# Cartes combinatoires pour l'analyse d'images

Guillaume Damiand, Luc Brun

# Cartes combinatoires pour l'analyse d'images

Guillaume Damiand Luc Brun

## 1 Introduction

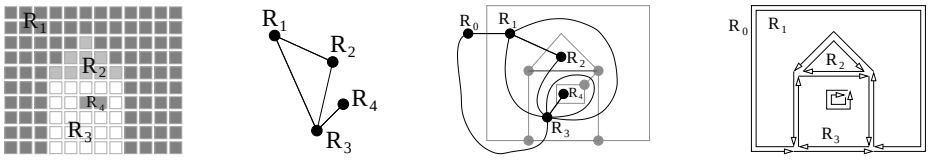
L'analyse d'images consiste à extraire des informations d'une ou plusieurs images. Pour cela, il est nécessaire de dégager des mesures pertinentes pour l'application visée afin d'avoir une certaine "compréhension" du contenu de l'image. Dans ce but, l'image est souvent décomposée en zones connexes, appelées régions, plus proches d'un découpage sémantique de l'image. Ces régions doivent former une partition de l'ensemble des pixels. Si l'on impose à chaque région d'être homogène selon un certain critère, le processus de création de la partition est appelé une segmentation.

La segmentation permet d'analyser les régions et leurs relations (adjacence, inclusions...) pour obtenir des informations pertinentes sur le contenu de l'image. La définition d'une segmentation peut être basée sur une première partition en régions qui est ensuite modifiée par des opérations de découpe et/ou de fusion de régions afin de fournir une partition en un nombre minimal de régions qui préserve les informations utiles à l'étape d'analyse.

L'étude des relations entre les régions d'une partition peut donc intervenir lors de la construction d'une segmentation ou lors de son analyse. Trois types de modèles sont généralement utilisés pour coder les relations entre les régions.

1. Les Graphes d'Adjacence de Régions (Region Adjacency Graph ou RAG) sont basés sur les graphes simples (sans arêtes multiples ni boucles). Ces graphes sont construits en

Figure 1 – Exemples de modèles utilisés en analyse d'images. (a) Une image 2D partitionnée en régions. (b) Le graphe d'adjacence des régions. (c) Le graphe dual. Le graphe primal qui est un multi-graphe d'adjacence est dessiné en noir et son graphe dual en gris. (d) La carte combinatoire.




(a) (b) (c) (d)

associant un sommet à chaque région de la partition et en connectant par une arête tout couple de sommets correspondant à des régions adjacentes (cf. exemple figure 1(b)).

2. Les graphes duaux [7] sont définis à partir d'un premier graphe, dit "graphe primal", qui est un RAG étendu pour représenter les adjacences multiples : sur l'exemple figure 1(c), le sommet $R_3$ associé à la façade maison est relié deux fois au sommet $R_1$ représentant le fond, car la façade est en contact deux fois avec le fond par deux murs différents. Le modèle des graphes duaux stocke, de plus, le dual du graphe primal afin de représenter toutes les cellules de la partition (faces et arêtes dans le graphe primal, et sommets et arêtes dans le dual).
3. Les cartes combinatoires sont définies dans ce chapitre (figure 1(d)). Comparés aux graphes d'adjacence de régions et aux graphes duaux, les modèles basés sur des cartes combinatoires présentent des propriétés intéressantes :
   (a) Elles peuvent être définies en dimension quelconque : on peut donc avec des formalismes similaires définir des partitions d'images 2D, 3D ou de dimension supérieure (cf. section 4 pour l'extension 3D).
   (b) Elles permettent un accès efficace à la géométrie des frontières et des régions.
   (c) Contrairement aux autres modèles, les cartes combinatoires représentent toutes les relations d'adjacences et d'incidences entre les éléments de la partition considérée, quelle que soit la dimension.

## 2 Présentation des cartes combinatoires

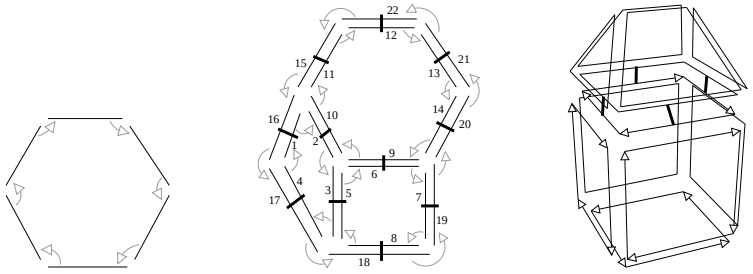
Une carte combinatoire est un modèle mathématique représentant une subdivision de l'espace ainsi que toutes les relations d'indicence et d'ajacence entre les éléments de cette subdivision. Une carte combinatoire permet de représenter les quasi-variétés [1] orientables fermées de dimension $n$. Elles ont été généralisées [8] afin de pouvoir représenter les quasi-variétés en dimension $n$, orientables ou non, fermées ou non.

Il est possible de définir les cartes combinatoires de manière progressive par dimension croissante. Une carte combinatoire en dimension 1 permet de représenter une face fermée (cf. figure 2(a)). Pour cela, elle est composée d'un ensemble $B$ d'éléments appelés brins. Chaque brin correspond ici à une arête de la face. Une application définie sur ces éléments va donner, pour chaque brin, le brin suivant de la face. Cette application est une *permutation* [2]. À l'aide des brins et de la permutation, il est possible de parcourir la face. De plus, la permutation (et son inverse) représente la relation d'adjacence entre les arêtes.

Construire une carte combinatoire 2D revient à prendre plusieurs cartes combinatoires 1D et à les "coller" les unes par rapport aux autres. Chaque carte combinatoire 1D représente une face, ces faces sont mises en relation à l'aide d'une *involution* [3] qui représente la relation d'adjacence

1. Intuitivement, une quasi-variété en dimension $n$ est un objet $n$D que l'on peut obtenir en collant des cellules de dimension $n$ le long de cellules de dimension $(n-1)$. De ce fait, une $(n-1)$-cellule ne peut pas appartenir au bord de plus de deux cellules.
2. Une permutation sur un ensemble $B$ est une bijection de $B$ dans $B$.
3. Une involution $f$ sur un ensemble $B$ est une bijection de $B$ dans $B$ telle que $f = f^{-1}$.

FIGURE 2 – Exemples de cartes combinatoires. Les brins sont représentés par les segments noirs. (a) En dimension 1 : la permutation est représentée par les flèches grises. (b) En dimension 2 : l'involution est représentée par les petits segments épais recouvrant les 2 brins en relation. (c) En dimension 3 : représentation partielle, il manque le volume englobant les 2 objets. Ici la permutation est représentée de manière partielle par une flèche sur les brins, l'involution utilisée en 2D n'est pas représentée, et la nouvelle involution est représentée par les petits segments épais recouvrant les 2 brins en relation.




(a) (b) (c)

entre les faces (cf. figure 2(b)). Cette relation d'adjacence est une involution car, étant donné un brin d'une face $f$, il existe une seule autre face adjacente à $f$ le long de l'arête.

Ce type de construction peut s'étendre en n'importe quelle dimension. Par exemple en dimension 3, il suffit de prendre plusieurs cartes combinatoires 2D et de les positionner les unes par rapport aux autres à l'aide d'une nouvelle involution représentant la relation d'adjacence entre les volumes. Comme les cartes représentent uniquement des quasi-variétés, deux volumes sont forcément adjacents par une face (et non par une arête ou un sommet). Cela se traduit par une contrainte sur les applications (cf. définition 1).

Les premiers travaux utilisant les cartes combinatoires étaient uniquement en 2D [4], et utilisaient une permutation $\sigma$ (**s**igma pour sommet) et une involution $\alpha$ (**a**lpha pour arête). Toutefois ces notations se prêtent mal à une définition générique des cartes combinatoires en dimension quelconque. Pour cela, nous préférons la notation introduite par [8] où la permutation est notée $\beta_1$ (1 pour la dimension des cellules adjacentes, ici les arêtes), la première involution $\beta_2$ (adjacence entre faces) et la seconde involution $\beta_3$ (adjacence entre volumes).

**Définition 1. 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. $\beta_1$ est une *permutation* sur $B$ ;
3. $\forall\, 2 \leq i \leq n$, $\beta_i$ est une *involution* sur $B$ ;
4. $\forall\, 1 \leq i \leq n-2$, $\forall\, i+2 \leq j \leq n$, $\beta_i \circ \beta_j$ est une *involution*.

Les brins sont ici une notion abstraite, et servent uniquement de support pour les différentes applications. Seule $\beta_1$ est une permutation, les autres $\beta_i$ sont des involutions. La dernière ligne

de cette définition fixe des contraintes sur la manière dont les brins sont mis en relation afin de garantir la validité des objets représentés. Par exemple, en dimension 3, la contrainte ajoutée est que $\beta_1 \circ \beta_3$ doit être une involution, ce qui revient à dire que lorsque deux brins de deux faces différentes sont en relation pour $\beta_3$, tous les autres brins de ces deux faces sont également en relation deux à deux par $\beta_3$.

Lorsque deux brins $b_1$ et $b_2$ sont tels que $\beta_i(b_1) = b_2$, nous disons que $b_1$ est *i-cousu* à $b_2$. Étant donné que les $\beta_i$, pour $i \neq 1$, sont des involutions, si $b_1$ est *i-cousu* à $b_2$ alors $b_2$ est *i-cousu* à $b_1$. La permutation $\beta_1^{-1}$ est souvent donnée de manière explicite dans les programmes afin d'optimiser les algorithmes et de retrouver directement le brin précédant un brin dans une face sans avoir à en faire le tour.

Une carte combinatoire représente une subdivision d'un objet $n$D, soit toutes les cellules de la subdivision (sommets, arêtes, faces, volumes...). Ces cellules ne sont pas représentées explicitement mais données implicitement par la notion d'orbite.

**Définition 2. orbite** Soit $\Phi = \{f_1, \ldots, f_k\}$ des permutations sur un même ensemble $B$. Nous notons $< \Phi >$ le groupe de permutations engendré par $\Phi$. C'est l'ensemble des permutations qu'il est possible d'obtenir de $\Phi$ par application de la composition et de l'inverse. L'*orbite* de $b \in B$ relativement à $\Phi$, notée $< \Phi > (b) = \{\phi(b) | \phi \in < \Phi >\}$, est l'ensemble des éléments de $B$ qu'il est possible d'atteindre par application, à partir de $b$, de n'importe quelle suite de $f_i$ et $f_i^{-1}$.

De manière intuitive, une orbite $< \Phi > (b)$ dans une carte combinatoire peut-être vue comme l'ensemble des brins qu'il est possible d'atteindre par un parcours en largeur d'origine $b$ en utilisant toutes les permutations $f_i$ et $f_i^{-1}$.

Les cellules de la subdivision sont des orbites spécifiques (cf. définition 3 et figure 2), les autres orbites pouvant être vu comme des cellules partielles (par exemple un sommet à l'intérieur d'une face et d'un volume).

**Définition 3. i-cellule** Soient $C$ une $n$-carte, $b$ un brin de cette carte, et $i \in \{0, \ldots, n\}$. La $i$-cellule incidente à $b$ est :

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

Nous avons deux cas : l'un pour la définition des 0-cellules (les sommets), et l'autre pour les cellules de dimension supérieure. Pour les 0-cellules, nous parcourons seulement un brin sur deux, afin d'atteindre uniquement les brins ¡¡ sortants ¿¿ du sommet incident à $b$. Les autres $i$-cellules forment simplement l'orbite composée de tous les $\beta$ sauf $\beta_i$.

Chaque ensemble de $i$-cellules est une partition de l'ensemble des brins de la carte. Chaque brin appartient donc exactement à une seule $i$-cellule, $\forall i \in 0, \ldots, n$. Par définition, on dit qu'un brin est incident à une cellule s'il appartient à l'orbite correspondante. De plus, deux cellules sont incidentes si les orbites correspondantes ont une intersection non vide, c'est-à-dire s'il existe au moins un brin appartenant aux deux cellules.

La carte figure 2(b) est composée de 9 sommets (0-cellules) $\{\{1, 17\}, \{2, 11, 16\}, \{3, 6, 10\}, \{4, 5, 18\}, \{7, 9, 20\}, \{8, 19\}, \{12, 15\}, \{13, 22\}, \{14, 21\}\}$, de 11 arêtes (1-cellules) $\{\{1, 16\}, \{2, 10\}, \{3, 5\}, \{4, 17\}, \{6, 9\}, \{7, 19\}, \{8, 18\}, \{11, 15\}, \{12, 22\}, \{13, 21\}, \{14, 20\}\}$ et de 4 faces (2-cellules) $\{\{1, 2, 3, 4\}, \{5, 6, 7, 8\}, \{9, 10, 11, 12, 13, 14\}, \{15, 16, 17, 18, 19, 20, 21, 22\}\}$.

Pour parcourir la face incidente à un brin, par exemple le brin 1, il suffit d'effectuer un parcours en largeur d'origine ce brin en utilisant $\beta_1$ et $\beta_1^{-1}$. Pour les sommets, on part d'un brin, par exemple le brin 2, et on utilise $\beta_1 \circ \beta_2$ et son inverse $\beta_2 \circ \beta_1^{-1}$. À partir du brin 2, on atteint le brin 11 en utilisant $\beta_1 \circ \beta_2(2)$ et le brin 16 à l'aide de $\beta_2 \circ \beta_0(2)$. Ici le parcours s'arrête à une profondeur de un car tous les brins atteignables ont été trouvés, mais ce n'est pas toujours le cas selon le nombre de brins du sommet.

# 3 Cartes pour l'analyse d'images 2D

## 3.1 Codage de la topologie et de la géométrie d'une image

Dans le cadre de l'analyse d'images, une partition présente trois types d'informations distinctes :

1. Des informations *topologiques* codant les relations entre les régions de la partition (adjacences, inclusions).
2. Des informations *géométriques* codant la géométrie des régions. On peut par exemple vouloir retrouver l'ensemble des pixels d'une région où l'ensemble de ses points frontières.
3. Des informations *photométriques* concernant les mesures attachées à l'ensemble des pixels d'une région (couleur moyenne, variance,...).

Les informations photométriques peuvent être retrouvées à partir d'un parcours de l'ensemble des pixels d'une région. La principale difficulté consiste donc à fournir un codage efficace des informations géométriques et topologiques et à maintenir la cohérence entre ces deux types d'informations.

Dans le cadre de modèles basés sur les cartes combinatoires, la géométrie de la partition est généralement codée en distinguant : les *segments* codant les frontières entre les régions et les *nœuds* définis comme l'intersection de plusieurs segments. Un segment correspond à une suite maximale [4] de points frontières ; il sera ici orienté. Le contour d'une région est alors défini comme une suite alternée de segments et de nœuds. De tels contours doivent vérifier la propriété de Jordan (cf. chapitre 1) afin de délimiter correctement les régions.

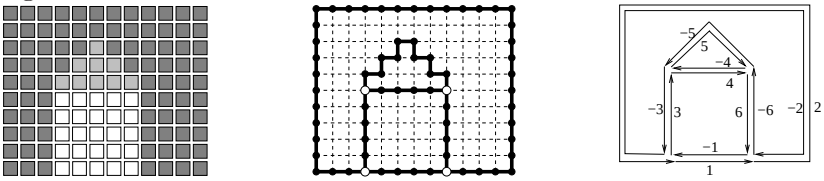
Comme chaque frontière sépare des régions, une approche naturelle consiste à définir les segments dans la grille demi-entière présentée au chapitre 1. Un nœud correspond alors à un pointel. Les segments peuvent être stockés en codant les coordonnées du nœud initial et une suite de déplacements (nord, sud, est, ouest) permettant d'aboutir au nœud extrémité. Diverses optimisations (codage de plages de déplacements, reconnaissance de droites discrètes cf. chapitre 6 ...) peuvent réduire, si nécessaire, la taille d'un tel codage. Cette approche peut être aisément généralisée à d'autres types de pavages.

Comme nous pouvons le voir sur la figure 3, le lien entre la géométrie et la topologie de la partition est établi en construisant une carte combinatoire où :

— chaque brin est associé à un segment et
— chaque sommet de la carte est associé à un nœud.

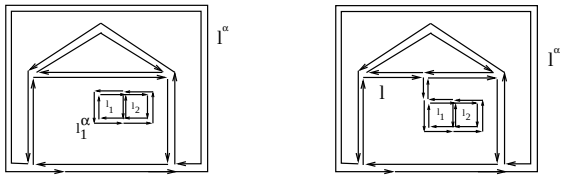
4. Maximale car il n'est pas possible de rajouter un point frontière pour prolonger un segment tout en gardant la notion de frontière.

FIGURE 3 – Codage de la partition (a) par un modèle combinant un codage de sa géométrie (b) et de sa topologie (c). Les lignels sont représentés sur la figure (b) par des traits pointillés. Les pointels appartenant aux segments, les nœuds et les lignels des segments sont quant à eux représentés par des ronds pleins (•), des cercles (○) et des traits pleins. Les pixels correspondent aux carrés de la grille.




(a) image (b) géométrie (c) topologie

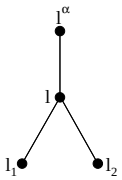
FIGURE 4 – Codage des relations d'inclusion (a) par une arête fictive (b), des relations parent/enfant (c) et un arbre d'inclusion (d).




(a) Inclusion de deux cartes (b) Arête fictive

$$Parent(l_1^\alpha) = l$$
$$Enfant(l) = \{l_1^\alpha\}$$

(c) Relations Parent / Enfant




(d) Arbre d'inclusions

Notons que deux brins 2-cousus $b$ et $\beta_2(b)$ définissent deux orientations opposées sur la même arête (par exemple 4 et $-4$ sur la figure 3). Chaque cycle de $\beta_1$ codant une face définit également un contour fermé muni d'une orientation. Tous les contours associés aux faces d'une carte combinatoire *sauf un* sont parcourus dans le sens négatif. Le cycle de $\beta_1$ correspondant au parcours dans le sens positif, est appelé la face infinie d'une carte (notée ici $l^\alpha$) et correspond à la face codant l'ensemble du plan moins le domaine défini par l'union des faces appartenant à la carte (cf. figure 3(c) où la face infinie est composée des deux brins {1,2}).

Le problème de ce formalisme est qu'il ne permet de coder qu'une seule composante connexe. Si nous rajoutons par exemple une fenêtre à la maison (figure 4(a)), nous créons une nouvelle carte combinatoire sans lien avec la carte précédente, et nous ne pouvons pas décider dans quelle face de la carte englobante est contenue la fenêtre.

Une première solution à ce problème consiste à ajouter une arête fictive (ou pseudo-arête) connectant chaque composante connexe à sa face englobante (figure 4(b)). Chaque composante connexe est alors reliée "fictivement" à sa face englobante et l'ensemble de la partition peut être codé par une seule carte combinatoire. Ce type de solution est généralement utilisé dans les méthodes de segmentation ascendantes ou hiérarchiques [2, 9]. En effet, ces méthodes construisent la partition par fusion de régions et la création d'arêtes fictives s'effectue dans ce cas naturellement en évitant de supprimer certaines arêtes.

Une seconde solution consiste à représenter explicitement les relations d'inclusion. Plusieurs solutions équivalentes ont été proposées à ce jour : une première solution s'appuie sur les fonctions de labels et consiste à définir deux fonctions : une fonction *Parent* qui indique pour chaque face infinie la face finie qui la contient (une face finie est forcément incluse dans exactement une face infinie) et une fonction *Enfant* qui indique pour toute face finie la liste des faces infinies qu'elle contient (une face finie contient exactement une face infinie par composante connexe, ou "trou", qu'elle contient) (figure 4(c)). Une solution sensiblement équivalente est basée sur un arbre d'inclusion où les relations portent ici uniquement sur les faces finies. Toute face finie a pour enfant dans l'arbre l'ensemble des faces finies qu'elle contient (figure 4(d)).

## 3.2 Construction du modèle à partir d'une image

Nous présentons ici la manière de construire la carte combinatoire associée à une partition d'image 2D. Cette partition peut-être donnée soit de manière explicite, sous la forme d'une image étiquetée où chaque pixel contient l'étiquette de sa partition, soit de manière implicite au moyen d'une fonction prenant deux pixels en paramètre et retournant vrai ou faux selon que les deux pixels sont dans la même région ou non. Nous nous intéressons uniquement ici à la construction du modèle représentant la topologie de la partition à l'aide d'une carte combinatoire. En effet, la construction du modèle représentant la géométrie ne pose pas de problème particulier.

### 3.2.1 Algorithme naïf

L'algorithme 1 présente une première manière naïve pour construire la carte topologique à partir d'une partition d'image. Son principe consiste à construire une carte combinatoire représentant tous les pixels de l'image au moyen de faces carrées, plus une face englobant ces pixels représentant la région infinie (figure 5(b)), puis à simplifier cette carte progressivement jusqu'à obtenir la représentation minimale en nombre de cellules.

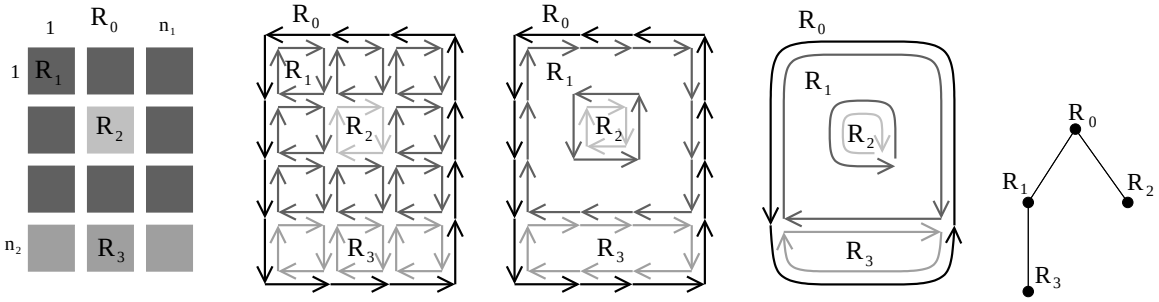
**Algorithm 1:** Extraction naïve de la carte combinatoire d'une image.

**Entrée**: Une image étiquetée $I$ de $n_1 \times n_2$ pixels.
**Sortie**: La carte combinatoire correspondant à $I$.

1 Construite la 2-carte composée de $n_1 \times n_2$ carrés 2-cousus entre eux;
2 Supprimer chaque arête $(b, \beta_2(b))$ telle que $b$ et $\beta_2(b)$ ont même étiquette;
3 Supprimer chaque sommet de degré deux;
4 Calculer l'arbre d'inclusion des régions;

Pour cela, les arêtes internes se trouvant à l'intérieur d'une même région sont supprimées (figure 5(c)) afin de ne conserver que les arêtes représentant les frontières de l'image. Puis les sommets de degré deux sont supprimés afin de ne conserver que les sommets représentant des nœuds (aux intersections de plusieurs frontières, cf. figure 5(d)). Cette carte est minimale en nombre de cellules car aucune cellule restante ne peut être supprimée sans entraîner une perte d'information topologique.

FIGURE 5 – Le principe de construction naïf d'une carte minimale représentant une image 2D étiquetée. (a) Une image 2D étiquetée. (b) La carte initiale correspondante. (c) La carte obtenue après suppression des arêtes internes. (d) La carte finale obtenue après la suppression des sommets de degré deux et l'arbre d'inclusion associé.




L'avantage principal de cet algorithme est qu'il est très simple. Son principal inconvénient est qu'il commence par créer beaucoup de cellules, et donc de brins, avant d'en supprimer la majorité dans les passes successives de simplification. De plus, même si cet algorithme est linéaire en nombre de brins de la carte initiale, les nombreuses allocations et destructions mémoire entraînent un surcoût non négligeable en temps d'exécution.

### 3.2.2 Algorithme incrémental

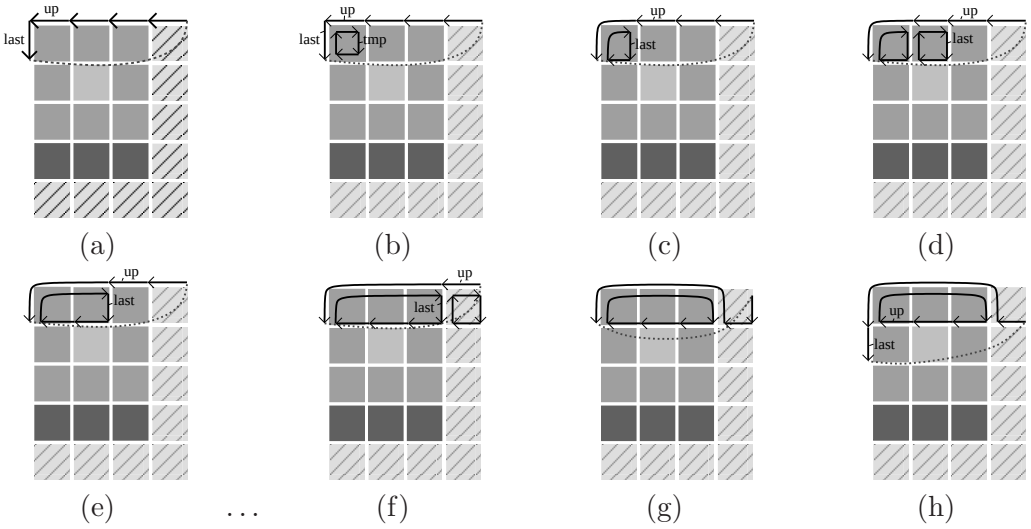
L'algorithme 2 présente une seconde manière de construire la carte topologique à partir d'une partition d'image, mais cette fois de manière incrémentale. Il s'agit de balayer l'image ligne à ligne et colonne à colonne en créant localement la carte représentant le pixel courant. Les cellules incidentes à ce pixel sont ensuite simplifiées.

L'invariant principal de cet algorithme est que la carte plus ancienne par rapport à l'ordre de parcours des pixels doit avoir été déjà construite. En effet, cet invariant garantit de pouvoir raccrocher le carré représentant le pixel courant au pixel situé à sa gauche (et désigné dans la carte par le brin appelé *last*) et au pixel situé au dessus de lui (et désigné par le brin appelé *up*) (cf. figure 6).

Afin de garantir cet invariant pour le pixel $(1, 1)$, nous commençons par créer un bord supérieur et gauche dans la carte avant de commencer le balayage de l'image (présenté figure 6(a) pour l'image de la figure 5(a)). Ce bord initial est plaqué sur un cylindre en reliant le brin à gauche du premier pixel et le brin au-dessus du dernier pixel de la première ligne. Ceci permet de parcourir l'image de manière unique sans avoir de traitement particulier pour les pixels se trouvant au bord de l'image. De plus, toujours afin d'éviter des traitements particuliers au bord de l'image, nous augmentons la fenêtre de balayage d'un pixel à droite et en-dessous de l'image (les pixels hachurés sur la figure 6).

Pour chaque pixel de l'image, le traitement consiste à créer un carré représentant ce pixel

FIGURE 6 – Déroulement de quelques étapes de la construction incrémentale d'une carte minimale représentant une image 2D étiquetée. Les pixels hachurés appartiennent à la région infinie. Le trait pointillé représente la relation $\beta_1$ permettant de "fermer" le bord. (a) La carte initiale représentant le bord supérieur et gauche (correspondante à l'image de la figure 5(a)). (b) Création et couture du carré représentant le pixel $(1, 1)$. (c) Les cellules sont localement simplifiées autour du pixel $(1, 1)$ lorsqu'elles vérifient les conditions. (d) Création et couture du carré représentant le pixel $(2, 1)$. (e) Carte obtenue après le traitement du pixel $(2, 1)$. (f) Création du carré associé à un pixel $(4, 1)$ appartenant à la dernière colonne. (g) Carte obtenue après le traitement du pixel $(4, 1)$. Comme l'image est plaquée sur un cylindre, cette configuration est équivalente à la carte présentée en (h).



**Algorithm 2:** Extraction incrémentale de la carte combinatoire d'une image.

**Entrée**: Une image étiquetée $I$ de $n_1 \times n_2$ pixels.
**Sortie**: La carte combinatoire correspondant à $I$.

```
1 last ← Construire le bord supérieur et gauche de l'image;
2 pour j=1 à n2+1 faire
      pour i=1 à n1+1 faire
3         tmp ← créer un carré correspondant au pixel (i,j) ;
          up ← ComputeUpFromLast(last);
          2-coudre ce carré avec les brins last et up ;
4         si last est dans la même région que β2(last) alors
              Supprimer l'arête incidente à last;
          si up est dans la même région que β2(up) alors
              Supprimer l'arête incidente à up;
5         si le sommet incident à last est de degré deux alors
              Supprimer le sommet incident à last;
          last ← tmp;
6 Calculer l'arbre d'inclusion des régions;
```

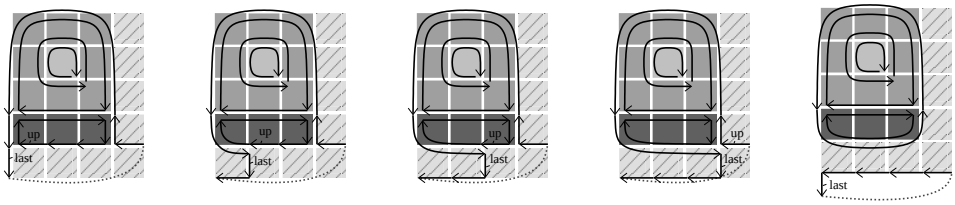
et à le relier par $\beta_2$ à la carte déjà construite (cf. figure 6(b)). Nous utilisons pour cela les deux brins particuliers *last* et *up*. Au cours de l'algorithme, la connaissance du brin *last* suffit car le brin *up* se calcule simplement à partir de ce brin en appliquant $\beta_1^{-1} \circ \beta_2$ jusqu'à trouver un brin 2-libre (c'est-à-dire un brin, sans brin en liaison par $\beta_2$).

Après avoir créé le carré représentant le pixel courant, la carte est localement simplifiée afin d'obtenir au final la carte minimale. Pour cela, il suffit d'étudier les cellules qui ont été fermées par l'ajout du carré : ce sont les deux arêtes incidentes aux brins *last* et *up* et le sommet incident au brin *last*. Pour savoir si une cellule doit être supprimée ou pas, nous utilisons les mêmes règles que pour l'algorithme naïf, à savoir les deux brins dans la même région pour la suppression d'arête et le sommet de degré deux pour la suppression de sommet. Après avoir simplifié localement la carte, il suffit de mettre à jour le brin *last* comme désignant le brin à gauche du prochain pixel, puis d'itérer le traitement sur ce prochain pixel.

La figure 6 présente quelques étapes de l'algorithme de construction incrémentale de la carte minimale représentant une image 2D étiquetée. Figure 6(a) est la carte initiale représentant le bord créé avant de commencer l'extraction. Figure 6(b) est la carte obtenue après création et couture du carré associé au pixel $(1, 1)$. Cette carte est localement simplifiée. Ici, les 2 arêtes sont conservées et le sommet est supprimé car il est de degré deux. Nous obtenons la carte présentée figure 6(c), avec les brins *last* et *up* correctement positionnés afin de traiter le prochain pixel. Le même processus est réalisé pour les autres pixels : création et couture d'un carré (figure 6(d) pour le pixel $(2, 1)$), puis simplification locale (figure 6(e) pour le pixel $(2, 1)$).

Nous pouvons vérifier sur cet exemple qu'il n'y a pas de traitement particulier pour les pixels de la dernière colonne de l'image grâce à la région infinie et au repliement de l'image sur un

FIGURE 7 – Déroulement de l'algorithme de construction incrémentale pour les pixels de la dernière ligne. (a) Carte obtenue après avoir traité tout les pixels de l'image et avant de commencer le balayage de la dernière ligne. (b) à (e) Cartes obtenues après le traitement de chacun des pixels de la dernière ligne. L'arête incidente à *last* est toujours supprimée car le pixel courant et son voisin de gauche appartiennent à la région infinie, et l'arête incidente à *up* n'est jamais supprimée, excepté pour le dernier pixel de la dernière ligne.(e) Résultat obtenu à la fin du balayage de l'image. La carte combinatoire minimale a été correctement construite et le bord initial est maintenant inutile.




(a) (b) (c) (d) (e)

cylindre. La figure 6(f) montre la configuration de la carte avant de traiter le pixel $(4, 1)$ qui est le dernier pixel de la première ligne. Comme ce pixel appartient à la région infinie, l'arête incidente au brin *up* est toujours supprimée et celle incidente au brin *last* jamais. Nous obtenons alors la configuration présentée figure 6(f) qui est équivalente à celle présentée figure 6(h) car le bord droit est identifié au bord gauche par le repliement de l'image.

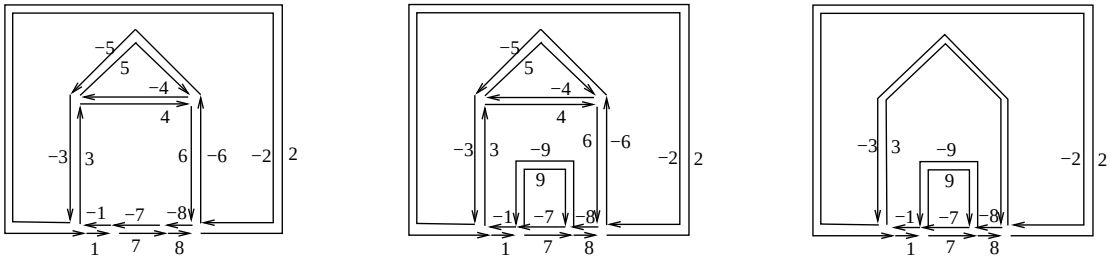
Après avoir traité tous les pixels (y compris ceux de la ligne supplémentaire en-dessous de l'image), on obtient la carte minimale représentant l'image (cf. les différentes étapes de l'algorithme d'extraction présentées figure 7). Le bord créé initialement est maintenant déconnecté du reste de la carte et se trouve en-dessous de l'image (cf. figure 7(e)). Il est alors inutile et peut être détruit.

Cet algorithme incrémental a une complexité linéaire en fonction du nombre de pixels de l'image car chaque opération peut être réalisée en un nombre borné d'opérations. De plus, il résout les inconvénients de l'algorithme naïf en évitant la création de la carte combinatoire complète. Par contre, il n'évite pas complètement la création momentanée d'éléments pour les détruire par la suite. Pour résoudre ce problème, un algorithme d'optimal d'extraction a été proposé dans [5]. Son principe est similaire à celui de l'algorithme incrémental mais il utilise la notion de *précode* (une configuration locale de l'image) et fixe le traitement à réaliser pour chaque configuration.

### 3.3 Édition d'une partition

Les méthodes vues en section 3.2 permettent de coder une partition par des cartes combinatoires à l'aide d'une fonction indiquant la région d'appartenance de chaque pixel. Toutefois, ce type d'information n'est pas toujours aisément accessible. De plus, le codage d'une parti-

FIGURE 8 – Découpe (a) et (b) d'une face et fusion (c) de deux faces.




(a) Découpe (étape 1) (b) Découpe (étape 2) (c) Fusion

tion à l'aide de cartes combinatoires n'est souvent que la première étape d'un processus visant à analyser l'image (Section 1). L'étude d'une première partition à l'aide des cartes combinatoires permet d'affiner cette première description en découpant davantage certaines régions ou au contraire en fusionnant des régions. Afin de continuer l'analyse de l'image après chaque opération de découpage ou de fusion, il convient de mettre en œuvre des méthodes efficaces de mise à jour de la partition après chacune de ces opérations.

Un cas simple d'insertion et de suppression est représenté sur la figure 8. Le découpage d'un face finie (figure 8(a) et (b)) se déroule en deux étapes. La première étape (figure 8(a)) consiste à insérer un ou deux sommets au niveau de l'intersection entre l'arête à insérer et les arêtes existantes. Dans ce cas, cette opération se traduit par le découpage de l'arête $(1, -1)$ en trois sous-arêtes $((1, -1), (7, -7)$ et $(8, -8))$. L'arête $(9, -9)$ est alors insérée au niveau des sommets ainsi créés (figure 8(b)), les permutations $\beta_1$ et $\beta_2$ sont modifiées en conséquence. La fusion de deux faces finies consiste à supprimer une des arêtes qui les relient. Dans le cas de la fusion de la façade et du toit de la maison (figure 8(c)) cette opération se traduit par la suppression de l'arête $(4, -4)$. Les deux sommets incidents à cette arête deviennent alors de degré 2 et sont considérés comme redondants. La suppression de ces sommets est équivalente à la fusion des brins $((-3, -5, -6)$ et $(6, 5, 3)$ en deux brins 3 et $-3$, 2-cousus. Ces opérations sur la carte combinatoire doivent être reportées sur le modèle géométrique associé. De plus, ces opérations impliquent la mise à jour des relations d'inclusions. Une description plus détaillée de ces opérations de découpage et de fusion peut être trouvée dans [1, 3].

# 4 Cartes pour l'analyse d'images 3D

Nous étudions maintenant la manière de représenter et de manipuler une image 3D étiquetée à l'aide d'une carte combinatoire. Nous allons principalement montrer comment les concepts présentés en 2D, section 3, s'étendent en 3D, mais sans détailler complètement le modèle.

## 4.1 Codage de la topologie et de la géométrie d'une image

Étant donnée une image 3D étiquetée, nous souhaitons représenter chaque région contenue dans l'image par un volume dans la carte, puis les différentes relations d'adjacences :

— une face par surface maximale de contact entre deux régions ;

— une arête par courbe maximale de contact entre deux faces ;
— un sommet par point de jonction entre plusieurs arêtes.

Ces concepts correspondent à l'extension immédiate des notions de segments et de nœuds utilisés en 2D. Le point important est à nouveau de vouloir représenter l'information de manière compacte en ayant une seule cellule dans la carte associée à chaque élément de contact dans l'image.

Le problème de cartes composées de plusieurs composantes connexes, déjà présenté en 2D, se pose également en 3D lorsque nous souhaitons représenter un objet comportant un trou (par exemple un cube dans lequel on enlève une boule). Les deux solutions présentées en 2D s'étendent sans difficulté particulière en 3D. Il est possible d'utiliser un arbre d'inclusion des régions donnant pour chaque région l'ensemble des régions totalement incluses, ou d'utiliser des éléments fictifs (ici des faces fictives) conservant la carte connexe.

La solution à l'aide d'éléments fictifs est plus complexe à mettre en œuvre que celle à l'aide d'un arbre d'inclusion car l'ajout d'éléments fictifs nécessite de rajouter des traitements particuliers, entre autre afin de garantir l'obtention d'une représentation minimale en nombre de cellules. Par contre, il faut noter que contrairement à la 2D où un arbre d'inclusion permettait de retrouver directement toutes les informations topologiques entre les différentes composantes connexes, cela n'est plus le cas en 3D. En effet, lorsque deux objets 3D sont complètement inclus dans une région englobante, l'arbre d'inclusion ne contient pas d'information sur la position relative de ces deux objets. Il en résulte que l'on ne saura pas différencier de manière immédiate deux tores côte à côte de deux tores entrelacés.

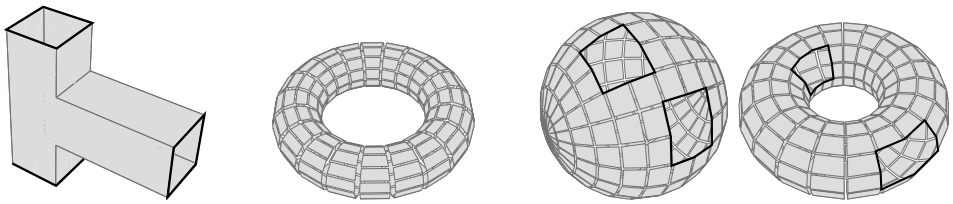
Un deuxième type de problème peut se poser en 3D, lorsque l'on souhaite représenter une face comportant plusieurs bords (par exemple si une sphère est en contact avec trois volumes distincts et non adjacents entre eux. Dans ce cas, chaque surface de contact est représentée par un bord et la sphère contient plusieurs bords). Dans ce cas, la solution à base d'inclusion n'est pas adaptée pour trois raisons :

1. la relation entre les différents bords de la face n'est pas forcément une relation d'inclusion (cf. exemple figure 9(a)) ;
2. les faces sans bord ne seraient pas du tout représentées dans la carte (cf. exemple figure 9(b)) ;
3. la donnée des différents bords de la face ne contient pas toutes les informations topologiques. En effet, deux faces peuvent être identiques pour leurs bords mais pas topologiquement équivalentes (cf. exemple figure 9(c) où les surfaces n'ont pas le même genre).

La résolution de ces problèmes peut se faire de manière directe en choisissant la solution déjà présentée en 2D et basée sur les éléments fictifs. Nous relions chaque bord d'une même face par des arêtes fictives. En conservant les arêtes fictives, chaque face est homéomorphe à un disque. De plus, le genre de chaque surface est correctement conservé dans la carte et peut être retrouvé en comptant le nombre de cellules et en utilisant la formule d'Euler.

Concernant la géométrie, la solution présentée en 2D s'étend immédiatement en 3D. Il est en effet possible de représenter la géométrie de chaque élément dans une grille d'éléments intervoxels et de faire le lien entre chaque brin de la carte et un triplet (pointel, lignel, surfel).

FIGURE 9 – Le problème de représentation de faces selon leur nombre de bords (en noir et épais sur les figures). (a) Une face comportant 3 bords, sans relation d'inclusion entre eux. (b) Une face sans bord. (c) Deux surfaces avec les mêmes bords mais pas la même topologie.



(a) (b) (c)

### 4.2 Construction du modèle à partir d'une image

Les algorithmes présentés section 3.2 pour construire la carte combinatoire associée à une partition d'image 2D s'étendent directement afin de construire la carte combinatoire minimale représentant une image 3D étiquetée.

Le premier algorithme naïf s'étend en ajoutant une passe supplémentaire de suppression pour la dimension additionnelle. Le second algorithme incrémental s'étend de la même manière. Nous avons maintenant trois brins particulier, *last*, *up* et *behind*. Pour chaque voxel, un nouveau cube est créé et 3-cousu par trois de ses faces à l'aide de ces brins. Ensuite, il faut tester si ces trois faces doivent être supprimées ou non, si les trois arêtes incidentes à ces faces doivent être simplifiées et si le sommet incident à ces trois arêtes doit être simplifié.

Il faut noter que la conservation des arêtes fictives s'effectue durant l'extraction en testant, avant de supprimer une arête, si cela entraînerait une déconnexion. De plus, ces arêtes engendrent des traitements spécifiques selon les différentes configurations.

### 4.3 Édition d'une partition

De manière identique à la dimension 2, il est nécessaire de fournir des outils permettant de manipuler la carte combinatoire après son extraction. Les opérations principales sont à nouveau les opérations de découpe ou de fusion. Il faut noter que ces opérations sont ici plus délicates à mettre en œuvre de par la présence des arêtes fictives (cf. par exemple [6]).

## 5 Conclusion

Ce chapitre nous a permis de découvrir comment utiliser les cartes combinatoires pour l'analyse d'images. Ce type de modèle trouvera son intérêt dans de nombreux algorithmes de traitement d'images, par exemple pour calculer des mesures sur les régions de l'image (comme des invariants topologiques ou de régions, mais également des attributs de contour. En effet, la carte combinatoire permet de retrouver efficacement à la fois les informations topologiques sur les cellules composant l'image mais également de parcourir géométriquement le bord de ces régions.

Les cartes combinatoires pourront aussi servir lors d'algorithmes de modification d'une partition afin de rajouter un contrôle sur l'impact de ces modifications sur la topologie des objets (par exemple en imagerie médicale, cf. chapitre 19). Enfin, des travaux plus récents sur l'utilisation des cartes combinatoires pour coder des hiérarchies [2, 9] ouvrent de nombreuses perspectives dans des objectifs d'applications multi-échelles.

## Références


[1] Jean Pierre Braquelaire and Jean Philippe Domenger. Representation of segmented image with discrete geometric maps. *Image and Vision Computing*, 17 :715–735, 1999.

[2] Luc Brun and Walter Kropatsch. Receptive fields within the combinatorial pyramid framework. *Graphical Models*, 65 :23–42, 2003.

[3] Luc Brun, Myriam Mokhtari, and Jean Philippe Domenger. Incremental modifications on segmented image defined by discrete maps. *Journal of Visual Communication and Image Representation*, 14 :251–290, 2003.

[4] R. Cori. Un code pour les graphes planaires et ses applications. In *Astérisque*, volume 27. Soc. Math. de France, Paris, France, 1975.

[5] G. Damiand, Y. Bertrand, and C. Fiorio. Topological model for two-dimensional image representation : definition and optimal extraction algorithm. *Computer Vision and Image Understanding*, 93(2) :111–154, February 2004.

[6] G. Damiand and P. Resch. Split and merge algorithms defined on topological maps for 3d image segmentation. *Graphical Models*, 65(1-3) :149–167, May 2003.

[7] Walter G. Kropatsch. Properties of pyramidal representations. *Computing, Supplementum : Theoretical Foundations of Computer Vision*, No. 11 :pp. 99–111, 1996.

[8] P. Lienhardt. Topological models for boundary representation : a comparison with n-dimensional generalized maps. *Commputer Aided Design*, 23(1) :59–82, 1991.

[9] C. Simon, G. Damiand, and P. Lienhardt. nd generalized map pyramids : definition, representations and basic operations. *Pattern Recognition*, 39(4) :527–538, April 2006.