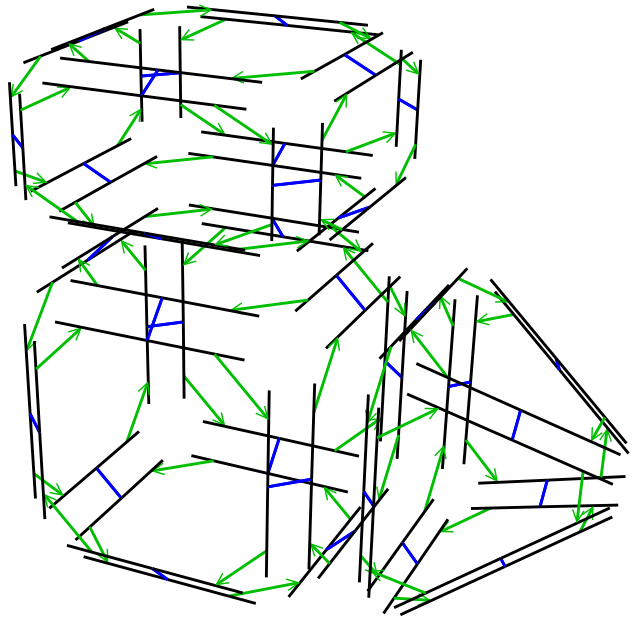
β_2 : brin $\rightarrow$ brin de l'autre face incidente à la même arête

- β_1 : **permutation** sur l'ensemble des brins
- β_2 : **involution** sur l'ensemble des brins

Les cartes en dimension 3

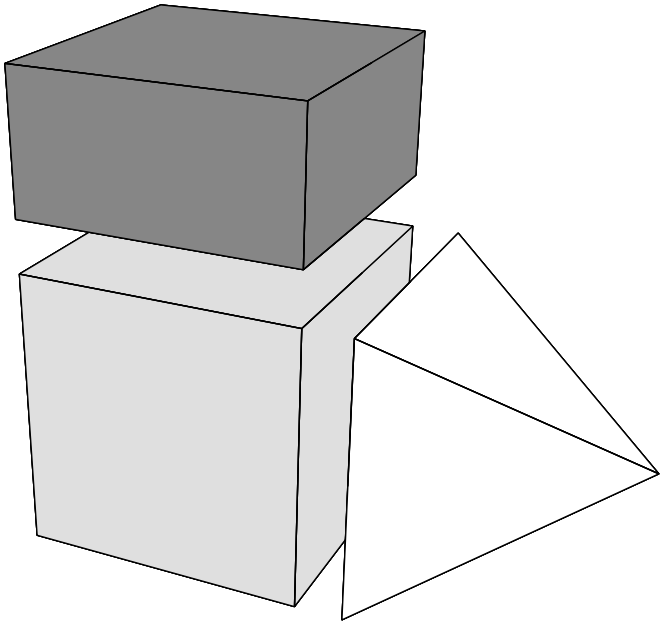
Ensemble de 2-cartes : chacune représente un volume

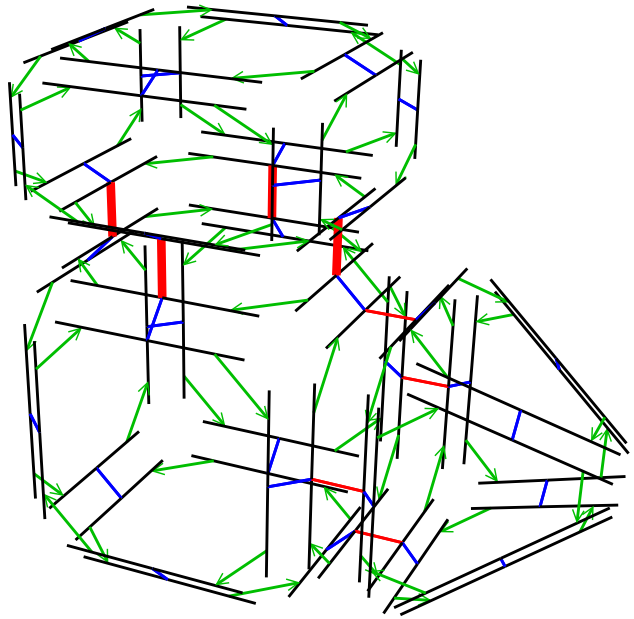




Les cartes en dimension 3

Ensemble de 2-cartes : chacune représente un volume





Ajout de β_3 : relie les volumes adjacents

- β_1 : permutation sur les brins
- β_2 et β_3 : involutions sur les brins

plus des contraintes de cohérences

Définition algébrique des cartes combinatoires *n*D

Definition (carte combinatoire)

Soit $n \geq 0$. Une n carte combinatoire (ou n-carte) est une algèbre $C = (B, \beta_1, \ldots, \beta_n)$ où :

1. B est un ensemble fini de brins ;
2. β_1 est une permutation sur B ;
3. $\forall i,\ 2 \leq i \leq n$, β_i est une involution sur B ;
4. $\forall i,\ 1 \leq i \leq n-2$, $\forall j,\ i+2 \leq j \leq n$, $\beta_i \circ \beta_j$ est une involution.

Remarque :

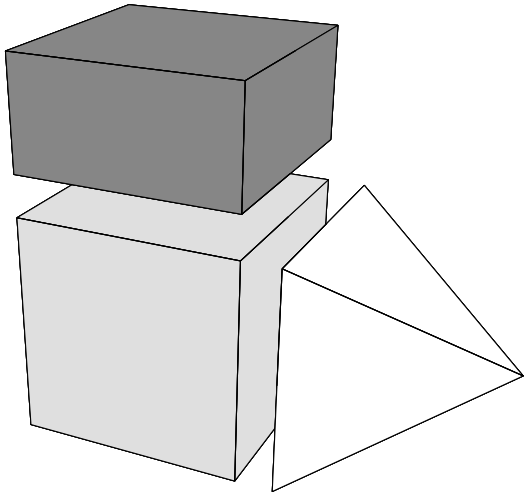
Ligne 4 : contraintes sur les relations

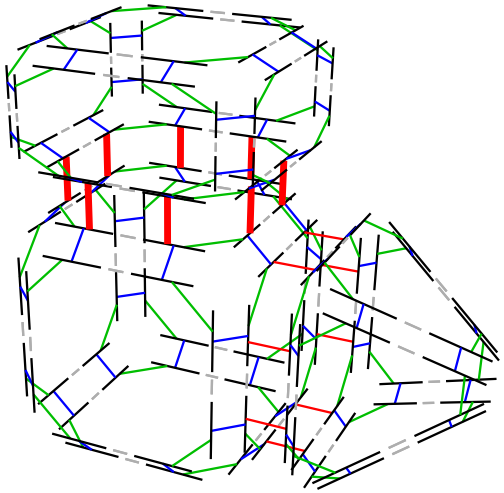
$\Rightarrow$ garantit la validité des objets

Exemple, en 3D : $\beta_1 \circ \beta_3$ involution

Les cartes généralisées

Cartes généralisées : une décomposition supplémentaire





$3G$-cartes : α_0, α_1, α_2, α_3 : involutions
plus des contraintes de cohérences

Les cartes généralisées

Définition algébrique des cartes généralisées *n*D

Definition (carte généralisée)

Soit $n \geq 0$. Une n carte généralisée (ou nG-carte) est une algèbre $G = (B, \alpha_0, \ldots, \alpha_n)$ où :

1. B est un ensemble fini de brins ;
2. $\forall i,\ 0 \leq i \leq n$, α_i est une involution sur B ;
3. $\forall i,\ 0 \leq i \leq n-2, \forall j,\ i+2 \leq j \leq n$, $\alpha_i \circ \alpha_j$ est une involution.

Avantages :

- définition homogène pour toutes les dimensions
  $\Rightarrow$ facilitent les définitions car plus de cas particulier
- représentation d'objet avec ou sans bord, orientable/non-orientable

Propriétés

Une G-carte $G = (B, \alpha_0, \ldots, \alpha_n)$ est :

Définition (connexe)

G est connexe si pour $b \in B$, $\langle \alpha_0, \ldots, \alpha_n \rangle(b) = B$.

Définition (avec/sans bord)

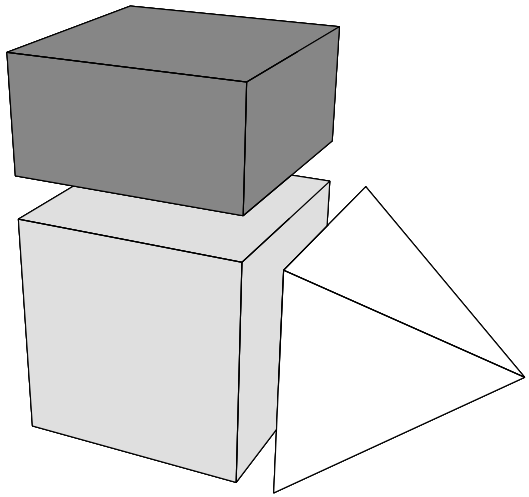
G est sans bord si $\forall b \in B$, $\forall i \in \{0, \ldots, n\}$, $\alpha_i(b) \neq b$.
G est dite à bord sinon.

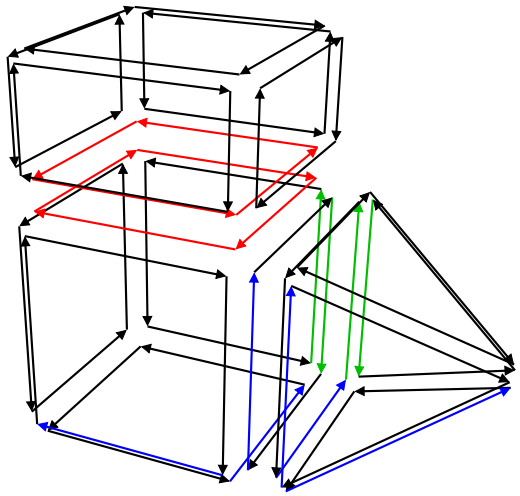
Définition (G-carte orientable)

Si G est connexe et sans bord. G est orientable si la n-carte $HG = (B, \alpha_{01}, \ldots, \alpha_{0n})$ a exactement deux composantes connexes distinctes. G est non orientable sinon.

Cellules dans les cartes

Exemple : 3-Cartes et Cellules





Cellules : représentation implicite : i-cellule = ensemble de brins
Utilise la notion d'orbite

Cellules dans les cartes

Représentation implicite : i-cellule = ensemble de brins

Definition (i-cellule)

Soit C une n-carte, b un brin de cette carte, et $i \in \{0, \ldots, n\}$. La i-cellule incidente à b est :

- Si $i = 0$: $< \beta_2 \circ \beta_0, \ldots, \beta_n \circ \beta_0 > (b)$;
- Sinon : $< \beta_1, \ldots, \beta_{i-1}, \beta_{i+1}, \ldots, \beta_n > (b)$.

Intuitivement : i-cellule incidente à un brin = l'ensemble des brins atteignables par un parcours en utilisant tout les β sauf β_i
Attention : définition différente pour les sommets

Incidence et Adjacence

Définition (Incidence)

Deux cellules c_1 et c_2 sont incidentes si elles sont de dimensions différentes, et $c_1 \cap c_2 \neq \emptyset$.

Définition (Adjacence)

Deux i-cellules c_1 et c_2 sont adjacentes s'il existe $b_1 \in c_1$ et $b_2 \in c_2$ tel que

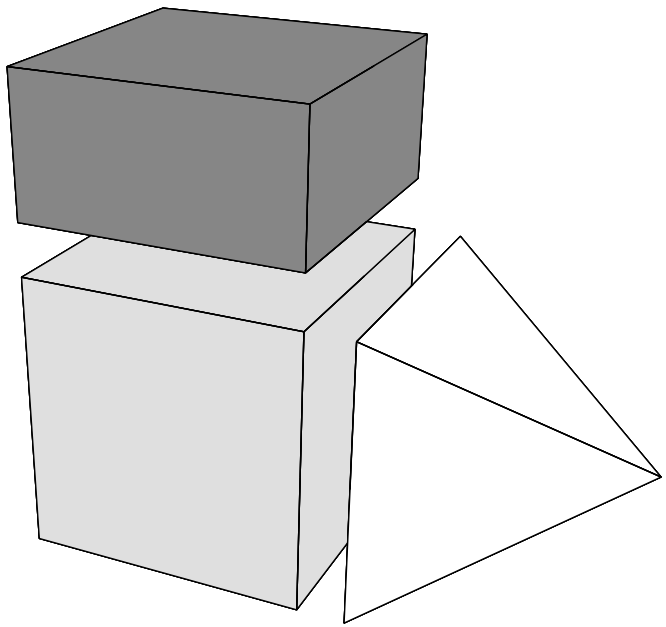
- *si $i = 0$: $d_1 = \beta_k(d_2)$ avec $k \in \{0, \ldots, n\}$;*
- *si $i > 0$: $d_1 = \beta_i(d_2)$ ou $d_2 = \beta_i(d_1)$.*

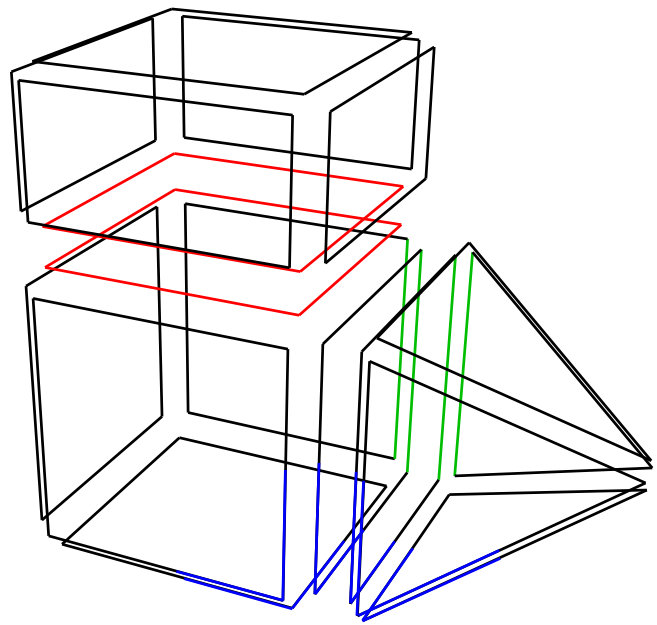
Cellules dans les G-cartes

plus de différence entre les 0-cellules et les autres cellules

Definition (i-cellule)

Soit G une nG-carte, b un brin de cette carte, et $i \in \{0, \ldots, n\}$. La i-cellule incidente à b est $< \alpha_0, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n > (b)$.





Codage en pratique

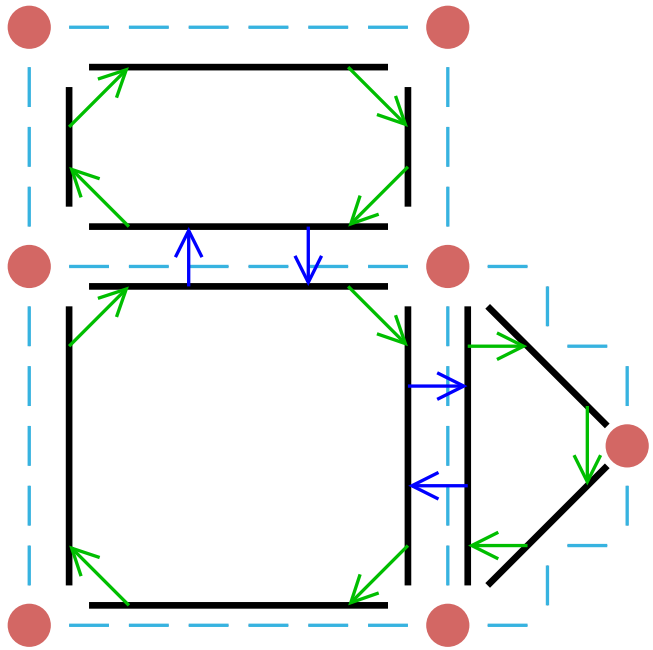
```
class CDart2D
{
  private:
    CDart* FBeta[3];
    bool FMarks[8];
}

class CMap2D
{
 private:
   std::list<CDart2D*> FListOfDarts;
}
```

Plongement des cartes

Représentation d'un objet :

- **topologie** : carte combinatoire
- **géométrie** : plongement

i-cellule topologique $\Longleftrightarrow$ **objet géométrique** de dimension i



2. Invariants Topologiques

Bases de topologie

Homéomorphismes

X et Y sont homéomorphes s'il $\exists f$ une bijection continue de X dans Y, tel que f^{-1} est continue.

Intuitivement :

- X peut se « déformer continuement » en Y
- X et Y ont la « même topologie »

Bases de topologie

Variété

X est une variété si en chaque point de X, il existe un voisinage homéomorphe à la boule unité ou à la demi-boule unité.

Bord

Soit X une variété. Son bord est l'ensemble des points de X dont le voisinage est homéomorphe à la demi-boule unité.

variété avec/sans bord.

Orientable

Une variété X est orientable s'il est possible de définir une direction $\ll$ gauche $\gg$ et $\ll$ droite $\gg$ globalement en tout point de la variété.

Invariants topologiques

Invariant topologique

Une propriété qui se conserve par homéomorphisme

2 objets homéomorphes ont les mêmes invariants topologiques

Exemples d'invariants :

- dimension, nombre de composantes connexes, orientabilité, caractéristique d'Euler, nombres de Betti, groupes d'Homologies. . .

Caractéristique d'Euler

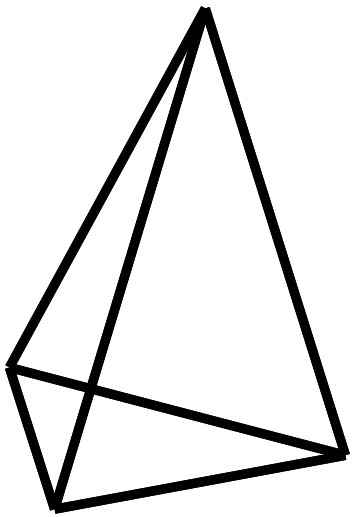
Caractéristique d'Euler

Caractéristique d'Euler χ de C un complexe cellulaire 2D sans bord et connexe :

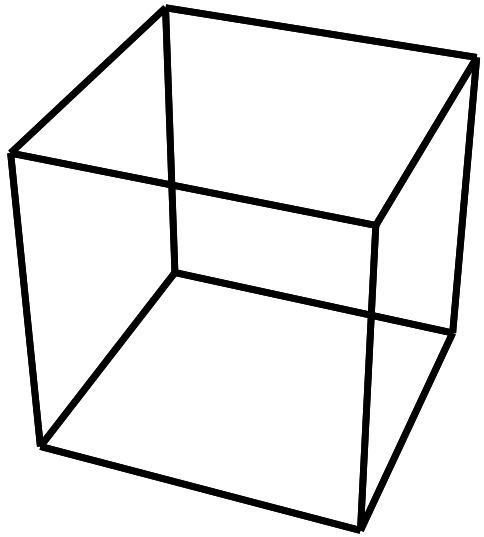
$$\chi(C) = k_0(C) - k_1(C) + k_2(C)$$

avec $k_i(C)$ le nombre de cellules de dimension i de C.

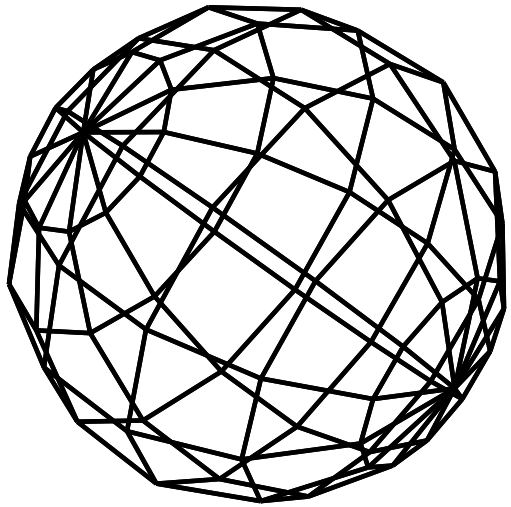
Autrement dit : *#sommets-#arêtes+#faces*



$4 - 6 + 4 = 2$



$8 - 12 + 6 = 2$



$74 - 156 + 84 = 2$

Caractéristique d'Euler

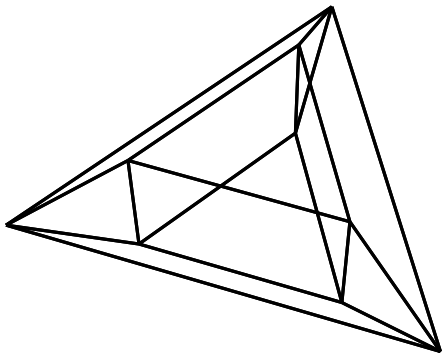
Caractéristique d'Euler

Caractéristique d'Euler χ de C un complexe cellulaire 2D sans bord et connexe :

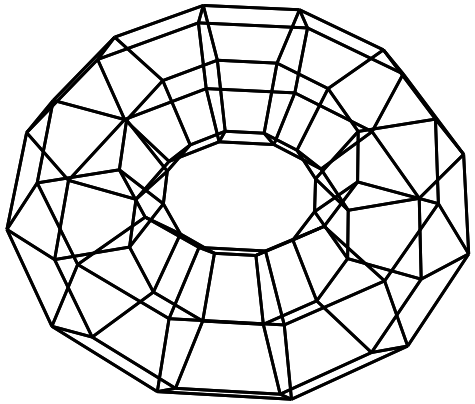
$$\chi(C) = k_0(C) - k_1(C) + k_2(C)$$

avec $k_i(C)$ le nombre de cellules de dimension i de C.

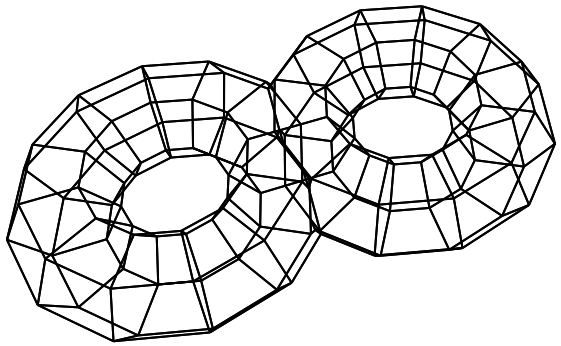
Autrement dit : *#sommets-#arêtes+#faces*



$9 - 18 + 9 = 0$



$72 - 144 + 72 = 0$



$148 - 298 + 148 = -2$

Genre

Autre invariant topologique :
nombre maximum de courbes simples fermées non déconnectantes

Lien genre $\leftrightarrow$ caractéristique d'Euler

- $\chi(C) = 2 - 2g(C)$ si C orientable
- $\chi(C) = 2 - g(C)$ si C non orientable

Extension aux objets à bord/non connexe

- $\chi(C) = 2\#cc(C) - 2g(C) - \#b(C)$ si C orientable
- $\chi(C) = 2\#cc(C) - g(C) - \#b(C)$ si C non orientable

avec $\#b(C)$ le nombre de bords
et $\#cc(C)$ le nombre de composantes connexes.

Caractéristique d'Euler-Poincaré

Généralisation en nD de la caractéristique d'Euler

Caractéristique d'Euler-Poincaré

Caractéristique d'Euler χ de C un complexe cellulaire nD sans bord :

$$\chi(C) = \sum_{i=0}^{n} (-1)^i \times k_i(C)$$

avec $k_i(C)$ le nombre de cellules de dimension i de C.

Classification des surfaces

Théorème de classification des surfaces

Toute surface fermée est homéomorphe à une des trois surfaces suivantes :

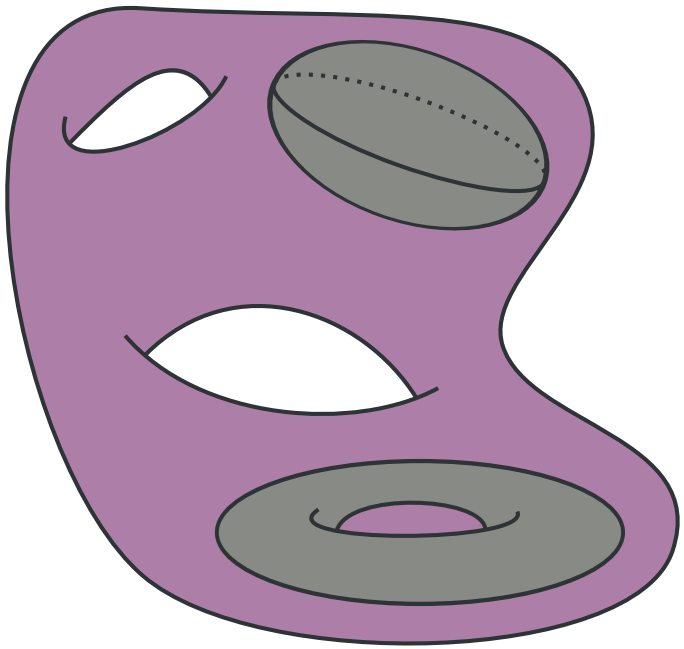
- S^2, la sphère de dimension 2 ;
- la somme connexe de g tores (ou tore à g trous) ;
- la somme connexe de k plans projectifs.

Classification complète des surfaces fermées par caractéristique d'Euler + orientabilité

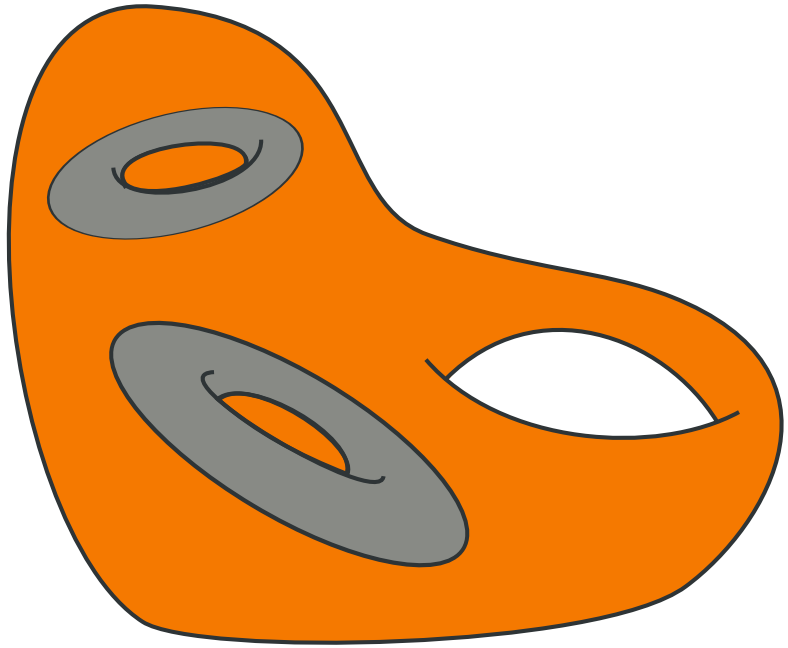
Extension directe aux surfaces à bord/non connexe :
ajouter nombre de bords et nombre de composantes connexes.

Pas de classification pour $n > 2$

2 objets non homéomorphe avec mêmes caractéristiques :
Euler, nb composantes connexes, nb bords



$\chi = 0, \#cc = 1, \#b = 3$



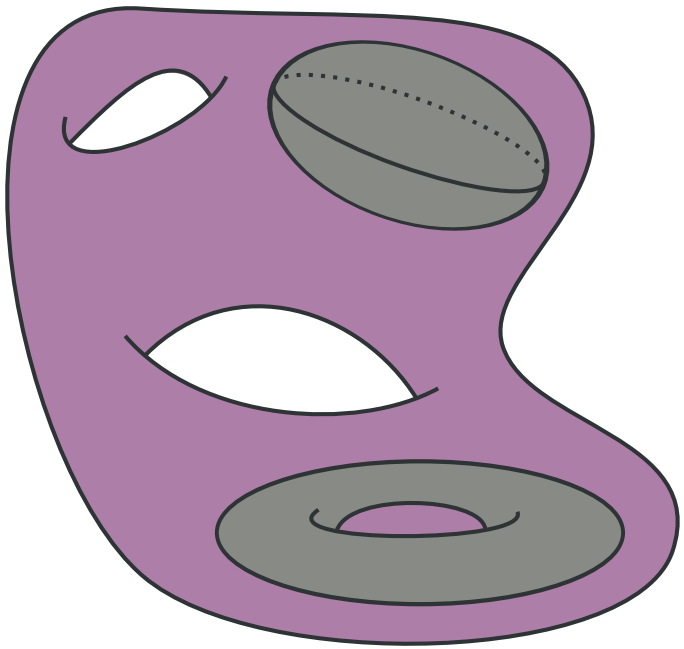
$\chi = 0, \#cc = 1, \#b = 3$

Nombres de Betti

Nombres de Betti

$\mathfrak{b}_0, \ldots, \mathfrak{b}_n$ (après tous nul)
Chaque $\mathfrak{b}_i$ est le rang du $i^{ème}$ groupe d'homologie.

Intuitivement : $\mathfrak{b}_i$ nombre de composantes connexes de i-surfaces



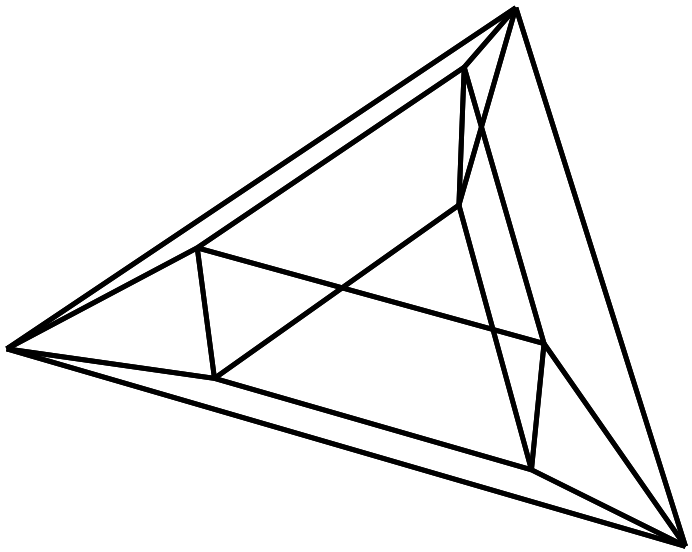
$$\mathfrak{b}_0 = 1$$
$$\mathfrak{b}_1 = 3$$
$$\mathfrak{b}_2 = 2$$

Nombres de Betti

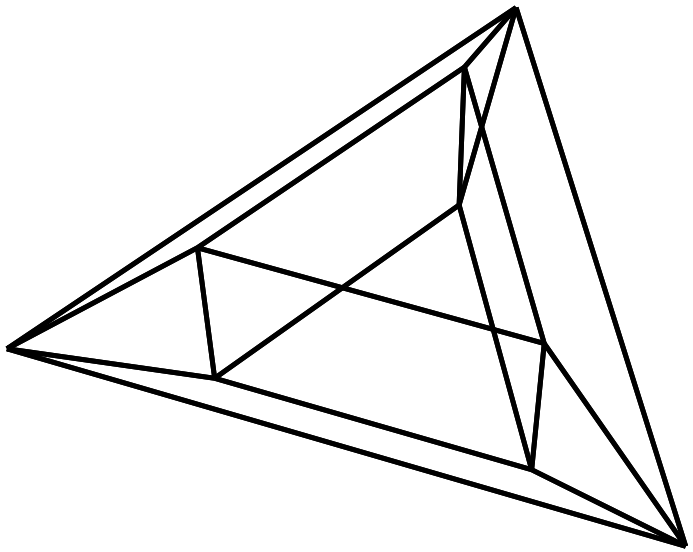
Lien avec caractéristique d'Euler-Poincaré

$$\chi(C) = \sum_{i=0}(-1)^i \times \mathfrak{b}_i(C)$$

Attention à la dimension :



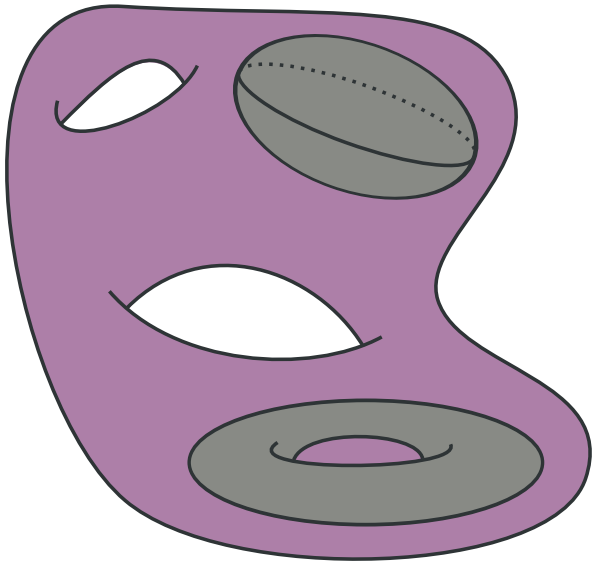
2D : $\mathfrak{b}_0 = 1\ \mathfrak{b}_1 = 2\ \mathfrak{b}_2 = 1 \Rightarrow \chi(C) = 0.$

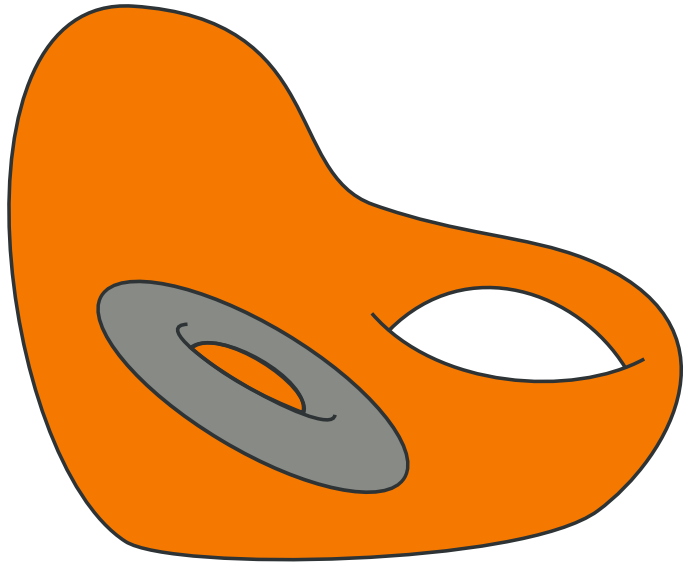


3D : $\mathfrak{b}_0 = 1\ \mathfrak{b}_1 = 1\ \mathfrak{b}_2 = 0$
$\mathfrak{b}_3 = 0 \Rightarrow \chi(C) = 0.$

Nombres de Betti

Invariant ≪ plus fort ≫ que la caractéristique d'Euler-Poincaré





$\mathfrak{b}_0 = 1, \mathfrak{b}_1 = 3, \mathfrak{b}_2 = 2$ $\qquad$ $\mathfrak{b}_0 = 1, \mathfrak{b}_1 = 2, \mathfrak{b}_2 = 1$

$$\chi = \mathfrak{b}_0 - \mathfrak{b}_1 + \mathfrak{b}_2 = 0$$

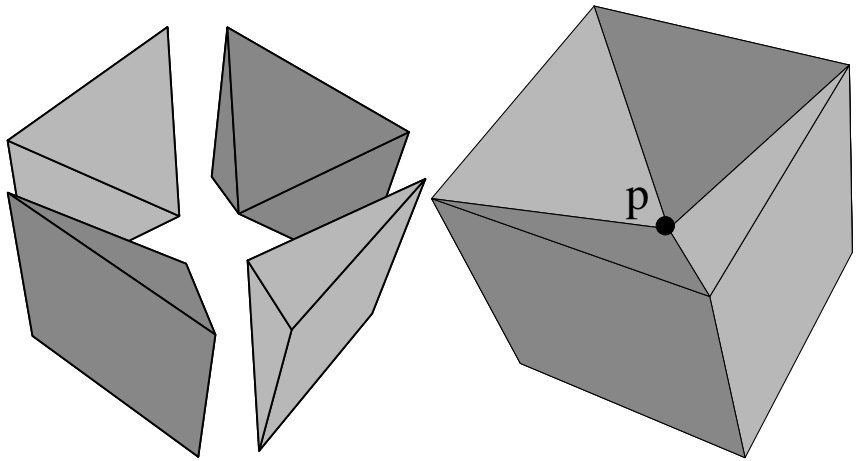
Liens avec cartes

- n-Carte : quasi variété nD avec ou sans bord orientable ;
- nG-Carte : quasi variété nD avec ou sans bord orientable ou non-orientable.

quasi variété nD

Des quasi variété $(n-1)$D collées entre elles le long de $(n-1)$-cellules.

En 2D, quasi-variété = variété ;
Pas en dimension supérieure ;
PB : $\nexists$ définition combinatoire de variété.



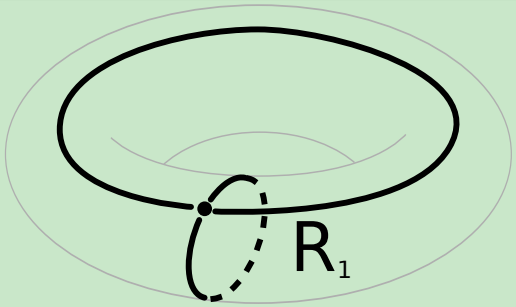
Liens avec cartes

Problème

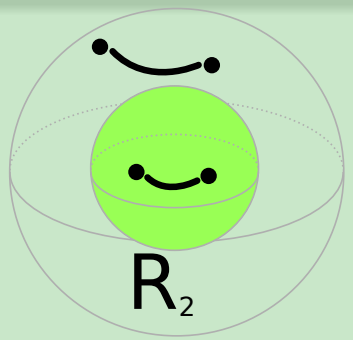
Les cellules ne sont pas forcément homéomorphe à des boules

Alors que c'est une condition des complexe cellulaires
$\Rightarrow$ pas possible d'utiliser directement les définitions précédentes

Exemples



$k_0 = 1, k_1 = 2, k_2 = 1, k_3 = 1$
1-2+1-1=-1 mais $\chi(C) = 0$



$k_0 = 4, k_1 = 2, k_2 = 2, k_3 = 1$
4-2+2-1=3 mais $\chi(C) = 2$

Heureusement il y a des solutions (cf. plus tard)

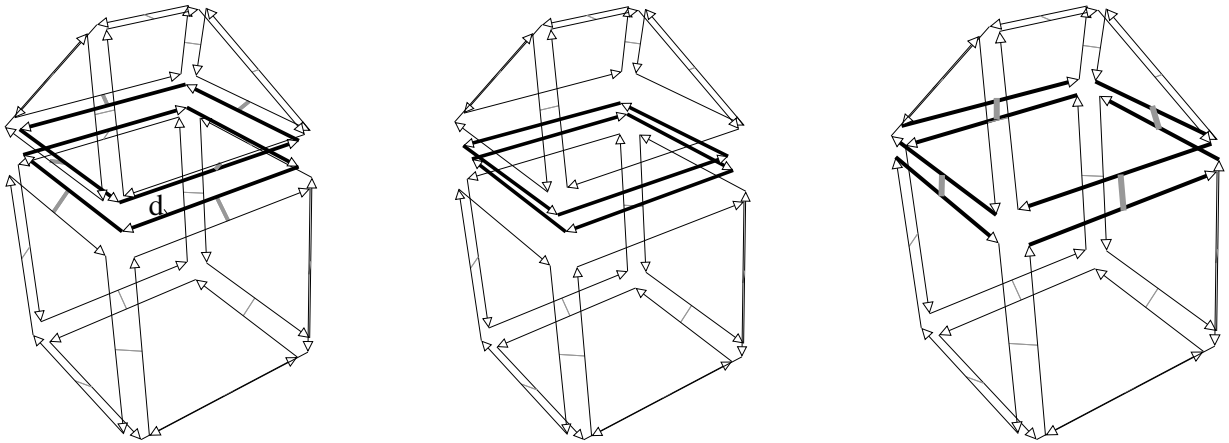
3. Opérations de base

Opérations de suppression

Suppressions

- Définies en dimension quelconque
- Contrainte : cellules localement de degré 2
- Suppression d'une i cellule $\Rightarrow$ fusion des (i+1) cellules

2-Suppression

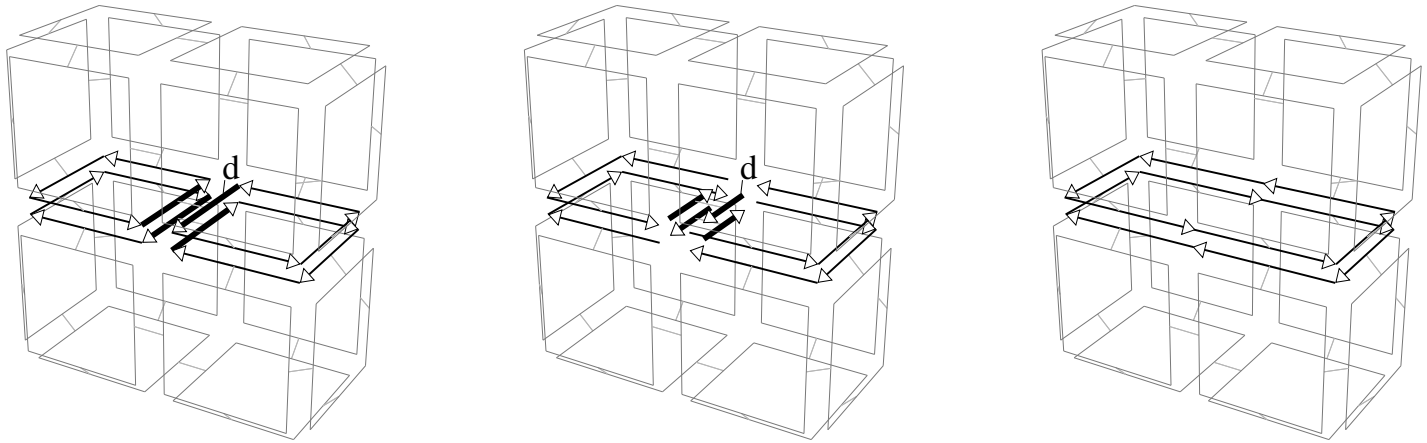


Opérations de suppression

Suppressions

- Définies en dimension quelconque
- Contrainte : cellules localement de degré 2
- Suppression d'une i cellule $\Rightarrow$ fusion des (i+1) cellules

1-Suppression

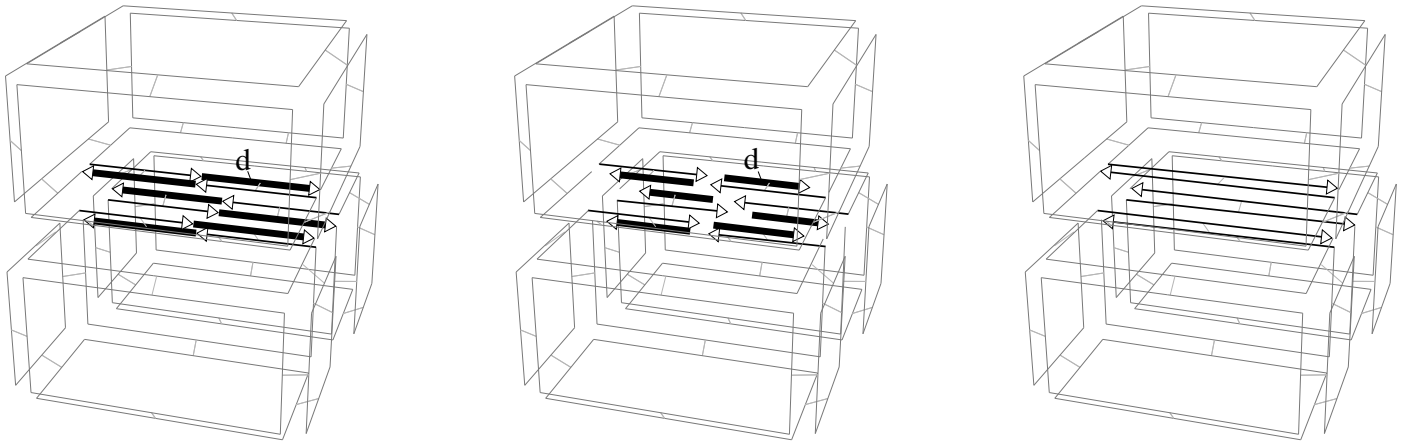


Opérations de suppression

Suppressions

- Définies en dimension quelconque
- Contrainte : cellules localement de degré 2
- Suppression d'une i cellule $\Rightarrow$ fusion des (i+1) cellules

0-Suppression



Autres opérations de base : contraction, insertion, éclatement

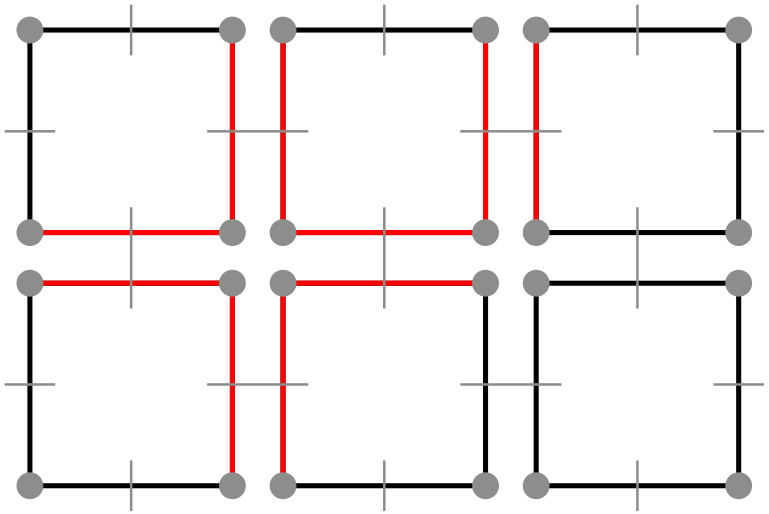
Suppression en dimension n

$C =< \alpha_0, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n > (b)$

Préconditions :

- $i \in \{0, \ldots, n-1\}$
- $\forall b' \in C$,
  $\alpha_{(i+1)}\alpha_{(i+2)}(b') = \alpha_{(i+2)}\ \alpha_{(i+1)}(b')$
  (sauf pour $i = n-1$)

$B^S = \alpha_i(C) \setminus C$
$\forall b' \in B^S,\ \alpha'_i(b') = (\alpha_i\alpha_{(i+1)})^k \alpha_i(b')$
k le plus petit entier



2-G-carte initiale ; arêtes à supprimer

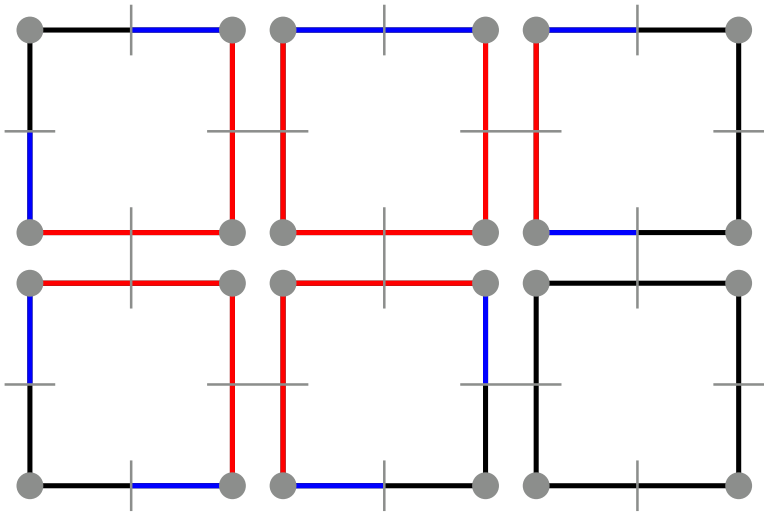
Suppression en dimension n

$C =< \alpha_0, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n > (b)$

Préconditions :

- $i \in \{0, \ldots, n-1\}$
- $\forall b' \in C$,
  $\alpha_{(i+1)}\alpha_{(i+2)}(b') = \alpha_{(i+2)}\ \alpha_{(i+1)}(b')$
  (sauf pour $i = n-1$)

$B^S = \alpha_i(C) \setminus C$
$\forall b' \in B^S,\ \alpha'_i(b') = (\alpha_i\alpha_{(i+1)})^k \alpha_i(b')$
k le plus petit entier



$B^S = C\alpha_1 - C$

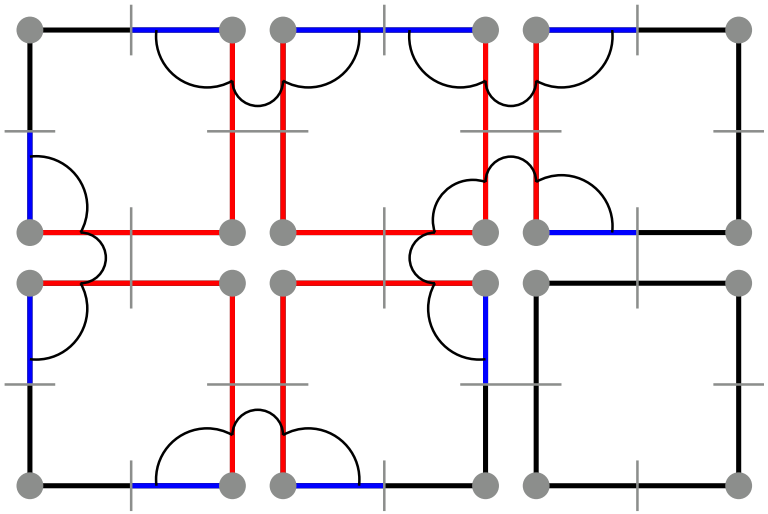
Suppression en dimension n

$C =< \alpha_0, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n > (b)$

Préconditions :

- $i \in \{0, \ldots, n-1\}$
- $\forall b' \in C$,
  $\alpha_{(i+1)}\alpha_{(i+2)}(b') = \alpha_{(i+2)}\ \alpha_{(i+1)}(b')$
  (sauf pour $i = n-1$)

$B^S = \alpha_i(C) \setminus C$
$\forall b' \in B^S,\ \alpha'_i(b') = (\alpha_i\alpha_{(i+1)})^k\alpha_i(b')$
k le plus petit entier



$\forall b' \in B^S,\ b'\alpha'_1 = b'(\alpha_1\alpha_2)^k\alpha_1$
k plus petit entier

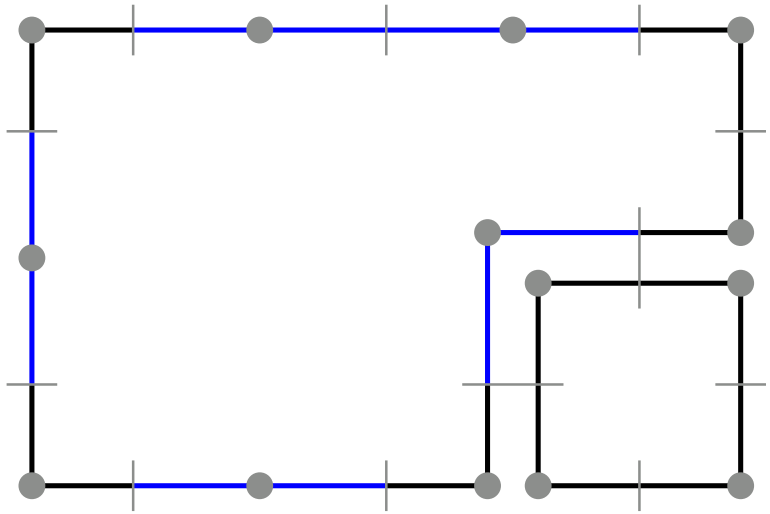
Suppression en dimension n

$C =< \alpha_0, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n > (b)$

Préconditions :

- $i \in \{0, \ldots, n-1\}$
- $\forall b' \in C$,
  $\alpha_{(i+1)}\alpha_{(i+2)}(b') = \alpha_{(i+2)}\ \alpha_{(i+1)}(b')$
  (sauf pour $i = n-1$)

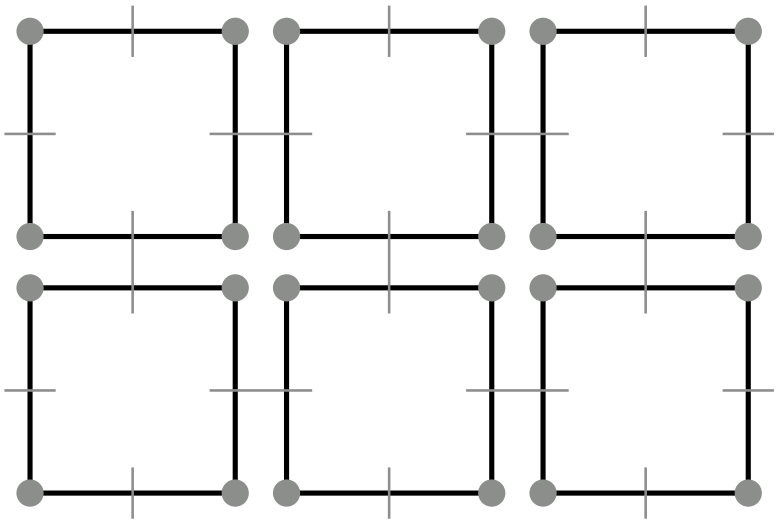
$B^S = \alpha_i(C) \setminus C$
$\forall b' \in B^S,\ \alpha'_i(b') = (\alpha_i\alpha_{(i+1)})^k \alpha_i(b')$
k le plus petit entier



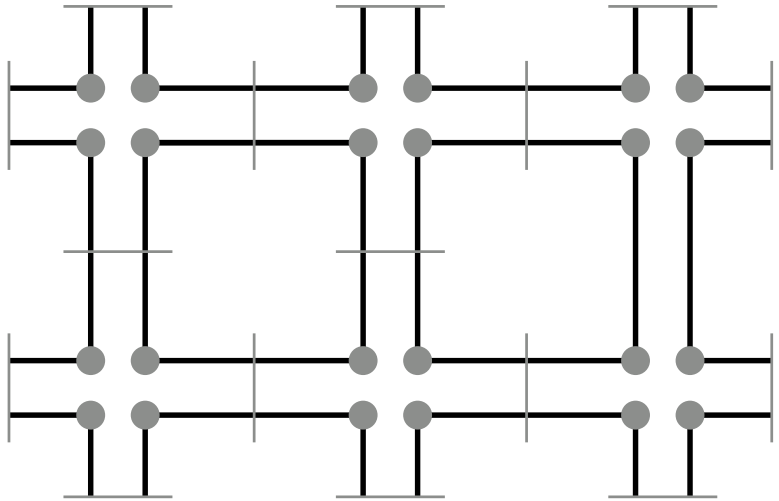
2-G-map finale

Contraction : dual de la suppression

$G = (B, \alpha_0, \ldots, \alpha_n) \Rightarrow$ dual $G' = (B, \alpha_n, \ldots, \alpha_0)$



2-G-carte initiale



2-G-carte duale

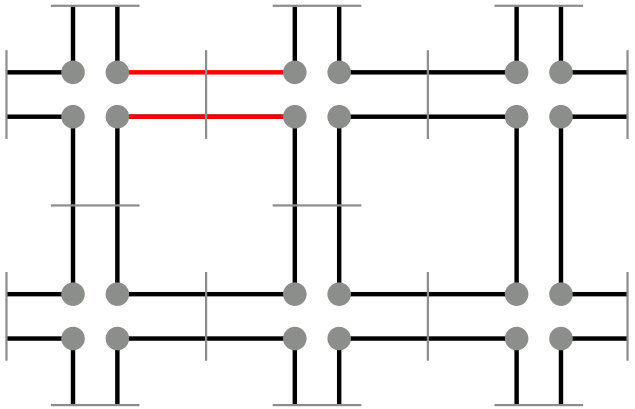
Contraction en dimension n

$C =< \alpha_0, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n > (b)$

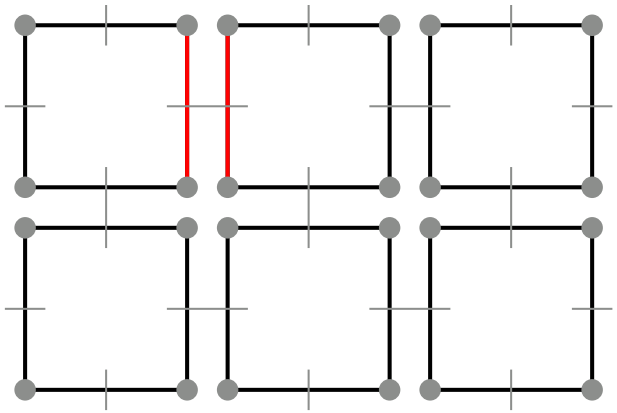
Préconditions :

- $i \in \{1, \ldots, n\}$
- $\forall b' \in C$,
  $\alpha_{(i-1)}\alpha_{(i-2)}(b') = \alpha_{(i-2)}\ \alpha_{(i-1)}(b')$
  (sauf pour $i = 1$)

$B^S = \alpha_i(C) \setminus C$
$\forall b' \in B^S,\ \alpha'_i(b') = (\alpha_i\alpha_{(i-1)})^k\alpha_i(b')$
k le plus petit entier



2-G-carte initiale ; arêtes à contracter



2-G-carte duale ; arêtes à supprimer

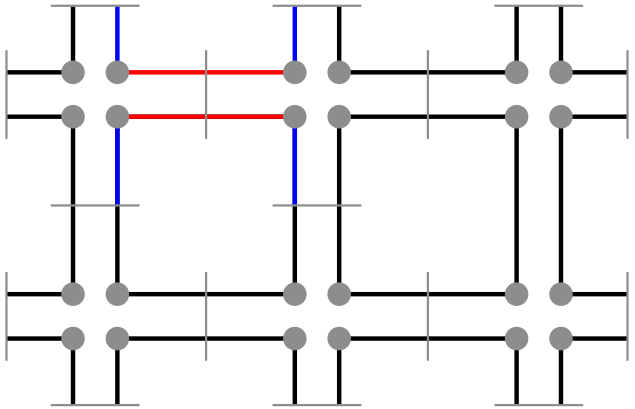
Contraction en dimension n

$C =< \alpha_0, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n > (b)$

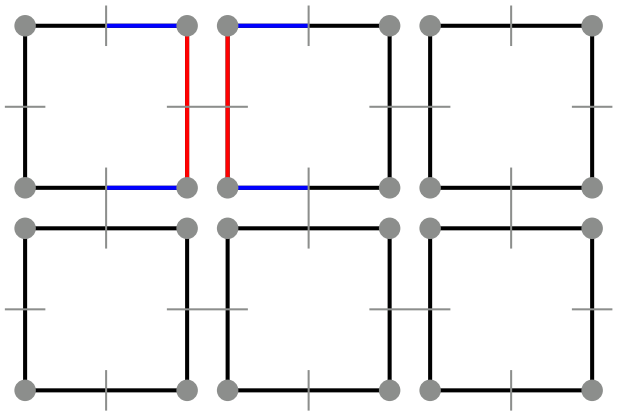
Préconditions :

- $i \in \{1, \ldots, n\}$
- $\forall b' \in C$,
  $\alpha_{(i-1)}\alpha_{(i-2)}(b') = \alpha_{(i-2)}\ \alpha_{(i-1)}(b')$
  (sauf pour $i = 1$)

$B^S = \alpha_i(C) \setminus C$
$\forall b' \in B^S,\ \alpha'_i(b') = (\alpha_i\alpha_{(i-1)})^k\alpha_i(b')$
k le plus petit entier



$B^S = C\alpha_1 - C$



$B^S = C\alpha_1 - C$

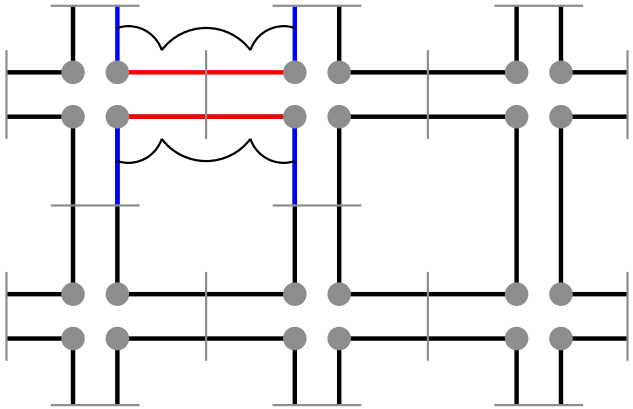
Contraction en dimension n

$C =< \alpha_0, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n > (b)$

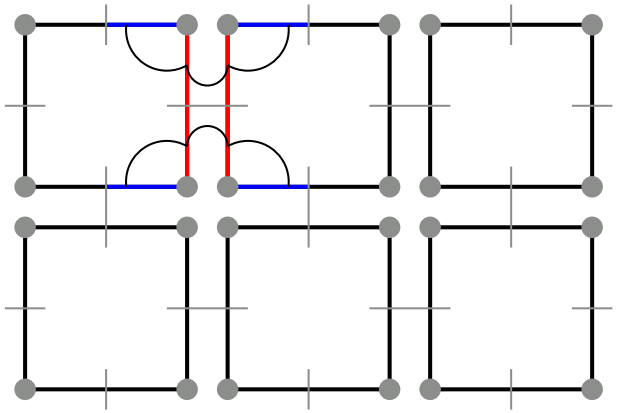
Préconditions :

- $i \in \{1, \ldots, n\}$
- $\forall b' \in C$,
  $\alpha_{(i-1)}\alpha_{(i-2)}(b') = \alpha_{(i-2)}\ \alpha_{(i-1)}(b')$
  (sauf pour $i = 1$)

$B^S = \alpha_i(C) \setminus C$
$\forall b' \in B^S, \alpha'_i(b') = (\alpha_i\alpha_{(i-1)})^k\alpha_i(b')$
k le plus petit entier



$\forall b' \in B^S, b'\alpha'_1 = b'(\alpha_1\alpha_0)^k\alpha_1$
k plus petit entier



$\forall b' \in B^S, b'\alpha'_1 = b'(\alpha_1\alpha_2)^k\alpha_1$
k plus petit entier

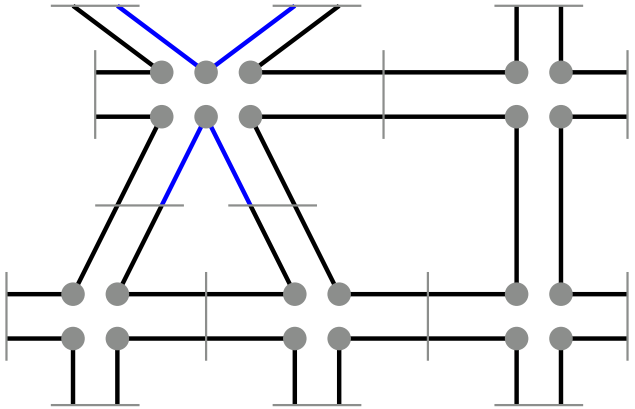
Contraction en dimension n

$C =< \alpha_0, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n > (b)$

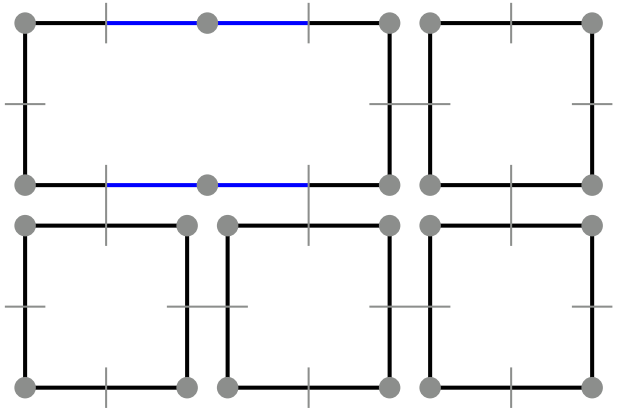
Préconditions :

- $i \in \{1, \ldots, n\}$
- $\forall b' \in C$,
  $\alpha_{(i-1)}\alpha_{(i-2)}(b') = \alpha_{(i-2)}\ \alpha_{(i-1)}(b')$
  (sauf pour $i = 1$)

$B^S = \alpha_i(C) \setminus C$
$\forall b' \in B^S,\ \alpha'_i(b') = (\alpha_i\alpha_{(i-1)})^k\alpha_i(b')$
k le plus petit entier



2-G-carte finale



2-G-carte finale

Part II : Applications

4. Segmentation d'Images

La segmentation

C'est regrouper les pixels en **zones homogènes**
$\Rightarrow$ Besoin de caractéristiques sur ces zones

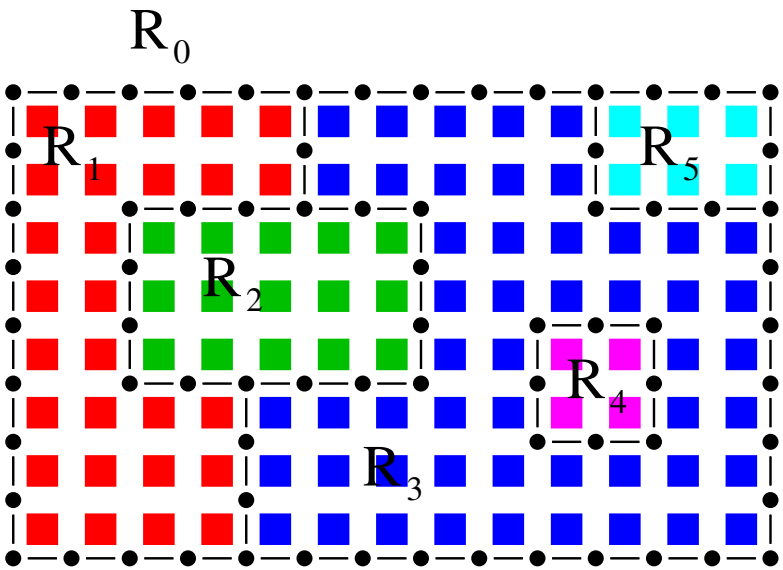
- **Moyenne des couleurs** des pixels
- Détection de **ruptures** dans l'image (frontières)
- Critères sur la **forme** des régions
- ...

Pour chaque critère $\Rightarrow$ besoin d'informations

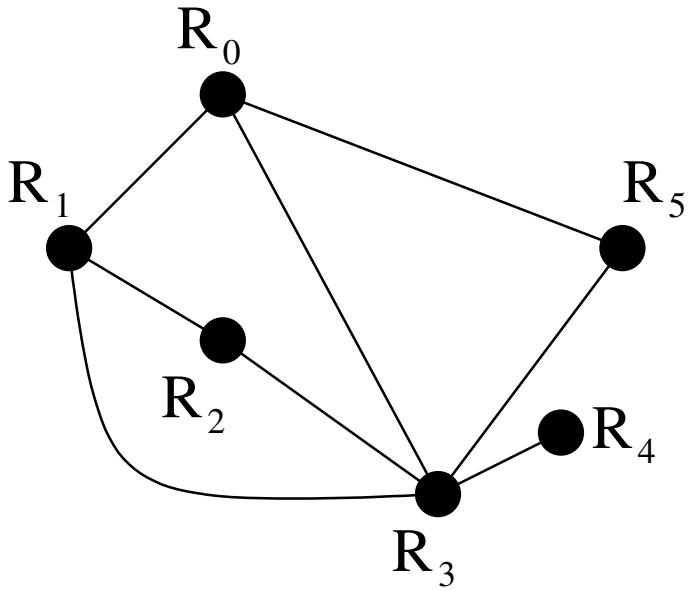
- un modèle de représentation
- des algorithmes

Un modèle de représentation : le RAG

Graphe d'Adjacence des Régions [Rosenfeld74]



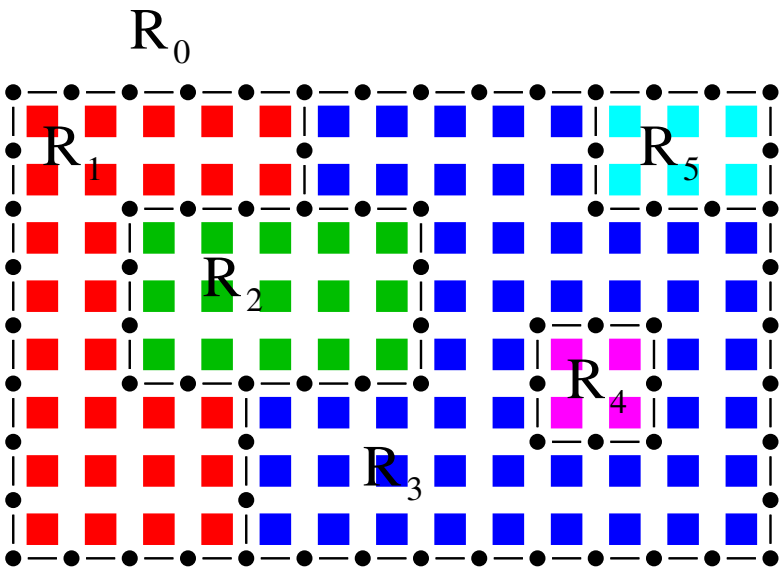
Image



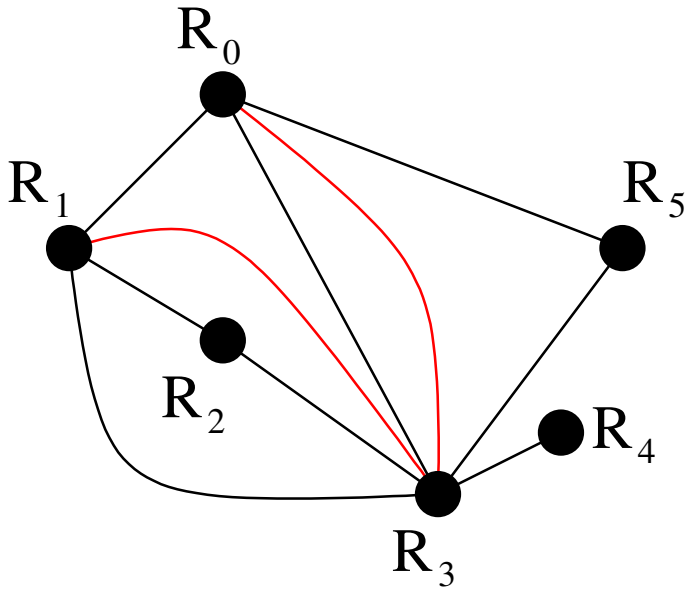
RAG

Un modèle de représentation : le RAG

Graphe d'Adjacence des Régions [Rosenfeld74]



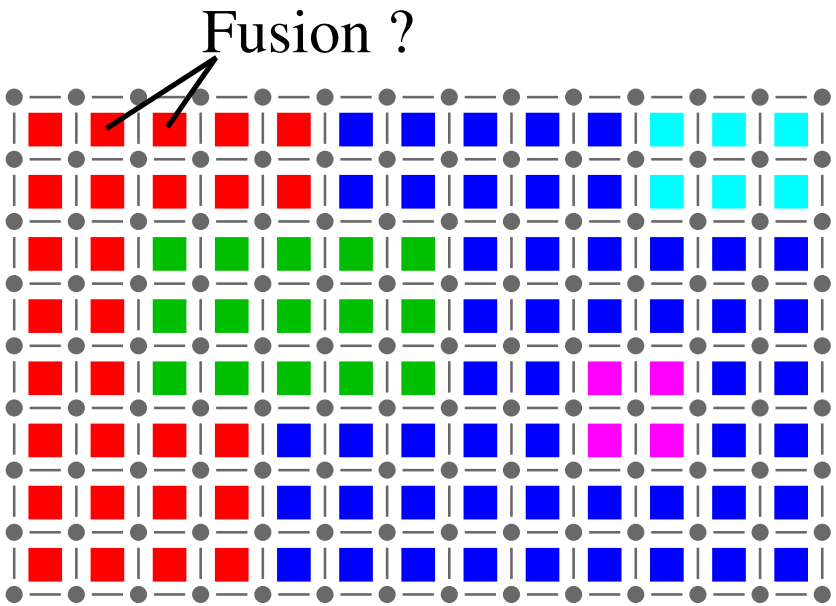
Image



multi-RAG

Exemple d'algo. : segmentation bottom-up

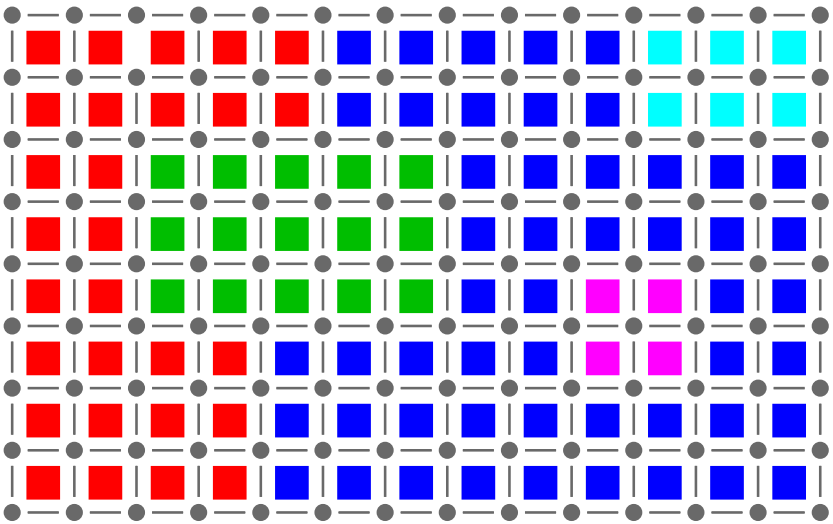
- Initialisation : chaque pixel est une région
- Tant que non fin
  - pour chaque région *r* faire
    - voir s'il faut fusionner *r* avec ses voisins



Exemple d'algo. : segmentation bottom-up

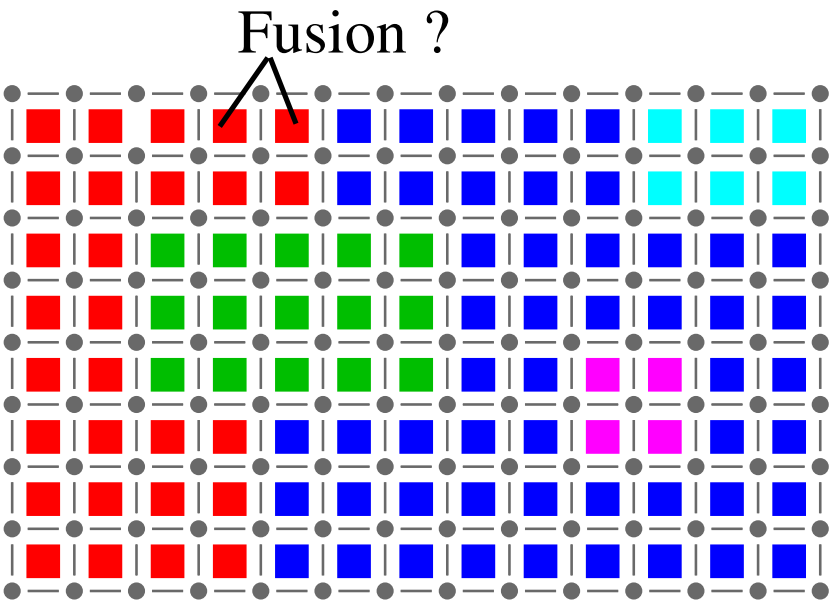
- Initialisation : chaque pixel est une région
- Tant que non fin
  - pour chaque région *r* faire
    - voir s'il faut fusionner *r* avec ses voisins

Fusion : oui



Exemple d'algo. : segmentation bottom-up

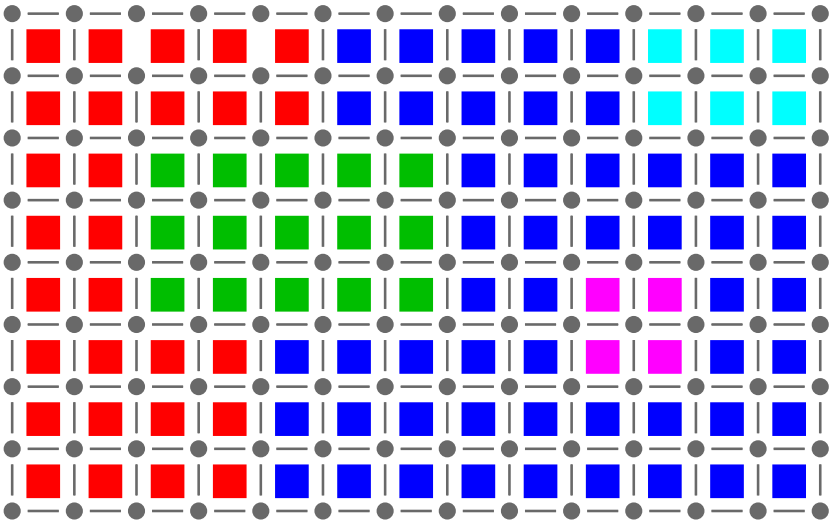
- Initialisation : chaque pixel est une région
- Tant que non fin
  - pour chaque région *r* faire
    - voir s'il faut fusionner *r* avec ses voisins



Exemple d'algo. : segmentation bottom-up

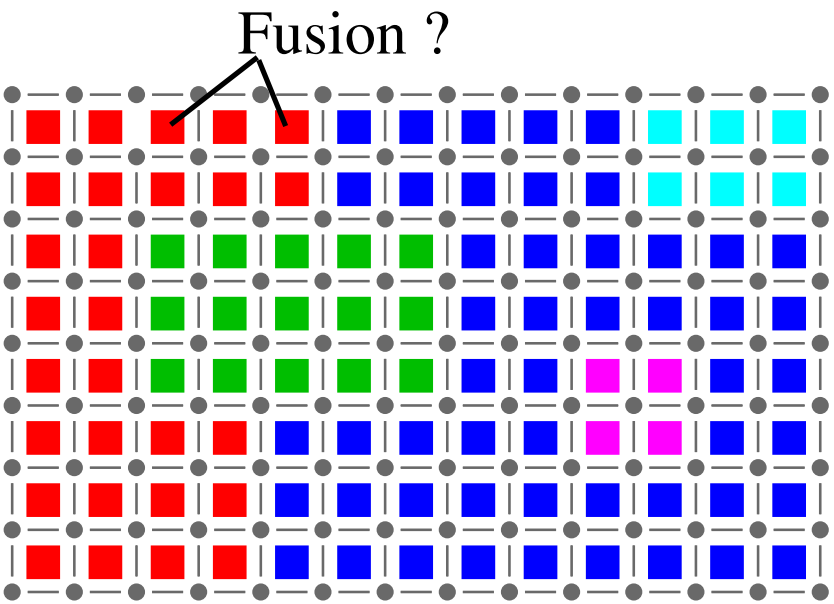
- Initialisation : chaque pixel est une région
- Tant que non fin
  - pour chaque région *r* faire
    - voir s'il faut fusionner *r* avec ses voisins

Fusion : oui



Exemple d'algo. : segmentation bottom-up

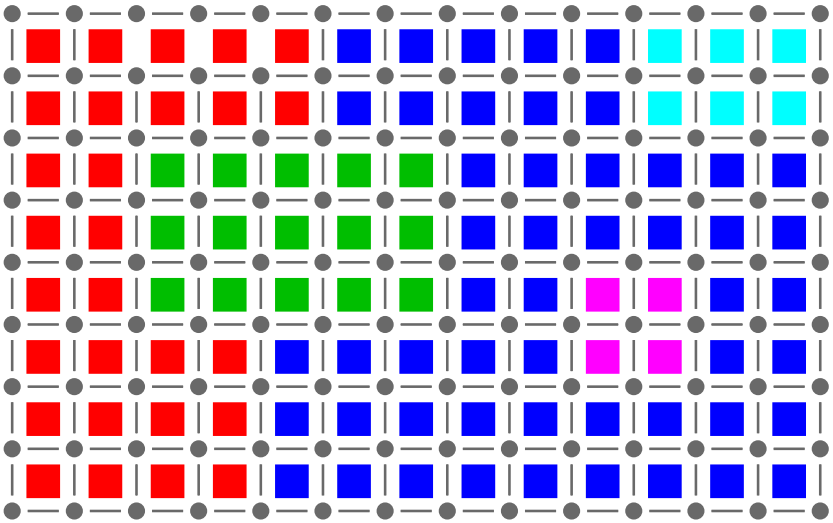
- Initialisation : chaque pixel est une région
- Tant que non fin
  - pour chaque région *r* faire
    - voir s'il faut fusionner *r* avec ses voisins



Exemple d'algo. : segmentation bottom-up

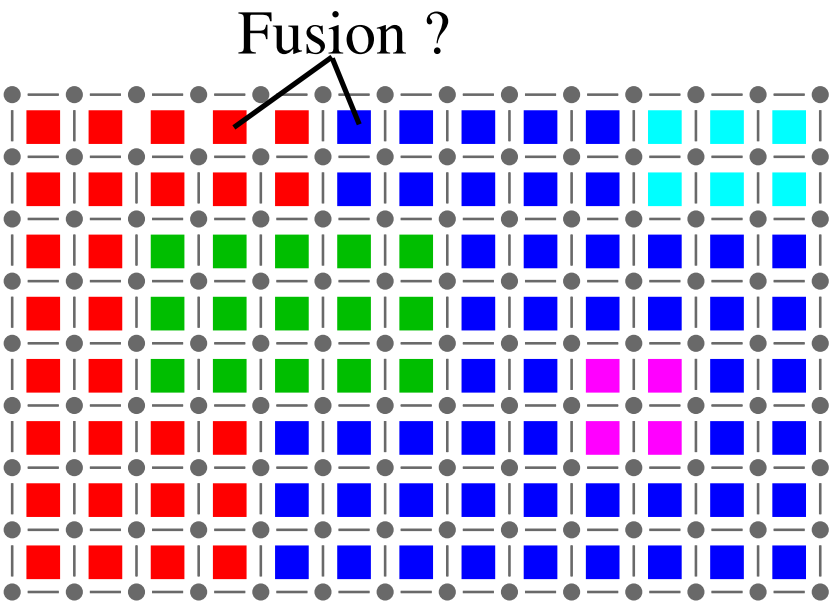
- Initialisation : chaque pixel est une région
- Tant que non fin
  - pour chaque région *r* faire
    - voir s'il faut fusionner *r* avec ses voisins

Fusion : oui



Exemple d'algo. : segmentation bottom-up

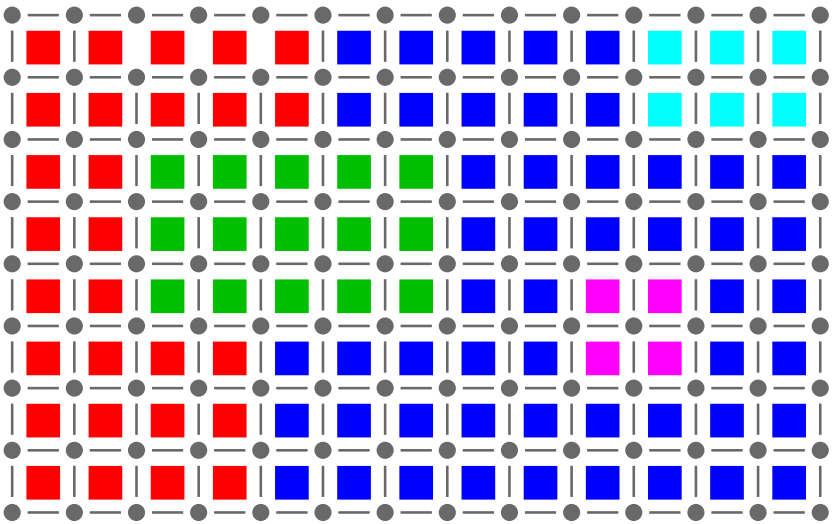
- Initialisation : chaque pixel est une région
- Tant que non fin
  - pour chaque région *r* faire
    - voir s'il faut fusionner *r* avec ses voisins



Exemple d'algo. : segmentation bottom-up

- Initialisation : chaque pixel est une région
- Tant que non fin
  - pour chaque région *r* faire
    - voir s'il faut fusionner *r* avec ses voisins

Fusion : non

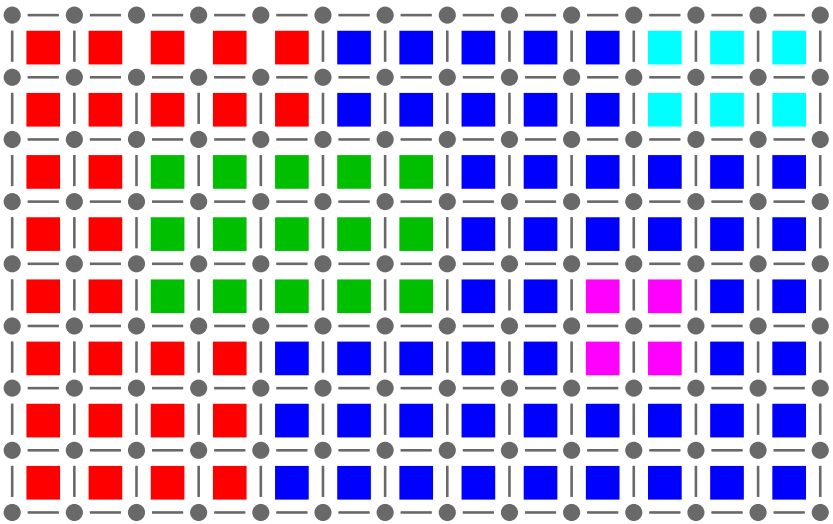


Complexité ?

Exemple d'algo. : segmentation bottom-up

- Initialisation : chaque pixel est une région
- Tant que non fin
  - pour chaque région *r* faire
    - voir s'il faut fusionner *r* avec ses voisins

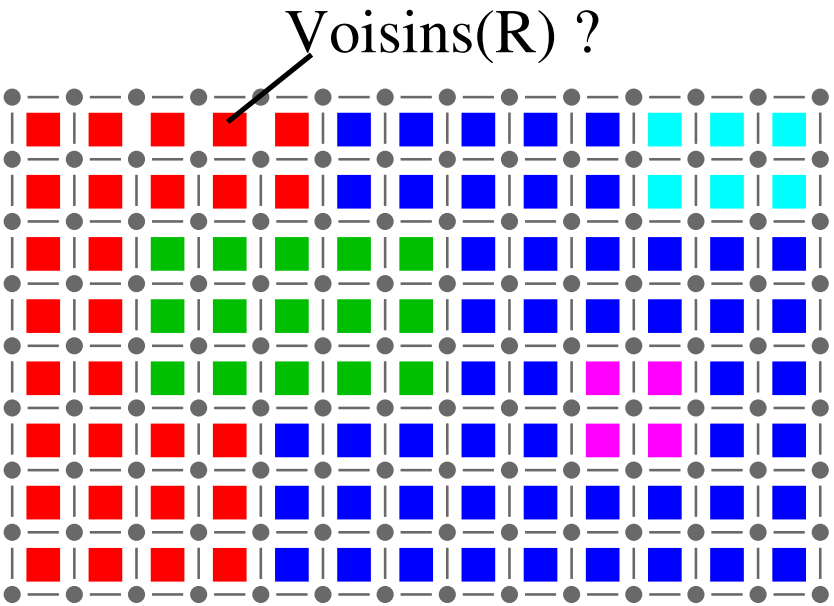
Fusion : non



Complexité : $O(n^2)$
Si test de tout les couples de régions

Exemple d'algo. : segmentation bottom-up

- Initialisation : chaque pixel est une région
- Tant que non fin
  - pour chaque région *r* faire
    - voir s'il faut fusionner *r* avec ses voisins

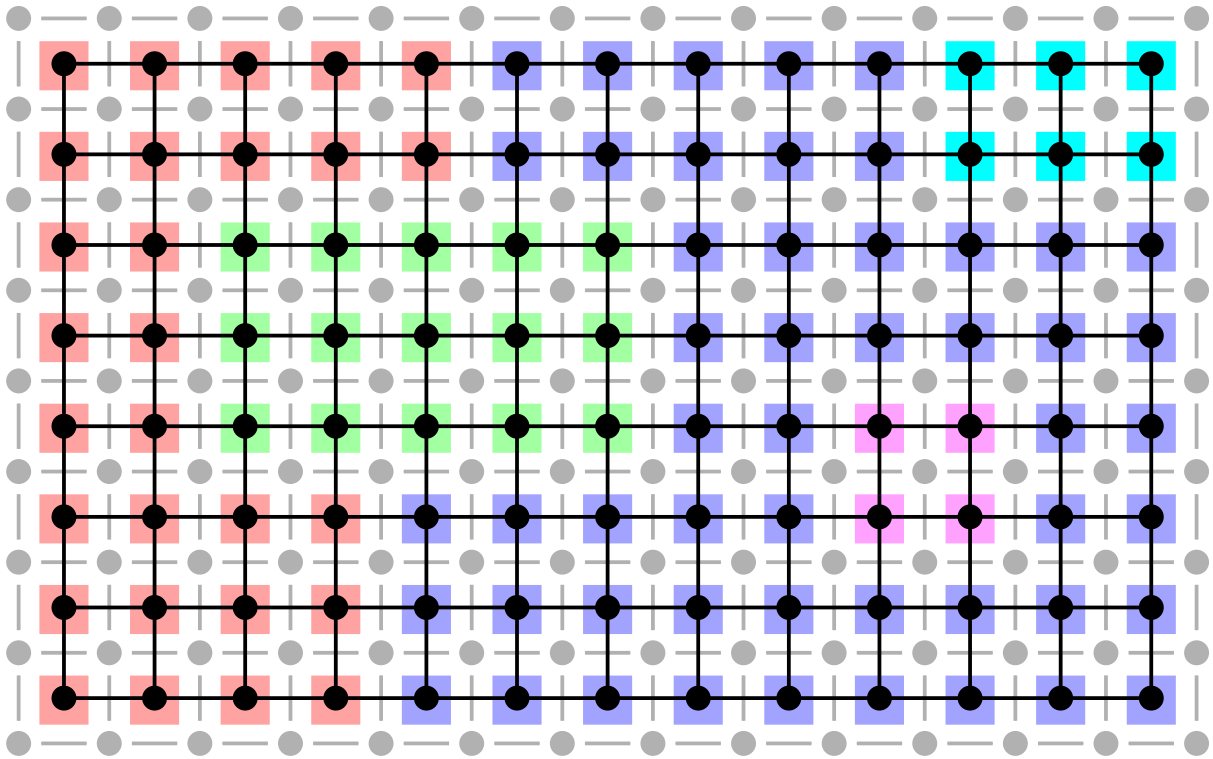


Question :
Comment récupérer les voisins d'une région *R* ?
Réponse :
Avec un RAG

Segmentation bottom-up et RAG

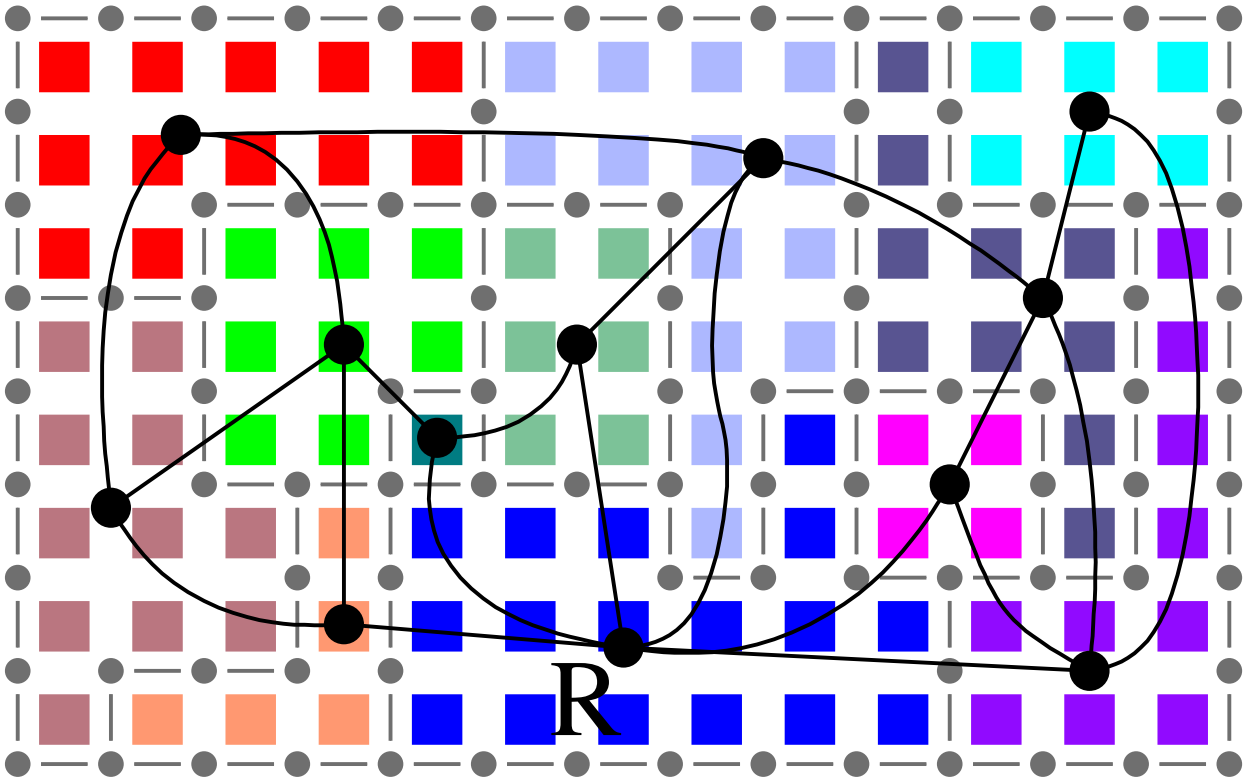
Initialisation :

1. pixel $\Rightarrow$ sommet dans le graphe
2. Deux pixels voisins $\Rightarrow$ arête dans le graphe



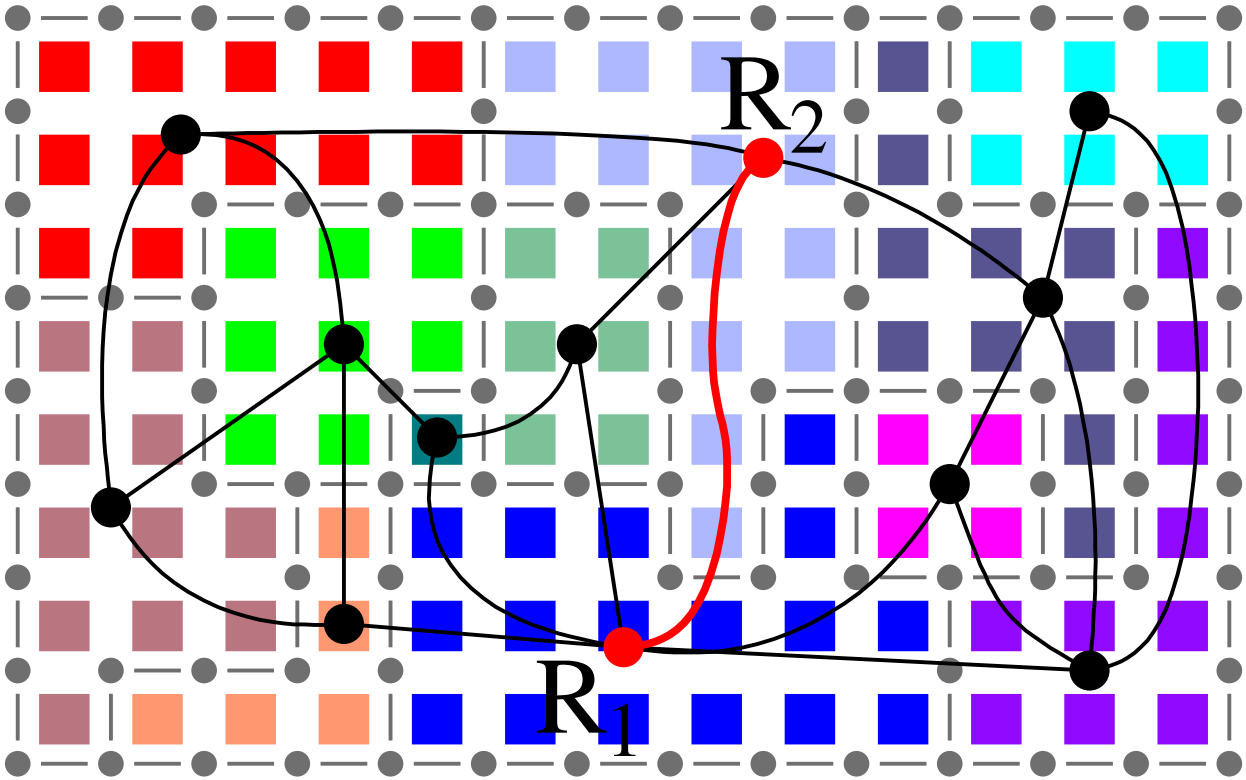
Segmentation bottom-up et RAG

Parcours :
Les voisins d'une région $R \Rightarrow$ voisins dans le graphe



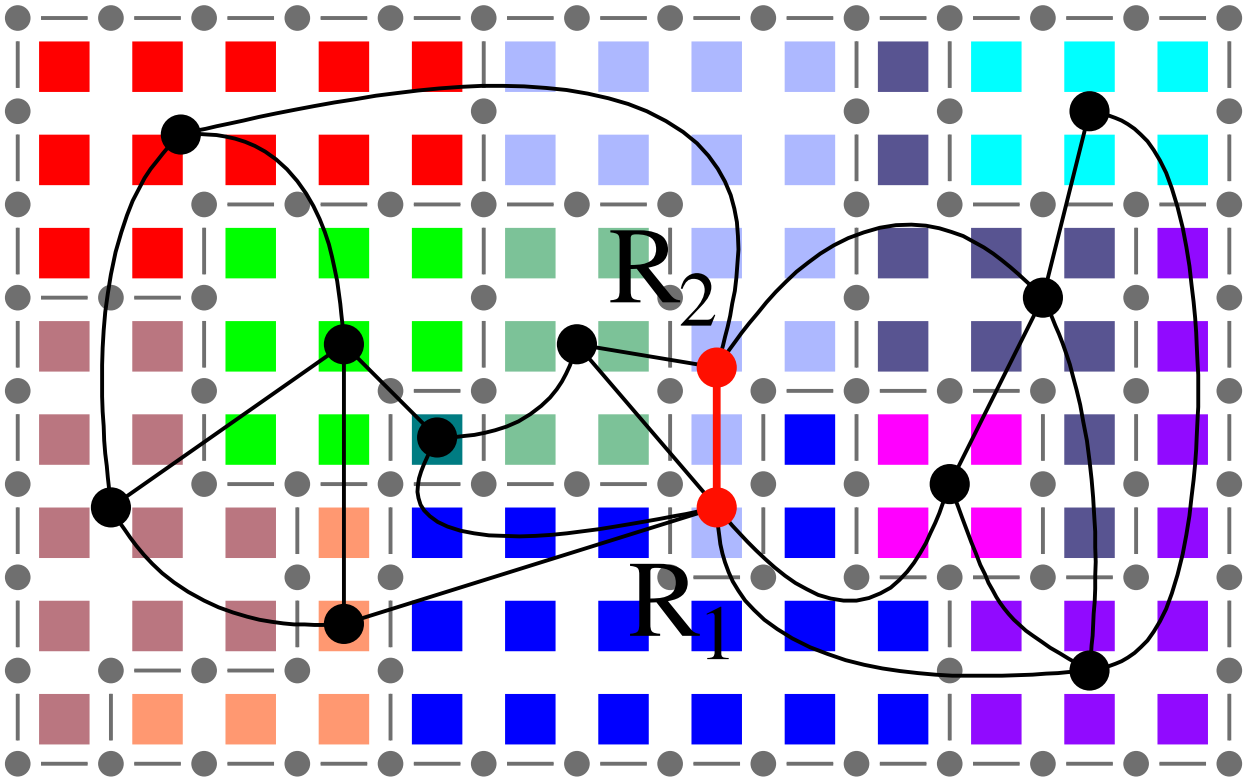
Segmentation bottom-up et RAG

Modification :
Fusionner deux régions R_1 et $R_2 \Rightarrow$ contraction d'arête



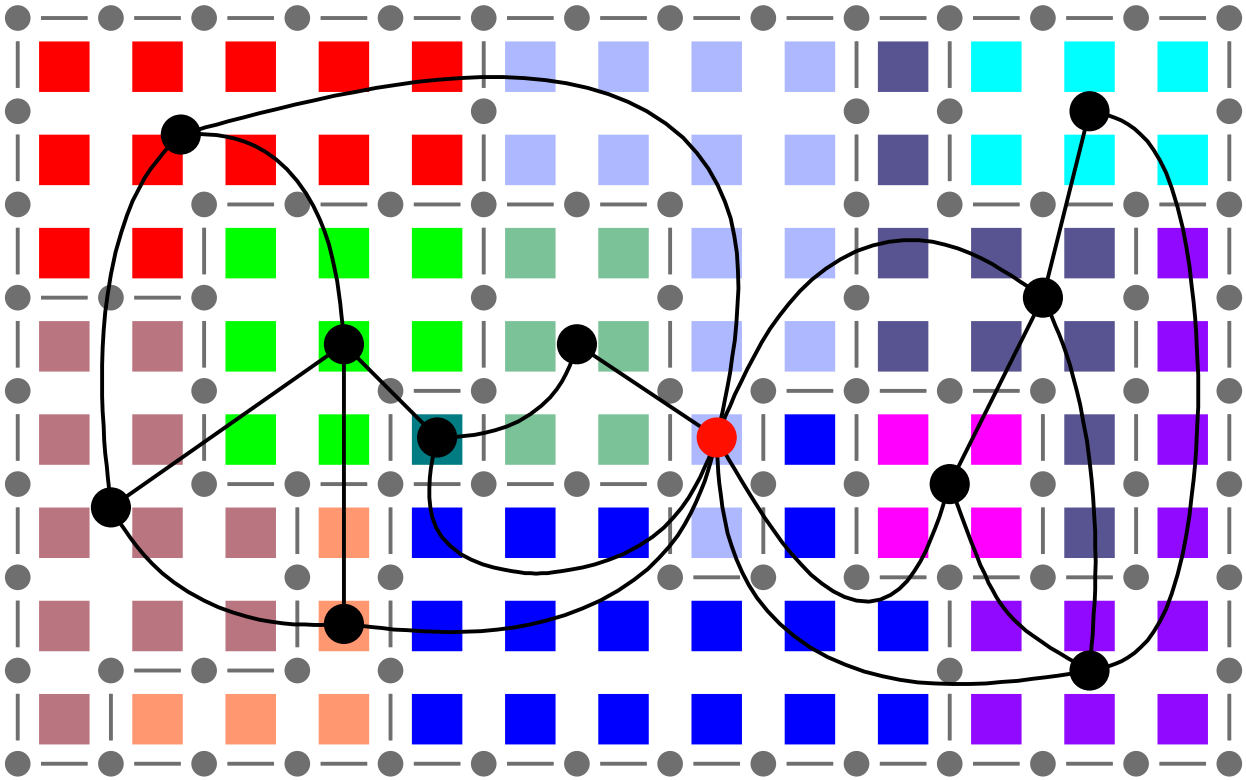
Segmentation bottom-up et RAG

Modification :
Fusionner deux régions R_1 et $R_2 \Rightarrow$ contraction d'arête



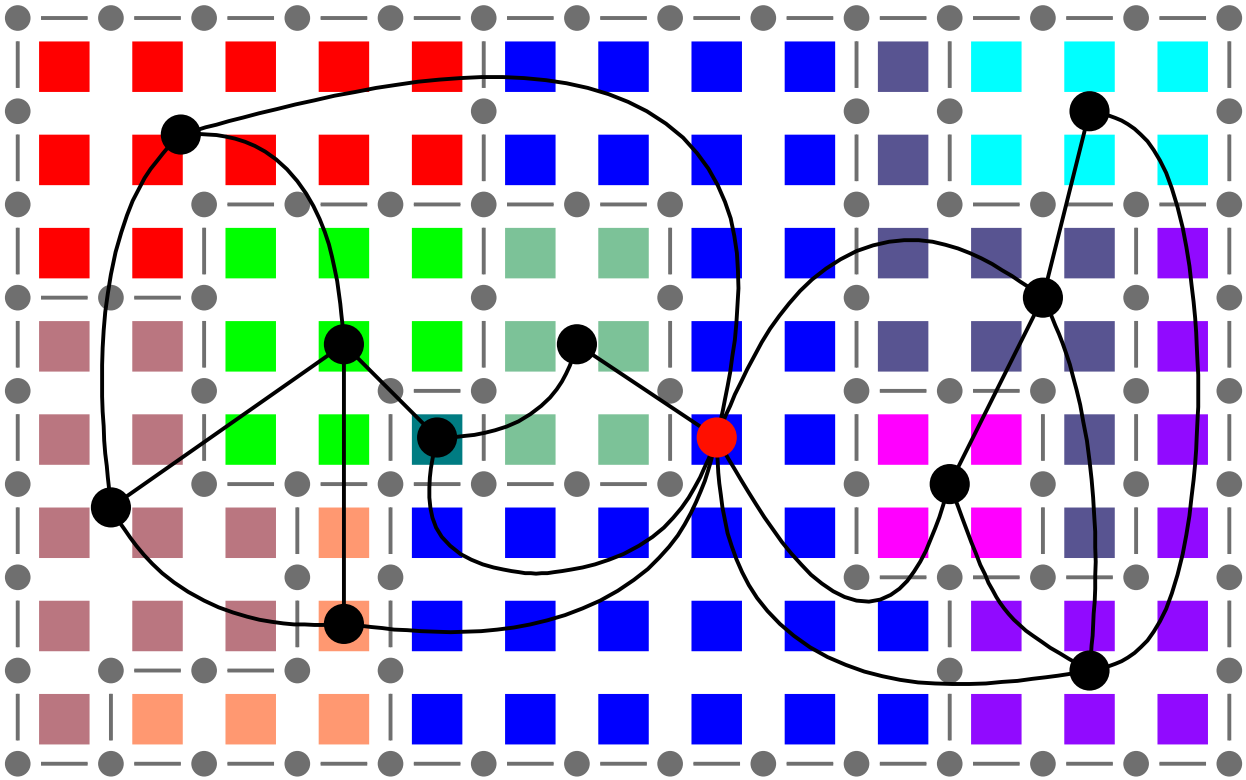
Segmentation bottom-up et RAG

Modification :
Fusionner deux régions R_1 et $R_2 \Rightarrow$ contraction d'arête

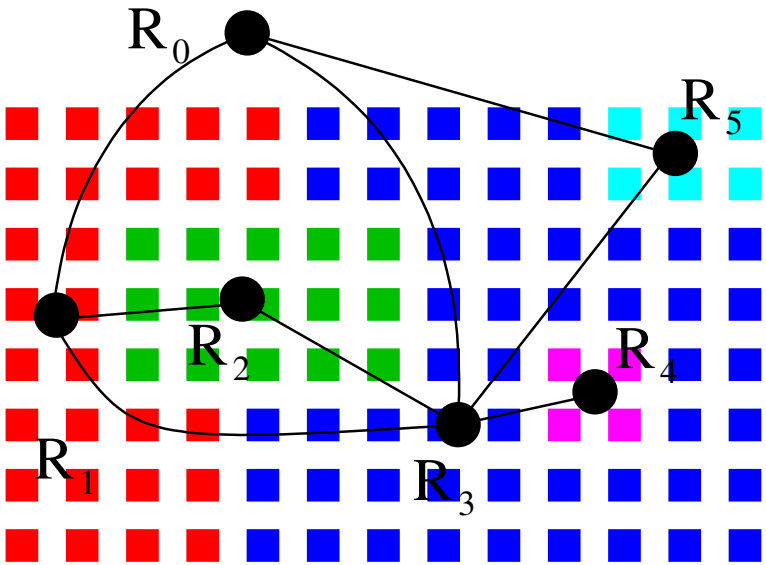


Segmentation bottom-up et RAG

Modification :
Fusionner deux régions R_1 et $R_2 \Rightarrow$ contraction d'arête



Les problèmes du RAG



Problèmes

- pas de multi-adjacence
- pas de représentation des sommets
- pas d'ordre des arêtes
- pas de notion d'imbrication

$\Rightarrow$ On n'a pas toutes les informations

Note : C'est encore pire en dimension supérieure

Extensions du RAG

Plusieurs modèles résolvent +/- partiellement ces problèmes

- Graphe d'adjacence des régions [Rosenfeld74]
- Graphes duaux [Kropatsch95]
- Carte planaire et frontières interpixel [Domenger92]
- Graphe topologique des frontières [Fiorio95]

Problème : modèles **difficilement extensibles** en dimension supérieure

Définition d'un **nouveau modèle** : la carte topologique

Les niveaux de simplification

Principe

- construire la carte représentant tous les pixels de l'image
- simplifier progressivement cette carte

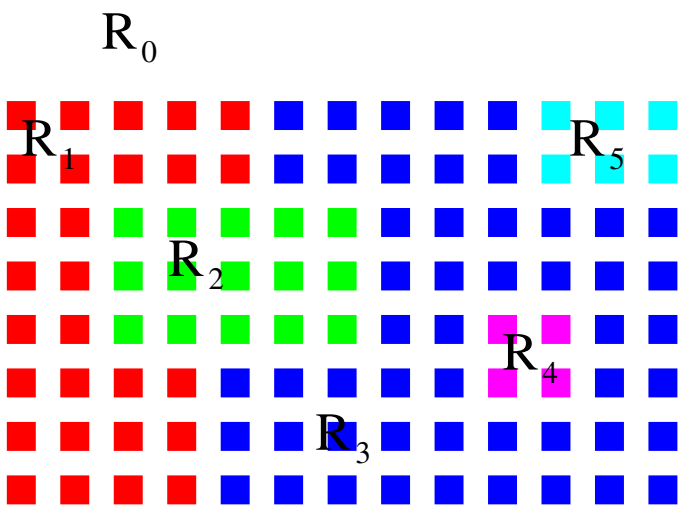
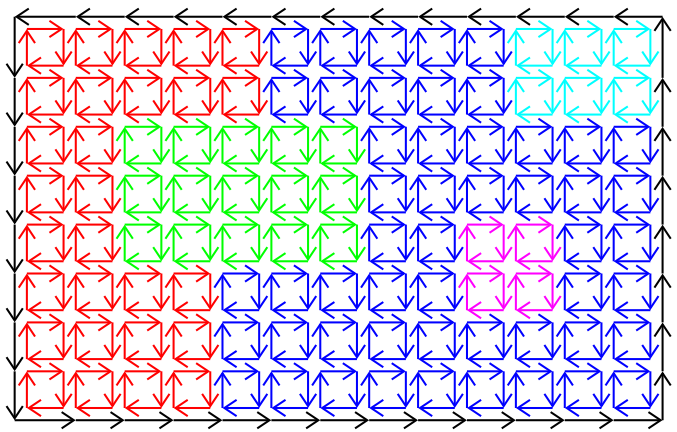


Image segmentée en régions

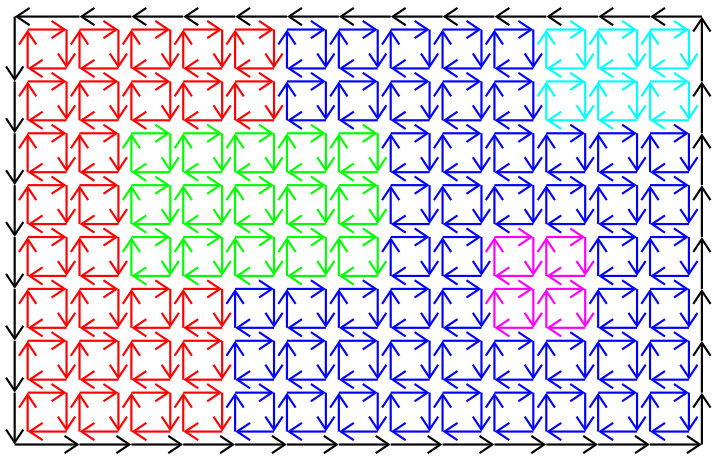


Niveau 0

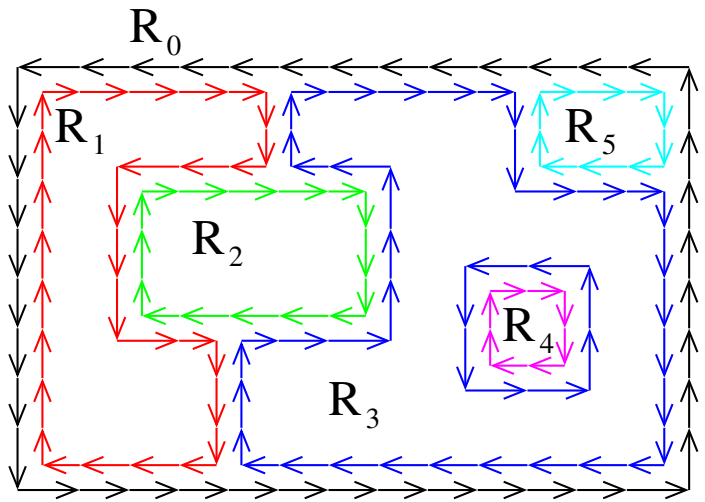
Niveau 1

Definition

Niveau 1 : Suppression des arêtes entre deux pixels d'une même région



Niveau 0

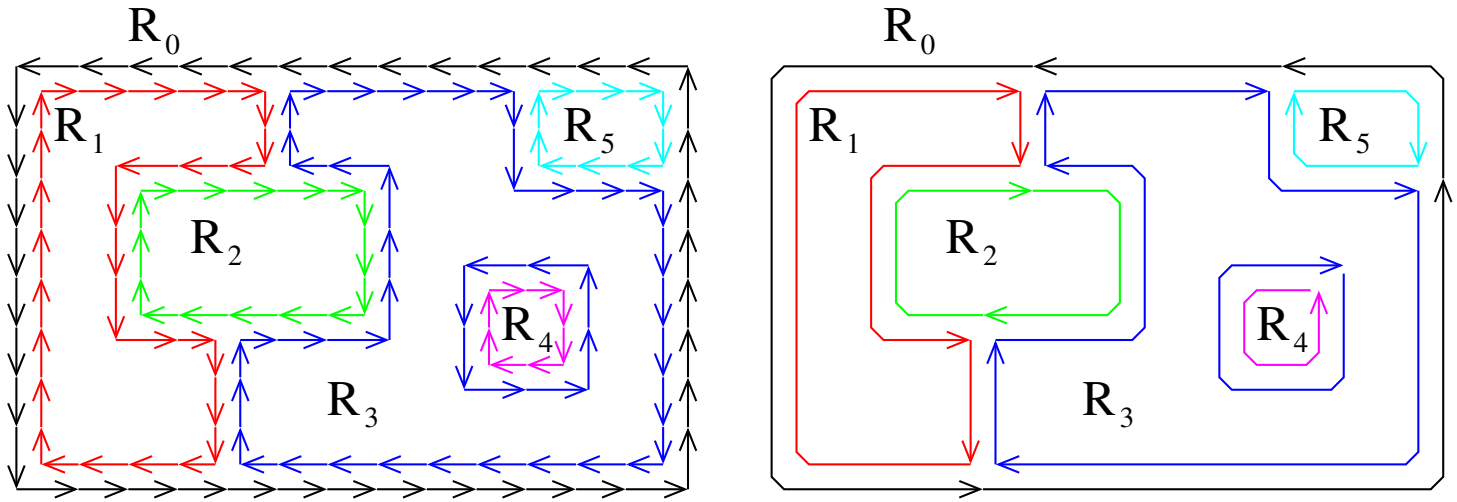


Niveau 1

Niveau 2

Definition

Niveau 2 : Suppression des sommets de degré 2

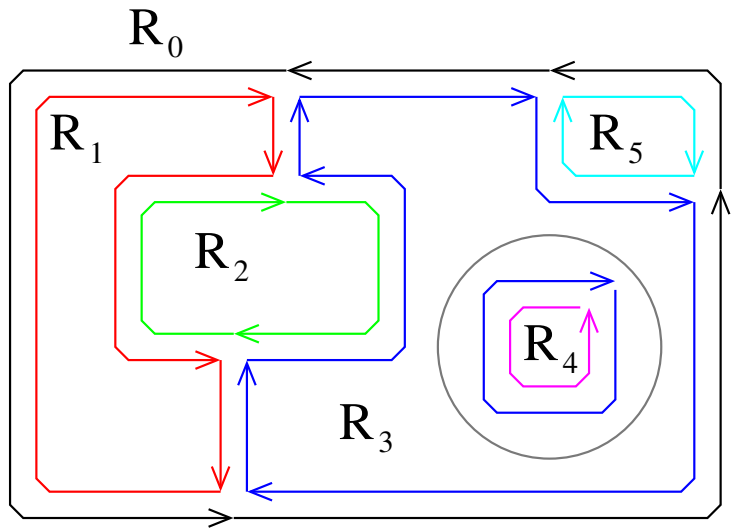


Niveau 1 Niveau 2

Remarque : ces définitions ⇒ algorithme d'extraction linéaire

Problème de déconnexion

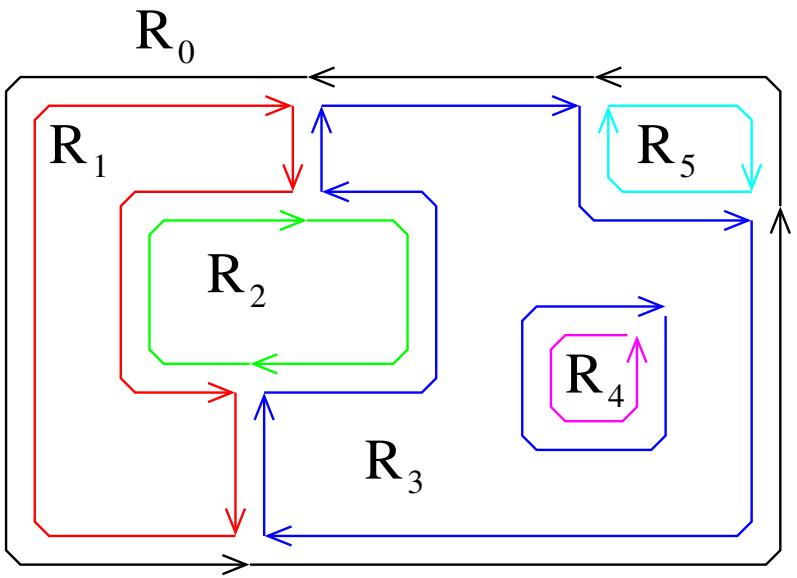
La carte peut avoir plusieurs composantes connexes :

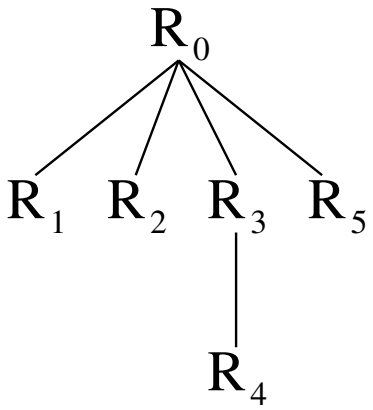


⇒ Perte d'information : comment positionner les différentes composantes ?

Notion de contour extérieur et intérieur

Une solution : l'arbre d'imbrication





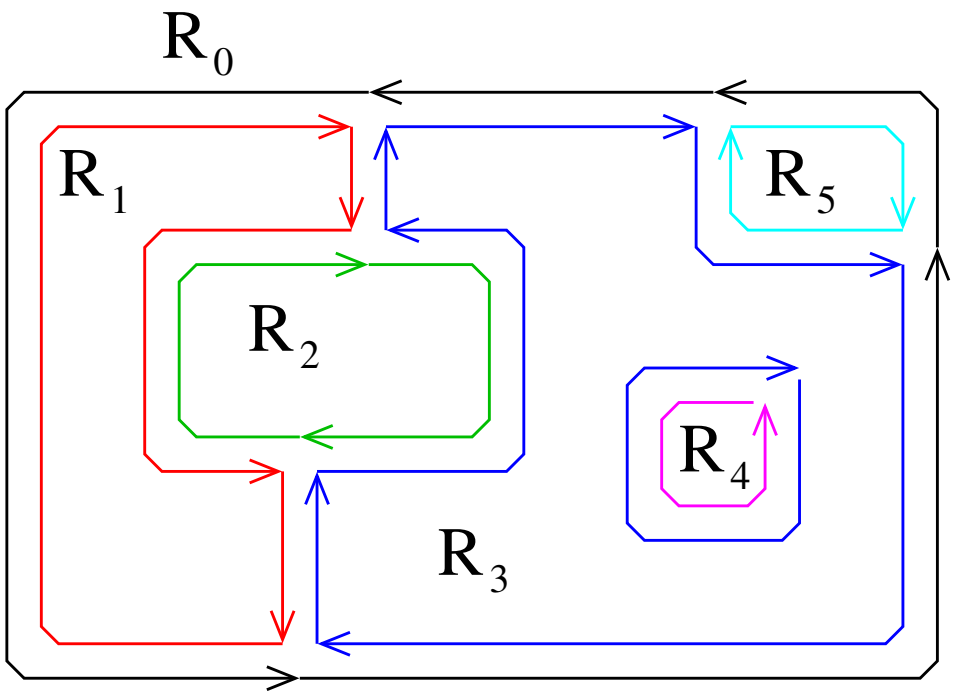
Pour conserver les informations topologiques :

- chaque brin connaît sa région d'appartenance
- chaque région connaît un brin du contour extérieur

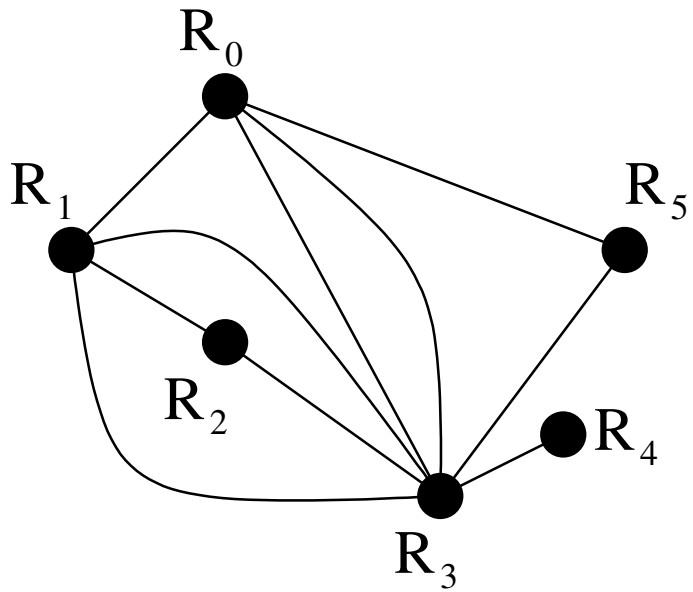
Niveau 3 + arbre d'imbrication = **carte topologique**

Carte topologique 2D

Carte topologique 2D = extension directe du multi-RAG



Carte topologique



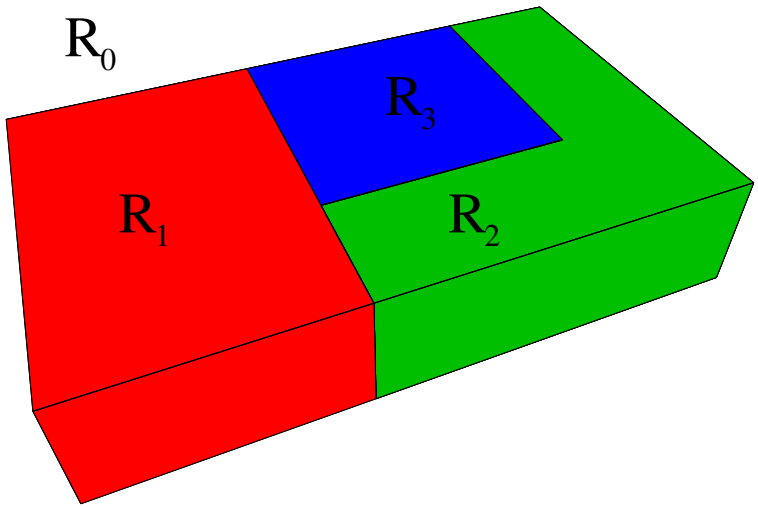
multi-RAG

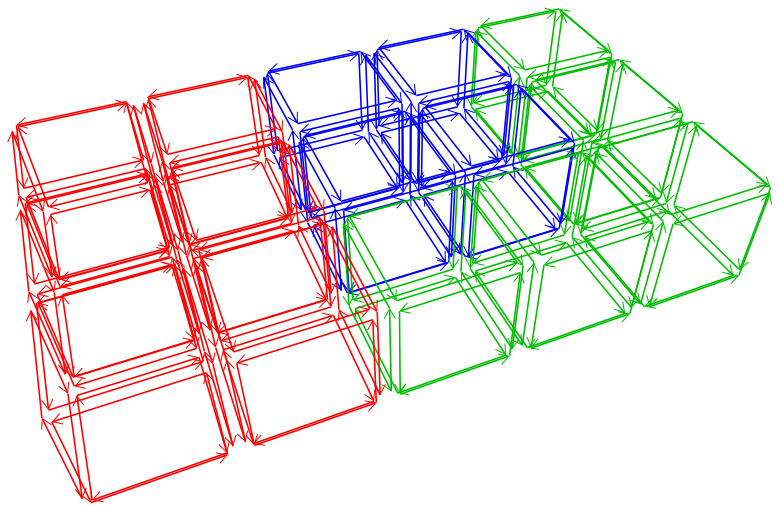
Définition progressive : Niveau 0

Niveaux de simplification $\Rightarrow$ extension directe en 3D

Definition

Niveau 0 : Tous les voxels.

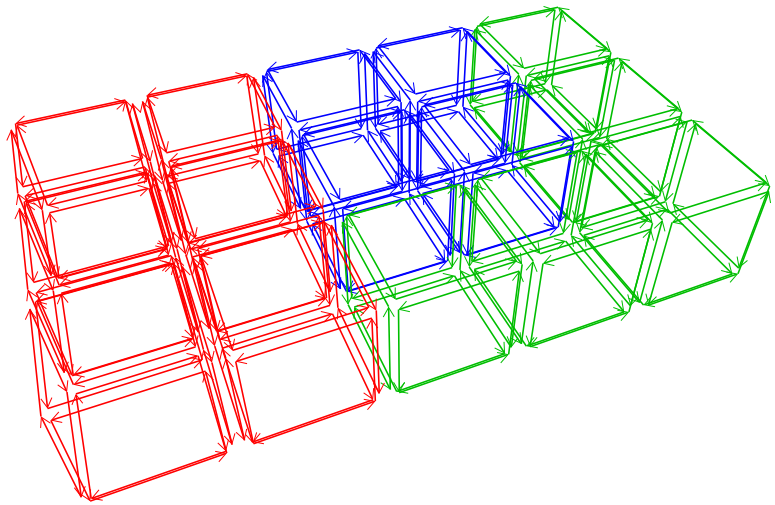


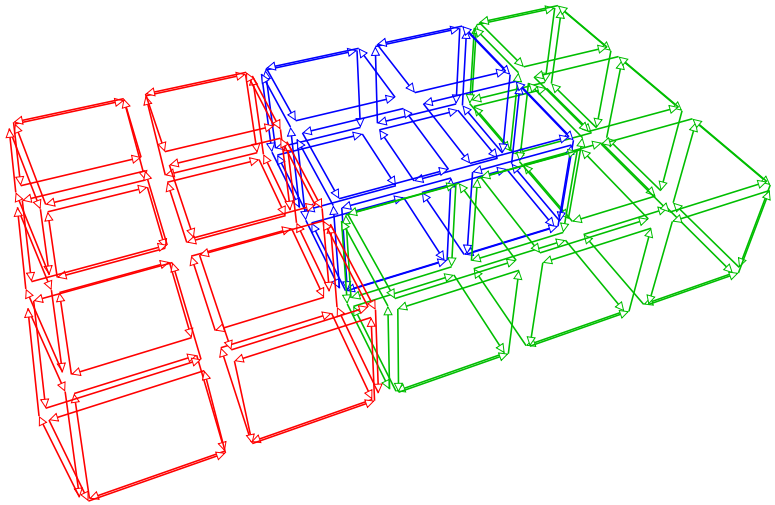


Niveau 1

Definition

Niveau 1 : Suppression des faces entre 2 voxels d'une même région

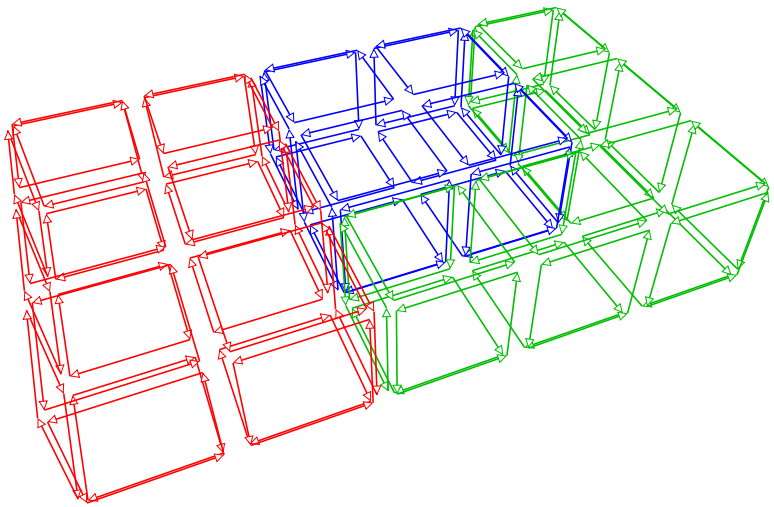


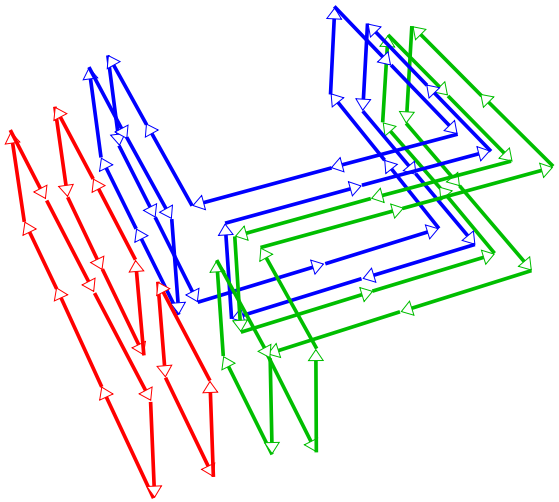


Niveau 2

Definition

Niveau 2 : Suppression des arêtes de degré 2 et arêtes pendantes

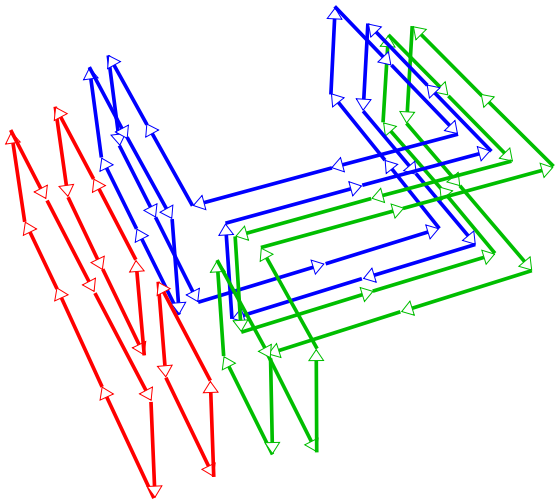


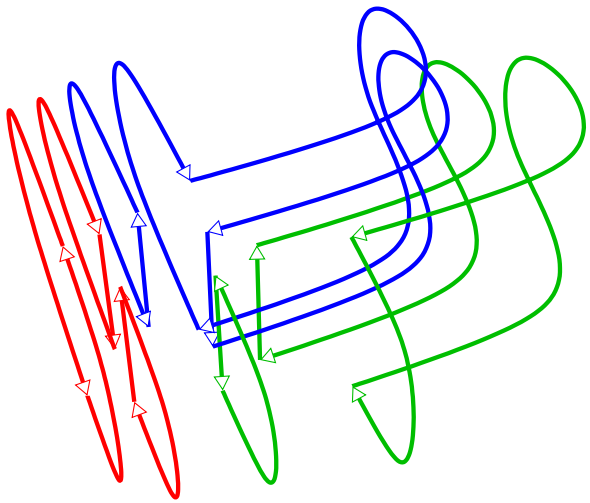


Niveau 3

Definition

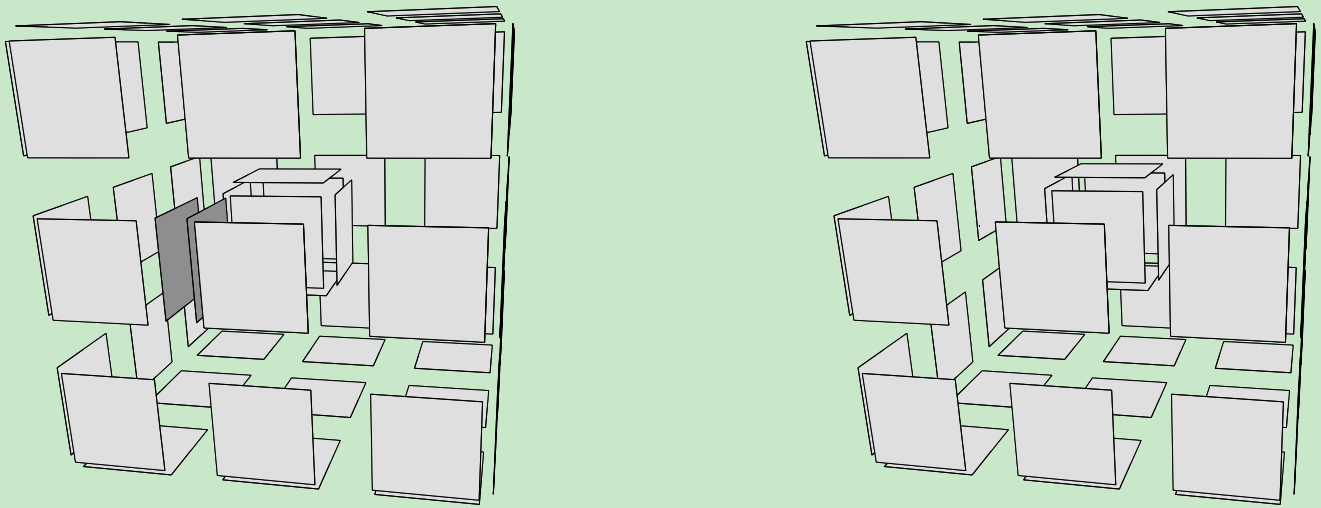
Niveau 3 : Suppression des sommets de degré 2





Déconnexion de volume

Exemple de déconnexion

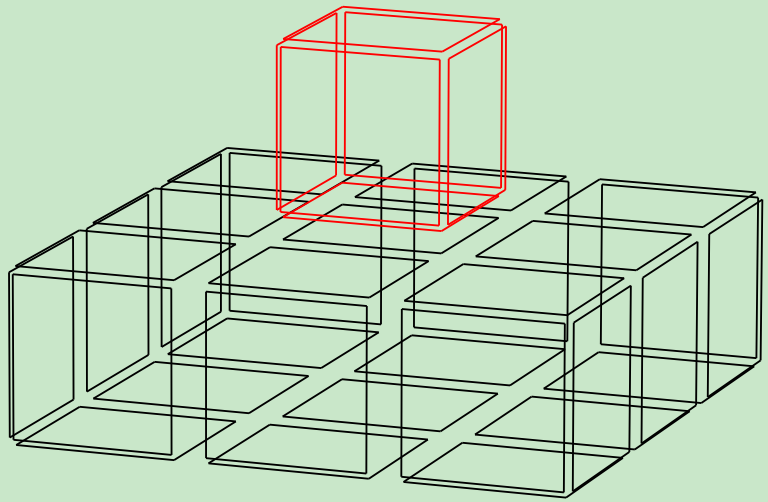


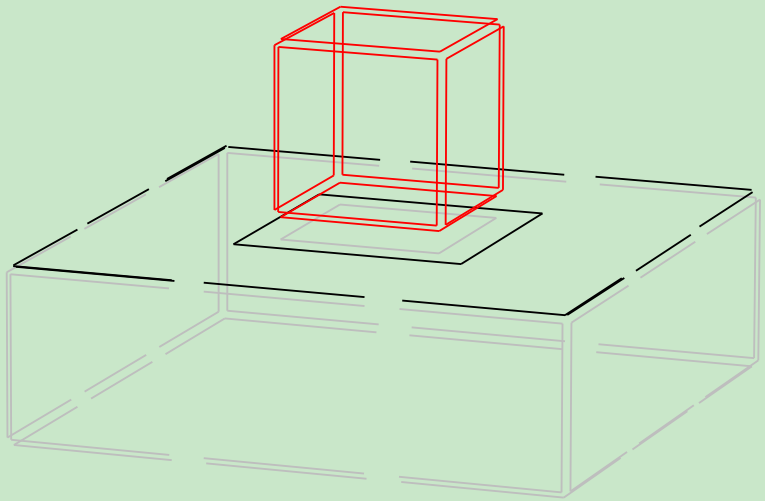
Solution

Ajout d'un arbre d'imbrication des régions

Déconnexion de face

Exemple





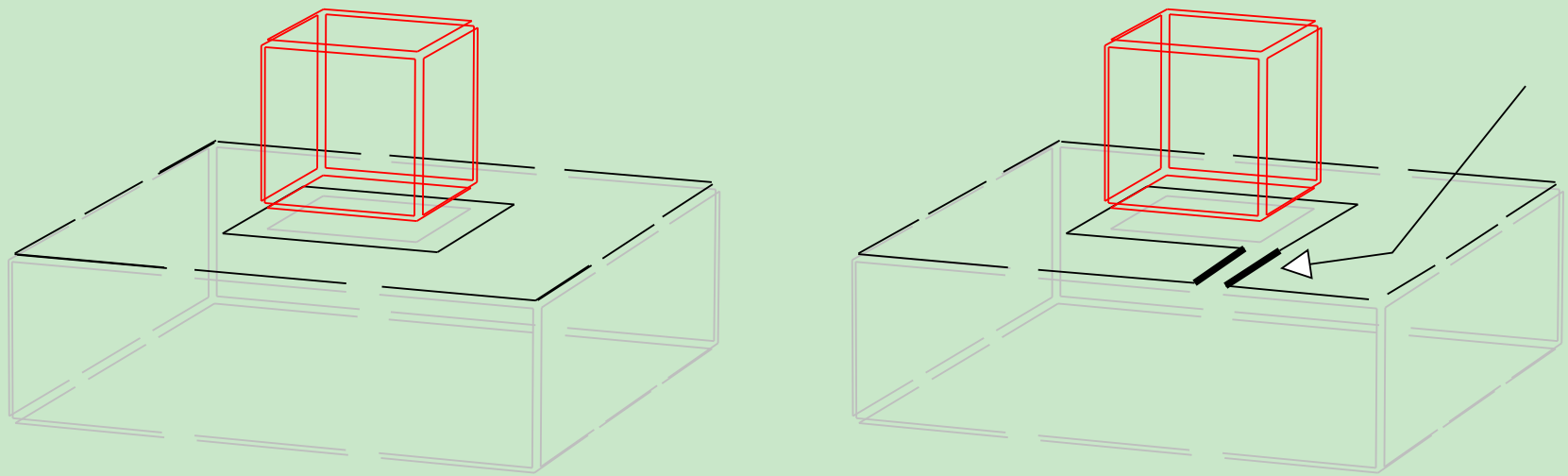
Solution

Garder des arêtes fictives

- arête de degré un dont la suppression $\Rightarrow$ déconnexion

Déconnexion de face

Exemple



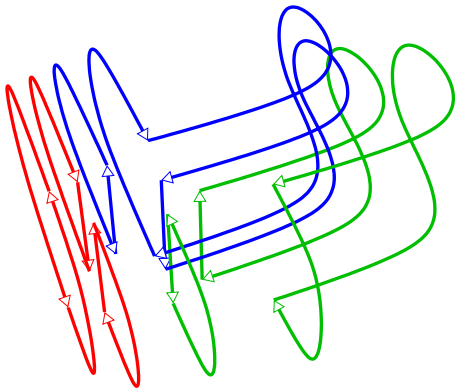
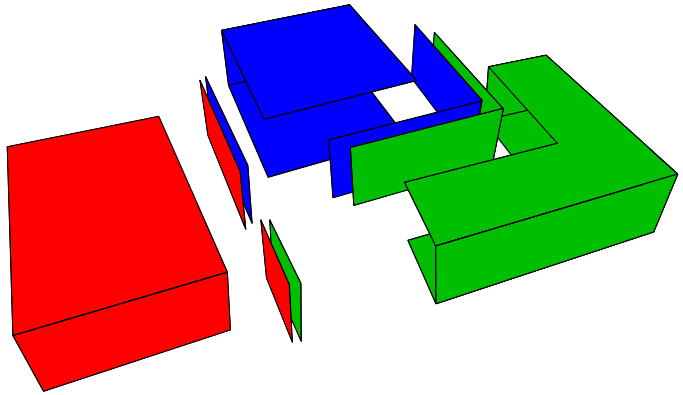
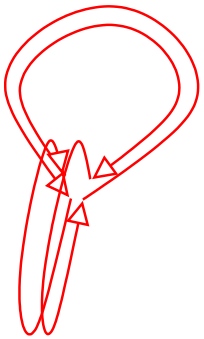
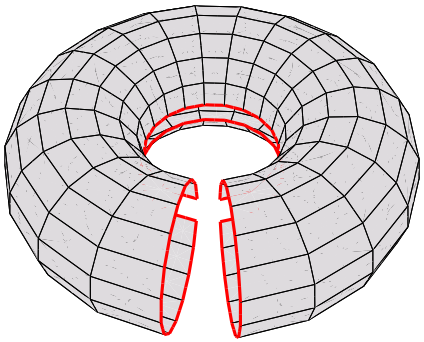
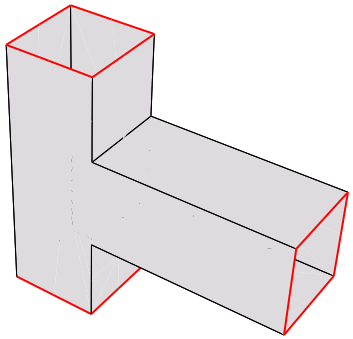
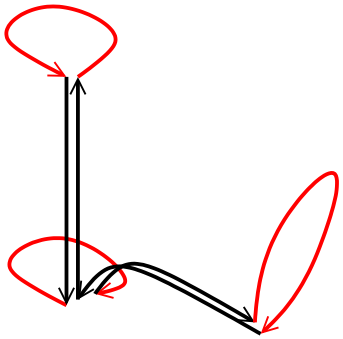
Solution

Garder des arêtes fictives

- arête de degré un dont la suppression ⇒ déconnexion

Carte topologique 3D

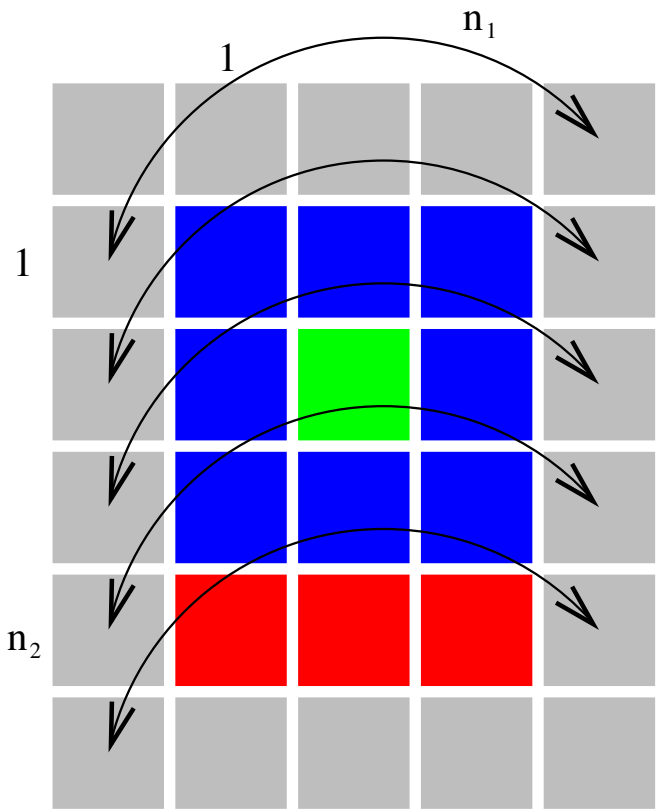
Carte topologique 3D : chaque face frontière de manière minimale
Arêtes fictives : arêtes de degré 1

Algorithme incrémental d'extraction

Principe

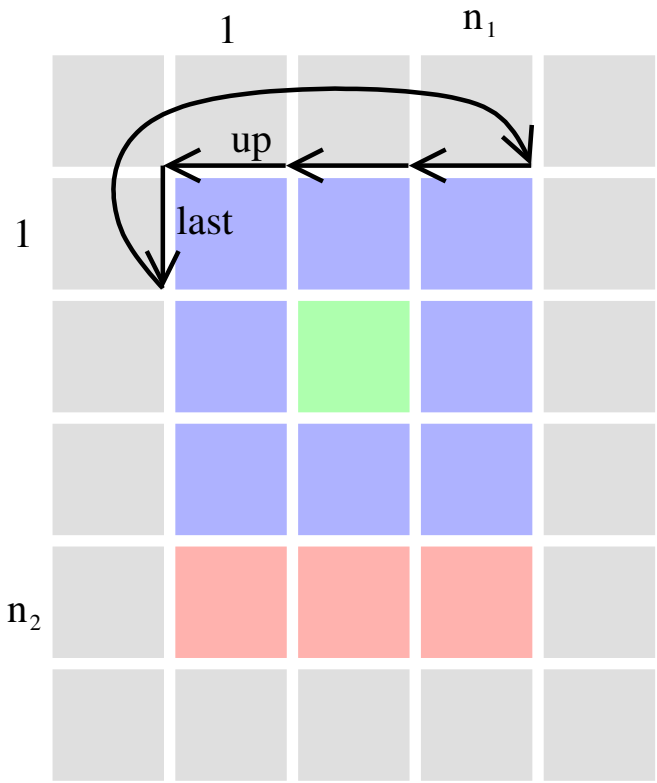
- Balayer l'image : parcours *scanline*
- Pour chaque pixel
  1. créer un carré dans la carte
  2. le coudre à la carte déjà existante
  3. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Principe

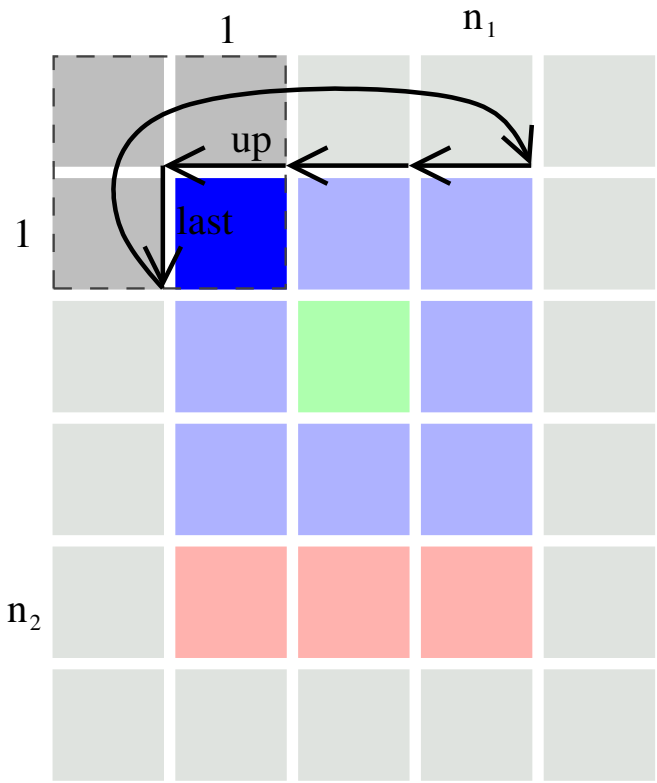
- Balayer l'image : parcours *scanline*
- Pour chaque pixel
  1. créer un carré dans la carte
  2. le coudre à la carte déjà existante
  3. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Principe

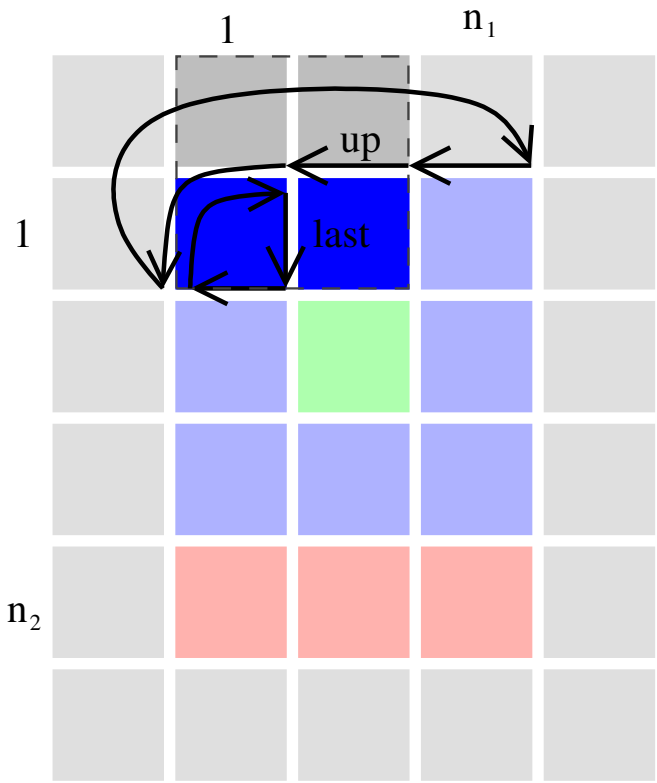
- Balayer l'image : parcours *scanline*
- Pour chaque pixel
  1. créer un carré dans la carte
  2. le coudre à la carte déjà existante
  3. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Principe

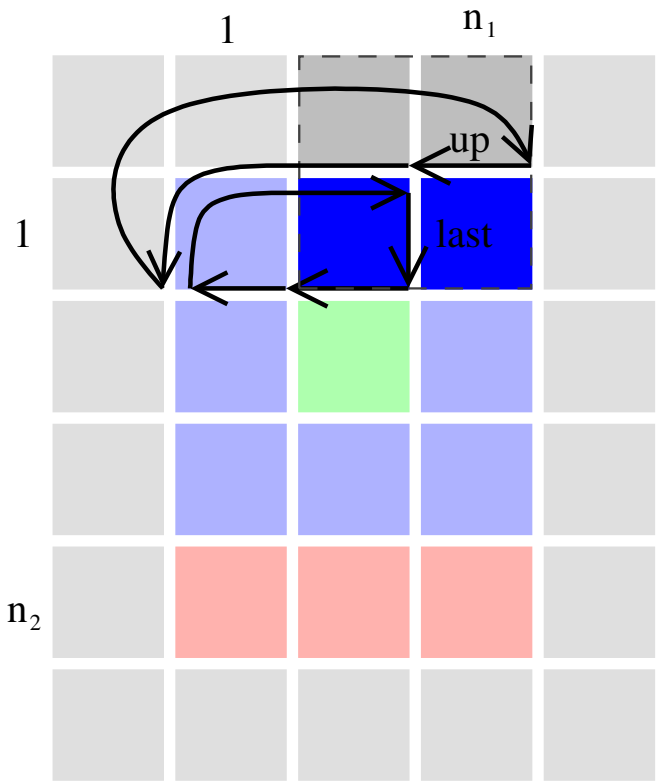
- Balayer l'image : parcours *scanline*
- Pour chaque pixel
  1. créer un carré dans la carte
  2. le coudre à la carte déjà existante
  3. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Principe

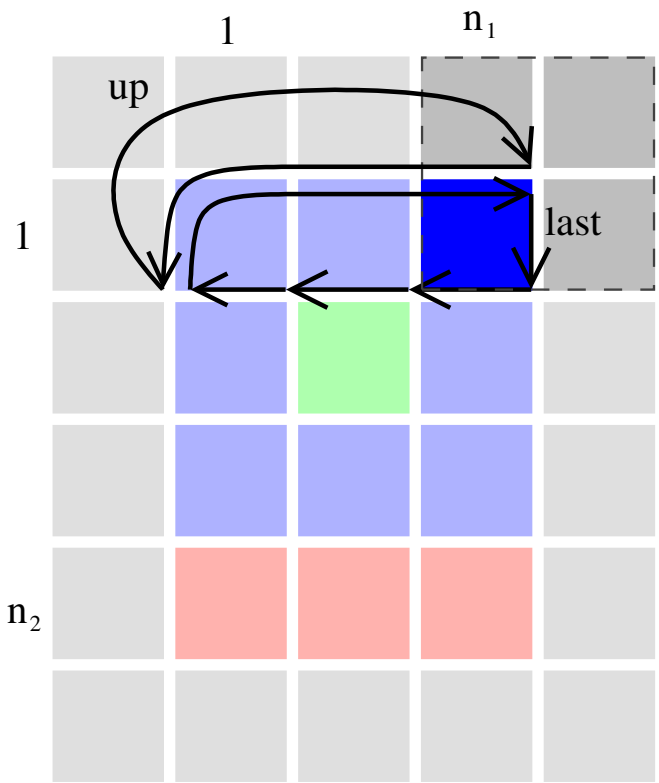
- Balayer l'image : parcours *scanline*
- Pour chaque pixel
  1. créer un carré dans la carte
  2. le coudre à la carte déjà existante
  3. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Principe

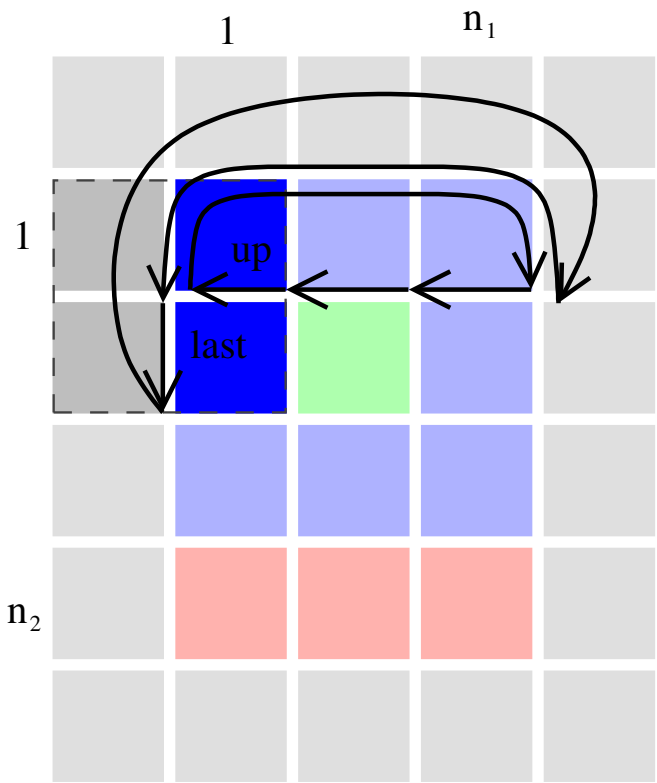
- Balayer l'image : parcours *scanline*
- Pour chaque pixel
  1. créer un carré dans la carte
  2. le coudre à la carte déjà existante
  3. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Principe

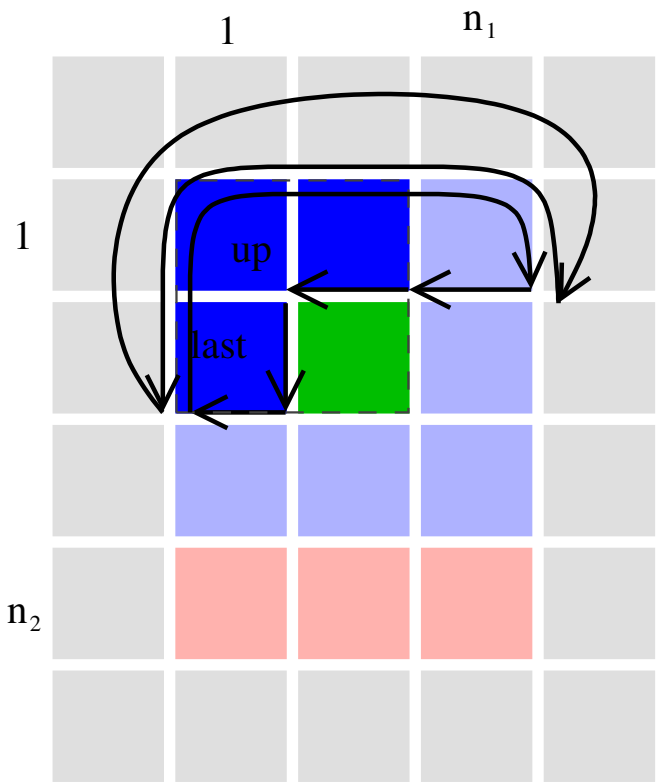
- Balayer l'image : parcours *scanline*
- Pour chaque pixel
  1. créer un carré dans la carte
  2. le coudre à la carte déjà existante
  3. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Principe

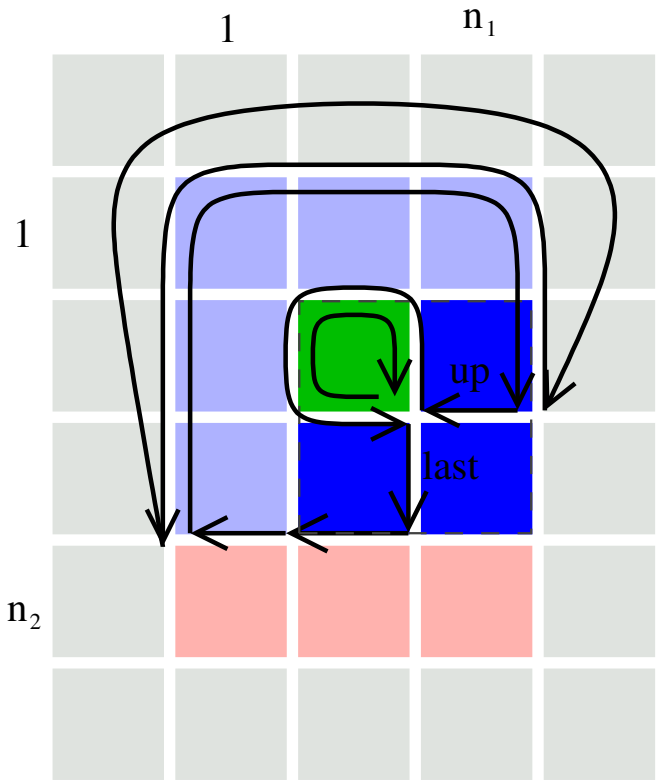
- Balayer l'image : parcours *scanline*
- Pour chaque pixel
  1. créer un carré dans la carte
  2. le coudre à la carte déjà existante
  3. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Principe

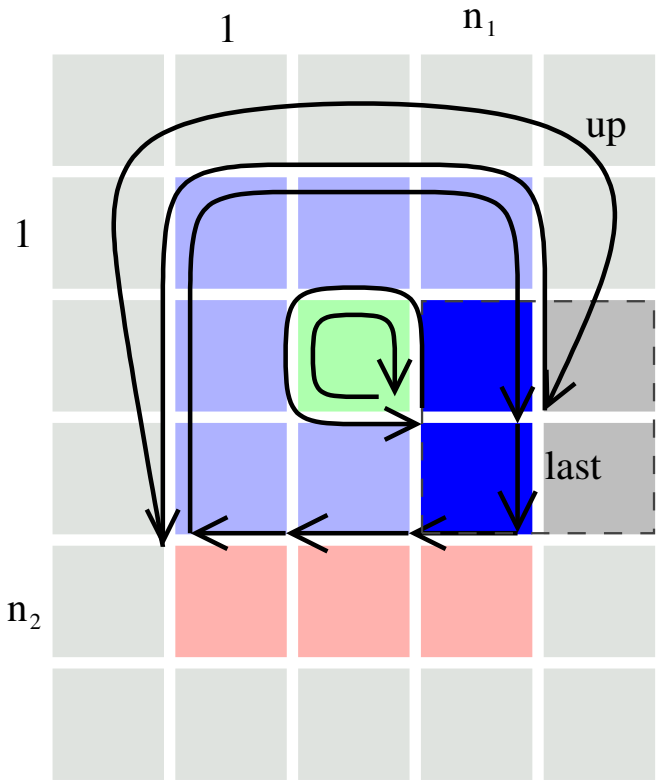
- Balayer l'image : parcours *scanline*
- Pour chaque pixel
  1. créer un carré dans la carte
  2. le coudre à la carte déjà existante
  3. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Principe

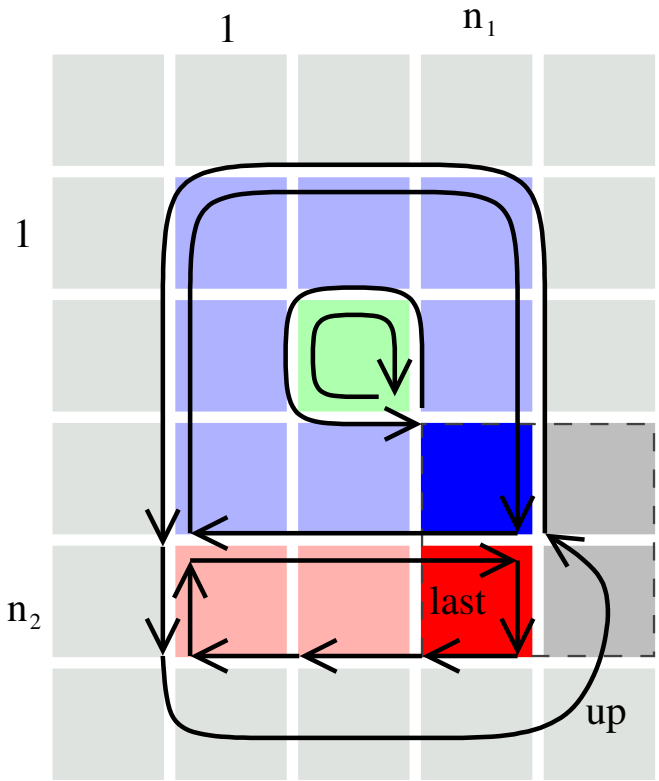
- Balayer l'image : parcours *scanline*
- Pour chaque pixel
  1. créer un carré dans la carte
  2. le coudre à la carte déjà existante
  3. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Principe

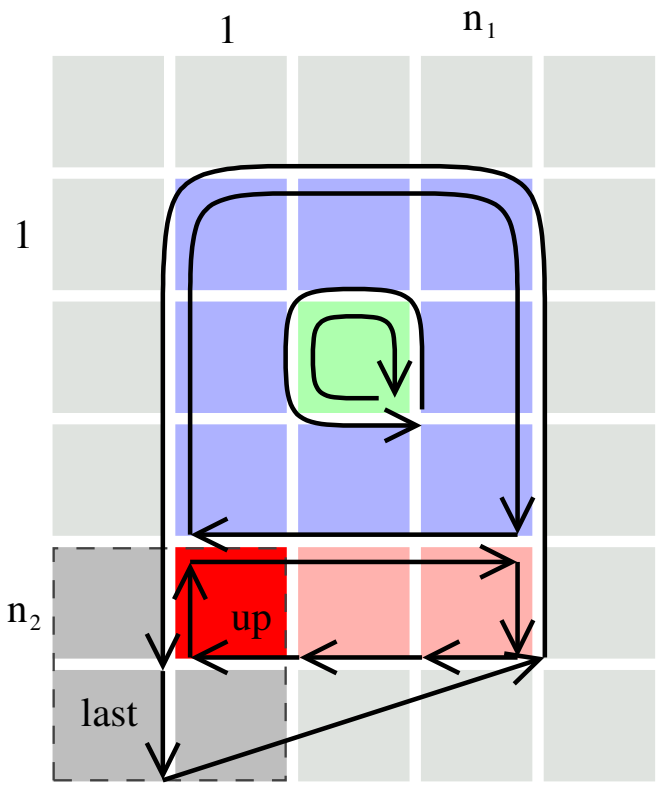
- Balayer l'image : parcours *scanline*
- Pour chaque pixel
  1. créer un carré dans la carte
  2. le coudre à la carte déjà existante
  3. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Principe

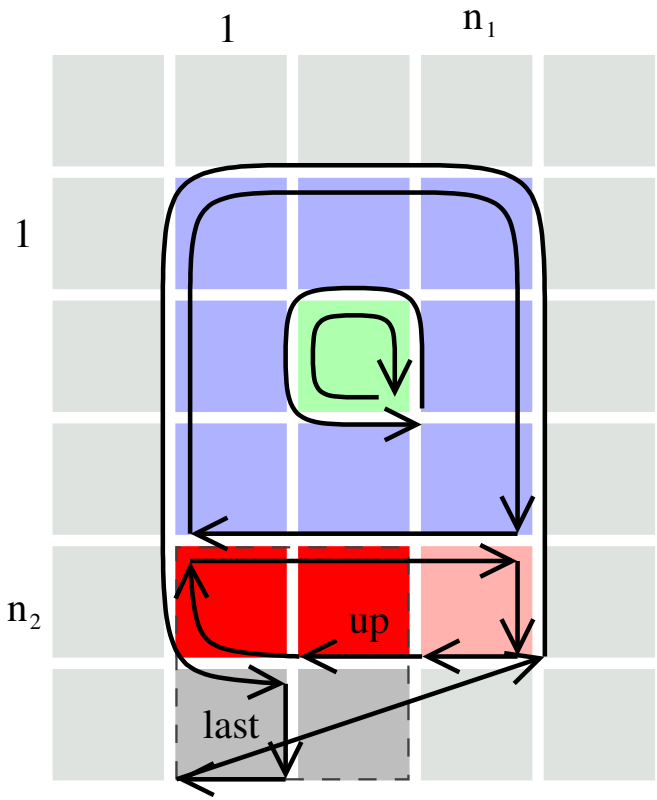
- Balayer l'image : parcours *scanline*
- Pour chaque pixel
  1. créer un carré dans la carte
  2. le coudre à la carte déjà existante
  3. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Principe

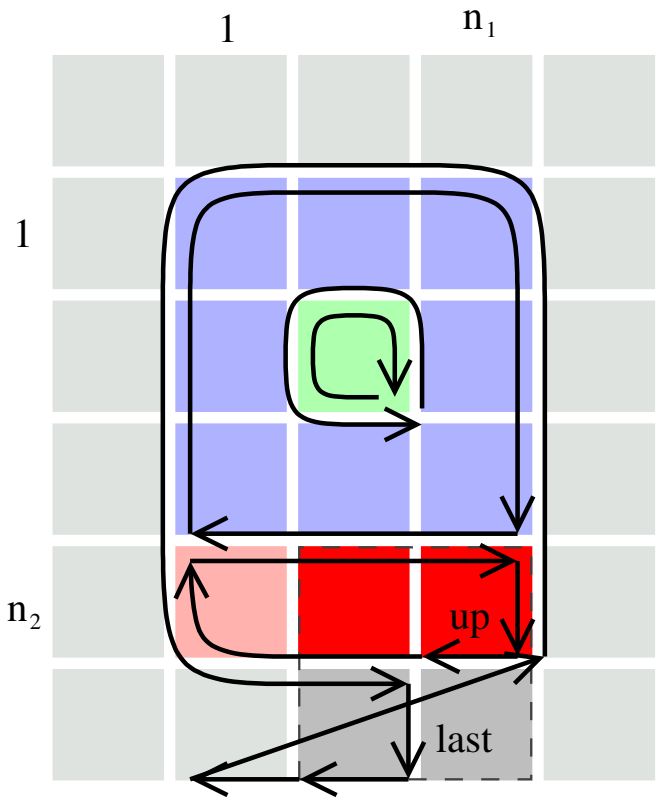
- Balayer l'image : parcours *scanline*
- Pour chaque pixel
  1. créer un carré dans la carte
  2. le coudre à la carte déjà existante
  3. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Principe

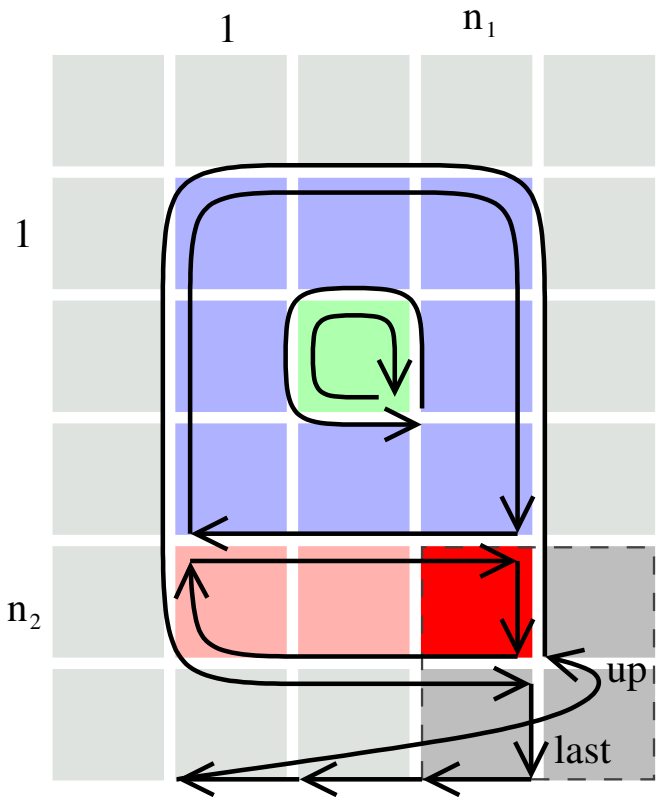
- Balayer l'image : parcours *scanline*
- Pour chaque pixel
  1. créer un carré dans la carte
  2. le coudre à la carte déjà existante
  3. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Principe

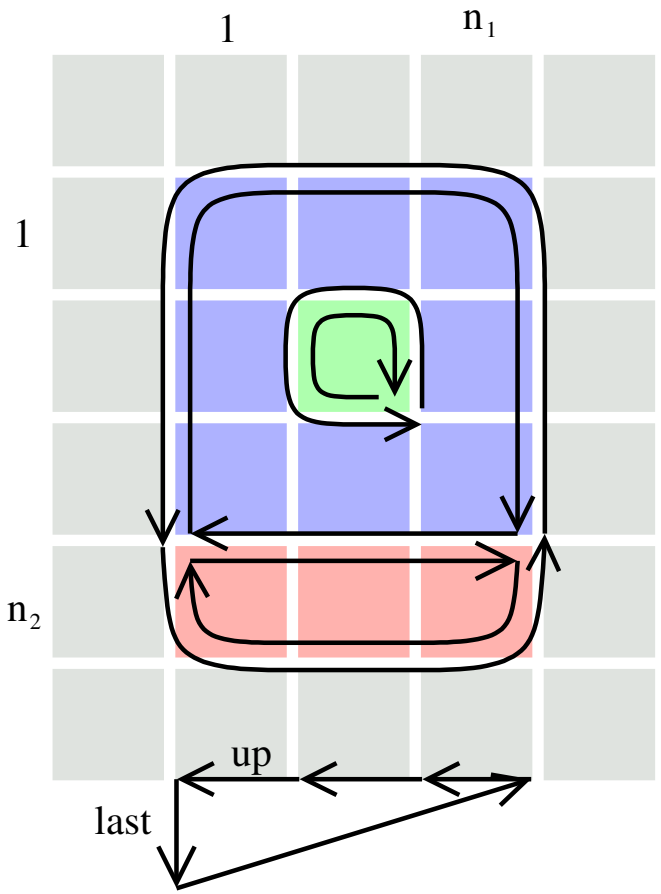
- Balayer l'image : parcours *scanline*
- Pour chaque pixel
  1. créer un carré dans la carte
  2. le coudre à la carte déjà existante
  3. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Principe

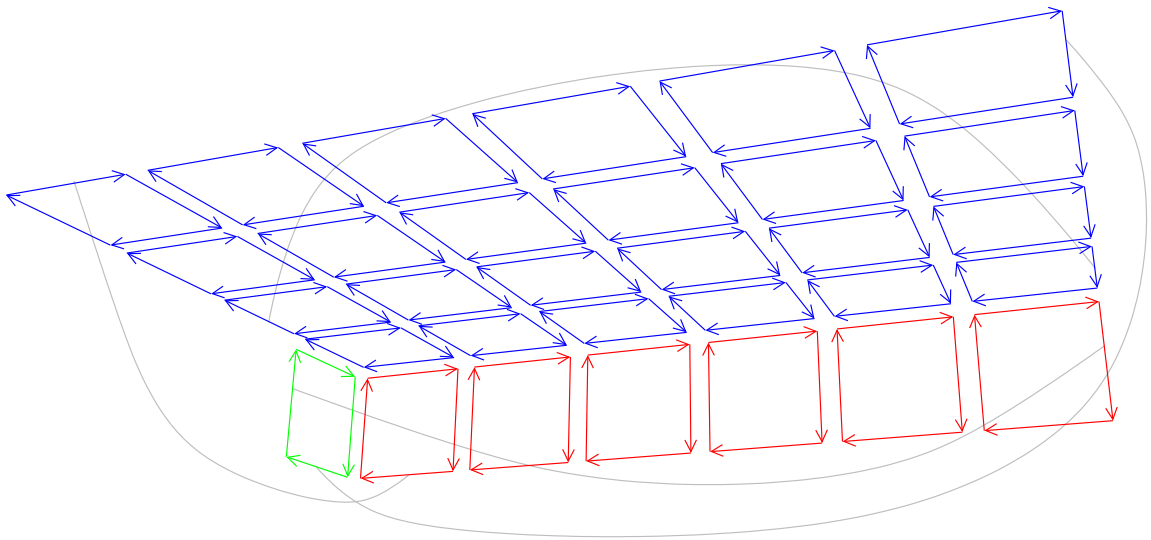
- Balayer l'image : parcours *scanline*
- Pour chaque pixel
  1. créer un carré dans la carte
  2. le coudre à la carte déjà existante
  3. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Même principe en 3D

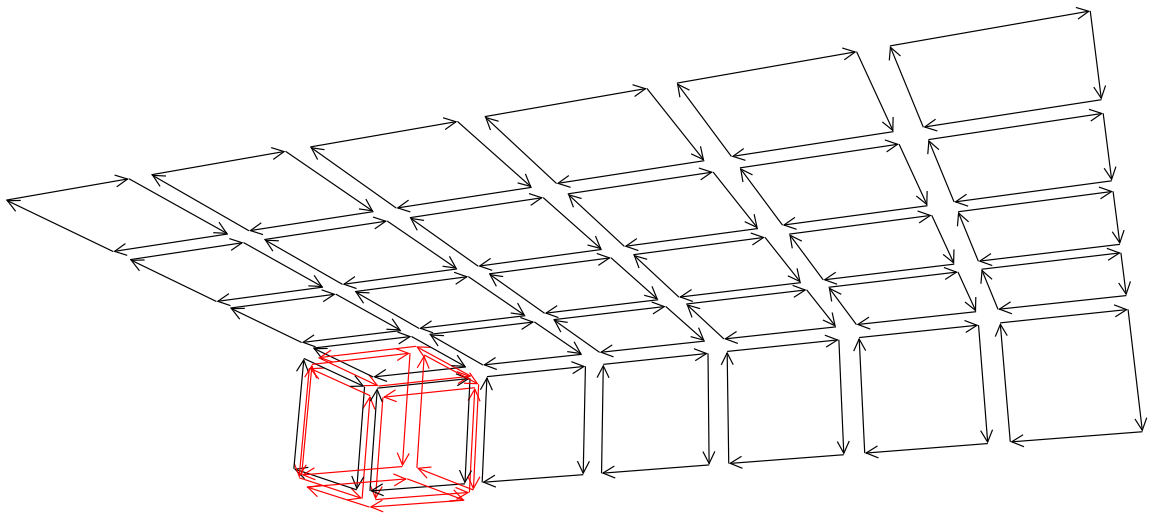
- Balayer l'image : parcours *scanline*
- Pour chaque voxel
  1. créer un cube et le coudre à la carte déjà existante
  2. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Même principe en 3D

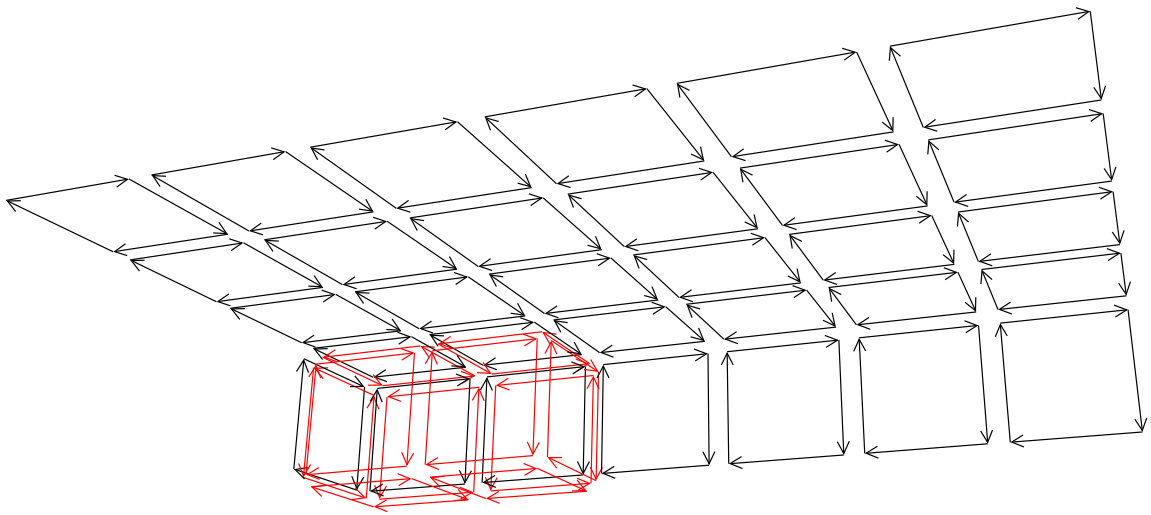
- Balayer l'image : parcours *scanline*
- Pour chaque voxel
  1. créer un cube et le coudre à la carte déjà existante
  2. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Même principe en 3D

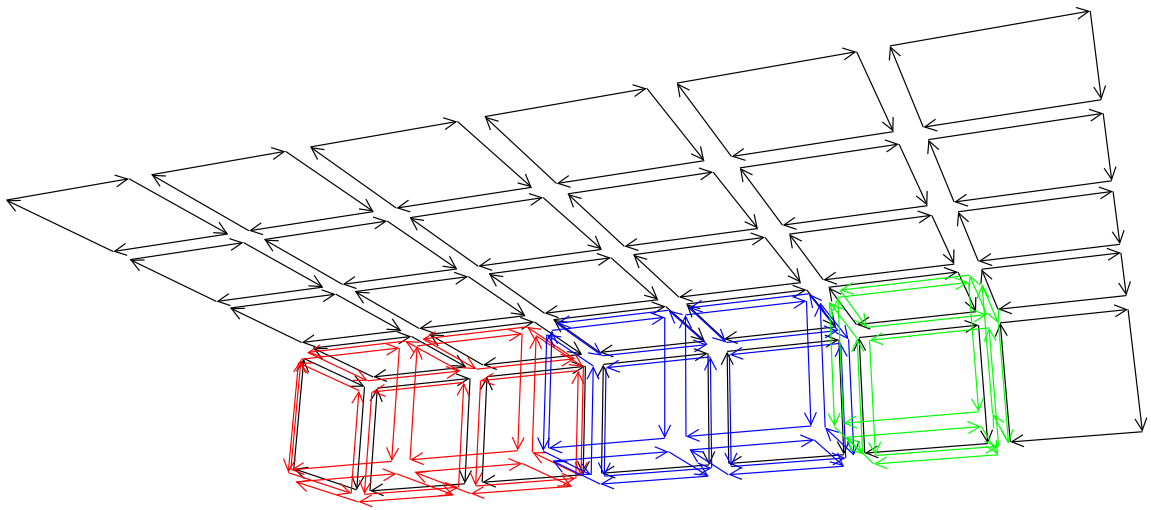
- Balayer l'image : parcours *scanline*
- Pour chaque voxel
  1. créer un cube et le coudre à la carte déjà existante
  2. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Même principe en 3D

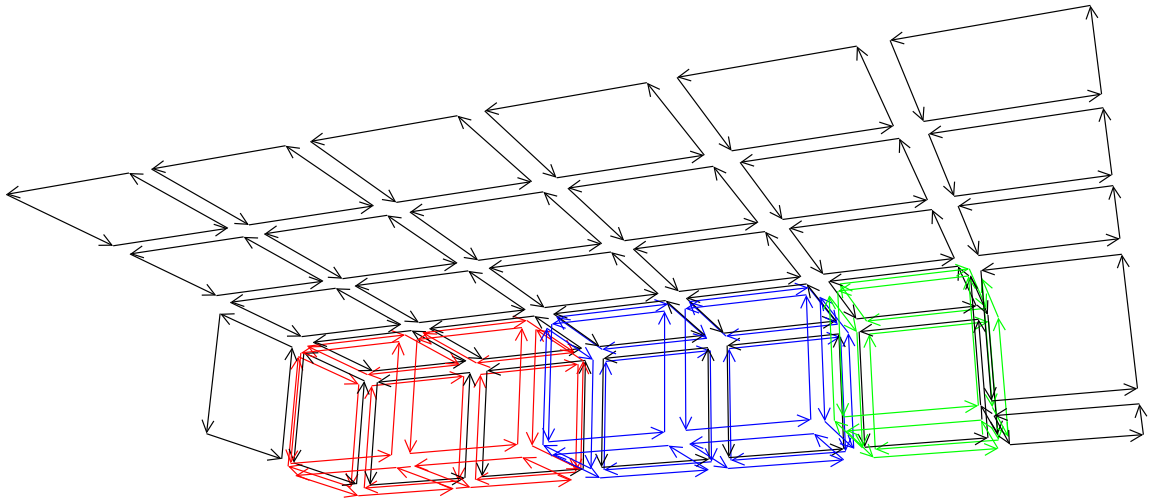
- Balayer l'image : parcours *scanline*
- Pour chaque voxel
  1. créer un cube et le coudre à la carte déjà existante
  2. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Même principe en 3D

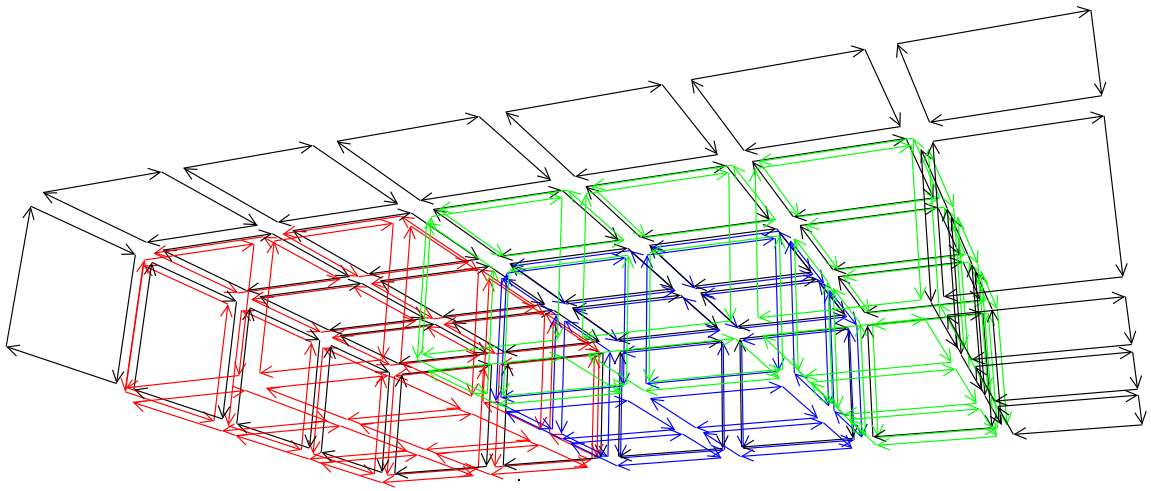
- Balayer l'image : parcours *scanline*
- Pour chaque voxel
  1. créer un cube et le coudre à la carte déjà existante
  2. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Même principe en 3D

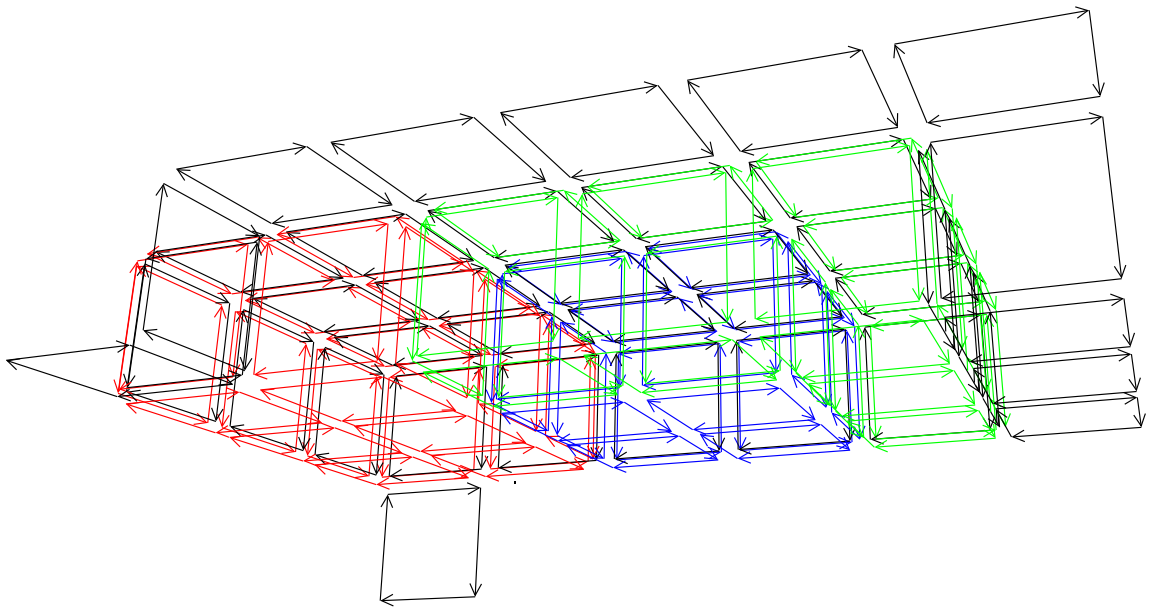
- Balayer l'image : parcours *scanline*
- Pour chaque voxel
  1. créer un cube et le coudre à la carte déjà existante
  2. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Même principe en 3D

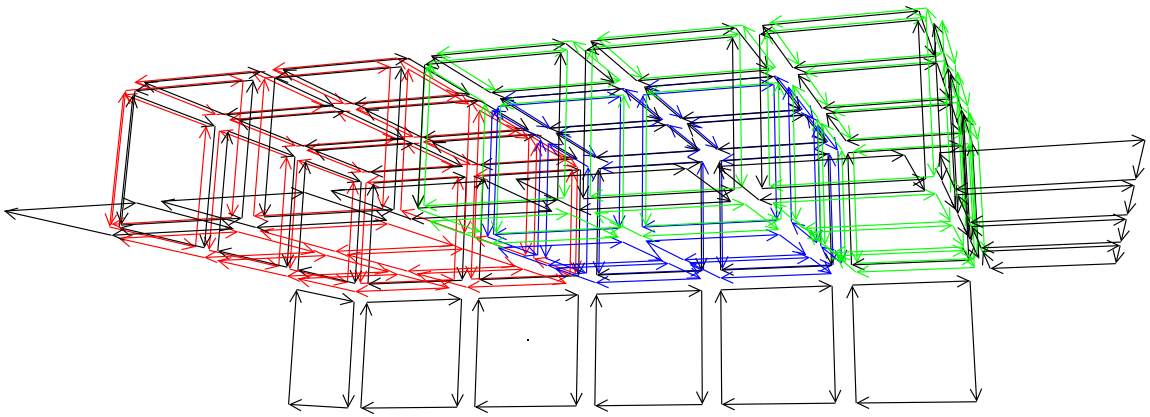
- Balayer l'image : parcours *scanline*
- Pour chaque voxel
  1. créer un cube et le coudre à la carte déjà existante
  2. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Même principe en 3D

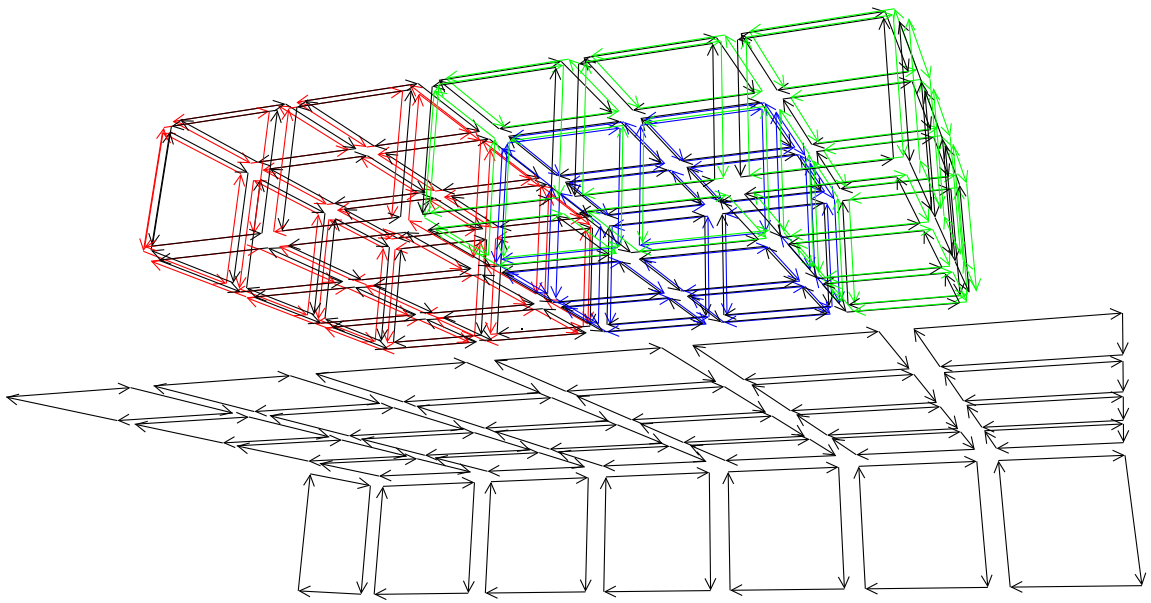
- Balayer l'image : parcours *scanline*
- Pour chaque voxel
  1. créer un cube et le coudre à la carte déjà existante
  2. chercher à supprimer les *cellules fermées*



Algorithme incrémental d'extraction

Même principe en 3D

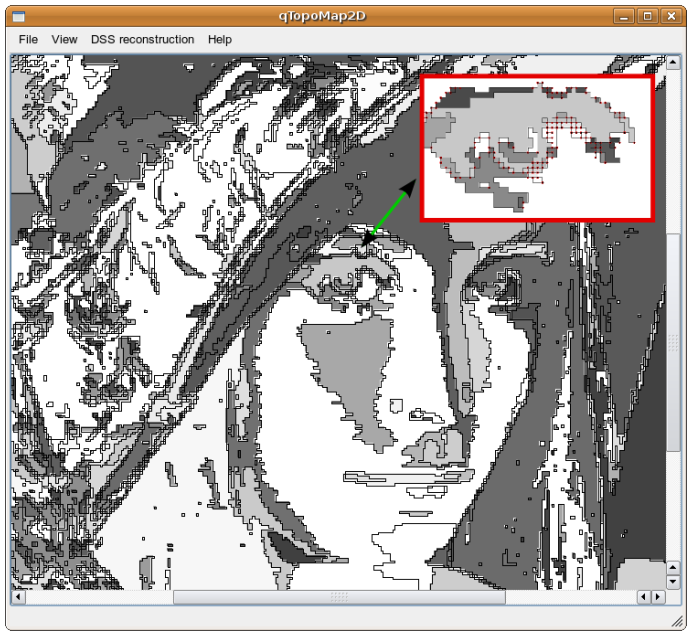
- Balayer l'image : parcours *scanline*
- Pour chaque voxel
  1. créer un cube et le coudre à la carte déjà existante
  2. chercher à supprimer les *cellules fermées*

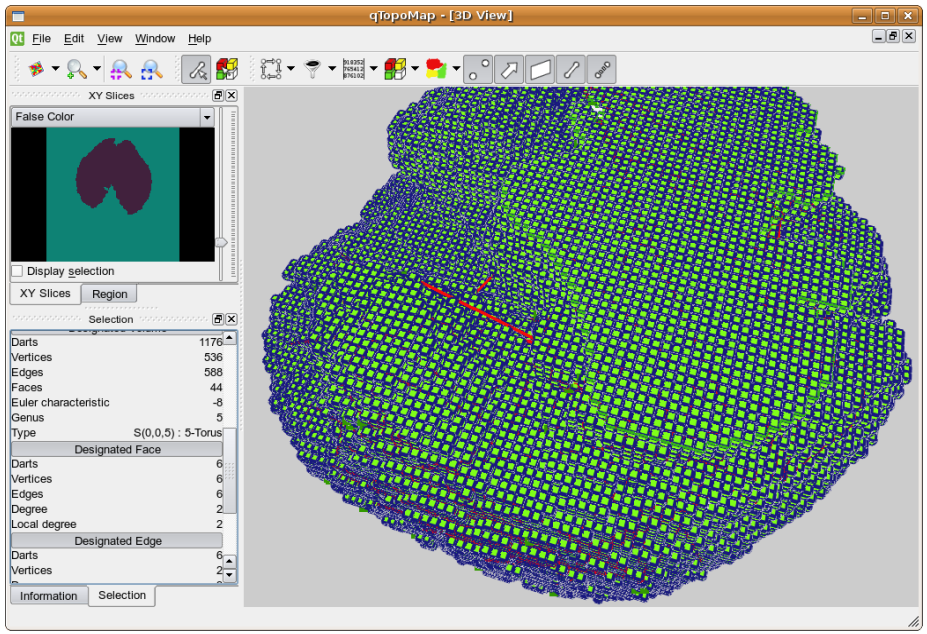


Segmentation d'images à base de cartes

Intérêt de la carte topologique

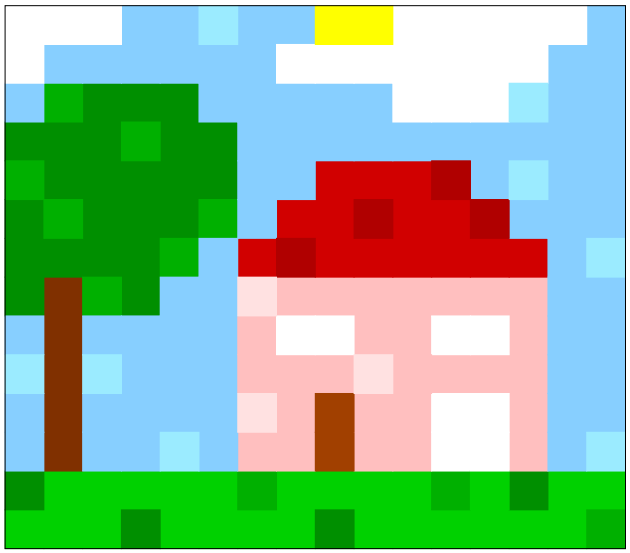
- Manipuler des régions
  - division, fusion, relations d'adjacences
- Calculer des caractéristiques topologique et géométrique
  - caractéristiques de haut niveaux (taille, forme, caractéristique d'Euler. . .)

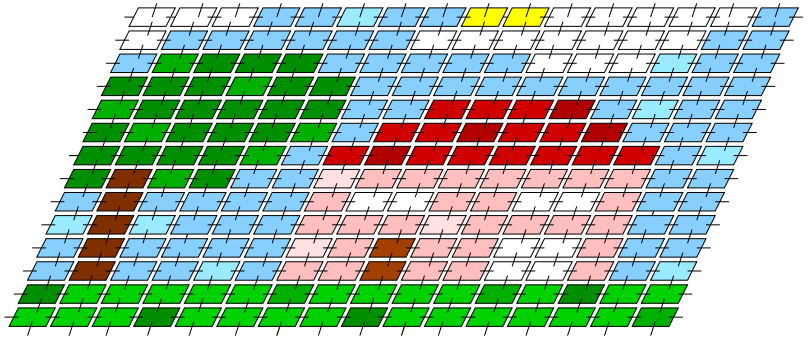




Segmentation bottom-up

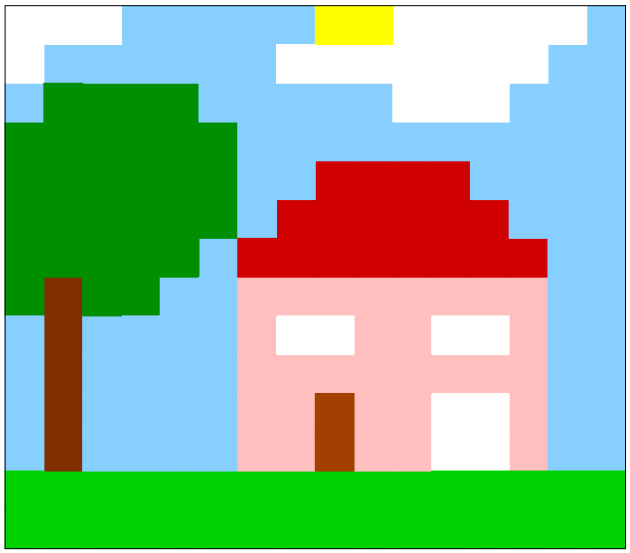
Carte de départ : image initiale

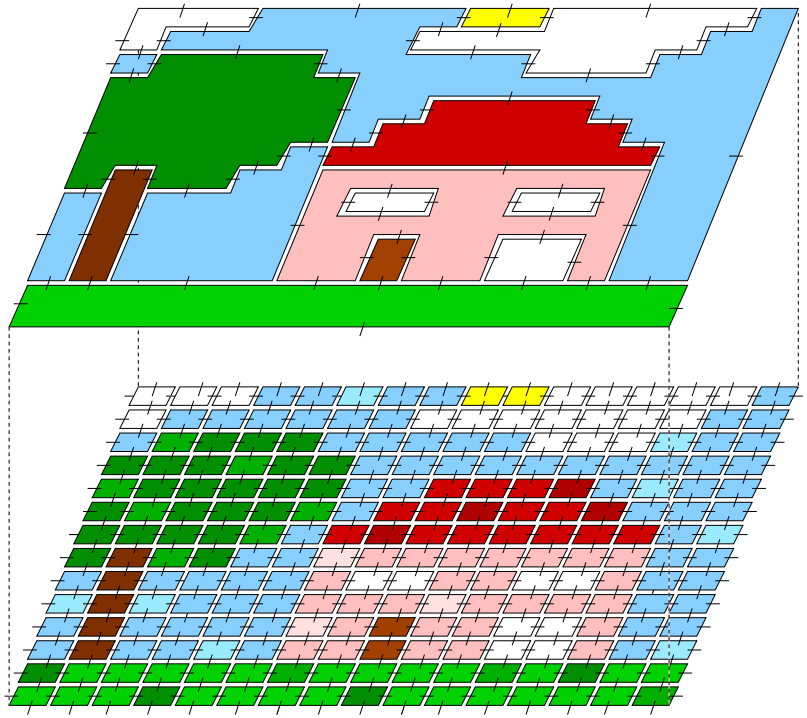




Segmentation bottom-up

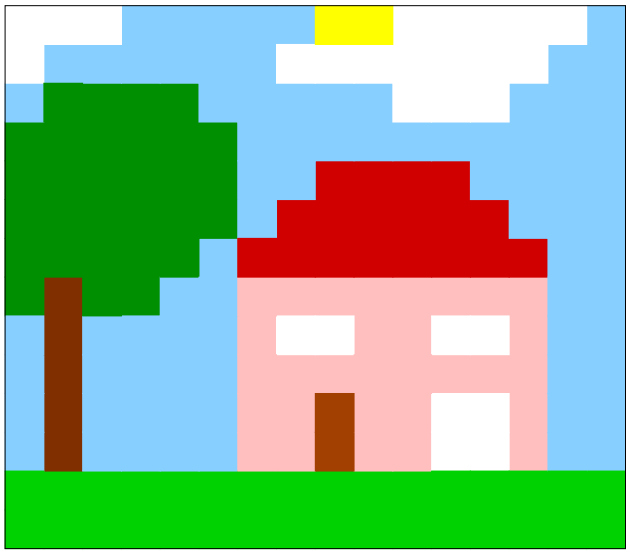
Supprimer les arêtes
$\Rightarrow$ fusion

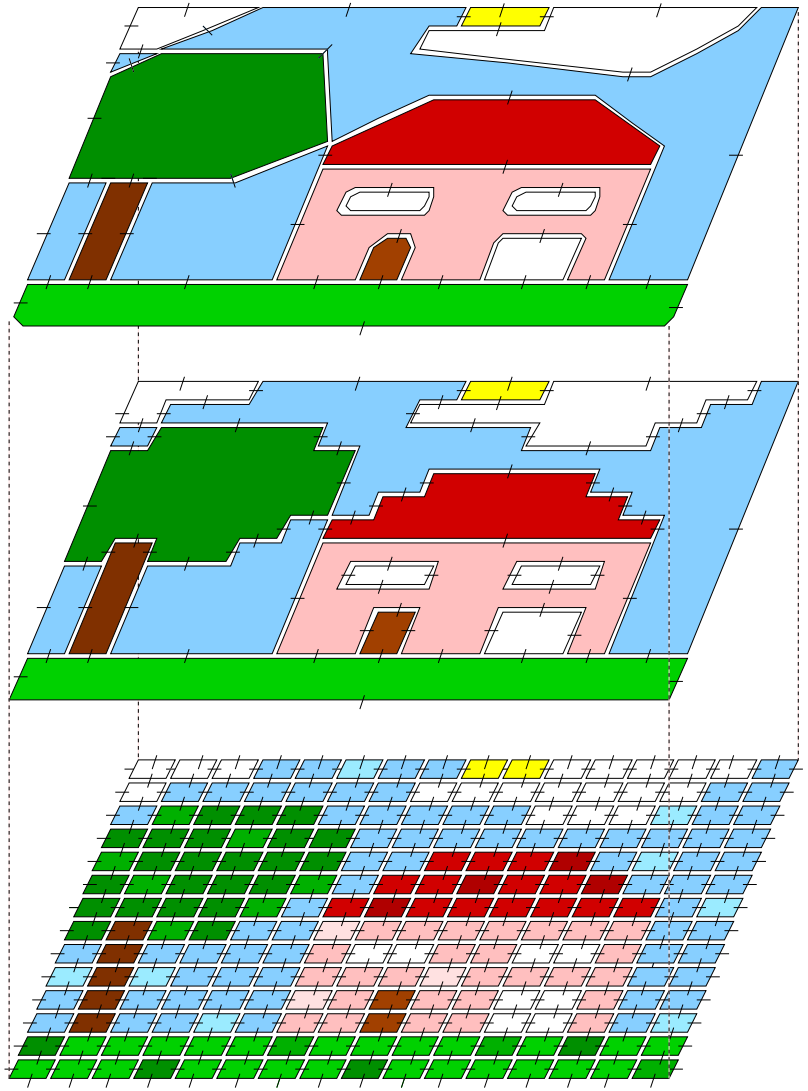




Segmentation bottom-up

Supprimer les sommets
$\Rightarrow$ simplification



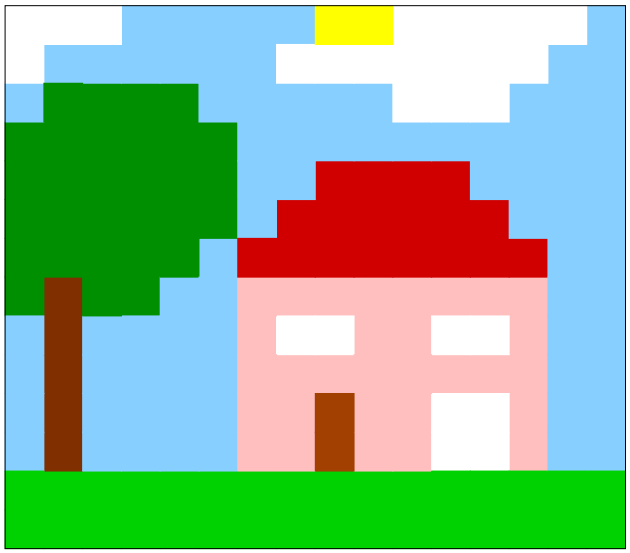


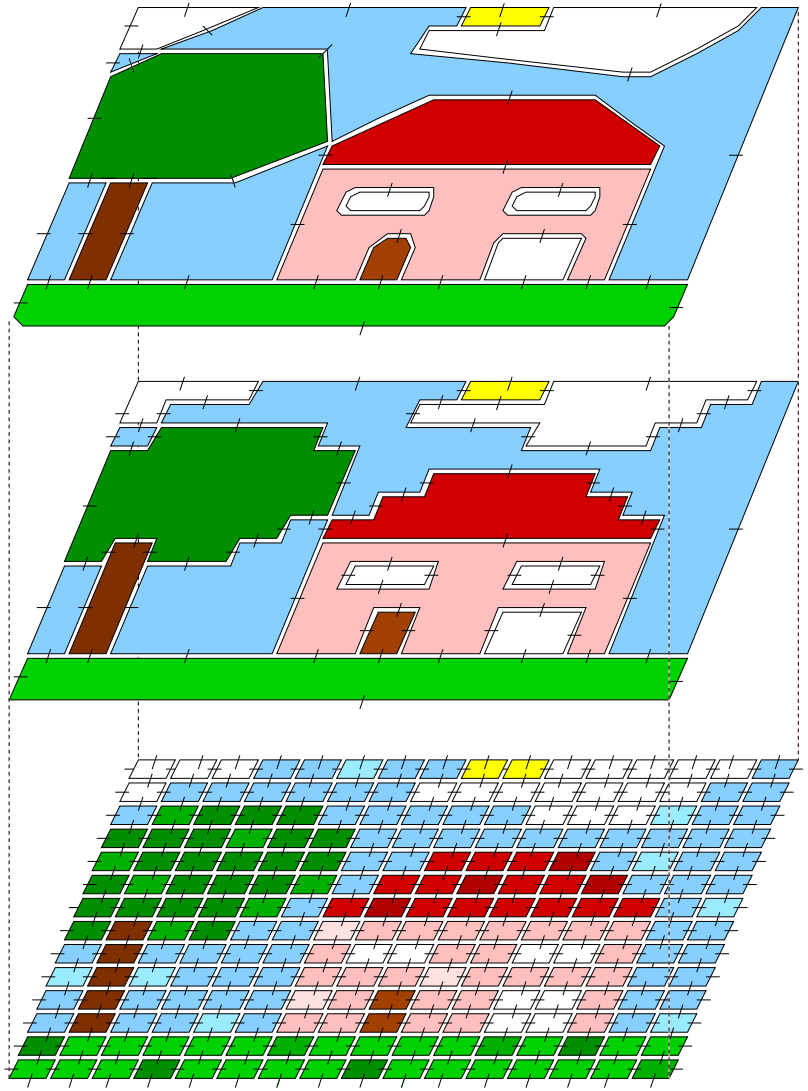
Segmentation bottom-up

Réitérer le processus
Critère de segmentation

⇒ marquage des arêtes

Couleurs, longueur des frontières, caractéristiques topologiques...





Algorithme de Segmentation 2D

Algorithme 1: Segmentation d'image 2D

Entrées : Une carte topologique 2D `C` décrivant une partition ;
Un `Oracle`.
Sorties : La partition segmentée selon l'oracle.
L une liste de de brins;
mettre dans L un brin par arête de `C`;
pour chaque *brin* b *de* L **faire**
  si `Oracle`$(b, \beta_2(b))$ **alors**
    fusionner $region(b)$ et $region(\beta_2(b))$;
    supprimer l'arête contenant b;
retourner C;

Intérêts :

- abstraction des pixels ; mixer différents critères dans l'oracle
  (ex. couleur moyenne des régions, longueur de l'arête, gradient de l'arête...)
- possibilité de segmenter n'importe qu'elle partition initiale
- ordre des fusions simple à modifier

Algorithme de Segmentation 2D

Algorithme 2: Segmentation d'image 2D

Entrées : Une carte topologique 2D `C` décrivant une partition ;
Un `Oracle`.
Sorties : La partition segmentée selon l'oracle.
L une liste de de brins;
mettre dans L un brin par arête de `C`;
trier L par gradient croissant;
pour chaque *brin* b *de* L **faire**
  si `Oracle`$(b, \beta_2(b))$ **alors**
    fusionner $region(b)$ et $region(\beta_2(b))$;
    supprimer l'arête contenant b;
retourner C;

Intérêts :

- abstraction des pixels ; mixer différents critères dans l'oracle
  (ex. couleur moyenne des régions, longueur de l'arête, gradient de l'arête...)
- possibilité de segmenter n'importe qu'elle partition initiale
- ordre des fusions simple à modifier

Critères « classiques »

Contraintes

Réflexif et Symétrique

Quelques critères

- différence absolue des couleurs moyennes des 2 régions
- intervalle [couleur min, couleur max] de l'union des 2 régions
- constraste externe et interne

  $int(r)$=gradient max de r ; $ext(r_1, r_2)$=gradient max entre r_1 et r_2 ;
  `Oracle` : $ext(r_1, r_2) \leq min(int(r_1) + \tau(r_1), int(r_2) + \tau(r_2))$

  ($\tau(r)$ fonction positive, par exemple $k/|r|$)

Des critères « mixtes »

Quelques critères

- taille min des 2 régions
- intervalle [couleur min, couleur max] de l'union des 2 régions, avec le seuil pondéré par la longueur de la zone de contact entre les 2 régions
- ...

Exemples

Critère : taille de l'intervalle [couleur min, couleur max]
Seuil : 100

Image originale.
Nb régions : 119514.

Sans tri.
Nb régions : 581.
Temps : 0.29s.

Avec tri.
Nb régions : 157.
Temps : 0.56s.

Exemples

Critère : taille de l'intervalle [couleur min, couleur max]
Seuil : 100

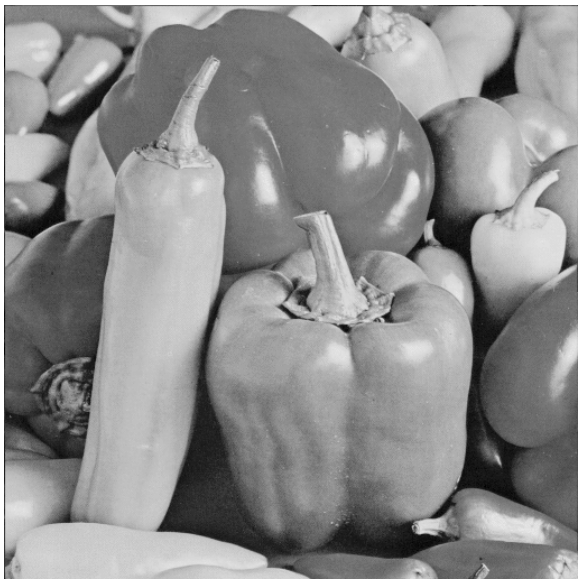
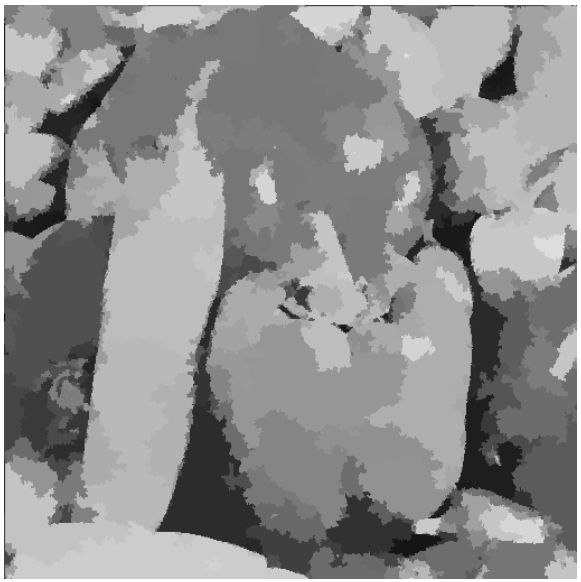


Image originale.
Nb régions : 139101.



Sans tri.
Nb régions : 780.
Temps : 0.35s.

Avec tri.
Nb régions : 394.
Temps : 0.66s.

Exemples

Critère : taille min des régions. Seuil : 6

Nb régions : 16040.

Nb régions : 1339.
Temps : 0.08s.

Exemples

Critère : intervalle de couleurs.

Sans pondération. Seuil : 100.
Nb régions : 61.
Temps : 0.58s.

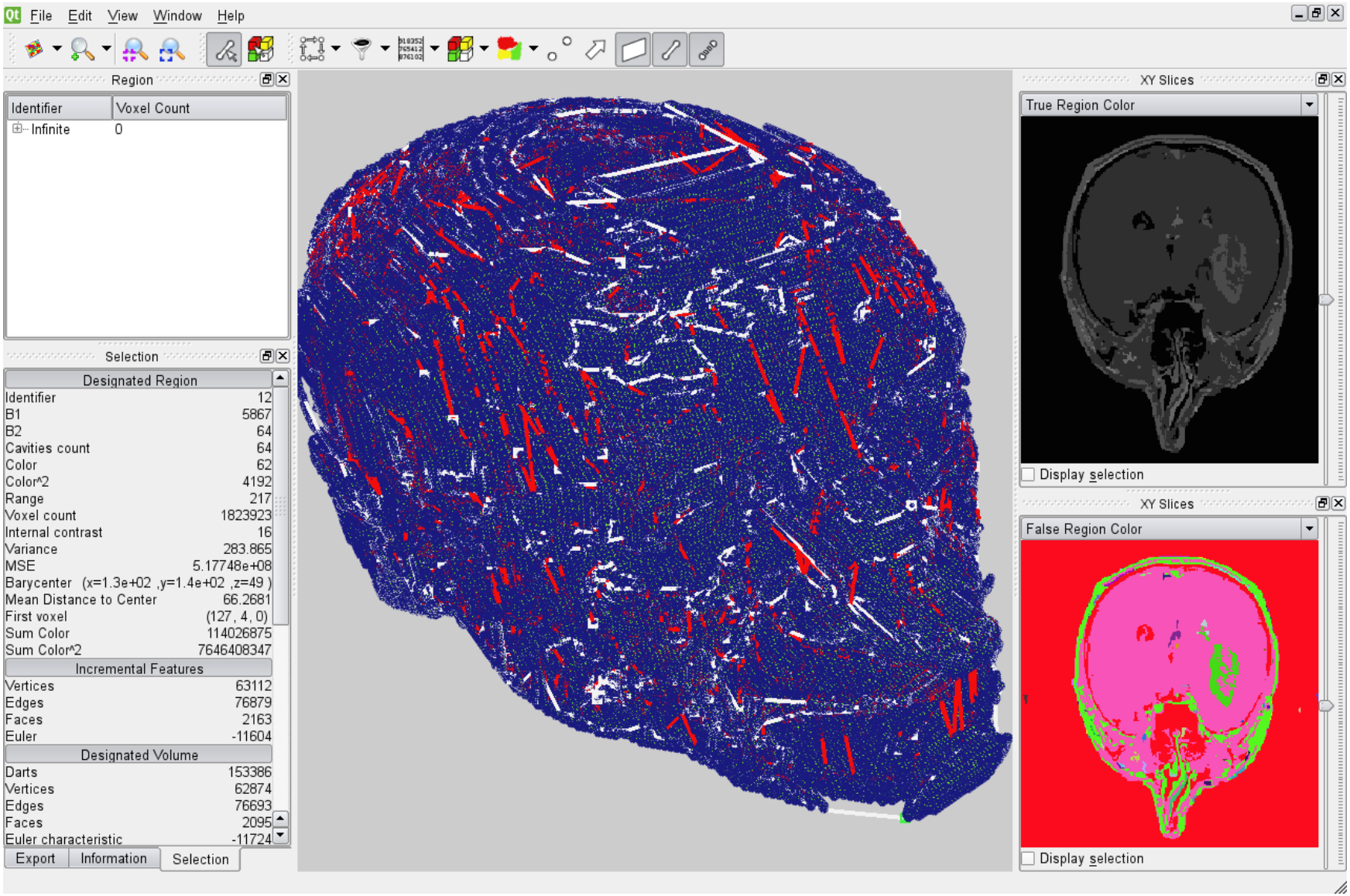
Avec pondération. Seuil : 680.
Nb régions : 61.
Temps : 1.96s.

Algorithme de Segmentation 3D

Algorithme 3: Segmentation d'image 3D

Entrées : Une carte topologique 3D `C` décrivant une partition ;
Un `Oracle`.
Sorties : La partition segmentée selon l'oracle.
L une liste de de brins;
mettre dans L un brin par face de `C`;
trier L par gradient croissant;
pour chaque *brin* b *de* L **faire**
  si `Oracle`$(b, \beta_3(b))$ **alors**
    fusionner $region(b)$ et $region(\beta_3(b))$;
    supprimer la face contenant b;
retourner C;

Exemples


File Edit View Window Help
Region
Identifier
Voxel Count
Infinite
0
Selection
Designated Region
Identifier 12
B1 5867
B2 64
Cavities count 64
Color 62
Color^2 4192
Range 217
Voxel count 1823923
Internal contrast 16
Variance 283.865
MSE 5.17748e+08
Barycenter (x=1.3e+02 ,y=1.4e+02 ,z=49)
Mean Distance to Center 66.2681
First voxel (127, 4, 0)
Sum Color 114026875
Sum Color^2 7646408347
Incremental Features
Vertices 63112
Edges 76879
Faces 2163
Euler -11604
Designated Volume
Darts 153386
Vertices 62874
Edges 76693
Faces 2095
Euler characteristic -11724
Export
Information
Selection
XY Slices
True Region Color
Display selection
XY Slices
False Region Color
Display selection

Calcul de la caractéristique d'Euler-Poincaré

1. Utilisation de la somme alternée des nombres de cellules dans le bord de R

Definition

Caractéristique d'Euler-Poincaré χ' du bord d'une région R :

$$\chi'(R) = \sum_{i=0}^{n-1} (-1)^i \times k_i(R)$$

2. On ajoute des cellules implicites :

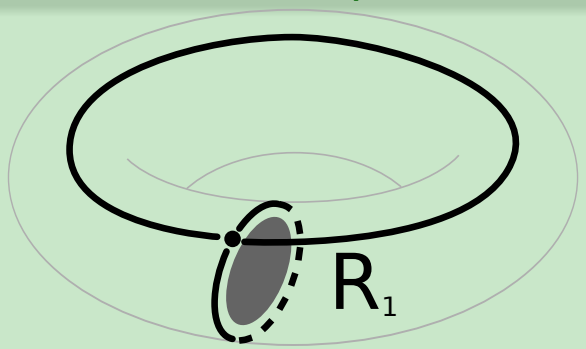
Cellules implicites

cellules qui doivent être ajoutées à R pour que chaque cellule de R devienne homeomorphe à une boule

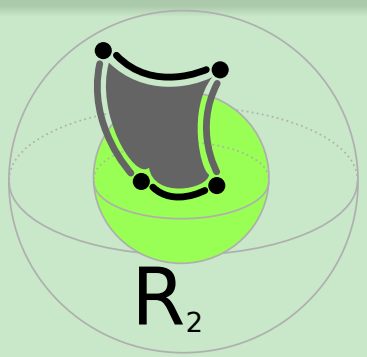
Cellules implicites

En 3D, seulement deux types de cellules implicites
(pour les tunnels et les cavitiés)

Avec les cellules implicites



$k_0 = 1, k_1 = 2, k_2 = 1 + 1, k_3 = 1$
$\Rightarrow \chi'(R_1) = 0$ et $\chi(R_1) = 0$



$k_0 = 4, k_1 = 2 + 2, k_2 = 2 + 1, k_3 = 1$
$\Rightarrow \chi'(R_2) = 4$ and $\chi(R_2) = 2$

Compter les cellules implicites $\Rightarrow$
la « bonne » caractéristique d'Euler-Poincaré

Cellules implicites

En 3D, seulement deux types de cellules implicites
(pour les tunnels et les cavitiés)

Compter les cellules implicites $\Rightarrow$
la $\ll$ bonne $\gg$ caractéristique d'Euler-Poincaré

Encore mieux

pas besoin de compter les cellules implicites grâce à la relation suivante :

$$\chi(R) = \frac{\chi'(R)}{2}$$

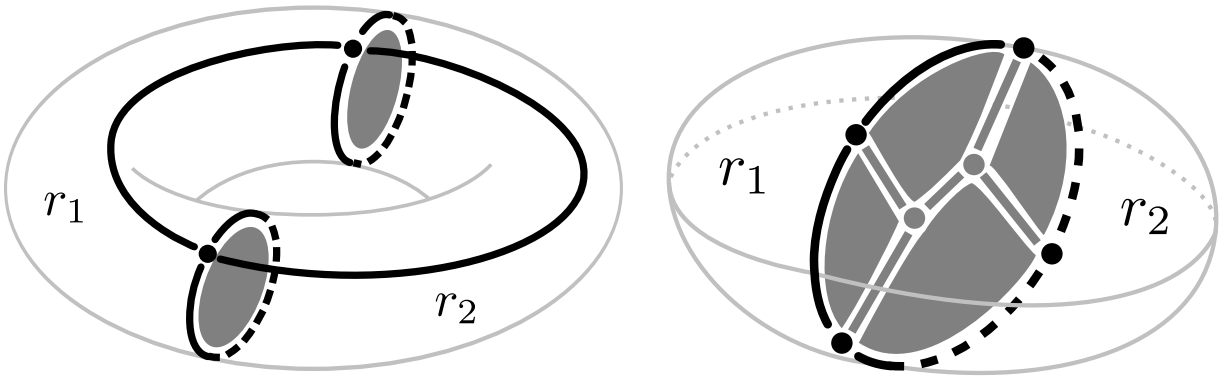
Comment calculer la caractéristique d'Euler-Poincaré

Mettre à jour localement les nombres k_i au cours de l'opération de suppression

$\Rightarrow$ Les nombres k_i peuvent être calculés au cours de la construction de la carte topologique

Principe de l'algorithme

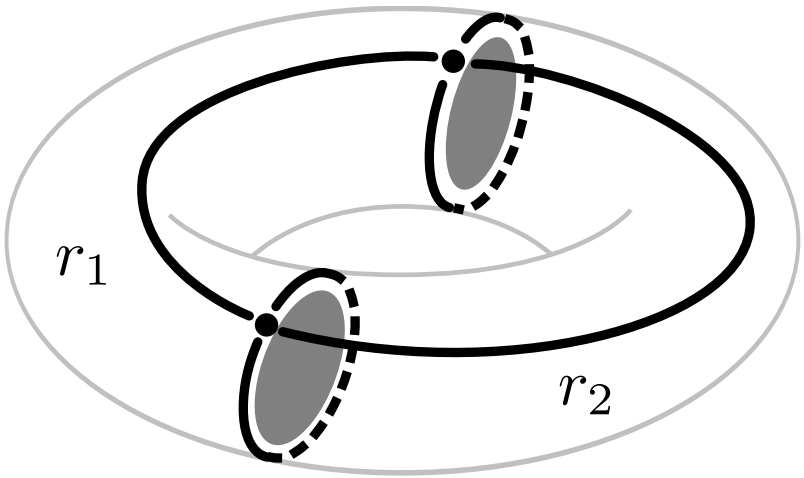
- Initialisation : création d'un pixel/voxel $\Rightarrow$ initialiser les k_i
- i-suppression $\Rightarrow$ mise à jour locale des k_i en utilisant r_1, r_2 et $r_1 \cap r_2$



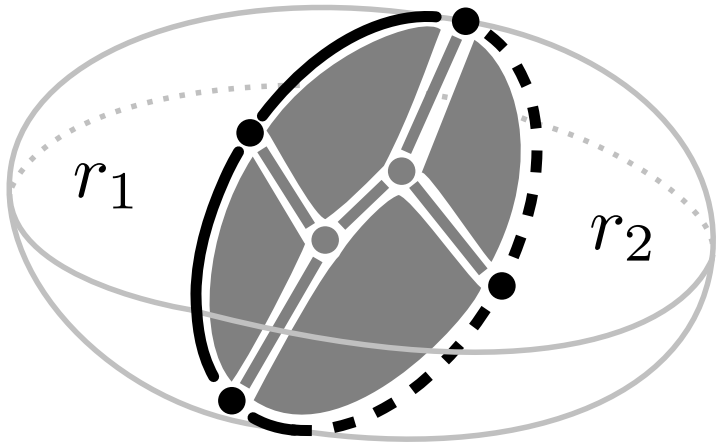
En gris, les cellules internes à $r_1 \cap r_2$

Mise à jour locale au cours de la i-suppression

$$\chi'(r_1 \cup r_2) = \chi'(r_1) + \chi'(r_2) - 2\chi'(r_1 \cap r_2)$$



$$\chi'(r_1) = 2 - 3 + 3 = 2\,;$$
$$\chi'(r_2) = 2 - 3 + 3 = 2$$
$$\chi'(r_1 \cap r_2) = 0 - 0 + 2 = 2$$
$$\Rightarrow \chi'(r_1 \cup r_2) = 0$$



$$\chi'(r_1) = 6 - 9 + 5 = 2\,;$$
$$\chi'(r_2) = 6 - 9 + 5 = 2$$
$$\chi'(r_1 \cap r_2) = 2 - 5 + 4 = 1$$
$$\Rightarrow \chi'(r_1 \cup r_2) = 2$$

Comment calculer les nombres de Betti

Liens entre les nombres de Betti et la caractéristique d'Euler-Poincaré

$$\chi(C) = \sum_{i=0}(-1)^i \times \mathfrak{b}_i(C)$$

Les nombres de Betti d'une région R en 3D

- $\mathfrak{b}_0(R) = 1$;
- $\mathfrak{b}_2(R) = |\text{surfaces}(R)| - 1$.
- $\mathfrak{b}_1(R) = \mathfrak{b}_0(R) + \mathfrak{b}_2(R) - \chi'(R)/2$;

$\Rightarrow$ On peut utiliser le même algorithme que pour calculer la caractéristique d'Euler-Poincaré :

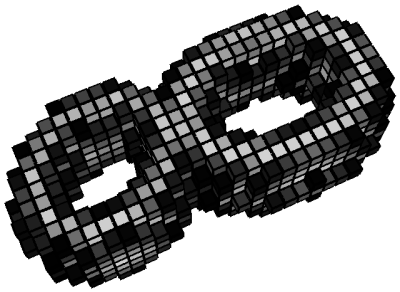
- initialiser les pixels/voxels $\Rightarrow$ valeurs constantes pour $\mathfrak{b}_i$
- i-suppression $\Rightarrow$ mise à jour locale de $\text{surfaces}(R)$ et $\chi'(R)$

Segmentation guidée par la topologie

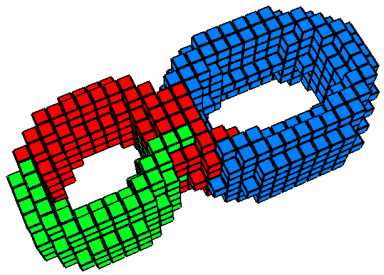
Idée : segmentation supervisée où l'on connait la topologie d'un objet (ex. cerveau $\Rightarrow$ sphère avec éventuellement des cavités)

Contrôle des nombres de Betti durant la segmentation

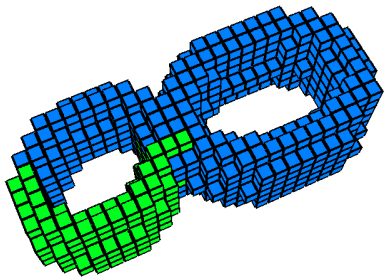
- Autoriser ou non les changements
- Favoriser ou pénaliser certaines fusions



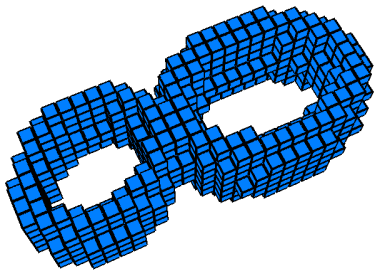
Image



$b_1 \to 0$

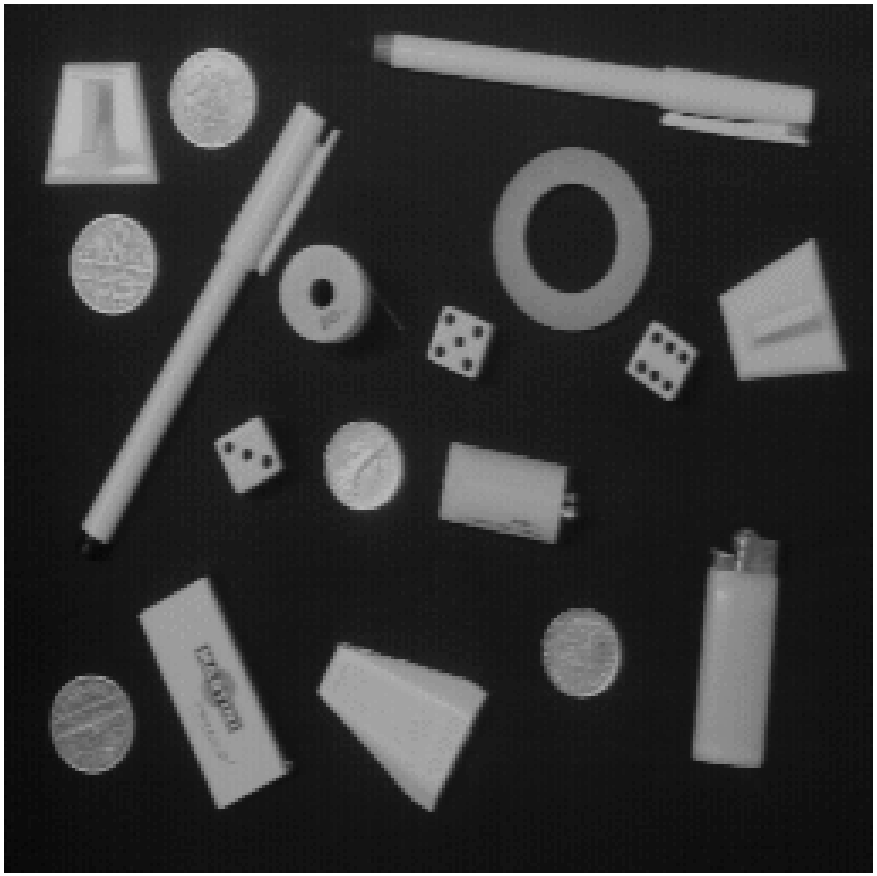


$b_1 \to 1$

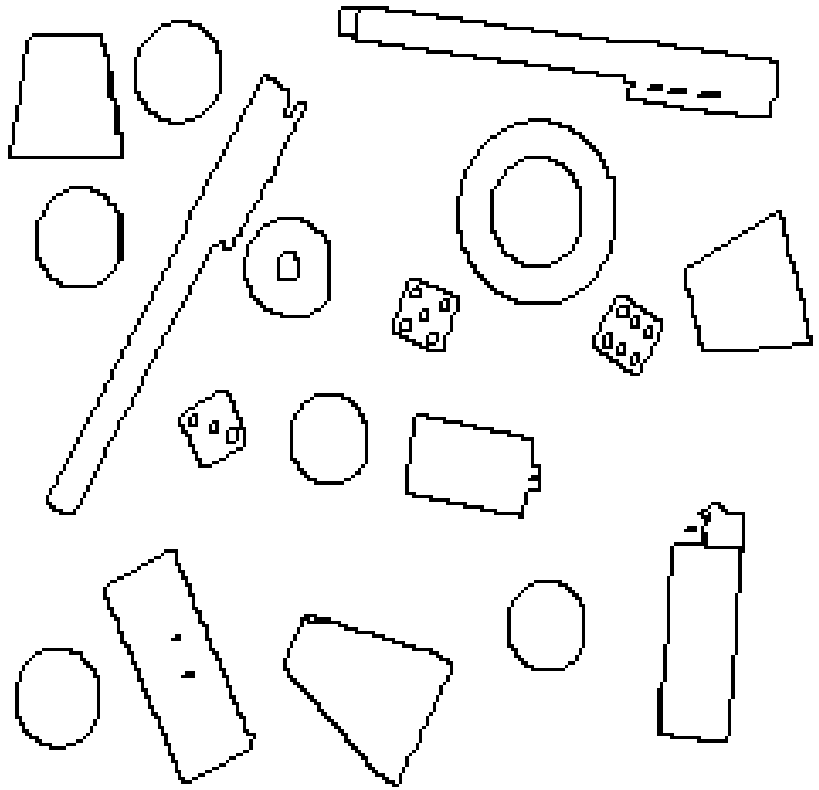


sans contrainte

Résultats en 2D

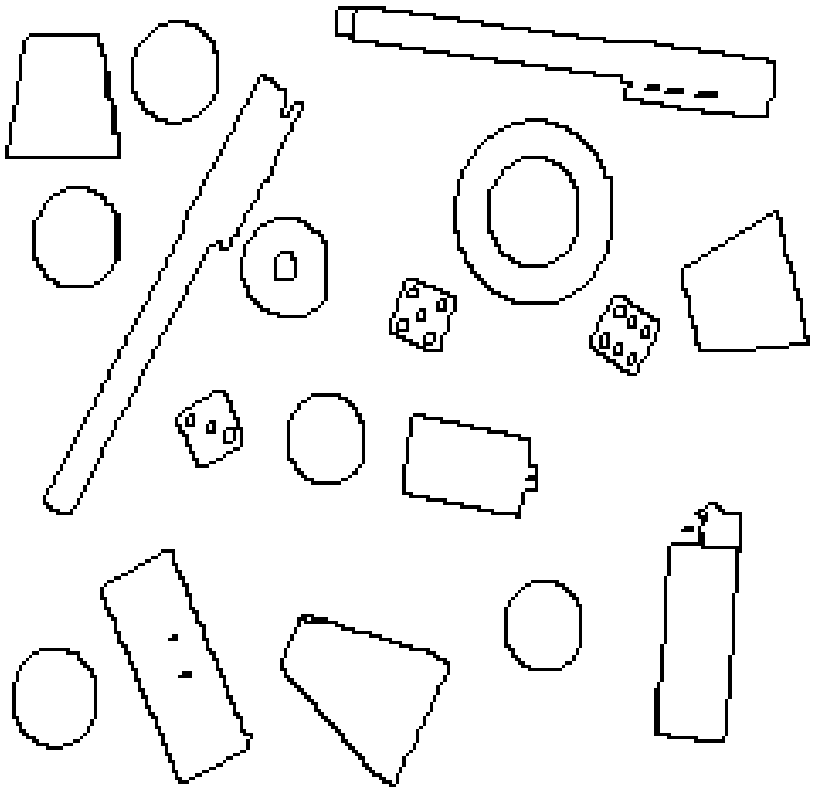


Image

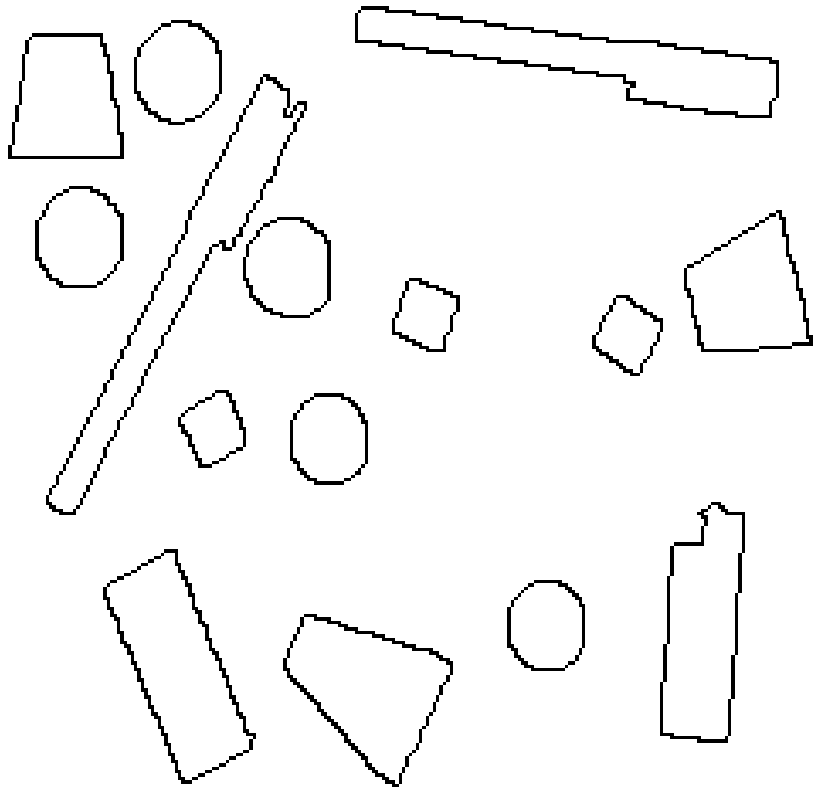


Résultat avec le critère intervalle (seuil=100), sans critère topologique

Résultats en 2D

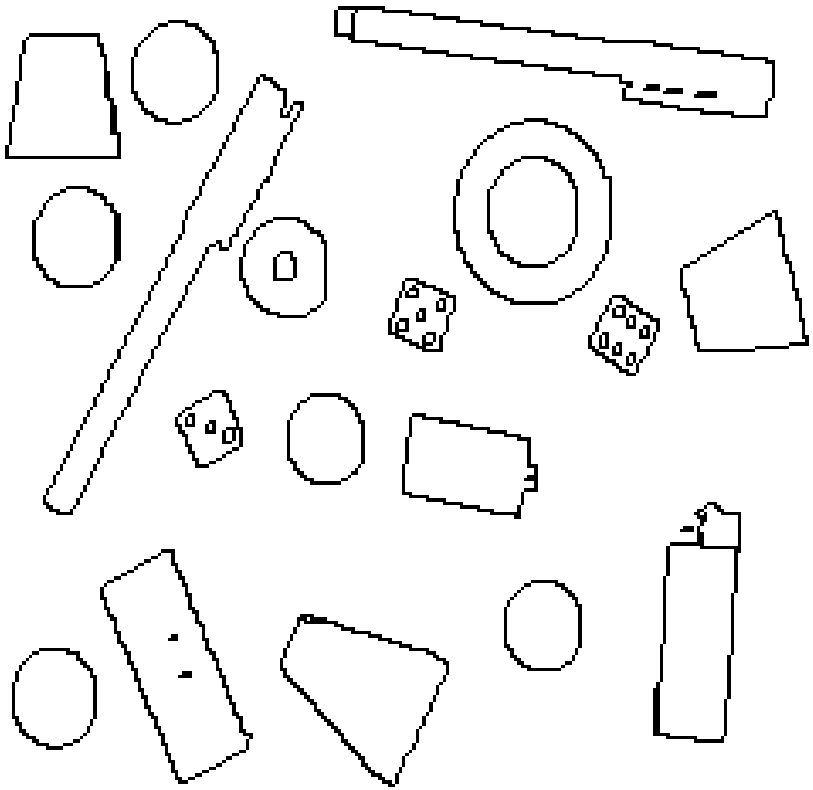


Résultat avec le critère intervalle (seuil=100), sans critère topologique

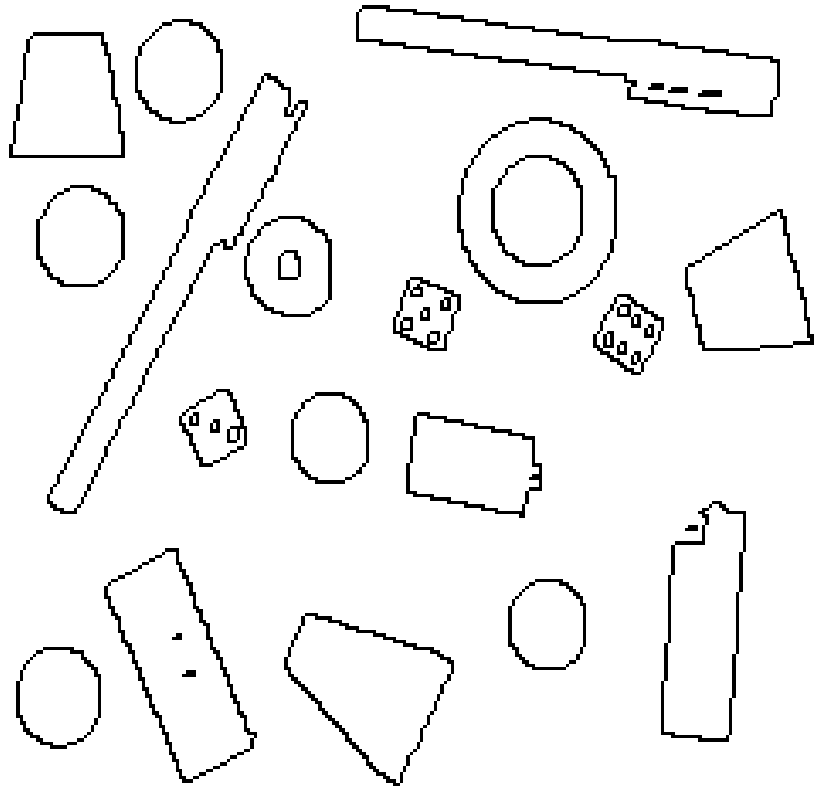


Résultat obtenu à partir de la partition précédente, avec seuil=150

Résultats en 2D



Résultat avec le critère intervalle (seuil=100), sans critère topologique



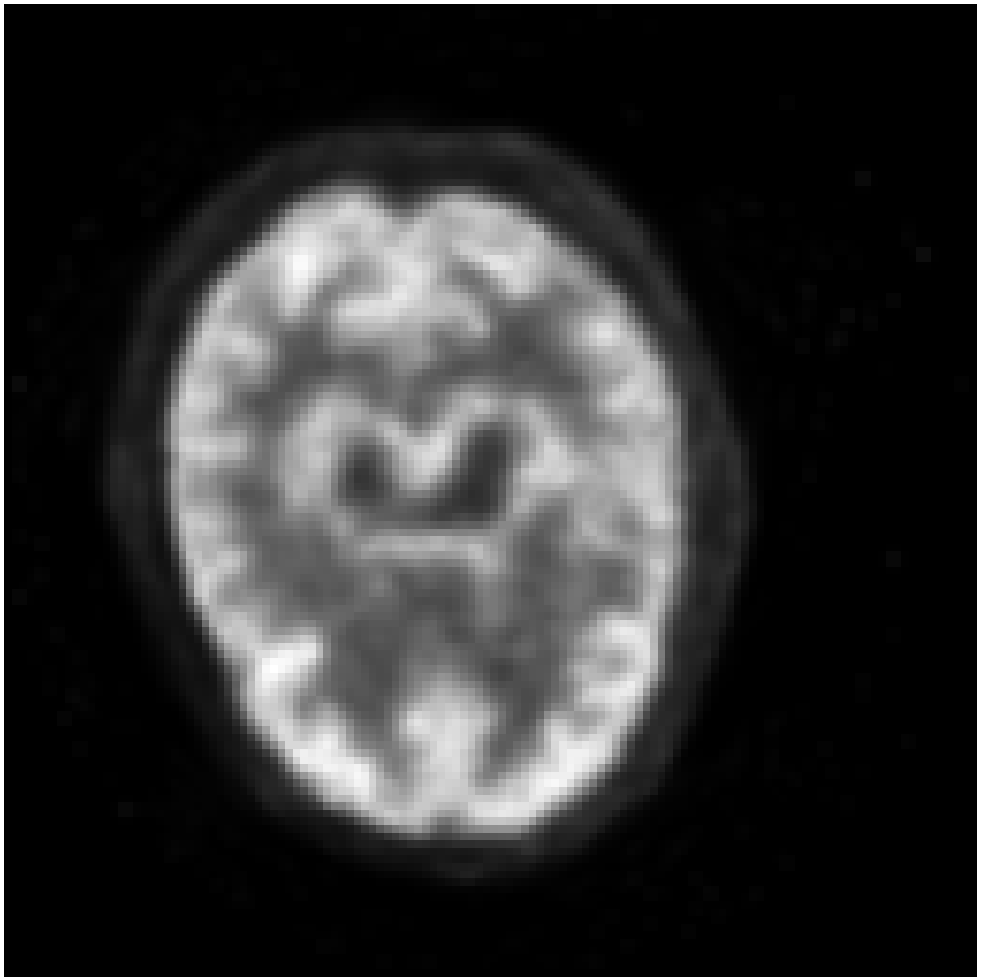
Résultat obtenu à partir de la partition précédente, avec seuil=150, et nombres de Betti contants

Exemple en 3D d'un processus complet de traitement

Tomographie par émission de positron (PET)

Exemple :

1. extraction
2. division : sphère ZI
3. division : intensité $< t_{necrosis}$
4. croissance : nécrose
5. division : 3 surfaces
6. fusion : régions adjacentes
7. croissance : région active
8. fusion : fond
9. déformation



Exemple en 3D d'un processus complet de traitement

Exemple :

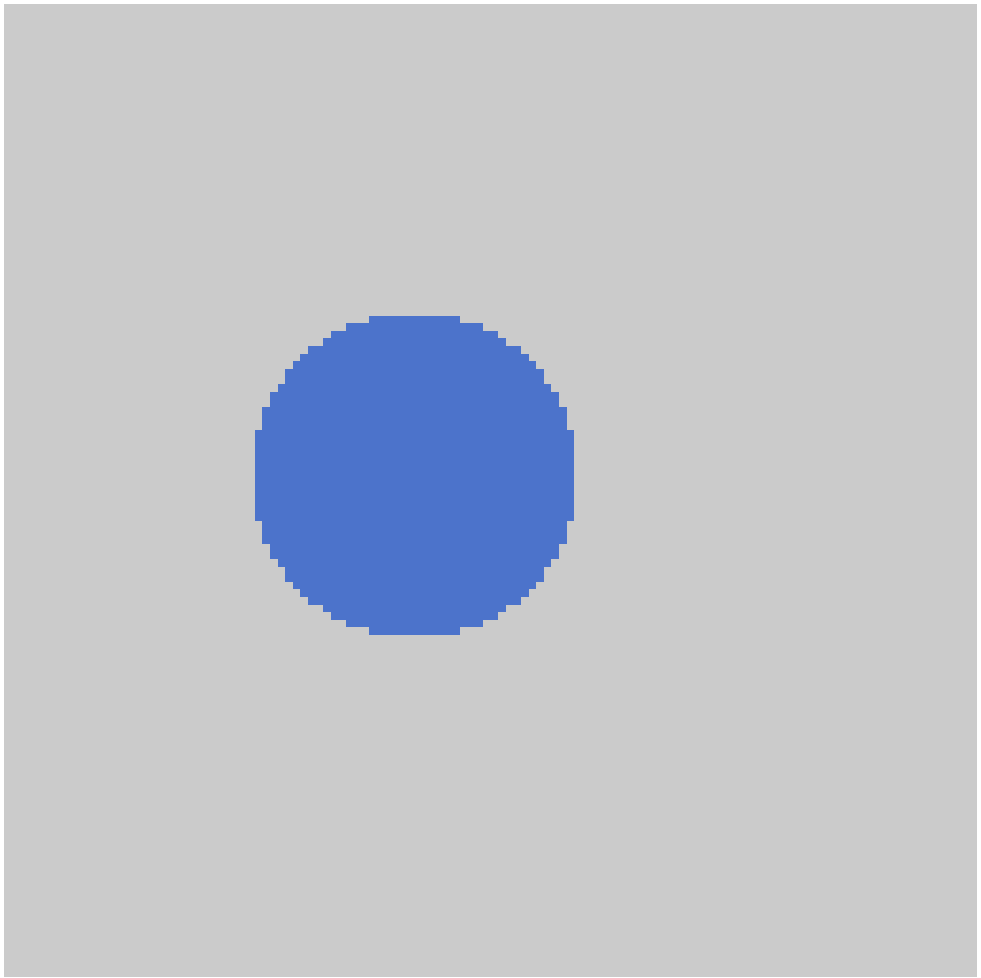
1. extraction
2. division : sphère ZI
3. division : intensité $< t_{necrosis}$
4. croissance : nécrose
5. division : 3 surfaces
6. fusion : régions adjacentes
7. croissance : région active
8. fusion : fond
9. déformation

extraction : image contient une seule région

Exemple en 3D d'un processus complet de traitement

Exemple :

1. extraction
2. division : sphère ZI
3. division : intensité $< t_{necrosis}$
4. croissance : nécrose
5. division : 3 surfaces
6. fusion : régions adjacentes
7. croissance : région active
8. fusion : fond
9. déformation

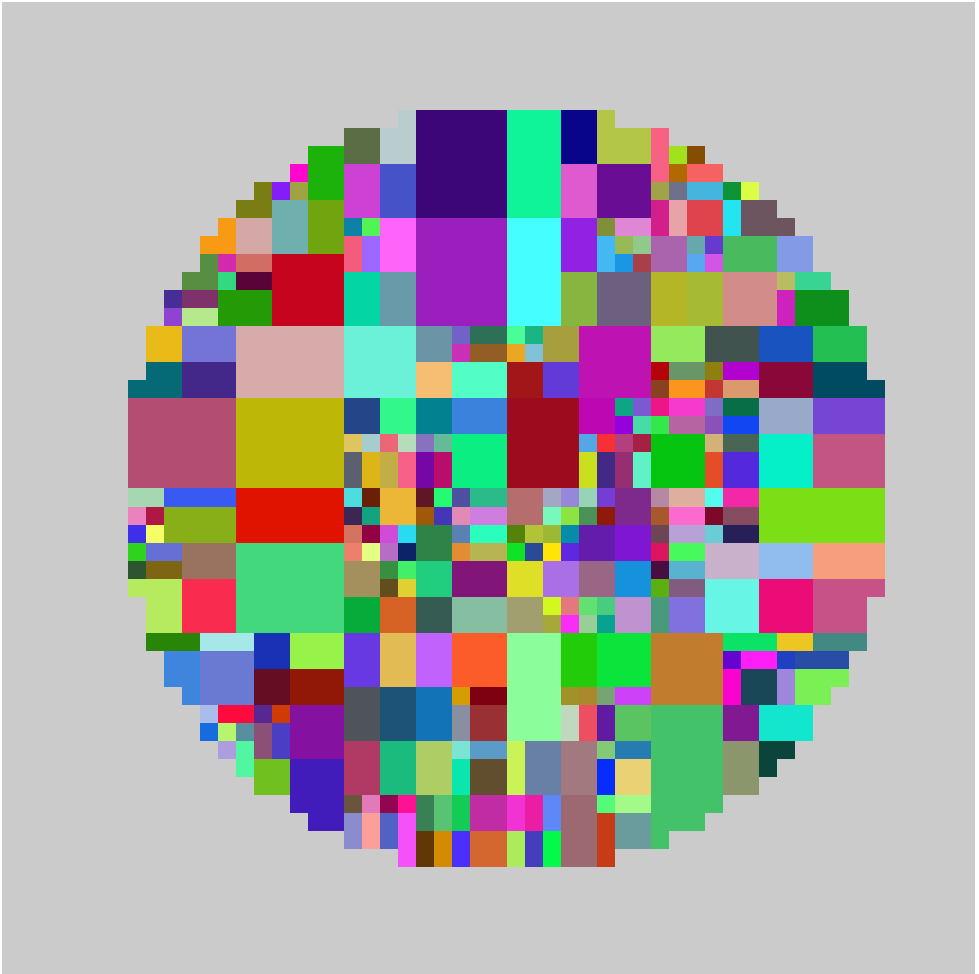


division : sphère atour de la région d'intérêt (ZI)

Exemple en 3D d'un processus complet de traitement

Exemple :

1. extraction
2. division : sphère ZI
3. division : intensité $< t_{necrosis}$
4. croissance : nécrose
5. division : 3 surfaces
6. fusion : régions adjacentes
7. croissance : région active
8. fusion : fond
9. déformation

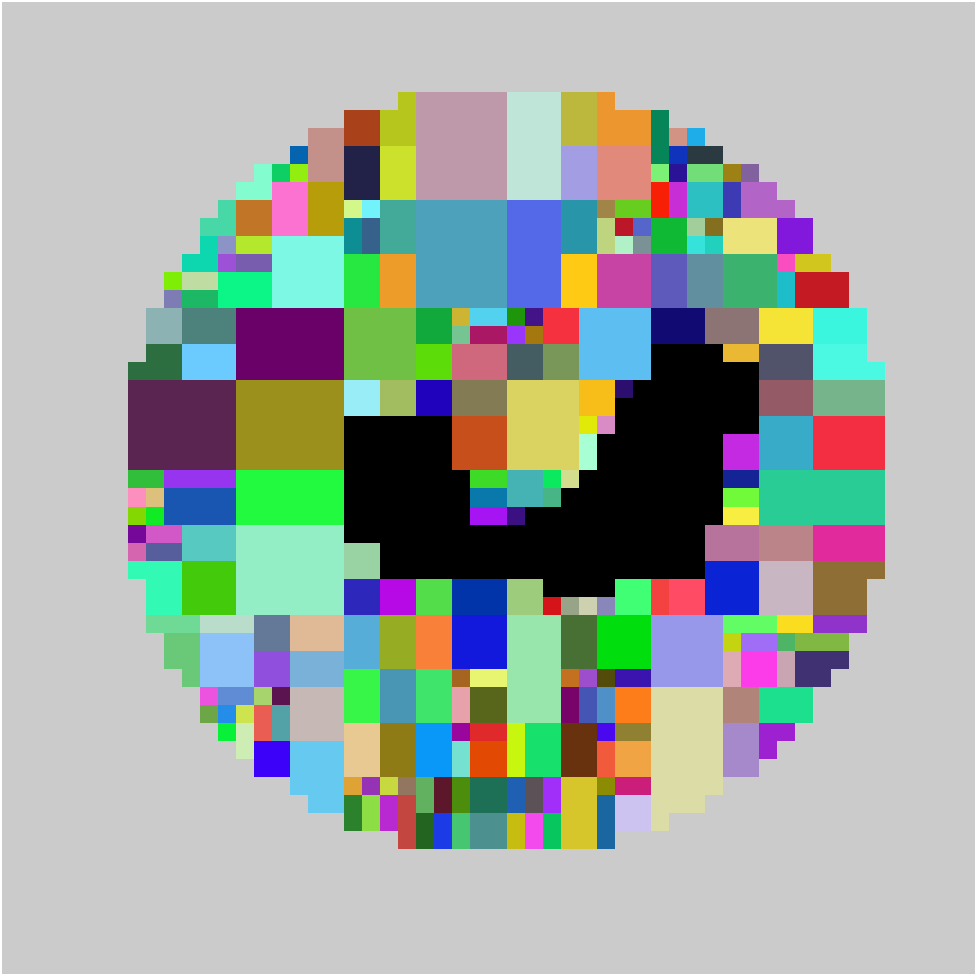


division : région d'intérêt par un critère d'intensité

Exemple en 3D d'un processus complet de traitement

Exemple :

1. extraction
2. division : sphère ZI
3. division : intensité $< t_{necrosis}$
4. croissance : nécrose
5. division : 3 surfaces
6. fusion : régions adjacentes
7. croissance : région active
8. fusion : fond
9. déformation



croissance : nécrose ($\mathfrak{b}_1 \to 0$ et $\mathfrak{b}_2 \to 0$)

Exemple en 3D d'un processus complet de traitement

Exemple :

1. extraction
2. division : sphère ZI
3. division : intensité $< t_{necrosis}$
4. croissance : nécrose
5. division : 3 surfaces
6. fusion : régions adjacentes
7. croissance : région active
8. fusion : fond
9. déformation

division : régions adjacente à la nécrose par des surfaces imbriquées

Exemple en 3D d'un processus complet de traitement

Exemple :

1. extraction
2. division : sphère ZI
3. division : intensité $< t_{necrosis}$
4. croissance : nécrose
5. division : 3 surfaces
6. fusion : régions adjacentes
7. croissance : région active
8. fusion : fond
9. déformation

fusion : régions adjacente à la nécrose $\Rightarrow$ zone active

Exemple en 3D d'un processus complet de traitement

Exemple :

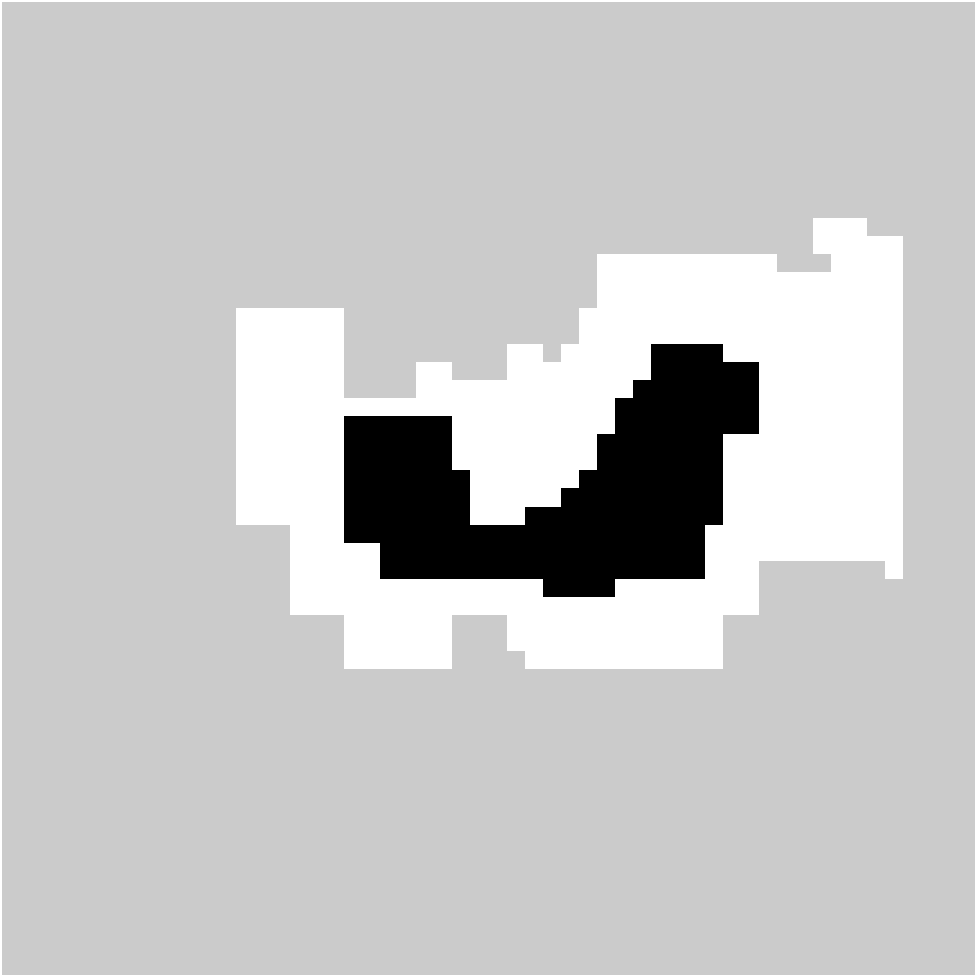
1. extraction
2. division : sphère ZI
3. division : intensité $< t_{necrosis}$
4. croissance : nécrose
5. division : 3 surfaces
6. fusion : régions adjacentes
7. croissance : région active
8. fusion : fond
9. déformation

croissance : zone active ($\mathfrak{b}_1 \to 0$ et $\mathfrak{b}_2 \to 1$)

Exemple en 3D d'un processus complet de traitement

Exemple :

1. extraction
2. division : sphère ZI
3. division : intensité $< t_{necrosis}$
4. croissance : nécrose
5. division : 3 surfaces
6. fusion : régions adjacentes
7. croissance : région active
8. fusion : fond
9. déformation

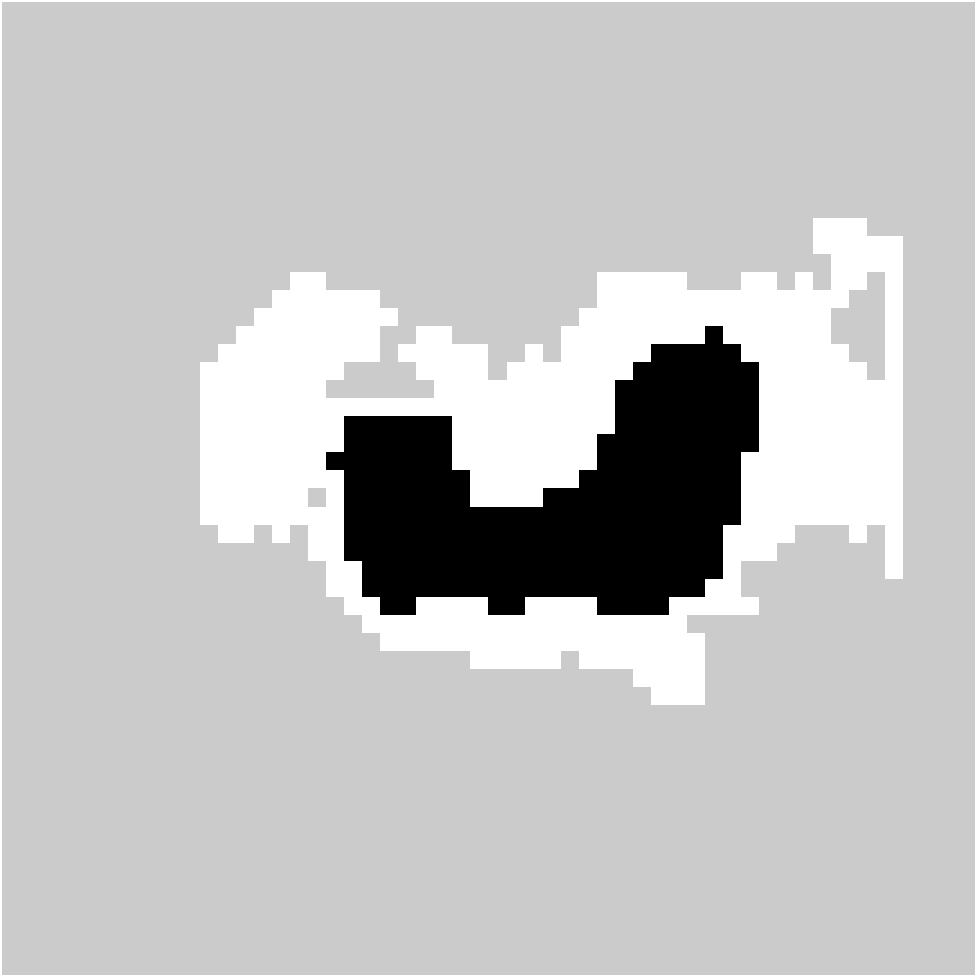


fusion : autres régions que la nécrose et zone active

Exemple en 3D d'un processus complet de traitement

Exemple :

1. extraction
2. division : sphère ZI
3. division : intensité $< t_{necrosis}$
4. croissance : nécrose
5. division : 3 surfaces
6. fusion : régions adjacentes
7. croissance : région active
8. fusion : fond
9. déformation



déformation : surfaces de la nécrose et zone active

5. Simulation Physique

Motivations

Maillages adaptatifs pour la simulation

Adapter localement le maillage $\Rightarrow$ réduit espace mémoire et temps de calculs

Problème

Utilisation de maillages réguliers (tétraèdres/hexaèdres)

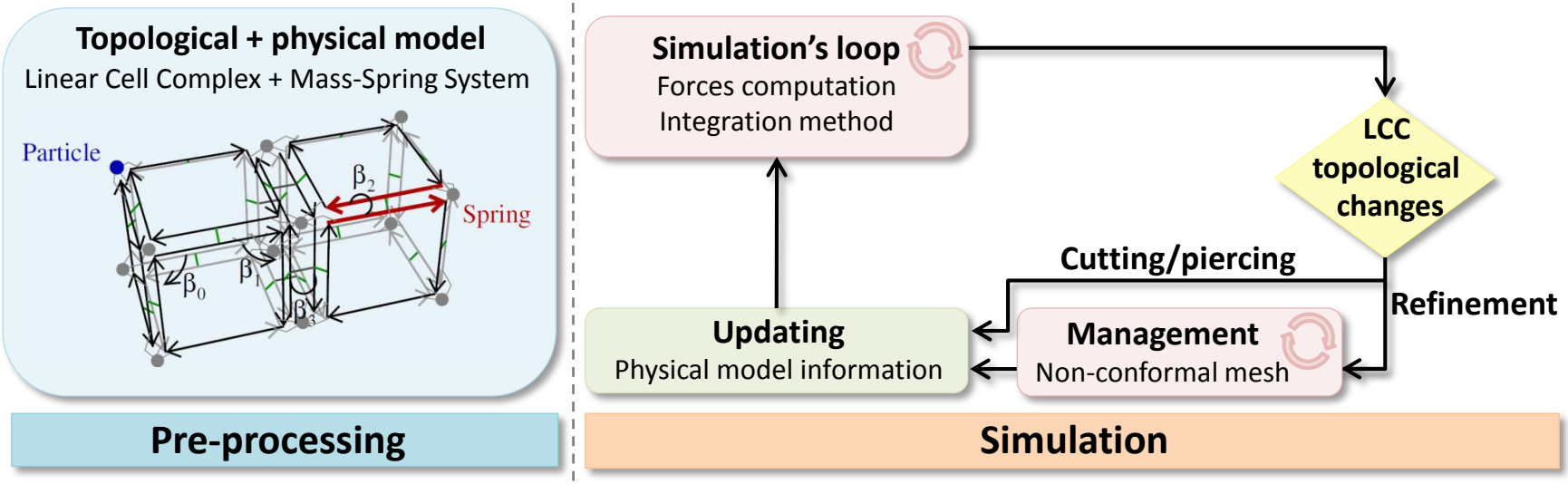
$\Rightarrow$ méthodes d'adaptation complexe, coûteuse, limitée, non locale

Proposer un cadre générique pour la simulation physique

- basé sur les cartes combinatoires
- n'importe quelle topologie de cellules
- mixer des cellules de différentes topologies
- associant les informations physiques au modèle

Avec Elsa Flechon ; Florence Zara ; Fabrice Jaillet ;

Principe Général



Spécialisation des LCC pour les systèmes masses ressorts

Systèmes masses ressorts (MSS) :
des *particules* connectées par des *resorts*

Simulation effectuée par la loi de Newton

$$m_i \frac{d^2}{dt^2}\vec{P}_i(t) = \vec{F}_i(t)$$

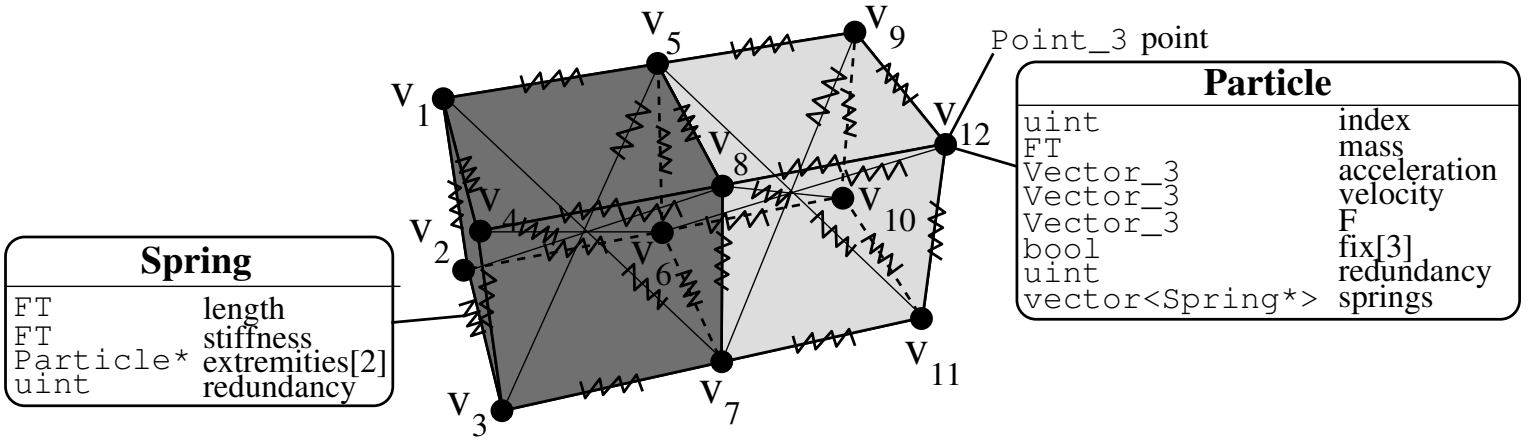
m_i et $\vec{P}_i(t)$ la masse et la position de la particule i
$\vec{F}_i(t)$ la somme des forces appliquées (ressorts, gravité, interactions...)

- on en déduit l'accélération
- on utilise un shéma d'intégration numérique pour calculer la vitesse et la position

Construction du LCC-MSS

Pour le moment, LCC limité à la représentation de tetraèdres et hexaèdres.

- associer une particule à chaque sommet du LCC
- associer un ressort à chaque arête ;
  plus les ressorts diagonaux pour les hexaèdres



Calcul des forces : formules

Force $\vec{F}_{ij}$ appliquée sur le ressort entre les particules i et j

$$\vec{F}_{ij}(t) = \vec{F}^e_{ij}(t) + \vec{F}^v_{ij}(t).$$

avec $\vec{F}^e_{ij}(t)$ la force d'élasticité

$$\begin{cases} \vec{F}^e_{ij}(t) & = k_{ij}\left(\|\vec{P}_j(t) - \vec{P}_i(t)\| - l_{ij}\right)\vec{u}_{ij}(t), \\ \vec{F}^e_{j,i}(t) & = -\vec{F}^e_{ij}(t), \end{cases}$$

et $\vec{F}^v_{ij}(t)$ la force de viscosité

$$\vec{F}^v_{ij}(t) = \gamma_{ij}\left(\vec{V}_j(t) - \vec{V}_i(t)\right) \cdot \vec{u}_{ij}(t)\ \vec{u}_{ij}(t),$$

k_{ij}=raideur, l_{ij}=longueur initiale, $\vec{P}_i(t)$ et $\vec{V}_i(t)$=position et vitesse de la particule i ; $\vec{u}_{ij}(t) = \frac{\vec{P}_j(t) - \vec{P}_i(t)}{\|\vec{P}_j(t) - \vec{P}_i(t)\|}$; γ_{ij}=coefficient de viscosité du ressort $2\sqrt{\frac{m_i + m_j}{2}\ k_{ij}}$

Calcul des forces pour le LCC-MSS

1. remettre à null le vector $\vec{F}$ de chaque particule
2. parcours des ressorts et calcul de chaque force $\vec{F}_{ij}$
   (formule du transparent précédent)
3. parcours des particules pour calculer chaque force $\vec{F}_i$
   somme des forces des ressorts ; plus forces externes (gravité)

Calcul de la vitesse et de la position de chaque particule

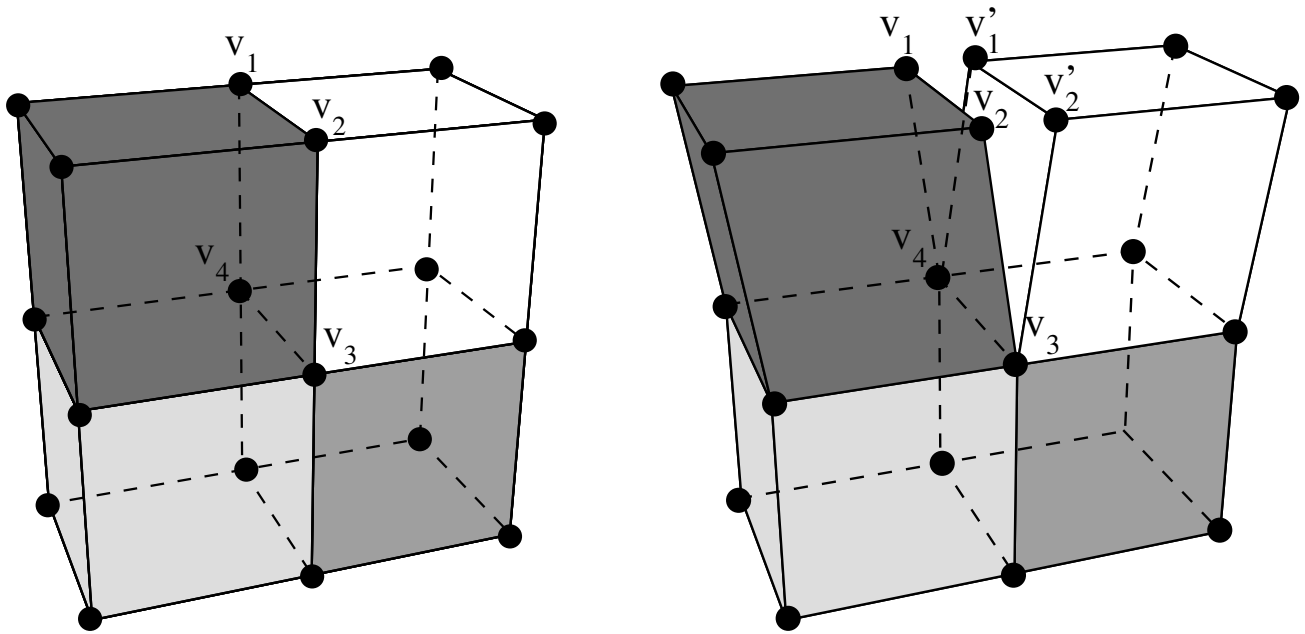
Utilisation d'un shéma d'intégration numérique

- *Euler semi-implicit*
$$\begin{cases} \frac{d}{dt}\vec{P}_i(t+h) &= \frac{d}{dt}\vec{P}_i(t) + h\frac{d^2}{dt^2}\vec{P}_i(t), \\ \vec{P}_i(t+h) &= \vec{P}_i(t) + h\frac{d}{dt}\vec{P}_i(t+h). \end{cases}$$
  Problème : h doit être très petit sinon simulation instable.
- *Euler implicit scheme*
$$\begin{cases} \frac{d}{dt}\vec{P}_i(t+h) &= \frac{d}{dt}\vec{P}_i(t) + h\frac{d^2}{dt^2}\vec{P}_i(t+h), \\ \vec{P}_i(t+h) &= \vec{P}_i(t) + h\frac{d}{dt}\vec{P}_i(t+h). \end{cases}$$

Opération `unsew<3>`

Détache deux volumes qui étaient collés le long d'une face

Peut entraîner la découpe de cellules incidentes
⇒`On_split` appelé pour chaque couple de j-cellule découpées

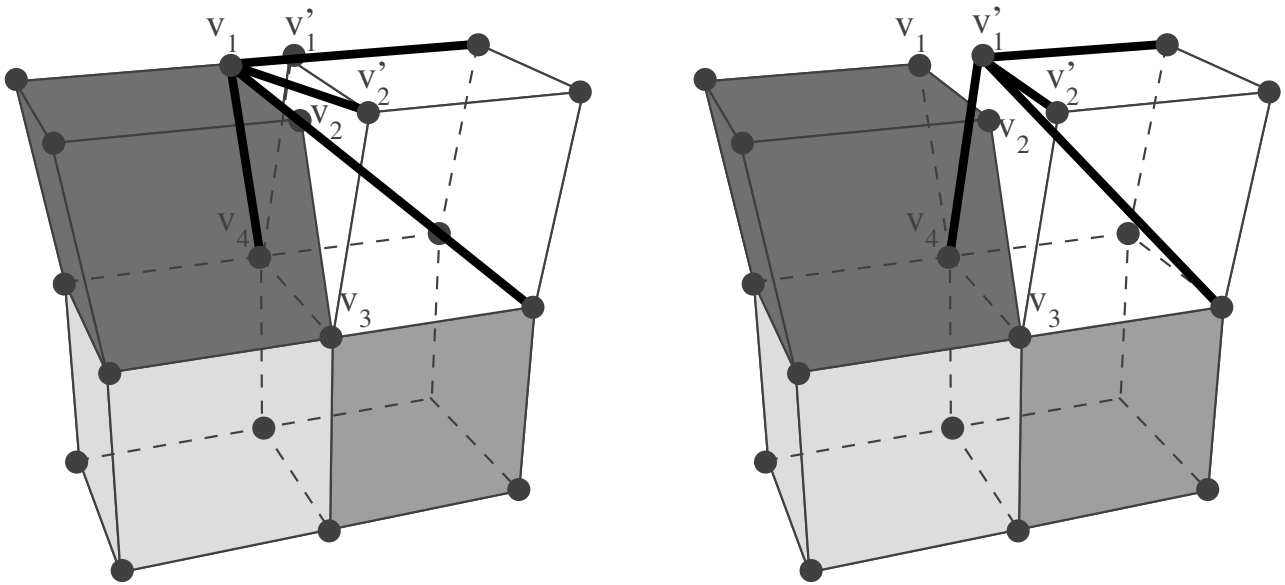


- particules `On_split` : (v_1, v_1') et (v_2, v_2')
- ressorts `On_split` : $((v_4, v_1), (v_4', v_1'))$, $((v_1, v_2), (v_1', v_2'))$ et $((v_2, v_3), (v_2', v_3'))$

Découpe pour les LCC-MSS

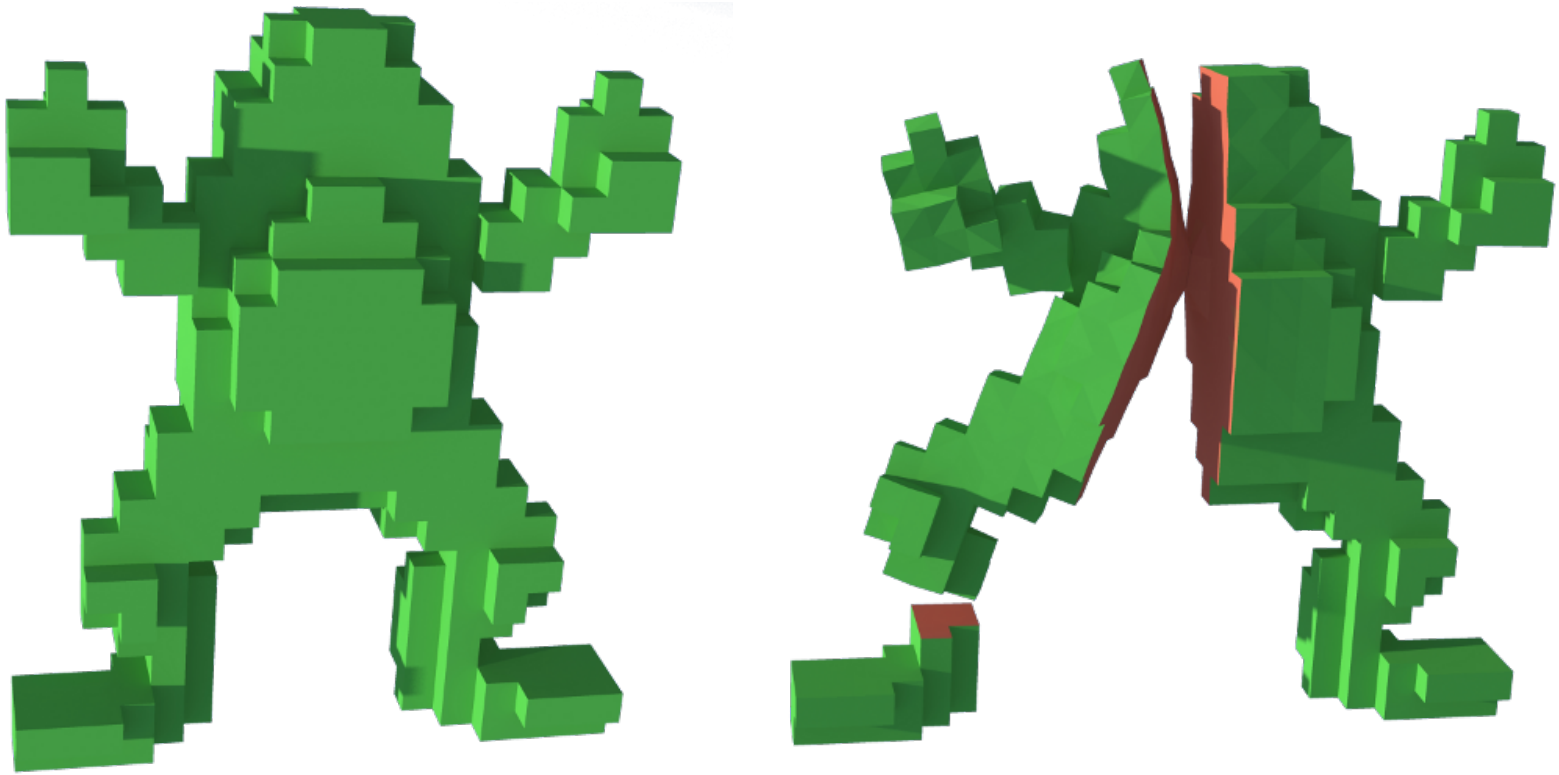
Utilisation de `unsew<3>` ; mise à jour des informations dans `On_split`

- particules (v_i, v_i') : v_i' :nouvel index ; v_i et v_i' :maj redondance et masse ; maj de la liste des ressorts : détacher de v_i et attacher à v_i'
- ressorts $((v_i, v_j), (v_i', v_j'))$: maj redondance des 2 ressorts ; maj des extrémité du nouveau ressort

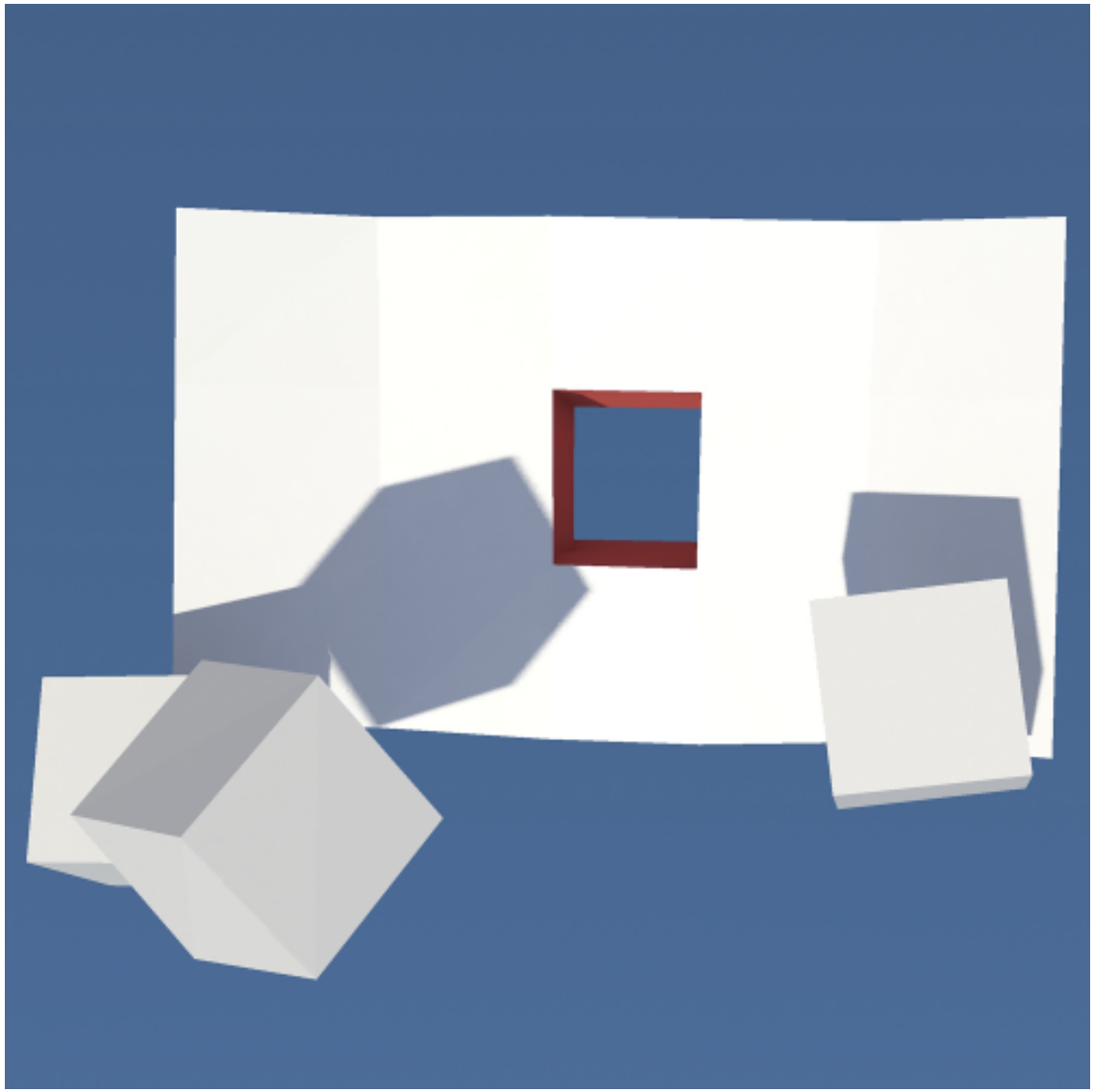


Découpe

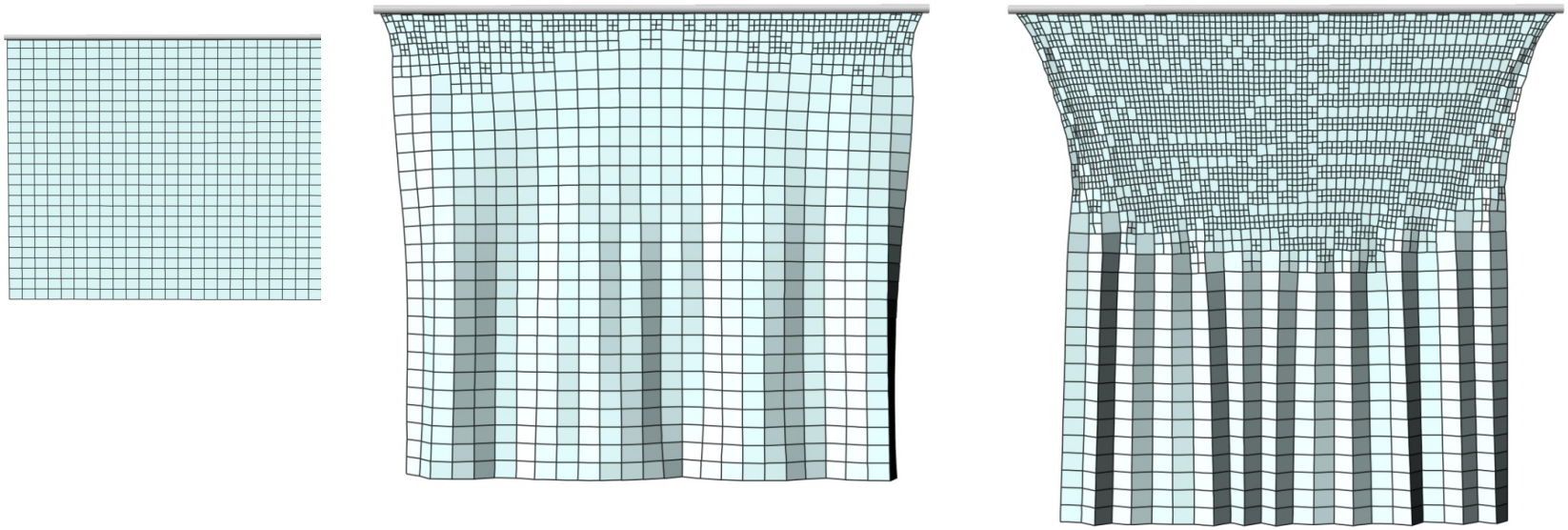
Découpe (2)



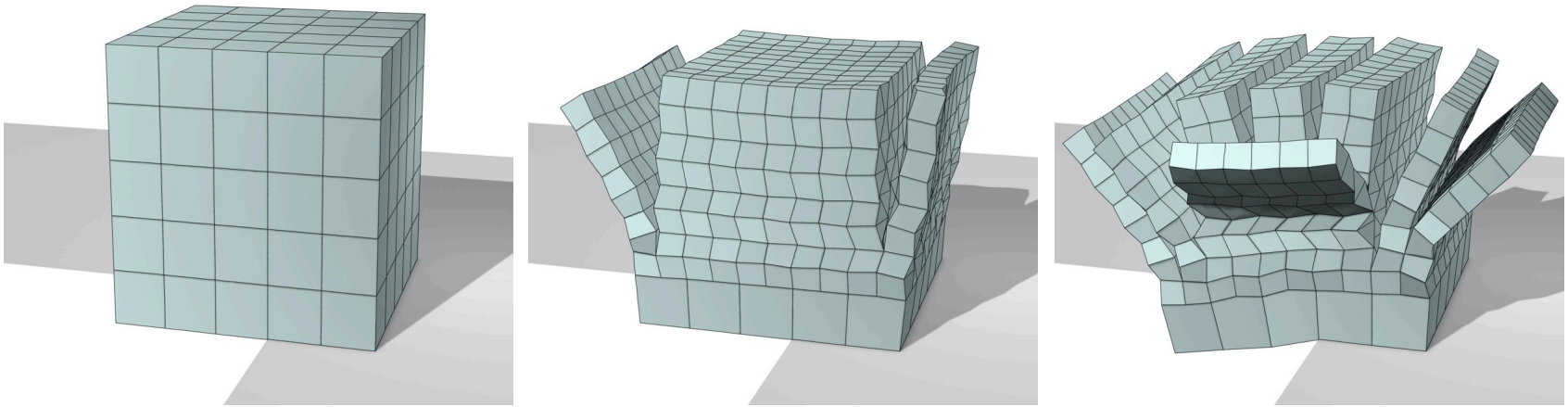
Découpe (3)



Raffinement Adaptatif (1)



Raffinement Adaptatif (2)

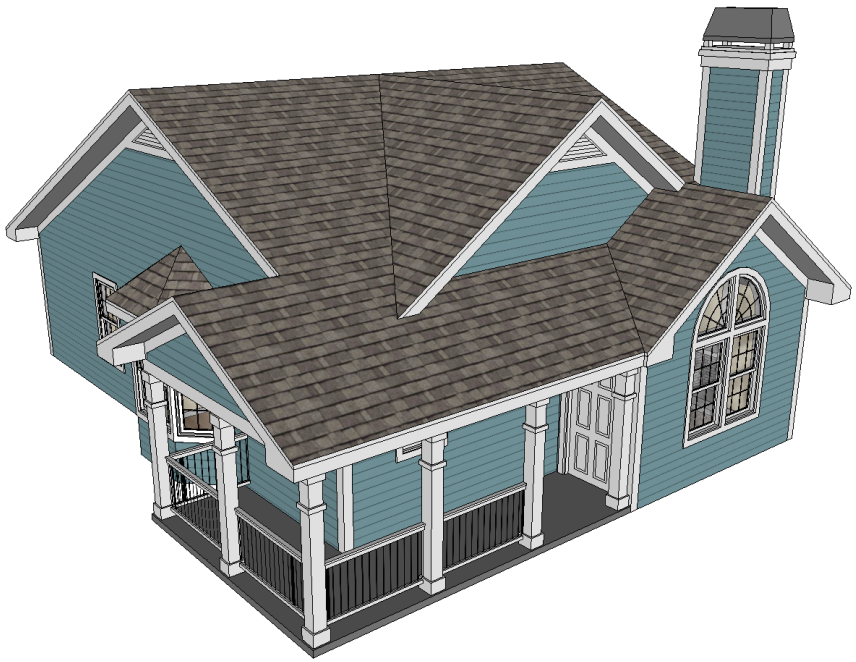


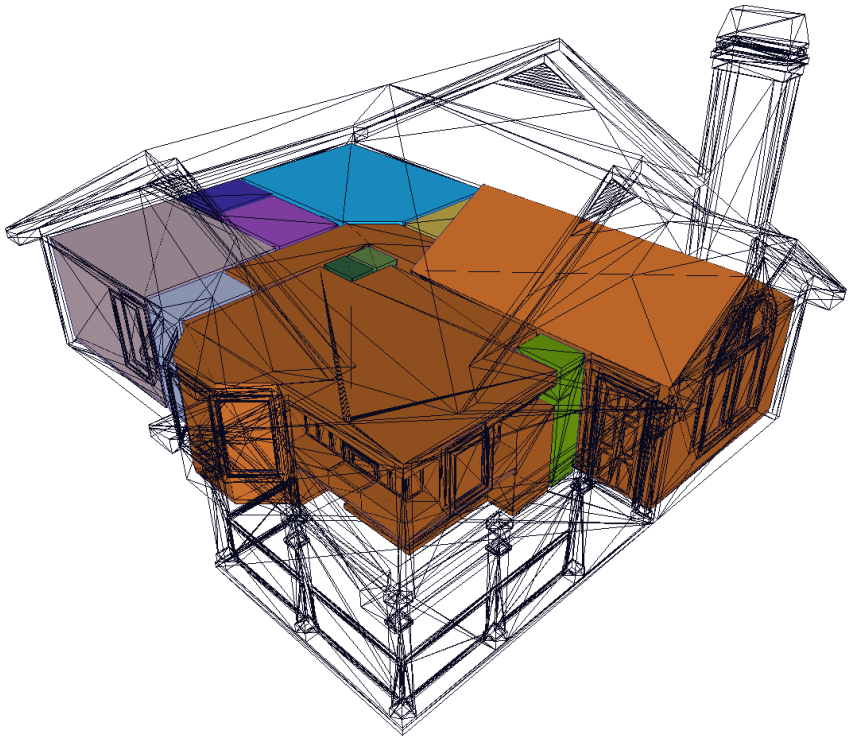
6. Autres Applications

TopoBuilding

Modèle unifié de bâtiments pour la simulation acoustique et énergétique

1. Reconstruction topologique à partir de la géométrie seule
2. Utilisation du modèle topologique pour des opérations dédiées





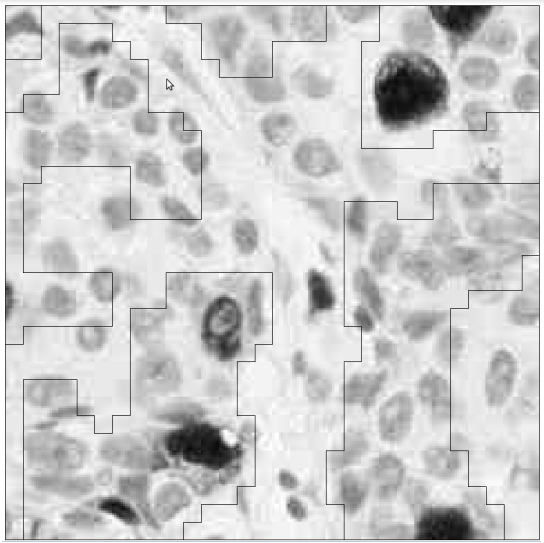
collaboration avec Abdoulaye Diakité, Dirk Van Maercke : CSTB, Grenoble

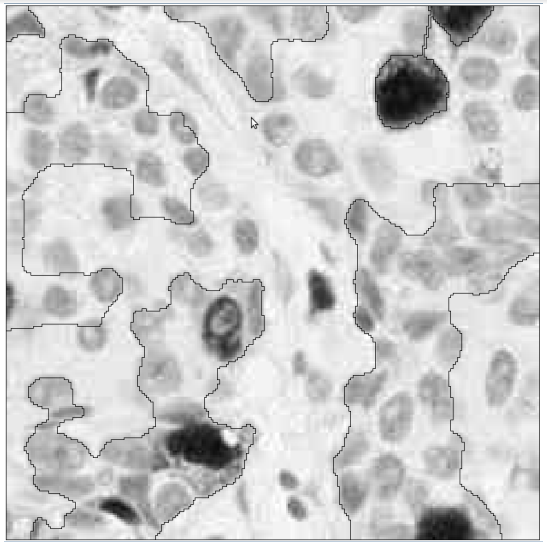
Partitions Déformables

Partitions déformables basées carte topologique et points simple multi-labels

Pixel $x \in X$ est ML-simple pour R si

1. $c(x, R)$ homotope à un 1-disque ;
2. $c(x, X)$ homotope à un 1-disque ;
3. $\forall O$ 4-adjacente à x, $O \neq X$ et $O \neq R$:
   $c^-(x, O)$ est simple pour $f(X, O)$; et add-simple pour $f(R, O)$.





Collaboration avec Jacques-Olivier Lachaud : LAMA, Chambery

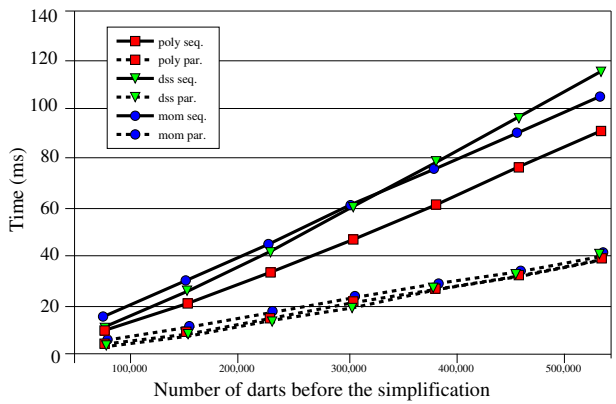
Approximation polygonale générique et parallèle

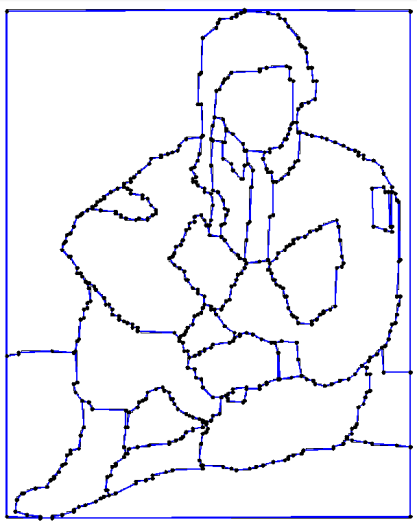
Approximation polygonale générique et parallèle basé cartes combinatoires

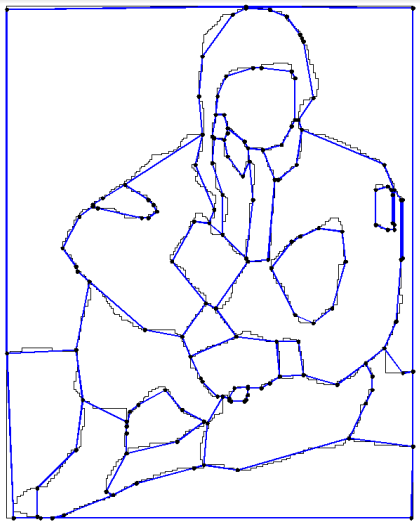
Principe

- Carte combinatoire 2D pour décrire les pointels/lignels
- Supprimer en parallèle les pointels selon un critère géométrique

La carte structure les contours et donne les simplifications pouvant s'effectuer en parallèle







Collaboration avec David Coeurjolly : LIRIS, Lyon

7. Conclusion

Conclusion

Utilisation des cartes

- traitement d'images 2D / 3D
  - segmentation d'images, plusieurs opérations
- modélisation géométrique
  - modeleur géométrique, calcul d'invariant topologiques, simulation physique

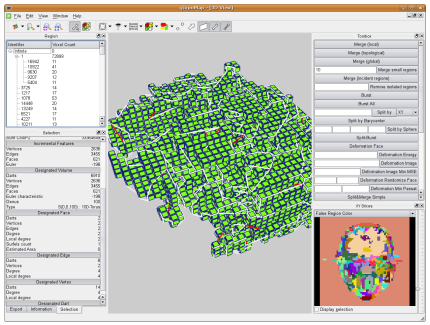
Avantages d'utiliser un modèle cellulaire

- manipulation des subdvisions
- calcul et contrôle de la topologie
- plus grande flexibilité des opérations
- utilisation de différent types de cellules
- algorithmes locaux et mises à jour incrémentales

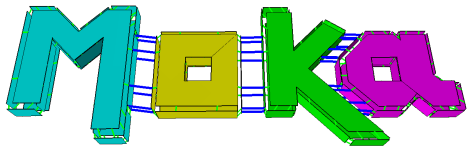
$\Rightarrow$ algorithmes efficaces et précis

Logiciels Existants

- **Carte Topo 2D/3D**
  segmentation d'images 2D/3D, calcul des nombres de Betti



- **Moka**
  Modeleur topologique 3D, 3G-cartes, calcul des groupes d'homologies
  `http://moka-modeller.sourceforge.net/`



- **CGAL**
  Cartes combinatoires nD, opérations de base
  `http://www.cgal.org/`