# Opérations et Algorithmes pour la Segmentation Topologique d'Images 3D

Alexandre Dupas

# Thèse

pour l'obtention du grade de

DOCTEUR DE L'UNIVERSITÉ DE POITIERS

**Faculté des Sciences Fondamentales et Appliquées**
(Diplôme National - Arrêté du 7 Août 2006)

*École Doctorale* : **Sciences et Ingénierie pour l'Information**
*Secteur de Recherche* : **Informatique et Applications**

présentée par :

Alexandre DUPAS

# Opérations et Algorithmes pour la Segmentation Topologique d'Images 3D

Directeurs de thèse :

Guillaume DAMIAND

Pascal LIENHARDT

Soutenue le 25 novembre 2009 devant le jury composé de :

Achille Braquelaire, Professeur, Université de Bordeaux I, LaBRI ..... Président du Jury

Michel Couprie, Professeur, ESIEE, Laboratoire d'Informatique Gaspard-Monge ..... Rapporteur
Annick Montanvert, Professeur, Université Pierre Mendès France, Gipsa-lab ..... Rapporteur

Yves Bertrand, Professeur, Université de Poitiers, XLim/SIC ..... Examinateur

Guillaume Damiand, Chargé de Recherche, CNRS, LIRIS ..... Directeur de Thèse
Pascal Lienhardt, Professeur, Université de Poitiers, XLim/SIC ..... Directeur de Thèse

# TABLE DES MATIÈRES

# INTRODUCTION

Les techniques d'imagerie permettant l'acquisition et le traitement des images ont connu dans les dernières décennies des progrès considérables. L'avènement des systèmes de capture d'images 3D par tomographie, technique qui consiste à reconstruire le volume d'un objet à partir d'une série de mesures réalisées depuis l'extérieur de l'objet, a permis l'étude de phénomènes jusque-là invisibles. La tomographie a des applications dans de nombreux domaines notamment en géologie, dans l'étude des matériaux et surtout en médecine. De nombreuses techniques de tomographie existent, les plus connues étant les scanners et l'Imagerie par Résonnance Magnétique nucléaire (IRM). Elles sont très utilisées à la fois pour l'aide au diagnostic mais aussi pour la recherche médicale. Dans ce contexte, les besoins pour des outils d'analyse d'images 3D sont croissants. Les appareils fournissant de plus en plus d'informations, le traitement complet des images par les médecins spécialistes - le traitement manuel - n'est plus envisageable d'où la nécessité de développer des algorithmes automatisés de traitement d'images.

L'une des opérations les plus importantes du traitement d'images est la segmentation. Elle a pour objectif de regrouper les voxels homogènes d'une image de manière à former des régions. La segmentation est généralement utilisée en préalable à d'autres opérations comme la reconnaissance de formes ou la visualisation. Il existe de très nombreuses techniques de segmentation principalement définies sur les images 2D, comme les photos, mais également sur les vidéos ou les images 3D. Parmi les différentes techniques, nous retrouvons quelques grandes catégories d'algorithmes. Celles qui nous intéressent particulièrement se classent en deux groupes : d'une part les approches de segmentation dérivées de la division-fusion et d'autre part les approches de segmentation basées sur les modèles déformables.

La segmentation par division-fusion modifie la partition en régions de l'image. Par un processus de fusion des régions adjacentes ou de division d'une région, l'algorithme fait évoluer la partition jusqu'à obtenir la segmentation optimale. Il existe deux variantes principales pour la segmentation par division-fusion. La première, dite de bas en haut (*bottom-up* en anglais), utilise comme point de départ une partition de l'image dans laquelle chaque voxel se trouve dans une région distincte. Seule la fusion des régions est alors utilisée afin d'agglomérer les régions adjacentes pour obtenir la partition finale. La seconde variante, dite de haut en bas (*top-down* en anglais), utilise une partition initiale composée d'une seule région contenant toute l'image. La division est appliquée successivement aux régions jusqu'à obtenir la partition finale. L'approche mixte est la méthode générale qui consiste à alterner les phases de division et les phases de fusion de régions afin d'obtenir la partition à convergence. Les outils de segmentation basés sur les modèles déformables considèrent généralement la partition de l'image comme étant composée de deux régions : l'une représentant l'objet et l'autre le fond. Le processus de segmentation déforme alors une partition initiale afin de l'adapter plus précisément à l'image d'après un critère énergétique.

Dans ces deux approches, la segmentation est dépendante d'un critère qui spécifie la manière dont l'image est partitionnée en régions. Les critères classiques utilisent la couleur ou l'intensité des éléments pour établir les regroupements. Cependant d'autres critères sont également définis comme par exemple les critères fréquentiels, géométriques ou de texture. Nous nous intéressons particulièrement à la catégorie des critères topologiques qui se basent sur la topologie des objets à segmenter. La topologie est une branche des mathématiques qui consiste à étudier les espaces et les relations entre ces espaces. Dans le contexte de l'imagerie, la topologie est utilisée pour caractériser les régions (en observant par exemple le nombre de cavités ou d'anses que possède la région) ou les relations entre les régions (en observant par exemple si deux régions sont adjacentes). Ces informations sont très utilisées notamment dans le cadre de la segmentation d'images médicales. En effet, les systèmes d'acquisition des organes internes du corps humain ne permettent pas d'avoir une grande résolution. Les organes représentés dans ces images ont de nombreux défauts topologiques. La topologie permet d'anticiper ou de corriger ces défauts lorsque nous segmentons l'image pour en extraire l'organe.

Afin d'utiliser ces informations de manière efficace dans le cadre du traitement d'images, de nombreux modèles représentant la topologie d'une partition ont été définis. Les premiers modèles sont apparus à la fin des années soixante-dix pour représenter les régions d'une image et les relations entre les régions comme l'adjacence. L'un de ces modèles est le graphe d'adjacence des régions [Ros74] qui utilise un graphe non orienté pour représenter la partition de l'image. Les nœuds du graphe représentent les régions et les arêtes symbolisent les relations d'adjacence entre les régions. Cette représentation, si elle est bien adaptée pour résoudre des problèmes de segmentation de type *bottom-up*, ne conserve pas toutes les informations sur les relations entre les régions. Par exemple, la relation d'imbrication n'est pas donnée directement par le modèle. Elle ne fait pas non plus de différence entre l'adjacence simple de deux régions et les adjacences multiples. Aussi de nombreux modèles sont apparus afin de combler les lacunes de leurs prédécesseurs. Nous trouvons parmi ces modèles différents résultats de travaux portant sur une représentation par graphe de la partition. Le multigraphe d'adjacence introduit la représentation des adjacences multiples entre les régions en utilisant une arête pour chaque adjacence entre deux régions. Les graphes duaux [KM95] sont une autre évolution de ce modèle. Ils permettent de retrouver en plus la relation d'imbrication entre les régions. Une famille de modèles est apparue par la suite : les modèles basés sur les cartes combinatoires. Une carte combinatoire est un modèle mathématique utilisé pour représenter une subdivision de l'espace. Les cartes combinatoires sont utilisées pour représenter la partition d'une image en régions. Différents travaux ont porté sur la définition d'un tel modèle, avec notamment les cartes discrètes [BD96], les graphes topologiques des frontières [Fio96] et les *GeoMap* [MK05] qui permettent de représenter des partitions d'images 2D. Par la suite, deux approches pour la représentation de partitions d'images 3D ont vu le jour en parallèle. D'une part, [Des01] qui propose une définition basée sur une 3-carte et utilisant l'intervoxel pour représenter la géométrie. Cependant le codage de la 3-carte utilisant la géométrie implique que deux structures qui sont topologiquement identiques peuvent avoir deux représentations qui ne soient pas isomorphes. D'autre part, [Dam01] qui après avoir proposé la définition d'une carte topologique en 2D a étendu son modèle en 3D en donnant un modèle représentant toutes les cellules d'une partition d'une image en régions et ainsi propose une solution qui représente toutes les informations de la partition. Dans ce modèle, deux partitions topologiquement identiques ont une représentation isomorphe.

L'objectif général de ce travail est de développer, dans les cartes topologiques 3D, les outils permettant la manipulation du modèle afin de définir des opérations de traitement d'images. Nous nous intéressons particulièrement aux processus de segmentation d'images et aux avantages que l'utilisation des cartes topologiques apporte à cette opération. Nous illustrons ces avantages en définissant de nouveaux critères et algorithmes de segmentation qui prennent en compte des informations liées à la topologie de la partition.

Nous étudions pour cela les méthodes de segmentation existantes et nous nous intéressons plus particulièrement aux algorithmes dérivant des méthodes de division-fusion [HP76]. Pour permettre ce type d'approche, nous commençons par définir les deux opérations de base que sont la fusion de régions et la division d'une région. Comme différentes approches sont possibles pour mettre en œuvre ces deux opérations, nous cherchons à explorer en détail plusieurs variantes afin de proposer les outils les plus adaptés aux problèmes rencontrés par les utilisateurs de ces opérations. Nous étudions ainsi deux approches de la fusion de régions, la méthode locale et la méthode globale, qui servent respectivement à fusionner ponctuellement des régions (par exemple dans le cadre d'une utilisation interactive), et à fusionner beaucoup de régions en même temps comme il est couramment nécessaire de le faire dans le cadre de processus automatisés. Nous développons également deux approches pour la division d'une région : d'une part l'éclatement d'une région qui conduit à créer une région par voxel, et d'autre part la division d'une région par un guide, une approche plus générale permettant de définir précisément comment une région doit être découpée à l'aide d'une ou plusieurs surfaces.

Nous ajoutons également la possibilité de déformer le modèle des cartes topologiques en cherchant à contrôler les changements de topologie dans l'image. Nous nous intéressons notamment à une opération de déformation qui préserve la topologie de la partition. Pour cela, nous étudions la notion de points simples [Ros70] qui permettent de contrôler la topologie en basculant des voxels entre l'objet et le fond dans une partition binaire d'une image. Cependant, les cartes topologiques offrent une partition plus détaillée de l'image en représentant plusieurs régions. Nous définissons donc la notion de points simples multilabels qui permettent de déformer une partition multirégions sans en changer la topologie.

Au fur et à mesure que les opérations de modification sont introduites dans le modèle des cartes topologiques, nous mettons en œuvre différents algorithmes de segmentation qui utilisent les opérations réalisées. Ces travaux préliminaires permettent de montrer comment nos outils peuvent être utilisés en segmentation. Nous réalisons pour cela diverses opérations avec dans un premier temps l'étude d'une segmentation de type *bottom-up* qui consiste à agglomérer les régions selon un critère. Afin de définir ce critère, nous utilisons les travaux de [FH98] qui se basent sur la notion de contraste. Nous étudions par la suite un nouveau critère qui tire parti des informations supplémentaires apportées par la carte topologique. Ce nouveau critère utilise des invariants topologiques, les nombres de Betti, qui de manière intuitive comptent le nombre de composantes connexes, de tunnels et de cavités d'une région. Nous développons les algorithmes permettant de calculer les nombres de Betti pour les régions représentées par une carte topologique et nous montrons comment intégrer ce critère à l'opération de segmentation d'image.

Nous développons également une méthode de déformation de la partition basée sur les points simples multilabels. Cette déformation est guidée par des énergies basées sur les valeurs des images et sur un critère géométrique lié à la surface des régions. Nous étudions la mise en œuvre de quelques méthodes trouvées dans la littérature à l'aide de la carte topologique et finalement nous proposons une méthode de déformation adaptée au contexte des partitions contenant plusieurs régions.

Enfin, nous étudions la réalisation d'une méthode de segmentation complète qui met en œuvre les outils de modification développés pour le modèle des cartes topologiques. Dans ce contexte, nous avons étudié un problème actuel dans le domaine de la segmentation d'images médicales : le traitement des images par tomographie à émission de positons représentant l'activité métabolique du cerveau des patients. Plus précisément, nous cherchons à résoudre la segmentation de tumeurs cérébrales dans ce type d'images avec comme objectif de pouvoir réaliser des études quantitatives sur la partition produite.

Ce manuscrit est divisé en trois grandes parties.

La Partie I décrit les différents résultats qui ont précédé les travaux de cette thèse. Dans le Chapitre 1, nous décrivons les différents modèles de représentation d'images ayant conduit à la définition des cartes topologiques. Le Chapitre 2 donne la définition des cartes topologiques et des opérations disponibles sur cette structure de données. Dans le Chapitre 3, nous discutons de la segmentation d'images en général

en présentant quelques algorithmes qui sont représentatifs des travaux que nous souhaitons réaliser avec les cartes topologiques. Nous évoquons en même temps les différents types de critères utilisés dans la segmentation. Nous mettons ainsi l'accent sur les critères topologiques en expliquant ce qu'est la topologie dans le contexte de l'analyse d'images.

La Partie II présente les différentes opérations sur les cartes topologiques qui ont été définies au cours de la thèse. Ces opérations sont les outils nécessaires à la réalisation d'algorithmes de segmentation. Nous présentons trois types d'opérations : la fusion de régions (Chapitre 4), la division de régions (Chapitre 5) et une opération de déformation d'une partition (Chapitre 6). Nous donnons pour chaque opération les algorithmes permettant d'accomplir la tâche selon différentes approches dans les cartes topologiques. Nous présentons également quelques expérimentations permettant de comparer les différentes approches entre elles afin d'évaluer ces opérations dans le cadre de la segmentation.

La Partie III présente les applications de segmentation définies à partir des opérations de modification. Dans le Chapitre 7, nous introduisons une opération de segmentation *bottom-up* utilisant un critère basé sur la notion de contraste qui donne un premier résultat de segmentation à l'aide des cartes topologiques. Dans le Chapitre 8, nous discutons de l'introduction d'un critère topologique dans cette segmentation. Enfin, le Chapitre 9 présente la chaîne de traitement d'images que nous avons mis en place pour résoudre un problème de segmentation sur des images médicales.

Nous terminons ce manuscrit par la conclusion de ce travail et nous discutons des perspectives pour de futurs travaux dans la continuité de cette thèse.

# Première partie

# Travaux Existants

CHAPITRE 1

# REPRÉSENTATION D'IMAGES

Dans le cadre du traitement d'images, le problème de la représentation des données est un point essentiel. Parmi les différents modèles de représentation d'images, nous pouvons distinguer en particulier les modèles considérant uniquement les données de l'image (pixels ou voxels) qui sont utilisés dans de nombreuses opérations de traitements d'images. Cependant, ces modèles ne représentent pas efficacement toutes les informations d'une image. La notion d'appartenance d'un élément de l'image à une zone, ou région, n'est pas mise en évidence par ces types de modèles. C'est pourquoi les modèles topologiques sont apparus dans les années 1970 avec l'apparition du premier modèle : les graphes d'adjacences de régions [Ros74]. Cependant, les graphes d'adjacences de régions ne représentant pas toutes les informations souhaitées pour mettre en œuvre des opérations d'analyse d'images, divers modèles ont été proposés de manière à les compléter et représenter ainsi de manière exhaustive les informations d'une image.

## 1.1 Modèles discrets

Les données manipulées en traitement d'images sont généralement des images numériques, terme qui désigne les images acquises, traitées ou stockées sous forme binaire. Une image est échantillonnée dans un sous-espace de $\mathbb{Z}^n$ où $n$ représente la dimension de l'image. Dans le cadre de ce travail, les données sont principalement des images 3D. Nous utilisons les images 2D pour illustrer certains algorithmes ou modèles.

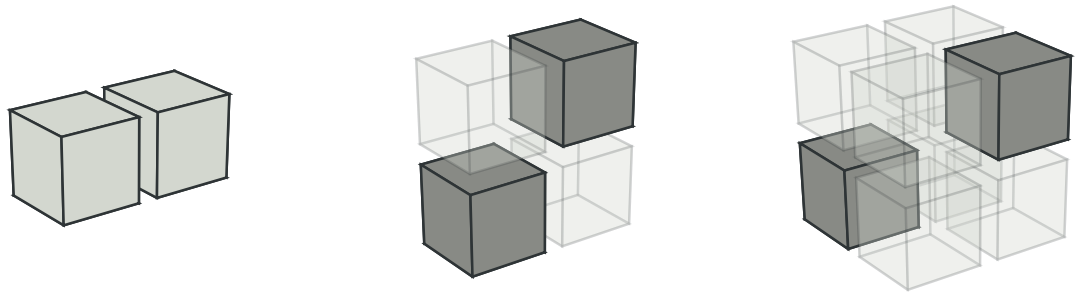
### 1.1.1 Image 3D

Une image 3D est un sous-espace de $\mathbb{R}^3$ partitionné en cubes (avec une correspondance naturelle avec $\mathbb{Z}^3$) où chaque cube de ce sous-espace est appelé un *voxel* (*volume element* en anglais). Ainsi une image $I$ est un couple $(I_d, I_f)$, où $I_d$ est un ensemble de voxels (le domaine de l'image) et $I_f$ est une application de $I_d$ vers un ensemble de couleurs ou d'intensités (les valeurs de l'image). Les voxels sont désignés par leurs coordonnées dans $\mathbb{Z}^3$. Le nombre de voxels est noté $|I_d|$, $|E|$ est la notation pour le nombre d'élément de $E$.

Les relations d'adjacences sont définies entre les voxels d'une image, deux voxels $v = (i, j, k)$ et $v' = (i', j', k')$ sont :

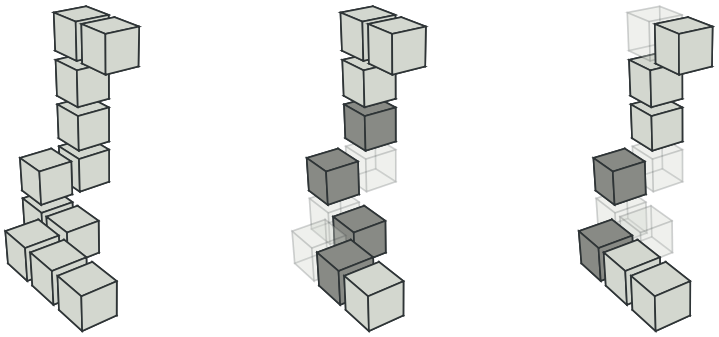
– 6-adjacents si

$$|i - i'| + |j - j'| + |k - k'| = 1;$$



(a) (b) (c)

FIG. 1.1: Relation d'adjacence entre deux voxels. Les voxels transparents sont dessinés pour mieux visualiser la configuration autour des deux voxels gris. Les différentes configurations représentent : (a) la 6-adjacence ; (b) la 18-adjacence ; (c) et la 26-adjacence.



(a) (b) (c)

FIG. 1.2: Les différentes chemins de voxels sont : (a) un 6-chemin ; (b) un 18-chemin ; (c) et un 26-chemin.

– 18-adjacents si

$$|i-i'|+|j-j'|+|k-k'| \leq 2 \text{ et } \max(|i-i'|,|j-j'|,|k-k'|) = 1;$$

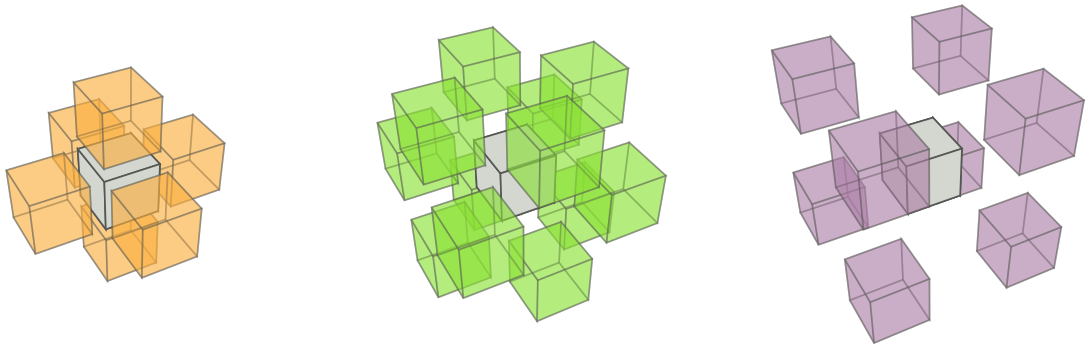
– 26-adjacents si

$$\max(|i-i'|,|j-j'|,|k-k'|) = 1.$$

Ces définitions, illustrées dans la Figure 1.1, étendent les notions de 4-adjacence et de 8-adjacence que nous retrouvons entre les pixels (*picture element* en anglais), les éléments de base des images 2D. D'une manière intuitive, deux voxels sont 6-adjacents si leurs cubes respectifs se touchent par une face. Ils sont 18-adjacents si ils se touchent par une face ou une arête. Enfin deux voxels sont 26-adjacents si ils se touchent par une face, une arête ou un sommet du cube.

L'adjacence permet de définir la notion de $\alpha$-chemin qui est illustrée dans la Figure 1.2. Un $\alpha$-chemin entre deux voxels $v_1$ et $v_2$ est une suite de voxels allant de $v_1$ à $v_2$ de sorte que chaque couple de voxels consécutifs soit $\alpha$-adjacent. Dans la Figure 1.2b, les voxels dessinés en gris foncés sont uniquement 18-adjacents, il s'agit donc d'un 18-chemin. Dans la Figure 1.2c, les voxels dessinés en gris foncés sont 26-adjacents, il s'agit d'un 26-chemin.

Un ensemble de voxels $C$ est dit $\alpha$-connexe si et seulement si il existe un $\alpha$-chemin entre tous les couples de voxels appartenant à $C$ de sorte que tous les voxels du chemin appartiennent à $C$.



(a) (b) (c)

FIG. 1.3: Les différents $\alpha$-voisinages d'un voxel : (a) les 6 voxels faisant partie du 6-voisinage du voxel central ; (b) les 12 voxels faisant partie du 18-voisinage du voxel central sans les voxels du 6-voisinage ; (c) les 8 voxels faisant partie du 26-voisinage du voxel central sans les voxels du 18-voisinage.

L'$\alpha$-voisinage d'un voxel $v$, noté $N^*_\alpha(v)$, est l'ensemble des voxels qui sont $\alpha$-adjacents à $v$. L'union d'un voxel $v$ et de son $\alpha$-voisinage est écrit : $N_\alpha(v) = N^*_\alpha(v) \cup \{v\}$. La Figure 1.3 montre les voxels appartenant aux différents voisinages : $N_6(v)$ dans la Figure 1.3a, $N_{18}(v) \setminus N^*_6(v)$ dans la Figure 1.3b et $N_{26}(v) \setminus N^*_{18}(v)$ dans la Figure 1.3c.

Dans la suite de ce travail, nous présentons des modèles ou des algorithmes qui fonctionnent aussi bien sur des images 2D que des images 3D. Les notions d'adjacence que nous utilisons en 2D sur les pixels sont les notions de 4-adjacence et de 8-adjacence. De ces notions dérivent de manière similaire les notions de connexité et de chemin en 2D.

### 1.1.2 Ordre sur les voxels d'une image

Le parcours des voxels de l'image est réalisé par un balayage de l'image ligne par ligne (*scanline* en anglais). Nous utilisons l'ordre induit par ce balayage pour définir un ordre sur les voxels d'une image. Nous utilisons un balayage qui commence par le voxel (0,0,0) puis incrémente sa valeur, d'abord selon l'axe des abscisses, puis l'axe des ordonnées et enfin l'axe de la hauteur (voir Algorithme 1) : il s'agit de l'ordre lexicographique sur les voxels.

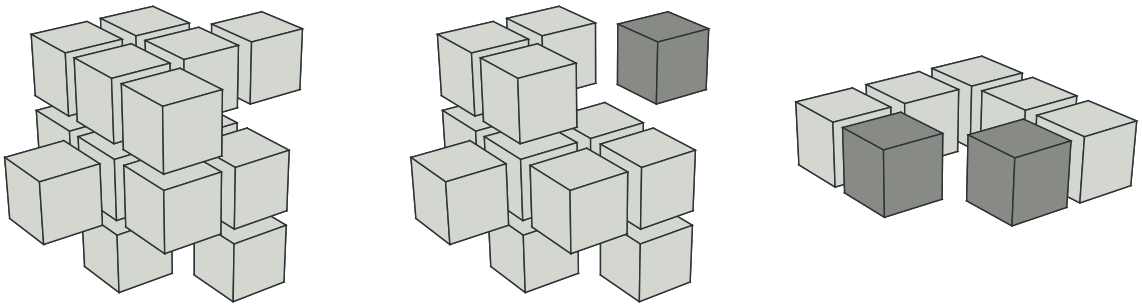
**Algorithme 1** : Balayage des voxels d'une image

**Données** : Image 3D $I = (I_d, I_f)$ de taille $(X, Y, Z)$
**Résultat** : Parcours des voxels dans l'ordre défini par le balayage

```
pour chaque z ∈ [0, Z[ faire
    pour chaque y ∈ [0, Y[ faire
        pour chaque x ∈ [0, X[ faire
            v = I_d(x, y, z);
            // Traitement du voxel v
```

Ainsi avec cet ordre, un voxel $v_1$ de coordonnées $(x_1, y_1, z_1)$ est plus petit qu'un voxel $v_2$ de coordonnées $(x_2, y_2, z_2)$ si et seulement si :

$$(z_1 < z_2) \vee ((z_1 = z_2) \wedge ((y_1 < y_2) \vee (y_1 = y_2) \wedge (x_1 \leq x_2))).$$



(a) (b) (c)

FIG. 1.4: Régions composées de voxels : (a) exemple d'une région valide contenant un ensemble 6-connexe de régions ; (b) exemple d'un ensemble de voxels ne pouvant pas composer une région : le voxel dessiné en gris foncé n'est pas 6-adjacent à un voxel gris clair. (c) exemple d'une région valide : les deux voxels dessinés en gris foncé sont 18-adjacents mais il existe un 6-chemin de voxels de la région permettant de les relier.

### 1.1.3 Notion de régions dans les images

Dans le cadre du traitement d'images, la notion de région (voir Définition 1) est très importante afin de décrire une zone homogène d'une image. Un exemple de région est illustrée par la Figure 1.4.

**Définition 1** (Région)**.** *Une région dans une image 3D est un ensemble* 6*-connexe de voxels qui sont homogènes selon un critère.*

Les critères permettant de définir l'homogénéité d'une région sont extrêmement variés : ils peuvent dépendre de la couleur ou de l'intensité des voxels mais aussi de leur localisation. Nous utilisons généralement un prédicat, parfois nommé oracle, qui détermine si deux voxels appartiennent ou non à une même région.

Nous définissons en même temps la notion de partition d'une image en régions comme étant un pavage de l'image par des régions. Il s'agit de représenter par des régions les différentes zones de l'image, les régions ne pouvant pas se chevaucher.

Dans ce travail, nous noterons les régions par la lettre $r$ et l'ensemble des voxels représentant une région $r$ est noté *voxels*$(r)$. Le premier voxel d'une région $r$, noté *premierVoxel*$(r)$, est le plus petit voxel de la région $r$ : *premierVoxel*$(r) = min($*voxels*$(r))$.

La notion d'$\alpha$-adjacence entre les voxels est étendue aux régions par la Définition 2. Lorsque le voisinage n'est pas précisé, nous utilisons le 6-voisinage et donc la notion de 6-adjacence pour parler de deux régions adjacentes (voir Figure 1.5).

**Définition 2** (Adjacence de régions)**.** *Deux régions* $r_1$ *et* $r_2$ *sont dites* $\alpha$*-adjacentes si il existe deux voxels* $v_1 \in$ voxels$(r_1)$ *et* $v_2 \in$ voxels$(r_2)$ *tels que* $v_1$ *et* $v_2$ *soient* $\alpha$*-adjacents.*

Pour généraliser les algorithmes sur les images, la Définition 3 introduit une région spéciale appelée région infinie qui entoure l'image. La région infinie est composée de l'ensemble des voxels n'appartenant pas à l'image.

**Définition 3** (Région infinie)**.** *La région infinie, notée* $r_0$*, est le complémentaire de l'image dans* $\mathbb{Z}^3$.

À l'aide de la définition de la région infinie, nous définissons la relation d'imbrication entre deux régions par la Définition 4. Cette relation est parfois nommée relation d'inclusion entre les régions (par exemple [Dam08a]).

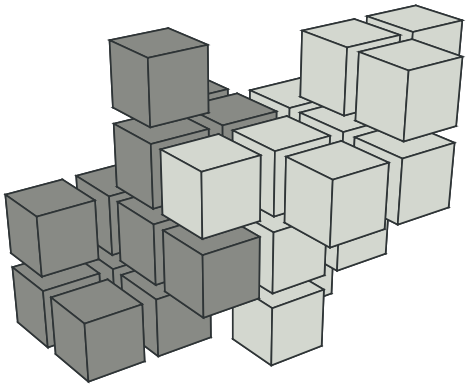


FIG. 1.5: Exemple de deux régions 6-adjacentes : un voxel de la région dessinée en gris foncé est 6-adjacent à un voxel de la région dessinée en gris clair.

**Définition 4** (Imbrication de régions)**.** *Une région $r_j$ est imbriquée dans une région $r_i$ si tous les* 18*-chemins allant d'un voxel de $r_j$ à un voxel de $r_0$ contiennent un voxel de $r_i$.*

La relation d'imbrication est transitive. Si une région $r_3$ est imbriquée dans une région $r_2$ et que la région $r_2$ est elle-même imbriquée dans la région $r_1$ alors la région $r_3$ est imbriquée dans la région $r_1$. À partir de cette remarque, la Définition 5 présente la notion d'imbrication directe ou de premier niveau. Par la suite, lorsque nous discutons de l'imbrication entre les régions, nous utilisons uniquement la relation d'imbrication directe.

**Définition 5** (Imbrication directe de régions)**.** *Une région $r_j$ est directement imbriquée dans une région $r_i$ si $r_j$ est imbriquée dans $r_i$ et s'il n'existe aucune région $r_k$ telle que $r_j$ soit imbriquée dans $r_k$ et $r_k$ soit imbriquée dans $r_i$.*

L'ordre des voxels de l'image est utilisé afin de définir un ordre sur les régions. La Définition 6 indique que les régions sont ordonnées par l'ordre de leur premier voxel dans le balayage de l'image.

**Définition 6** (Ordre des régions)**.** *Les régions sont ordonnées en utilisant l'ordre sur le premier voxel de chaque région :*

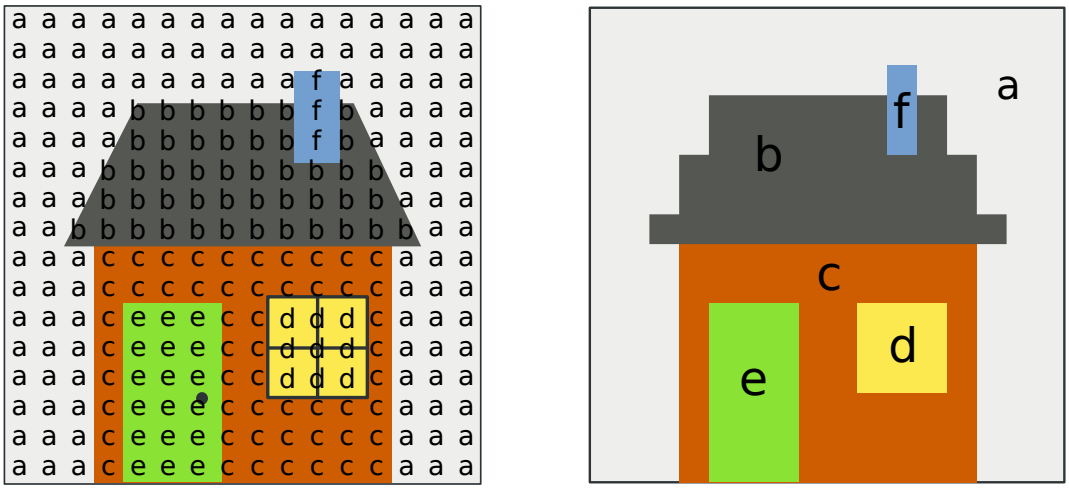
$$r_i < r_j \Leftrightarrow \text{premierVoxel}(r_i) < \text{premierVoxel}(r_j).$$

La région infinie $r_0$ est définie de telle sorte que pour chaque région $r$ appartenant à l'image, $r_0 < r$. Ainsi dans l'ordre des régions, la région la plus petite est la région infinie suivie par la région contenant le premier voxel $v = (0, 0, 0)$. L'ordre des autres régions dépend de la partition.

Cette relation d'ordre entre les régions permet d'établir une propriété entre les régions imbriquées et les régions englobantes en fonction de leur position dans l'ordre de balayage de l'image. Le Lemme 1 indique qu'une région ne peut pas englober une région plus petite dans l'ordre de balayage.

**Lemme 1.** *Si une région $r_2$ est imbriquée dans une région $r_1$ (avec $r_1 \neq r_2$), alors $r_1 < r_2$.*

*Démonstration.* La région $r_2$ est imbriquée dans la région $r_1$. Supposons que $r_2 < r_1$, d'après la Définition 6, nous avons la relation *premierVoxel*$(r_2) <$ *premierVoxel*$(r_1)$. À partir du premier voxel de $r_2$ , il est facile de construire un 18-chemin ne comportant que des voxels $v_k <$ *premierVoxel*$(r_2)$ permettant d'aller jusqu'au premier voxel de l'image. Il existe donc un 18-chemin de voxels $v_k$ allant d'un voxel de $r_2$ à un voxel de $r_0$ et qui ne contient pas de voxels de $r_1$ puisque $v_k <$ *premierVoxel*$(r_2) <$ *premierVoxel*$(r_1)$ pour tous les voxels $v_k$ appartenant au chemin. Il y a contradiction avec la Définition 4 et donc $r_2 > r_1$. □




(a) (b)

FIG. 1.6: Image 2D étiquetée : (a) image 2D où chaque pixel est associé à l'un des 6 étiquettes ; (b) image de étiquettes représentant la partition de l'image 2D. Une couleur est choisie pour chaque étiquette et chaque pixel est dessiné avec la couleur de son étiquette.

### 1.1.4 Image étiquetée

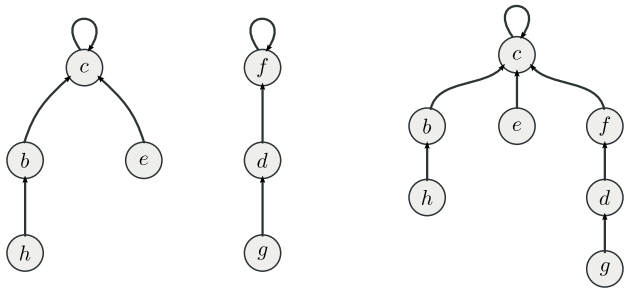
Afin de représenter la partition d'une image en régions, il faut associer à chaque élément (pixel, voxel, etc.) de l'image une étiquette décrivant l'appartenance de l'élément à une région donnée. Deux pixels (ou deux voxels) ayant une même étiquette appartiennent à la même région. Inversement, deux pixels (ou deux voxels) ayant des étiquettes différentes appartiennent à des régions différentes.

La Figure 1.6 présente un exemple d'étiquetage pour une image 2D. Chaque pixel de l'image originale Figure 1.6a est étiqueté par une lettre : l'étiquette de la région d'appartenance du pixel. En pratique, nous associons une couleur différente à chaque étiquette et nous représentons l'image avec ces couleurs comme nous le montre la Figure 1.6b. Cette image présente la partition de l'image en régions.

Cette représentation est très utilisée pour effectuer des traitements locaux sur les régions comme les opérateurs morphologiques [Ser83]. Basculer un élément dans une autre région revient à changer son étiquette. Cependant, cette approche est inadaptée lorsqu'il s'agit de faire des opérations de fusion ou de division de régions car ces opérations impliquent de nombreux changements d'étiquettes pour les éléments de l'image.

Plusieurs structures de données permettent de maintenir plus efficacement l'image étiquetée. L'une de ces structures, parfaitement adaptée à l'étiquetage d'une image et à la fusion de régions, est la forêt d'ensembles disjoints [CLR89] (*disjoint-sets forest* en anglais). Ainsi les ensembles disjoints, et les opérations de manipulation qui l'accompagnent, permettent de représenter la partition d'un ensemble de valeurs de sorte qu'une valeur ne peut appartenir qu'à un ensemble à la fois.

Dans une forêt d'ensembles disjoints, les ensembles sont représentés par des arborescences où chaque arbre correspond à un ensemble. Chaque élément de l'arbre pointe uniquement sur son parent. La racine de chaque arbre est le représentant de l'ensemble et elle est son propre parent. Les deux opérations disponibles sur les ensembles disjoints sont $trouver$ ($find$ en anglais) et $union$ ($union$ en anglais). La première utilise la relation de parent jusqu'à trouver le représentant de l'ensemble. La seconde opération force la racine d'un arbre à pointer vers la racine d'un autre arbre et ainsi fusionne les ensembles représentés. Deux heuristiques sont utilisées, l'union par rang et la compression de chemin,




(a) (b)

FIG. 1.7: Une forêt d'ensembles disjoints. (a) Représentation de deux ensembles $\{b, h, c, e\}$ et $\{f, d, g\}$ ayant comme représentants respectifs $c$ et $f$. (b) Union des deux ensembles par l'opération *union*$(e, g)$ : la racine de l'arbre contenant $g$ pointe maintenant sur la racine de l'arbre contenant $e$.

afin d'améliorer la complexité des opérations. La première optimisation consiste à toujours insérer le plus petit des deux arbres sous le plus grand lors de la fusion de deux ensembles. La deuxième optimisation consiste à aplatir l'arbre en remontant sous la racine de l'arbre tous les nœuds traversés lors de la recherche d'un élément. L'idée est de conserver en permanence un arbre qui soit le moins profond possible, permettant ainsi de diminuer la complexité. Avec ces heuristiques, le temps d'exécution est quasi linéaire par rapport au nombre d'opérations effectuées sur les ensembles disjoints. La complexité d'une opération $trouver$ ou $union$ est en $O(\alpha(n))$ [Tar75] où $n$ est le nombre de valeurs et $\alpha$ est l'inverse de la fonction d'Ackermann, une fonction à croissance très rapide. Ainsi pour toute valeur pratique de $n$, $\alpha(n)$ est bornée par 4. Nous considérons que ces opérations sont réalisées en temps constant et nous utilisons cette supposition lorsque nous étudions la complexité des opérations mettant en œuvre cette structure de données.

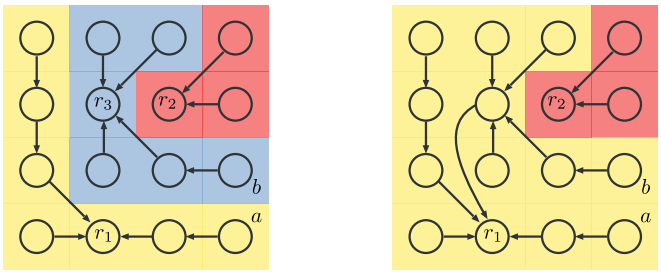
La Figure 1.7a présente un exemple (tiré de [CLR89]) contenant deux arborescences représentant deux ensembles de valeurs. L'arbre de gauche représente l'ensemble $\{b, h, c, e\}$ avec $c$ comme représentant et l'arbre de droite représente l'ensemble $\{f, d, g\}$ avec $f$ comme représentant. La Figure 1.7b illustre l'union des deux ensembles en utilisant l'opération *union*$(e, g)$.

Dans notre application, les éléments des ensembles sont les pixels de l'image et les ensembles représentent les éléments ayant une même étiquette : il s'agit des régions. Les ensembles disjoints permettent de représenter une partition de l'image en régions à l'aide d'une étiquette en autorisant une mise à jour efficace en cas de fusion de régions grâce à l'opération $union$. La Figure 1.8a illustre le principe d'une forêt d'ensembles disjoints utilisée pour représenter une partition d'une image. Dans cette exemple, il y a trois composantes connexes que nous étiquetons $r_1$, $r_2$ et $r_3$. Dans la Figure 1.8b, nous réalisons la fusion de la région $r_1$ avec la région $r_3$. Nous observons alors comment la racine de l'arbre qui représentait la région $r_3$ pointe maintenant vers la racine de l'arbre représentant la région $r_1$.

### 1.1.5 Utilisation d'un graphe pour représenter une image

Certaines applications de traitement d'images utilisent un graphe pour représenter les éléments de l'image et les relations d'adjacence entre ces éléments (voir par exemple [FH98]). Cette structure est équivalente à la représentation par une image étiquetée. Cependant la notion d'adjacence n'est pas forcément induite par la grille de l'image : les algorithmes de traitement travaillent sur le graphe et sont ainsi plus génériques.

Les nœuds du graphe représentent les pixels ou les voxels et les arêtes représentent la notion d'adjacence entre éléments. Dans ce type de représentation, les régions sont représentées par un ensemble de




(a) (b)

FIG. 1.8: Une forêt d'ensembles disjoints utilisée pour représenter la partition d'une image. (a) Représentation de la partition de l'image en trois régions étiquetées $r_1$, $r_2$ et $r_3$. (b) Fusion de la région $r_1$ avec la région $r_3$ par l'utilisation de l'opération *union*$(a, b)$. La racine de l'arbre qui représente $r_3$ pointe maintenant sur la racine représentant $r_1$.

nœuds du graphe qui partagent une même étiquette. Avec cette représentation, la connexité des pixels ou des voxels peut facilement varier, en changeant simplement la manière dont les nœuds sont reliés par des arêtes. Des informations supplémentaires permettant de faire des traitements sur l'image peuvent être ajoutées sur les sommets ou les arêtes, par exemple en donnant un poids aux arêtes. La Figure 1.9 présente un graphe étiqueté représentant la partition d'une image 2D en régions.

### 1.1.6 Intervoxels

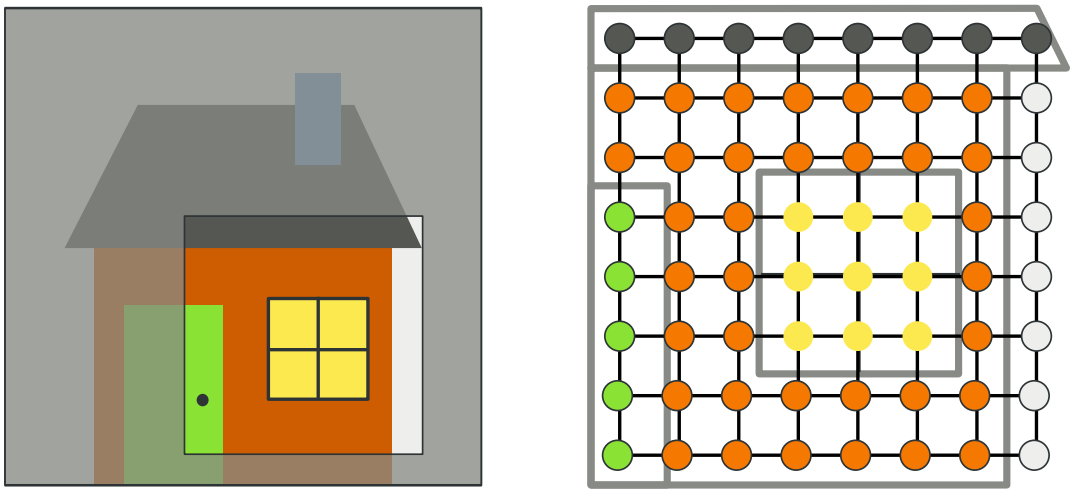
Dans le cadre de la représentation d'images par des modèles discrets, l'intervoxel [Kov89] propose une subdivision de l'espace $\mathbb{R}^3$ en éléments unitaires de dimension 0, 1, 2 et 3. Ainsi dans le cadre de l'intervoxel, les éléments unitaires de dimension 3, que nous représentons par des cubes, correspondent aux voxels. Entre deux voxels se trouve un élément unitaire de dimension 2 appelé *surfel* pour élément de surface. Nous représentons les surfels par des carrés. Les surfels sont séparés par des éléments unitaires de dimension 1, les *lignels*, qui sont représentés par des segments. Enfin, les lignels sont séparés par des *pointels*, les éléments unitaires de dimension 0, représentés par des points. L'ensemble de ces éléments unitaires permet de représenter les frontières de la partition d'une image (voir l'exemple Figure 1.10).

La notion d'incidence sur les éléments unitaires de dimension $i$ est définie de la manière suivante : un élément de dimension $i$ est incident à un élément de dimension $(i-1)$ si l'élément de dimension $(i-1)$ appartient au bord de l'élément de dimension $i$. Deux éléments unitaires de dimension $i$ sont adjacents si ils sont incidents à un même élément unitaire de dimension $(i-1)$ (voir les deux exemples Figure 1.11).

Les éléments de l'intervoxel sont bien adaptés à la modélisation d'espaces discrets [Kov08]. Ils permettent notamment de respecter des propriétés très importantes comme le théorème de Jordan pour les surfaces : une surface fermée partage l'espace en deux composantes connexes distinctes. L'intervoxel permet de représenter les frontières des régions. Ainsi dans le cadre de l'intervoxel, la frontière entre deux régions est donnée par la Définition 7.

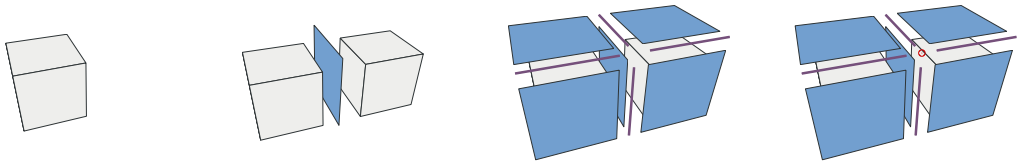
**Définition 7** (Frontières entre deux régions)**.** *La frontière entre deux régions $r_1$ et $r_2$ d'étiquettes $l_1$ et $l_2$ est l'ensemble des surfels séparant un voxel $v_1$ d'étiquette $l_1$ et un voxel $v_2$ d'étiquette $l_2$.*

L'union des frontières entre les régions adjacentes de l'image forme les frontières de la partition que nous appelons parfois frontières de l'image. De plus, l'intervoxel représente les faces mais aussi les



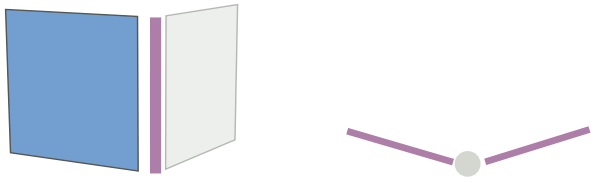
(a) (b)

FIG. 1.9: Image 2D représentée par un graphe étiqueté des pixels : (a) morceau d'une image 2D ; (b) graphe étiqueté des pixels représentant la partition en régions du morceau de l'image. Les couleurs associées aux nœuds du graphe représentent les différentes étiquettes. L'adjacence utilisée pour le graphe est la 4-connexité.



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

FIG. 1.10: Les différents éléments unitaires de l'intervoxel : (a) un cube représentant un voxel ; (b) un carré représentant un surfel entre deux voxels ; (c) des segments représentant des lignels entre les surfels ; (d) le point représentant un pointel entre les lignels.



(a) (b)

FIG. 1.11: Les relations entre les éléments de l'intervoxel : (a) deux surfels incidents à un lignel, les surfels sont adjacents ; (b) deux lignels incidents à un pointel, les lignels sont adjacents.

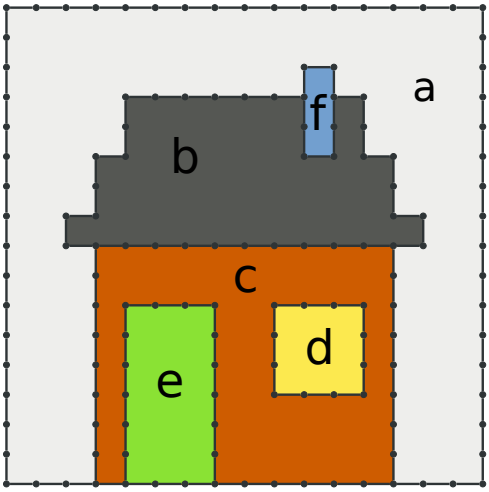



FIG. 1.12: Partition d'une image 2D représentée à l'aide de l'interpixel.

arêtes et les sommets qui appartiennent aux frontières des régions. En utilisant les éléments unitaires de l'intervoxel, la frontière d'une région est représentée par un ensemble de pointels, lignels et surfels. Cette décomposition naturelle de l'espace permet de donner une représentation unifiée du bord des régions que nous utilisons dans la suite de ce manuscrit. La Figure 1.12 représente les éléments de l'interpixel appartenant aux frontières de la partition de l'image 2D en 6 régions. Les segments représentent les lignels et les points représentent les pointels.

### 1.1.7 Modèles hiérarchiques

Les modèles hiérarchiques de représentation sont apparus afin de simplifier la représentation d'une image dans le cadre du traitement d'images. Le modèle à l'origine de cette classe de modèle est le *quadtree* [FB74, DRS80]. Il s'agit d'une structure d'arbre représentant un espace à deux dimensions en le subdivisant récursivement en quatre sous-espaces. Une image 2D est alors représentée par un arbre, chaque nœud de l'arbre pouvant être subdivisé en quatre *quadrants* égaux (voir exemple Figure 1.13).

L'extension des quadtrees en dimension 3 conduit à la définition des *octrees* [Sam84]. Les octrees sont une structure d'arbre représentant un espace à trois dimensions en le subdivisant récursivement en huit sous-espaces. Chaque nœud de l'arbre est alors subdivisé en huit *octants* égaux (voir exemple Figure 1.14).

Les quadtrees et octrees possèdent des définitions simples donnant une première représentation hiérarchique de la partition d'une image en régions. Cependant, pour modifier une partition stockée sous cette forme nous ne disposons que de deux opérations, soit fusionner des feuilles de l'arbre soit les diviser. Ce type de modèle est limité par le découpage régulier de l'espace qui ne permet pas par exemple de fusionner deux régions n'appartenant pas au même sous-arbre.

D'autres modèles hiérarchiques ont été développés pour répondre à différents besoins. Chacun de ces modèles utilise le même principe. Il s'agit de représenter une image par un arbre en utilisant une stratégie afin de subdiviser l'image en sous-espaces. Par exemple, nous pouvons citer les partitions binaires de l'espace (*Binary Space Partition* en anglais) [FKN80], ou les *Kd-trees* [BCKO08] qui sont utilisés classiquement en imagerie.

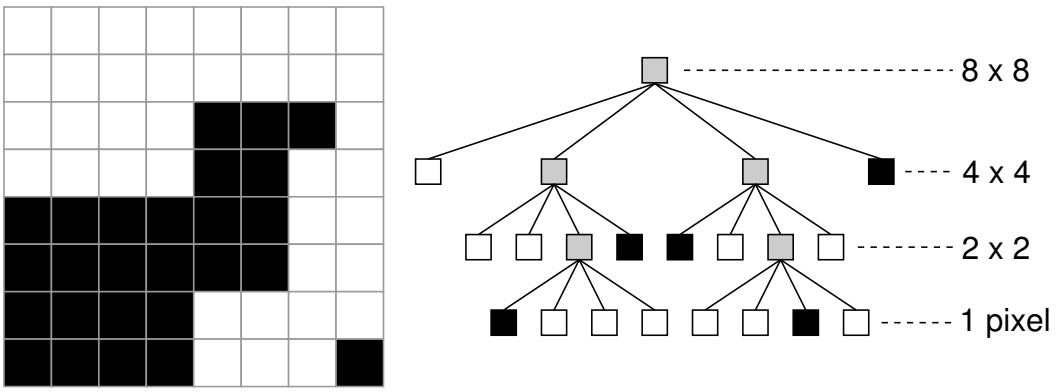



FIG. 1.13: Exemple de quadtree représentant une image binaire $8 \times 8$ pixels. Les nœuds gris dans l'arbre représentent les régions non homogènes qui sont alors subdivisées en quatre sous-régions. Les feuilles de l'arbre représentent les régions homogènes de l'image.

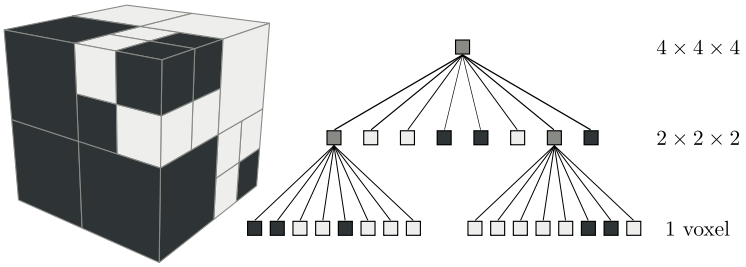
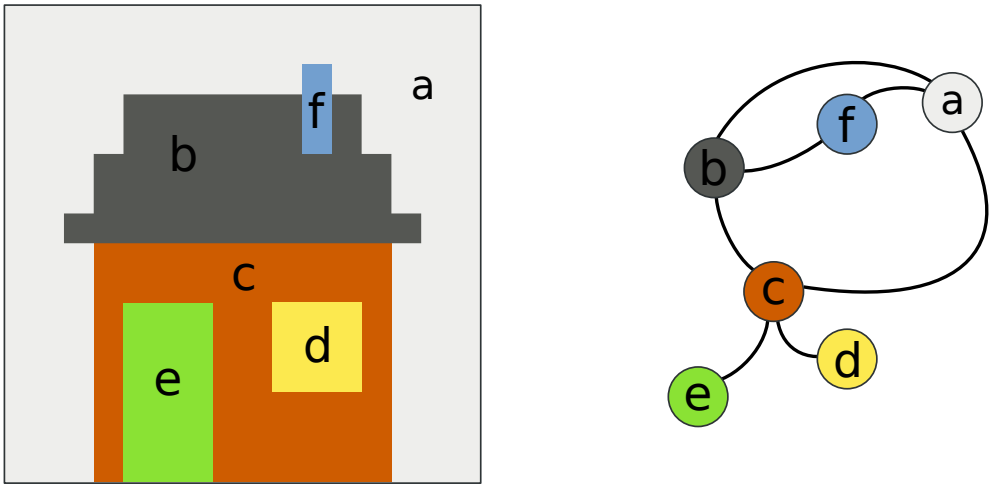



FIG. 1.14: Exemple d'octree représentant une image binaire de $4 \times 4 \times 4$ voxels. Les nœuds gris dans l'arbre représentent les régions non homogènes qui sont alors subdivisées en huit sous-régions.




(a) (b)

FIG. 1.15: Graphe d'adjacence des régions pour une image 2D (la région infinie n'est pas représentée) : (a) exemple d'image 2D avec 6 régions étiquetées de $a$ à $f$ ; (b) graphe d'adjacence des régions représentant la partition de l'image. Une arête indique que les deux nœuds reliés représentent des régions adjacentes.

## 1.2 Modèles topologiques

La partition d'une image en régions contient des informations topologiques comme l'adjacence ou l'imbrication de régions qui sont exploitables dans les opérations de traitement d'images. Nous cherchons à obtenir une représentation où l'élément de base est la région. Les modèles discrets présentés jusqu'ici ne représentent pas efficacement ces informations car chaque pixel ou voxel est considéré de manière indépendante. Aussi, pour permettre une manipulation plus aisée des partitions d'images en régions, les modèles topologiques sont apparus au cours des années 1970. Dans un premier temps, la définition et l'étude de ces modèles ont porté sur les applications au traitement d'images 2D. Certains de ces modèles ont été par la suite étendus en dimension supérieure, notamment en 3D, par exemple pour les applications de traitement d'images médicales.

### 1.2.1 Graphe d'adjacence des régions

Le graphe d'adjacence des régions [Ros74] (*Region Adjacency Graph ou RAG* en anglais) est l'un des premiers modèles topologiques de représentation d'une image partitionnée en régions. Il modélise les relations d'adjacence entre les régions d'une image. Un RAG est un graphe planaire non orienté où les nœuds représentent les régions de l'image et les arêtes symbolisent les relations d'adjacence entre les régions.

La Figure 1.15 donne l'exemple d'une image 2D avec 6 régions et présente le graphe d'adjacence des régions qui décrit cette partition. Nous observons que la région $a$ est adjacente aux régions $b$, $c$ et $f$, ainsi des arêtes relient le nœud $a$ avec les nœuds $b$, $c$ et $f$ dans le graphe.

La définition du graphe d'adjacence des régions s'étend sans problème en dimension supérieure, il est possible d'utiliser cette structure de données pour représenter des partitions d'images 3D. Un autre avantage du graphe d'adjacence des régions est la facilité de mise en œuvre de l'opération de fusion de

régions. En effet, pour fusionner deux régions connexes dans le graphe d'adjacence des régions, il faut fusionner les nœuds adjacents représentant les deux régions : c'est l'opération de contraction d'arêtes.

Cependant, le RAG souffre également de défauts importants. D'abord la structure ne fait pas de différence entre deux régions simplement adjacentes et deux régions multiadjacentes. Par exemple dans la Figure 1.15, les régions $a$ et $c$ sont adjacentes deux fois, le graphe d'adjacence ne représente pas cette information. D'autre part, le graphe d'adjacence des régions ne représente pas la relation d'imbrication. La région $d$ est imbriquée dans la région $c$ alors que la région $e$ n'est pas imbriquée dans la région $c$. Les deux régions $d$ et $e$ sont adjacentes à la région $c$ : c'est la seule information proposée par le graphe d'adjacence des régions, il n'est pas possible de différencier $d$ de $e$ par rapport à la région $c$. Ainsi, la représentation par graphes d'adjacence des régions ne permet pas de différencier deux partitions d'images pourtant topologiquement différentes.

La structure de graphe d'adjacence des régions est donc une structure simple et facile à manipuler dans le cadre du traitement d'images mais les problèmes de représentation rencontrés ont conduit différents auteurs à proposer des solutions qui résolvent tout ou partie des inconvénients des RAG.

### 1.2.2 Modèles dérivés à base de graphes

Parmi les modèles topologiques dérivés des graphes d'adjacence des régions, nous trouvons différentes solutions qui résolvent une partie des problèmes de représentation du RAG.

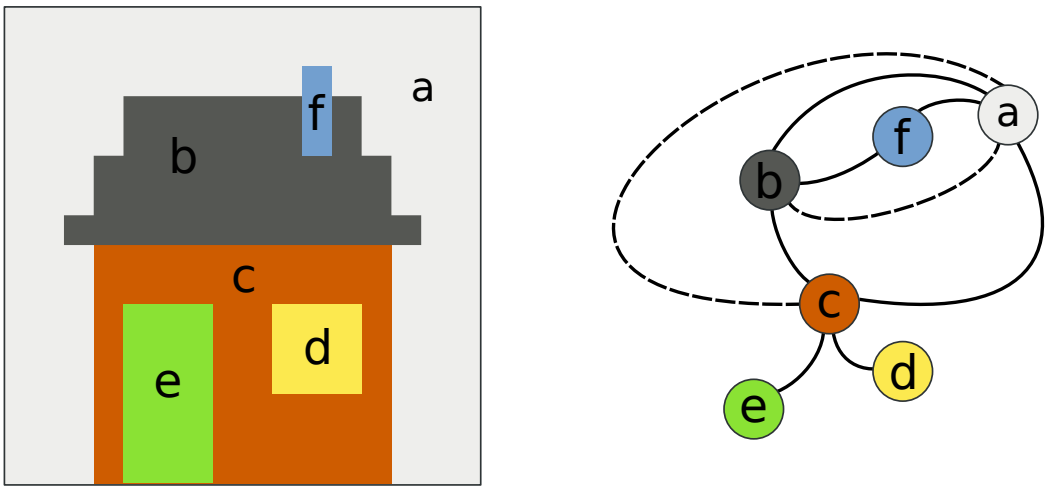
Un multigraphe d'adjacence des régions consiste à représenter chaque adjacence entre deux régions par une arête. La Figure 1.16 présente le multigraphe d'adjacence représentant l'image précédente. Nous observons les deux arêtes entre la région $a$ et la région $b$ ainsi que les deux arêtes entre la région $a$ et la région $c$ qui permettent de représenter les multiadjacences entre ces régions. Cette approche résout le problème lié à la multiadjacence mais ne traite pas les autres problèmes. La relation d'imbrication n'est toujours pas intégrée et deux partitions topologiquement différentes peuvent avoir le même graphe d'adjacence. Le multigraphe d'adjacence des régions apporte donc une nouvelle information et reste facilement utilisable avec des images de dimension supérieure.

La représentation d'une image basée sur les graphes duaux [KM95] répond plus complètement aux problèmes posés par les graphes d'adjacence de régions. L'idée des graphes duaux est de maintenir en parallèle deux graphes, une extension des graphes d'adjacence d'un côté et son graphe dual de l'autre. Le premier, appelé primal, est un multigraphe d'adjacence plongé dans $\mathbb{R}^2$ avec en plus des boucles pour représenter la relation d'imbrication. La Figure 1.17 illustre les graphes duaux sur une image 2D. Nous observons bien une boucle autour de la région imbriquée dans le graphe primal. Le graphe dual est représenté en gris.

En utilisant la géométrie des arêtes (boucles) dans le graphe primal, il est alors possible d'obtenir la relation d'imbrication entre les régions de la partition [KM95]. La structure des graphes duaux permet de distinguer deux images topologiquement différentes. Cependant les graphes duaux posent différents problèmes. D'une part, il faut étudier la géométrie des graphes pour obtenir les informations d'imbrication ce qui n'est pas souhaitable. D'autre part, le modèle des graphes duaux ne possède pas d'extension en dimension supérieure ce qui pose problème notamment pour la représentation des images 3D. C'est pour résoudre ces différents problèmes que des travaux se sont intéressés aux modèles basés sur les cartes combinatoires.

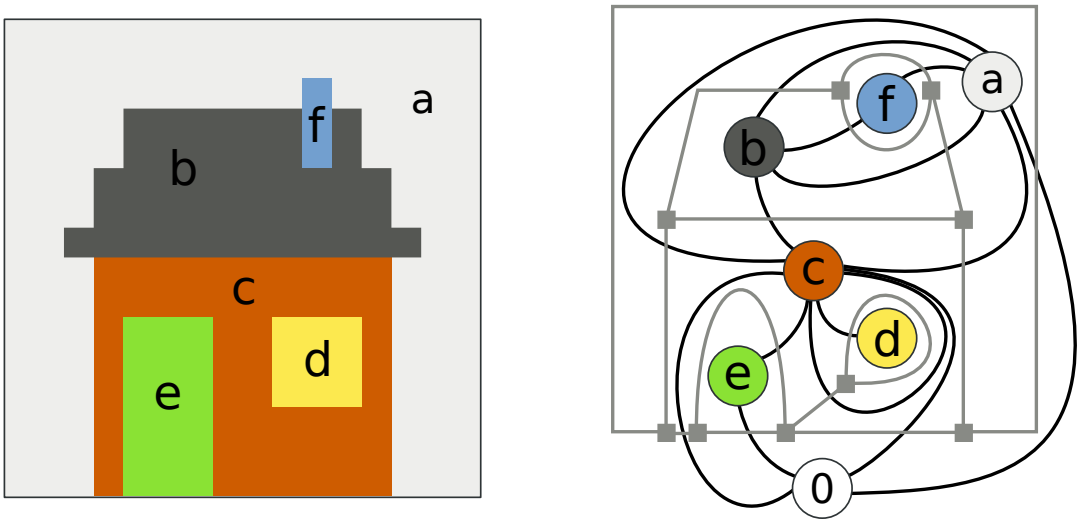
### 1.2.3 Cartes combinatoires

Une carte combinatoire est un modèle mathématique, issu des cartes planaires, qui décrit une subdivision d'un espace. Une subdivision de l'espace est une partition d'un espace de dimension $n$ en $(n+1)$ sous-ensembles, où chaque élément est appelé *cellule* de dimension 0, 1, 2, 3, ..., $n$ (désignés couramment par sommet, arête, face et volumes pour les dimension 0 à 3). Nous parlons de $i$-cellule




(a) (b)

FIG. 1.16: Multigraphe d'adjacence des régions pour une image 2D : (a) image 2D avec 6 régions étiquetées de $a$ à $f$ ; (b) multigraphe d'adjacence des régions représentant la partition de l'image. Une arête représente une adjacence entre deux régions (représentées par les nœuds). Les arêtes en pointillés représentent les nouvelles arêtes par rapport à un graphe d'adjacence des régions. Ainsi les régions $a$ et $b$ comme les régions $a$ et $c$ sont multiadjacentes.




(a) (b)

FIG. 1.17: Graphes duaux utilisés pour représenter la partition d'une image : (a) Image 2D avec 6 régions étiquetées de $a$ à $f$ ; (b) graphe primal (en traits noirs) et graphe dual (en traits gris) représentant la partition de l'image. La région infinie $0$ qui entoure l'image est ajoutée au graphe d'adjacence. Nous remarquons notamment la boucle dans le graphe primal représentant l'imbrication de la région $b$ dans la région $c$.

pour désigner une cellule de dimension $i$. Le bord d'une $i$-cellule est un ensemble de $(j < i)$-cellules. Deux cellules sont *incidentes* lorsque l'une appartient au bord de l'autre. Deux cellules sont *adjacentes* lorsqu'elles sont de même dimension $i$ et qu'elles sont incidentes à une même $(i-1)$-cellule. La carte encode toutes les cellules de la subdivision ainsi que les relations d'incidence et d'adjacence entre ces différentes cellules. Elle représente ainsi la topologie de l'espace. La Définition 8 présente la définition formelle d'une carte combinatoire en 3D.

**Définition 8** (Carte combinatoire 3D)**.** *Une carte combinatoire 3D, ou 3-carte, est un quadruplet* $M = (D, \beta_1, \beta_2, \beta_3)$ *où :*

1. $D$ *est un ensemble fini de brins ;*
2. $\beta_1$ *est une permutation*[1] *sur* $D$ *;*
3. $\beta_2$ *et* $\beta_3$ *sont deux involutions*[2] *sur* $D$ *;*
4. $\beta_1 \circ \beta_3$ *est une involution.*

Les éléments de base utilisés dans la définition des cartes combinatoires sont appelés *brins*. $\beta_i$ est une relation entre deux brins qui décrit une relation d'adjacence entre deux cellules de dimension $i$. Dans l'ensemble de ce travail, nous écrivons $\beta_0 = \beta_1^{-1}$ et nous notons la composition de deux relations $\beta_i$ et $\beta_j$ par $\beta_{ij} = \beta_j \circ \beta_i$.

Dans ce modèle, la notion de cellule est représentée par un ensemble de brins liés par des relations $\beta_i$ spécifiques. Par exemple, une face incidente à un brin $b$ est représentée par l'ensemble des brins accessibles en utilisant n'importe quelle combinaison des relations $\beta_1$ et $\beta_3$ à partir de $b$. Cette idée est formalisée dans la Définition 9 par la notion d'orbite d'un brin.

**Définition 9** (Orbite)**.** *Soit* $\Phi = \{f_1, \ldots, f_k\}$ *un ensemble fini de permutations sur* $D$*. Nous notons* $\langle\Phi\rangle$ *le groupe de permutations généré par* $\Phi$*. Il s'agit de l'ensemble des permutations obtenues par toutes compositions et inversions des permutations contenues dans* $\Phi$*. L'orbite d'un brin* $b$ *par rapport à* $\Phi$ *est définie par* $\langle\Phi\rangle(b) = \{\phi(b) | \phi \in \langle\Phi\rangle\}$.

Ainsi un sommet est l'ensemble des brins appartenant à une orbite $\langle\beta_{21}, \beta_{31}\rangle$, une arête est l'ensemble des brins appartenant à l'orbite $\langle\beta_2, \beta_3\rangle$, une face est l'ensemble des brins appartenant à l'orbite $\langle\beta_1, \beta_3\rangle$, et un volume est l'ensemble des brins appartenant à l'orbite $\langle\beta_1, \beta_2\rangle$. Un brin est incident à une cellule s'il appartient à l'orbite représentant cette cellule. Cette définition implique que deux cellules sont incidentes si l'intersection de leurs orbites est un ensemble non vide de brins. Étant donné un brin $b$ incident à une $i$-cellule $c$, un brin $b'$ de la $i$-cellule adjacente à $c$ le long de la $(i-1)$-cellule qui contient $b$ est obtenu en utilisant $b' = \beta_i(b)$.

La Figure 1.18 présente un exemple de cartes combinatoires représentant deux volumes. Les deux volumes de la Figure 1.18a sont représentés par la carte combinatoire dessinée dans la Figure 1.18b. Les brins sont représentés par des flèches dont l'orientation suit la relation $\beta_1$.

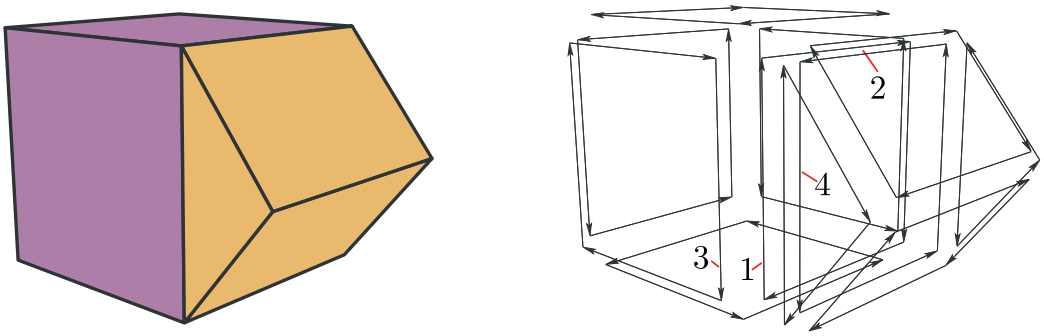
Le degré d'une cellule $c$ de dimension $i$ est défini par le nombre de cellules distinctes de dimension $(i+1)$ incidentes à $c$ (voir exemple Figure 1.19a). La notion de degré local d'une cellule $c$ de dimension $i$ est la somme sur toutes les $(i+1)$-cellules incidentes à $c$ du nombre de fois où chaque cellule distincte est incidente à $c$ (voir exemple Figure 1.19b).

### 1.2.4 Suppression de cellules

Dans une carte combinatoire, l'opération de suppression de cellules en dimension $i$ est appelée $i$-suppression. Cette opération n'est utilisable que si le degré local de la cellule est inférieur ou égal

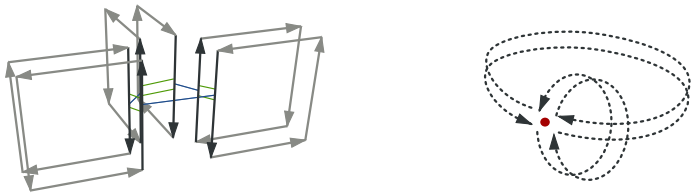
[1] Une permutation sur un ensemble $E$ est une bijection de $E$ sur lui-même.

[2] Une involution $f$ sur un ensemble $E$ est une bijection de $E$ sur lui-même telle que $f = f^{-1}$.



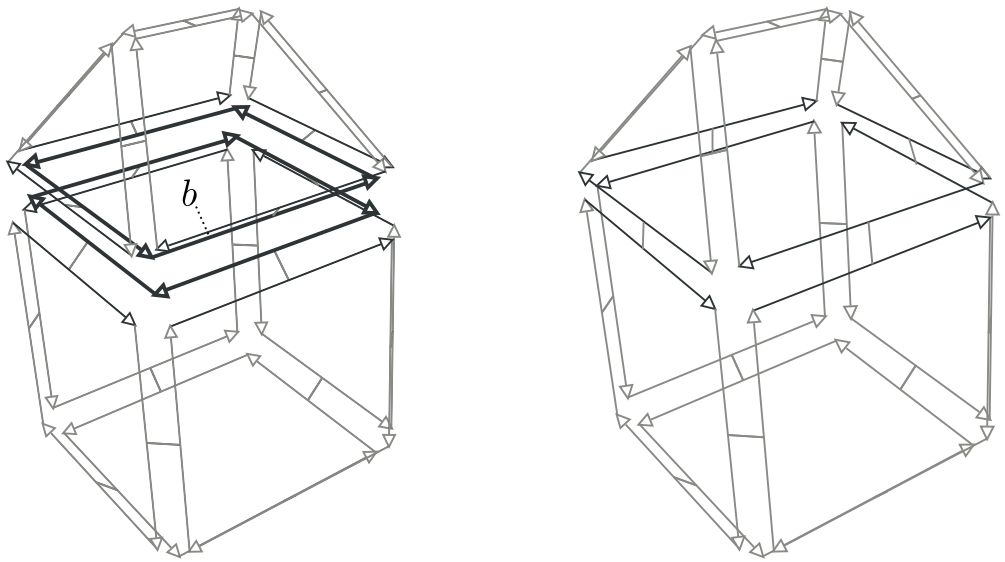

(a) (b)

FIG. 1.18: Exemple de carte combinatoire en dimension 3 : (a) un objet composé de deux volumes, un cube et un prisme ; (b) carte combinatoire représentant l'objet. Les brins sont représentés par des flèches indiquant la relation $\beta_1$. Les involutions $\beta_2$ et $\beta_3$ sont implicitement données par la géométrie utilisée dans la représentation. Deux brins proches incidents à la même face mais dans des volumes différents sont reliés par $\beta_3$ (par exemple $\beta_3(1) = 4$). Deux brins proches incidents au même volume mais situés dans des faces différentes sont liés par $\beta_2$ (par exemple $\beta_2(1) = 3$). Deux brins qui se suivent dans la même face et le même volume sont liés par $\beta_1$, la flèche donnant le sens de la permutation (par exemple $\beta_1(1) = 2$ et $\beta_0(2) = 1$).



(a) (b)

FIG. 1.19: Notion de degré et de degré local dans une carte topologique. Tous les brins ne sont pas représentés pour préserver la lisibilité. (a) L'arête représentée par les brins en noir est de degré 3. Son degré local est également 3. (b) Le sommet représenté est de degré 2. Son degré local est 4, il y a deux arêtes qui sont chacune incidente deux fois au sommet.




(a) (b)

FIG. 1.20: 2-suppression de la face incidente au brin $b$. (a) Configuration initiale avec deux volumes adjacents à la face désignée par $b$. (b) Carte obtenue après la suppression de faces : les deux volumes ont fusionné.

à deux. En effet, pour les cellules de degré supérieur il n'est pas possible de déterminer de manière automatique comment fusionner les cellules incidentes. Lors de la suppression, cette opération fusionne les deux $(i+1)$-cellules incidentes à la cellule supprimée. Pour plus d'informations, [DL03] propose une définition généralisée en toutes dimensions des opérations de $i$-suppression.

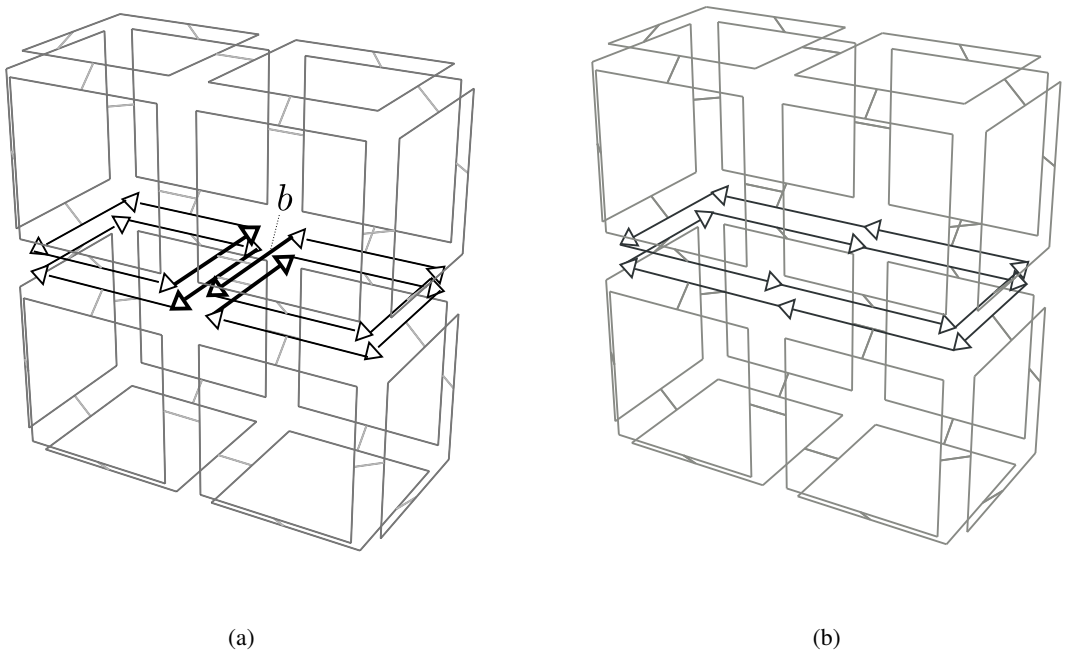
Dans les cartes combinatoires 3D, les trois opérations de suppression possibles sont la suppression de faces (2-suppression), la suppression d'arêtes (1-suppression) et la suppression de sommets (0-suppression). Nous rappelons ici les algorithmes de $i$-suppression définis dans [Dam08b] et nous illustrons leur fonctionnement.

Dans une carte combinatoire 3D, la suppression d'une face est réalisée sans contrainte. En effet, dans un espace de dimension 3, chaque face est au maximum de degré deux. L'Algorithme 2 réalise la suppression de la face incidente à $b$ dans la 3-carte $M$ en fusionnant les deux volumes incidents à la face, puis en supprimant celle-ci de la carte combinatoire. La Figure 1.20 présente un exemple de 2-suppression.

**Algorithme 2** : Suppression d'une face

**Données** : Une carte $M$ ;
Un brin $b$.
**Résultat** : Suppression dans $M$ de la face incidente à $b$.

**pour chaque** *brin* $b'$ *appartenant à l'orbite* $\langle\beta_1\rangle(b)$ **faire**
  coudre par $\beta_2$ les brins $\beta_2(b')$ et $\beta_{32}(b')$;
  supprimer les brins $b'$ et $\beta_3(b')$ de $M$;




(a) (b)

FIG. 1.21: 1-suppression de l'arête incidente au brin $b$. (a) Configuration initiale avec deux faces adjacentes à l'arête désignée par $b$. (b) Carte obtenue après la suppression de l'arête : les deux faces sont fusionnées.

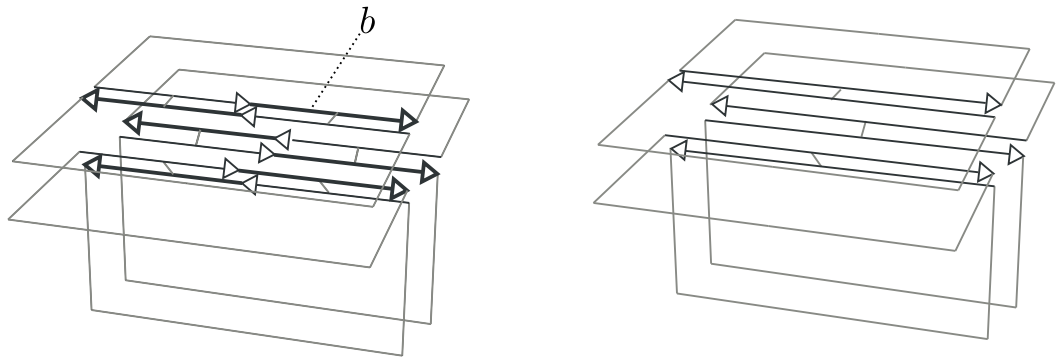
La suppression d'une arête dans une 3-carte n'est possible que pour une arête de degré inférieur ou égal à deux (voir [DL03] pour plus d'informations). L'Algorithme 3 présente la suppression de l'arête incidente à $b$ dans la 3-carte $M$. Il réalise dans un premier temps la fusion des faces incidentes à l'arête puis supprime l'arête. Contrairement aux autres opérations cet algorithme ne comporte pas de boucle car le nombre de brins concerné est toujours le même. La Figure 1.21 illustre le principe de la 1-suppression.

**Algorithme 3** : Suppression d'une arête

**Données** : Une carte $M$ ;
Un brin $b$.
**Résultat** : Suppression dans $M$ de l'arête de degré inférieur ou égal à 2 incidente à $b$.

$b_0 \leftarrow \beta_0(b), b_1 \leftarrow \beta_{21}(b), b_2 \leftarrow \beta_{20}(b), b_3 \leftarrow \beta_1(b)$;
coudre par $\beta_1$ les brins $b_0$ et $b_1$ ;
coudre par $\beta_1$ les brins $\beta_3(b_1)$ et $\beta_3(b_0)$ ;
coudre par $\beta_1$ les brins $b_2$ et $b_3$ ;
coudre par $\beta_1$ les brins $\beta_3(b_3)$ et $\beta_3(b_2)$ ;
supprimer les brins $b$, $\beta_2(b)$, $\beta_3(b)$ et $\beta_{23}(b)$ de $M$;

Enfin, l'opération de suppression d'un sommet dans une 3-carte est définie uniquement pour les sommets de degré inférieur ou égal à deux (voir [DL03]). L'Algorithme 4 présente la suppression du sommet incident à $b$ dans la 3-carte $M$. L'algorithme est plus compliqué que pour la suppression d'une arête à cause de la définition non homogène des cartes combinatoires, $\beta_1$ étant une permutation. Le principe reste le même que pour les autres opérations de suppression : dans un premier temps nous fu-




(a) (b)

FIG. 1.22: 0-suppression du sommet incident au brin $b$. (a) Configuration initiale avec deux arêtes adjacentes au sommet désigné par $b$. (b) Carte obtenue après la suppression de sommet : les deux arêtes sont fusionnées.

sionnons les arêtes incidentes au sommet et nous supprimons les brins représentant le sommet supprimé. Dans l'algorithme, $i$ désigne l'involution courante afin d'alterner entre l'involution $\beta_2$ et l'involution $\beta_3$. La Figure 1.22 illustre le principe de la 0-suppression dans une carte combinatoire.

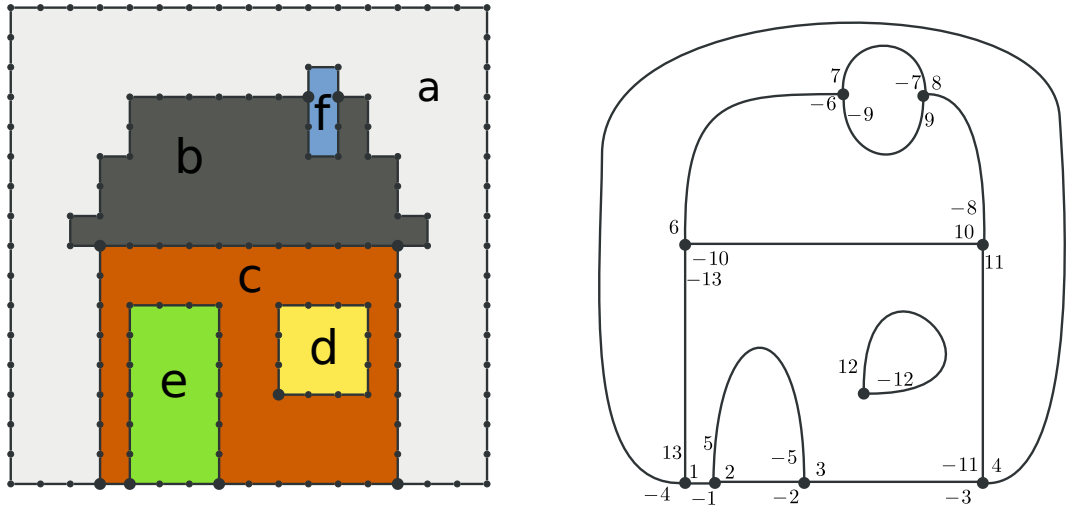
**Algorithme 4** : Suppression d'un sommet

**Données** : Une carte $M$ ;
Un brin $b$.
**Résultat** : Suppression dans $M$ du sommet de degré inférieur ou égal à 2 incident à $b$.

$i \leftarrow 2$;
$b_{courant} \leftarrow d$ ;
$b_{fin} \leftarrow \beta_1(b)$ ;
**répéter**
  $b_{suivant} \leftarrow \beta_1(\beta_i(b_{courant}))$ ;
  $b_0 \leftarrow \beta_0(b_{courant})$ ;
  $b_1 \leftarrow \beta_1(b_{courant})$ ;
  coudre par $\beta_1$ les brins $b_0$ et $b_1$ ;
  coudre par $\beta_i$ les brins $b_0$ et $\beta_0(b_{suivant})$ ;
  **si** $i = 2$ **alors** $i \leftarrow 3$ ;
  **sinon** $i \leftarrow 2$ ;
  supprimer le brin $b_{courant}$ de $M$ ;
  $b_{courant} \leftarrow b_{suivant}$ ;
**jusqu'à** $b_{courant} = b_{fin}$ ;

### 1.2.5 Modèles à base de cartes

Afin de résoudre les problèmes de représentation d'une partition d'une image par un graphe d'adjacence, une famille de modèles utilisant le formalisme des cartes combinatoires est apparue. Nous présentons dans un premier temps deux modèles : la carte discrète et le graphe topologique des frontières. Ces deux modèles sont le résultat de travaux qui précèdent la définition du modèle des cartes




(a) (b)

FIG. 1.23: Exemple d'une carte discrète représentant une image : (a) frontières en interpixel de la partition de l'image ; (b) carte planaire représentant la carte des frontières de la partition.

topologiques que nous utilisons dans ce travail et qui est décrit dans le Chapitre 2. Nous présentons également un modèle plus récent, la GeoMap, qui dérive des premiers modèles à base de cartes combinatoires. Il s'agit d'une variante à la solution proposée dans les cartes topologiques. Nous détaillons enfin un modèle représentant des partitions d'images 3D basé sur une 3-carte utilisant les éléments intervoxels.

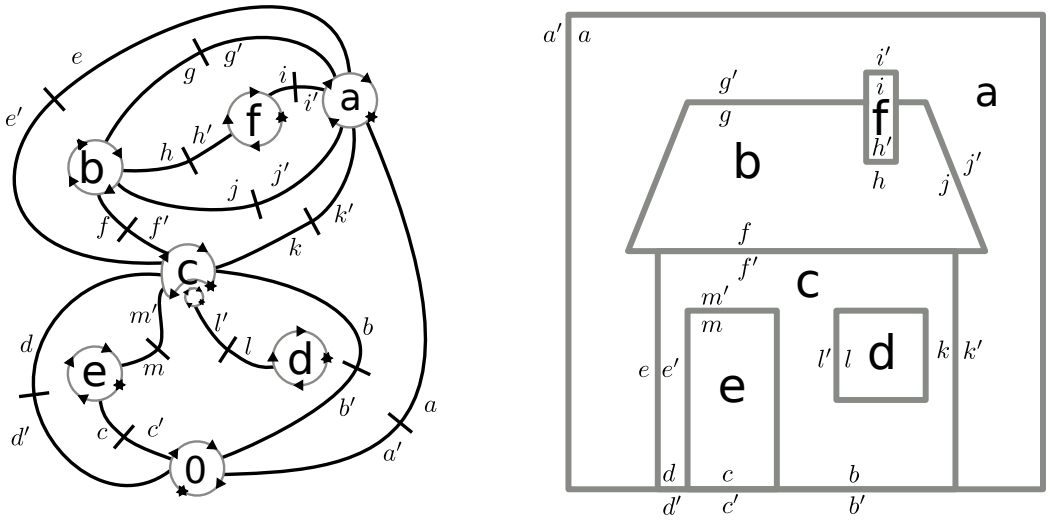
### 1.2.6 Cartes discrètes

Une carte discrète [BD96] encode la géométrie et la topologie des régions d'une image 2D dans le but d'obtenir une représentation unifiée qui ne nécessite pas le développement d'un modèle spécifique pour chaque problème à résoudre. Le modèle utilise une représentation des frontières de la partition par les éléments de l'interpixel associée à une carte planaire [Edm60] encodée sous forme d'une carte combinatoire. La carte planaire représente les cellules frontières de la partition de l'image en régions. La Figure 1.23 est un exemple de représentation d'une image 2D par une carte discrète.

Les cartes discrètes sont notamment utilisées pour définir des algorithmes de segmentation. Ainsi [BD97] définit deux opérations élémentaires permettant d'ajouter ou de supprimer une arête dans le modèle des cartes discrètes. Avec ces opérations, les auteurs définissent une opération de segmentation mélangeant des opérations de division et des opérations de fusion. Cependant l'extension en 3D du modèle des cartes discrètes est difficile, leur utilisation directe dans le contexte des images 3D n'est pas possible.

### 1.2.7 Graphe Topologique des Frontières (TGF)

Un graphe topologique des frontières [Fio95, Fio96] est également une structure basée sur les cartes combinatoires qui permet de représenter une partition d'une image 2D. Dans cette structure, les régions sont représentées par des sommets qui sont reliés par des brins. Deux opérateurs $\alpha$ et $\sigma$ sur ces brins permettent de retrouver les adjacences entre les régions ($\alpha$) ainsi que l'ordre des arêtes autour de chaque région ($\sigma$). La région infinie $\infty$, représentant l'extérieur de l'image, est ajoutée pour représenter l'ensemble des frontières de la partition. Cette approche est similaire aux cartes discrètes en y ajoutant




(a) (b)

FIG. 1.24: Exemple de graphe topologique des frontières représentant une image 2D : (a) TGF représentant la partition de l'image ; (b) autre représentation du même TGF.

la notion directe de régions. Les deux approches sont différentes dans le codage de l'imbrication. L'originalité de cette approche est son algorithme d'extraction optimal permettant d'obtenir un graphe topologique des frontières par un balayage ligne par ligne de l'image. La Figure 1.24 donne un exemple de graphe topologique des frontières représentant une image 2D.

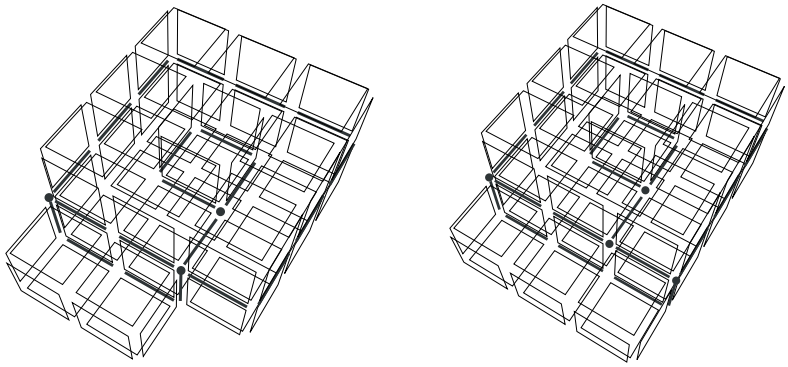
Une opération de fusion de régions est définie sur les TGF [AFG99] dans le but de réaliser des opérations de traitement d'images. Mais, si le modèle répond bien aux besoins de représentation d'une partition d'une image 2D, l'extension directe en 3D n'a pas été réalisée [Fio08] et semble difficile.

### 1.2.8 GeoMap

Une GeoMap [MK05] représente la topologie et la géométrie de manière unifiée à l'aide d'un modèle dérivé des cartes combinatoires et nommé XPMap [Köt02]. Le modèle, qui permet de représenter la partition en régions d'une image 2D, est utilisé dans des applications de traitement d'images mêlant la géométrie et la topologie. Nous trouvons notamment une approche classique de segmentation mais aussi des applications moins conventionnelles comme la mise en œuvre de ciseaux intelligents permettant de calculer de manière interactive un chemin qui épouse le contour d'un objet entre deux points donnés par un utilisateur. Les auteurs ont également travaillé sur différentes représentations de la géométrie avec notamment des plongements en sous-pixels d'une GeoMap [MK06]. Dans ce type de plongement, les frontières des régions ne sont plus limitées par une subdivision régulière de l'espace comme dans le modèle interpixel et peuvent ainsi proposer des résultats plus précis. Cependant les travaux sur les GeoMap n'ont pas d'extension en 3D.

### 1.2.9 3-carte avec des éléments intervoxel pour les frontières

Une approche développée par [Des01, BDD01] est d'utiliser une 3-carte pour représenter la topologie avec une représentation géométrique des frontières utilisant les éléments intervoxels. La particularité du modèle est d'encoder la 3-carte en utilisant des brins géométriques. Dans cette représentation, les éléments intervoxels sont utilisés pour représenter les surfaces qui séparent les régions dans la partition de l'image. Ainsi une surface est décomposée en éléments de dimension 0, 1 et 2. La correspondance



(a) (b)

FIG. 1.25: Exemple de représentation par une 3-carte de partitions d'une image 3D d'après [Des01]. (a) Exemple de représentation de la partition. Il y a trois sommets, cinq arêtes et deux faces. (b) Exemple de configuration topologiquement équivalente mais dont la représentation diffère. Il y a quatre sommets, six arêtes et deux faces.

entre la 3-carte et la géométrie est réalisée en utilisant un équivalent géométrique aux brins. Il est défini par un surfel orienté désigné par un triplet contenant un pointel, un lignel et un surfel. Les auteurs appellent cet équivalent un *g-brin*. Avec cette représentation, ils proposent des opérations d'insertion et de suppression de cellules ainsi qu'une opération de simplification permettant de conserver la cohérence du modèle. À partir de ces opérations élémentaires, ils définissent les opérations de plus haut niveau que sont la division et la fusion de régions. La Figure 1.25a donne un exemple de cette représentation pour une image 3D.

Les g-brins impliquent que la géométrie joue un rôle important dans le modèle représentant une partition. Cela provoque des incohérences comme le montre la Figure 1.25b. La partition illustrée est topologiquement équivalente à celle présentée dans la Figure 1.25a. Du fait de la définition des g-brins, il y a une arête et un sommet supplémentaires dans le modèle : les deux représentations ne sont pas isomorphes. Cette structure permet de réaliser des parcours sur les surfaces de la partition. Cependant, comme la représentation d'une partition n'est pas unique, le modèle ne permet pas d'identifier directement deux images topologiquement identiques. D'autre part, ce modèle ne permet pas de représenter des régions qui comportent au moins deux voxels 18-adjacents comme la région illustrée dans la Figure 1.4c, page 10. En effet, la géométrie des g-brins ne permet pas sur ce type de régions de différencier l'intérieur de l'extérieur de la région.

## 1.3 Conclusion

Dans ce chapitre, nous avons présenté ce que sont les images 3D et comment les partition en régions des images sont représentés dans les modèles existants. Parmis ces différents modèles, nous avons identifié la catégorie des modèles topologiques qui permettent de représenter les régions et les relations d'adjacence et d'imbrication entre les régions. Ces modèles sont bien adapté pour représenter les informations importantes nécessaire pour le traitement d'image. Dans notre travail, nous utilisons un modèle topologique particulier appelé *carte topologique* qui est détaillé dans le Chapitre 2.

CHAPITRE 2

# CARTE TOPOLOGIQUE 3D

La carte topologique 3D est une structure combinatoire qui représente les frontières de la partition en régions d'une image 3D. Cette structure, développée d'abord en 2D puis en 3D par Guillaume Damiand [Dam01, Dam08b], propose une représentation unifiant la géométrie et la topologie pour les images dans le but d'améliorer les algorithmes de traitement et d'analyse. Les cartes topologiques sont une extension des cartes combinatoires utilisant la représentation des frontières en intervoxels comme plongement géométrique pour les cellules. Cette structure représente l'ensemble des cellules du bord des régions et permet de conserver l'ensemble des relations d'adjacences et d'incidences entre ces cellules.

Nous présentons la définition des cartes topologiques 3D. Nous détaillons la construction par niveau liée à la contrainte de minimalité du modèle. Nous expliquons brièvement les problèmes de déconnexion qui se posent et nous détaillons les solutions retenues pour les corriger. Enfin les différentes parties d'un algorithme d'extraction permettant de construire la carte topologique à partir d'une image sont présentées. Cet algorithme permet d'obtenir la carte topologique représentant la partition en régions d'une image en fonction d'un critère.

## 2.1 Définition d'une carte topologique

Une carte topologique est composée de trois structures de données :
- une carte combinatoire minimale (en nombre de cellules) représentant les cellules du bord des régions et les relations d'adjacence et d'incidence entre ces cellules ;
- une matrice représentant les éléments intervoxels appartenant au plongement des cellules de la carte combinatoire ;
- un arbre de régions représentant la relation d'imbrication.

Ces trois structures de données représentent l'ensemble des cellules appartenant aux frontières de la partition sans perdre d'informations topologiques quant à l'adjacence ou l'imbrication entre les régions. Ces trois structures sont détaillées dans la suite de cette section.

### 2.1.1 Carte combinatoire minimale

Une carte combinatoire représente toutes les cellules de la subdivision en intervoxels de l'image. Avec les relations entre les brins, nous conservons l'ensemble des relations d'adjacence et d'incidence entre les cellules. Cependant, une carte topologique ne représente pas la subdivision en intervoxel de l'image mais la subdivision de l'image en régions. Ainsi une propriété de minimalité est définie afin de

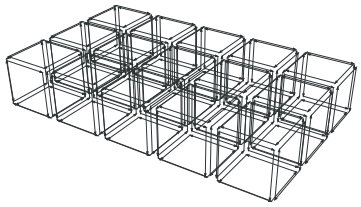


FIG. 2.1: Carte de niveau 0 représentant une image de taille $5 \times 3 \times 1$. C'est une représentation partielle, nous ne représentons pas la région infinie.

minimiser le nombre de cellules utilisées dans la représentation de l'image. Cette minimalité peut être introduite sous la forme d'une définition en quatre niveaux allant du niveau maximal, le niveau 0, au niveau minimal utilisé par la carte topologique, le niveau 3. Le passage d'un niveau à l'autre est réalisé en utilisant les opérations de suppression de cellules (voir Section 1.2.4, page 21).

### 2.1.2 Carte de niveau 0

La carte de niveau 0 est le point de départ de la minimisation : elle décrit l'ensemble des éléments intervoxels d'une image. Cette carte est définie par la Définition 10. La Figure 2.1 montre une carte combinatoire de niveau 0 représentant une image de taille $5 \times 3 \times 1$. Dans cette carte, chaque voxel est représenté par un cube. Nous ne représentons pas la région infinie.

**Définition 10** (Carte de niveau 0)**.** *La carte de niveau 0 correspondant à une image $I$ ayant $|I_d|$ voxels est une carte contenant $|I_d|$ cubes plus un volume représentant la région infinie qui entoure l'image. Chaque cube représente un voxel et est cousu par $\beta_3$ avec les cubes représentant ses voxels 6-voisins.*

La carte de niveau 0 ne décrit pas la partition de l'image définie par le critère d'homogénéité des régions de la partition puisqu'elle représente tous les voxels de l'image. Afin de faire correspondre la partition en volumes de la carte combinatoire avec la partition définie par le critère, nous devons simplifier le modèle en supprimant les faces qui séparent des voxels appartenant à la même région.

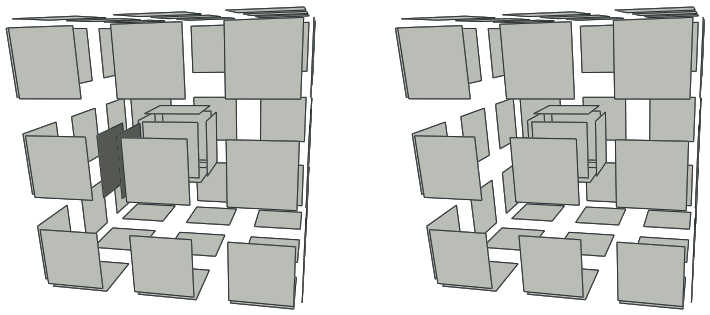
### 2.1.3 Carte de niveau 1

En utilisant l'opération de suppression de faces, nous retirons les faces intérieures de la carte de niveau 0 (c'est-à-dire les faces qui ne séparent pas deux voxels de régions différentes d'après le critère) afin d'obtenir le niveau 1 (voir Définition 11).

**Définition 11** (Carte de niveau 1)**.** *La carte de niveau 1 est construite à partir du niveau 0 en supprimant les faces qui séparent deux voxels appartenant à la même région d'après le critère.*

Durant la transformation du niveau 0 au niveau 1, la suppression d'une face peut entraîner une déconnexion entre deux surfaces de la carte. C'est le cas lorsqu'un ensemble de régions est imbriqué dans une autre région (voir exemple Figure 2.2). Le bord de la région englobante est alors composé de plusieurs surfaces déconnectées faisant perdre la relation d'imbrication entre les régions. Ce problème est résolu grâce à l'arbre des régions qui représente cette relation.

La Figure 2.3 montre une carte combinatoire de niveau 1 représentant la partition d'une image de taille $5 \times 3 \times 1$ en trois régions. Les faces séparant deux voxels d'une même région dans la carte de niveau 0 sont supprimées en utilisant l'opération de 2-suppression (Algorithme 2).



(a) (b)

FIG. 2.2: Exemple de déconnexion de surfaces : (a) configuration d'origine, la face en gris foncé est supprimée ; (b) configuration résultat, les deux surfaces du bord de la région sont déconnectées.

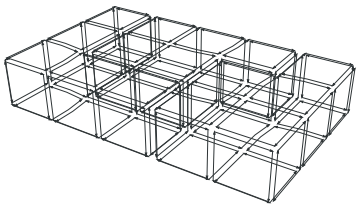
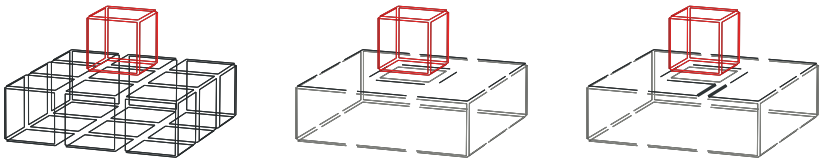


FIG. 2.3: Carte de niveau 1 représentant la partition d'une image de taille $5 \times 3 \times 1$ en trois régions. Par rapport au niveau 0, de la Figure 2.1, les faces qui séparent deux voxels d'une même région sont supprimées.



(a) (b) (c)

FIG. 2.4: Exemple de déconnexion de faces : (a) configuration d'origine, des arêtes de degré deux sont supprimées (pas toutes pour conserver la lisibilité du schéma), et notamment les arêtes de la face supérieure du volume du bas ; (b) la face supérieure a deux bords, il y a déconnexion ; (c) la dernière arête reliant les deux bords n'est pas supprimée mais transformée en arête fictive, il n'y a pas de déconnexion.

La carte de niveau 1 décrit la partition de l'image définie par le critère. Il n'existe plus de face intérieure dans ce niveau de la carte combinatoire. Cependant, elle contient beaucoup de brins qui représentent la même information concernant l'adjacence entre les régions, nous pouvons encore simplifier le modèle.

### 2.1.4 Carte de niveau 2

Le passage du niveau 1 au niveau 2 est une étape de simplification. Afin de diminuer le nombre de brins utilisés pour représenter la partition, nous réduisons le nombre de cellules qui décrivent les frontières intervoxels de la partition. Nous commençons donc par fusionner les faces en supprimant les arêtes de degré local 2 car deux faces incidentes à une arête de degré local 2 séparent les deux mêmes régions et représentent la même relation d'adjacence.

Lorsque nous supprimons une arête de degré 2, il est possible qu'une déconnexion des arêtes appartenant au bord des faces se produise : il s'agit de la déconnexion des bords de la face que nous appelons plus généralement déconnexion de face (voir exemple Figure 2.4). Lors de ces déconnexions, nous perdons une information sur les relations entre les bords de la face. Aussi nous introduisons la notion d'arêtes fictives par la Définition 12.

**Définition 12** (Arête fictive)**.** *Une arête fictive est une arête de degré 1 dont la suppression induit une déconnexion ou supprime complètement une face.*

La seconde condition concernant la suppression totale d'une face est nécessaire pour éviter la suppression des faces ne comportant qu'une seule arête (cas des sphères par exemple). La suppression de ce type d'arête provoque la disparition de la face qui engendre alors la disparition d'une information d'adjacence entre les deux régions incidentes. Avec les arêtes fictives les bords de chaque face sont connectés, la face reste équivalente au disque unitaire : le problème de la déconnexion de face est résolu. Les arêtes fictives n'ont pas de plongement géométrique. En effet, la localisation géométrique de l'arête dans la face peut être quelconque et ne contient pas d'information supplémentaire aussi nous choisissons de ne pas représenter leurs géométries plutôt que d'attribuer arbitrairement un plongement. Par opposition, une arête qui n'est pas fictive est dite réelle.

Le niveau 2 est défini d'après la Définition 13. Avec cette définition nous sommes certains que les arêtes fictives ne sont pas supprimées et qu'il n'y a donc pas de déconnexion de face.

**Définition 13** (Carte de niveau 2)**.** *La carte de niveau 2 est la carte obtenue à partir du niveau 1 en supprimant successivement chaque arête de degré local 2 qui n'induit pas de déconnexion de face ou qui ne supprime pas complètement une face.*

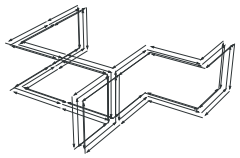


FIG. 2.5: Carte de niveau 2 extraite à partir de la carte de niveau 1 (Figure 2.3) en supprimant les arêtes incidentes à deux faces qui séparent les deux mêmes régions.

Un exemple de carte de niveau 2 est présenté par la Figure 2.5. Les arêtes séparant deux faces incidentes aux mêmes régions sont supprimées en utilisant l'opération de 1-suppression (Algorithme 3). Dans cet exemple, aucune arête n'entraîne de déconnexion de face, il n'y a pas d'arête fictive.

La carte de niveau 2 représente toujours la partition de l'image et ne contient pas d'arête superflue pour représenter la partition de l'image. Cependant, il reste des brins inutiles qui sont supprimés lors du passage au niveau suivant.

### 2.1.5 Carte de niveau 3

La carte de niveau 3 est construite à partir du niveau 2 en supprimant les sommets de degré 2 qui ne représentent pas d'informations d'adjacence entre les régions de la partition. Nous conservons les sommets incidents à des arêtes formant des boucles car la suppression d'un tel sommet fait disparaître la face incidente et supprime donc une information d'adjacence entre les régions. Ces sommets, à la manière des arêtes fictives, n'ont pas de plongement défini. Ils peuvent se situer n'importe où sur l'arête. La Définition 14 donne la définition d'une carte combinatoire minimale telle qu'elle est utilisée par la carte topologique.

**Définition 14** (Carte de niveau 3)**.** *La carte de niveau 3 est obtenue à partir de la carte de niveau 2 en supprimant successivement :*

- *les sommets de degré 2 incidents à deux arêtes qui ne sont pas des boucles en décalant au besoin les arêtes fictives ;*
- *les sommets incidents uniquement à des arêtes fictives après avoir décalé les arêtes de sorte que le sommet soit de degré 2.*

Dans la Définition 14, nous indiquons qu'il faut parfois décaler les arêtes fictives avant de pouvoir supprimer un sommet. Si nous ne décalons pas les arêtes fictives, le degré des sommets n'est pas compatible avec leur suppression et nous n'obtenons pas un modèle minimal. Le décalage d'une arête fictive est toujours possible puisque les sommets d'une arête fictive peuvent se situer n'importe où sur le bord des faces. Plus d'informations sur les arêtes fictives peuvent être trouvées dans [Dam01, Dam08b]. La Figure 2.6 donne un exemple de décalage d'arêtes fictives. Le sommet incident au brin $b_4$ est incident à deux arêtes réelles (arêtes incidentes à $b_3$ et $b_4$) et une arête fictive (incidente à $b_1$) représentée en pointillés. L'arête est décalée : les brins $b_1$ et $b_2$ sont insérés entre $b_5$ et $b_3$ (reliés par $\beta_1$) et les brins $b_3$ et $b_4$ sont reliés par $\beta_1$. Le sommet incident à $b_4$ est maintenant de degré deux et peut donc être supprimé par l'opération de 0-suppression.

La Figure 2.7 présente la carte combinatoire de niveau 3 représentant la partition de l'image. Cette carte est obtenue à partir du niveau 2 en supprimant les sommets qui sont incidents à deux arêtes qui ne sont pas des boucles et qui séparent les mêmes faces. Les sommets sont supprimés par l'opération de 0-suppression définie par l'Algorithme 4.

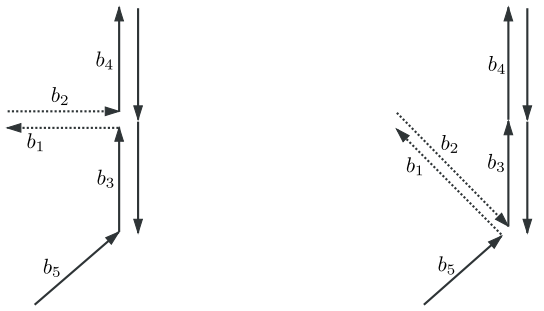



FIG. 2.6: Exemple de décalage d'une arête fictive pour pouvoir supprimer un sommet (pour plus de lisibilité seule une partie des brins est dessinée) : (a) configuration d'origine, un sommet est incident à deux arêtes réelles et une arête fictive ; (b) l'arête est décalée sur un sommet adjacent, le sommet initial est de degré local deux : il peut être supprimé.

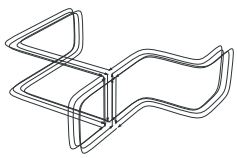
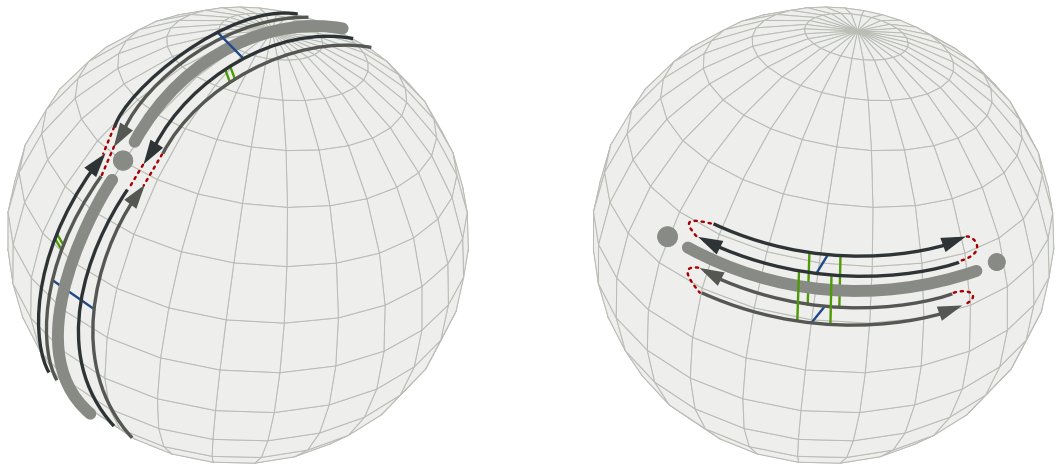


FIG. 2.7: Carte de niveau 3 extraite à partir de la carte de niveau 2 (Figure 2.5) en supprimant les sommets incidents à deux arêtes qui ne sont pas des boucles et qui séparent les deux mêmes régions.



(a) (b)

FIG. 2.8: Deux représentations minimales en nombre de cellules d'une sphère : (a) avec deux faces, une arête et un sommet ; (b) avec une face, une arête et deux sommets (représentation standard).

La carte de niveau 3 est la carte combinatoire minimale en nombre de cellules qui représente la partition de l'image d'après le critère d'homogénéité des régions. Une preuve de ce résultat peut être trouvée dans [Dam08b].

Dans la suite de ce travail, nous utilisons le terme d'oracle pour symboliser la fonction permettant de tester l'homogénéité des régions dans la partition.

### 2.1.6 Représentation des sphères

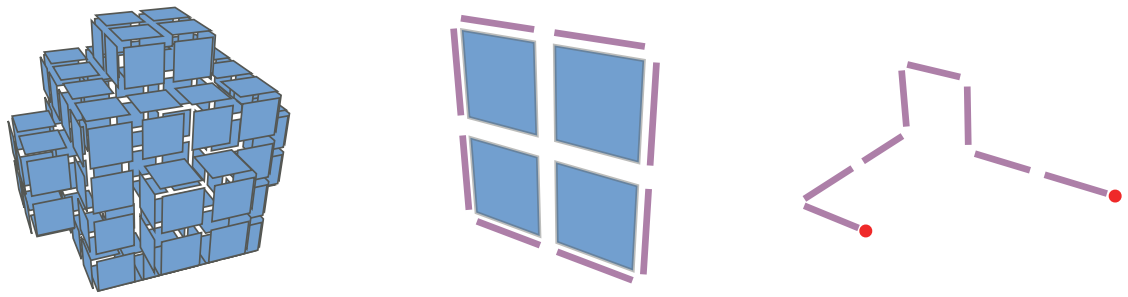
Avec cette définition par niveau, la représentation de toutes les partitions est minimale en terme de nombre de cellules. Cependant les sphères topologiques admettent deux représentations duales comportant le même nombre de cellules : d'une part celle utilisant deux faces, une arête et un sommet, illustrée dans la Figure 2.8a, et d'autre part celle utilisant une face, une arête et deux sommets, présentée dans la Figure 2.8b.

L'ordre des suppressions des cellules dans la définition par niveau de la carte combinatoire minimale conduit toujours à représenter les sphères en utilisant la deuxième solution. Cette représentation est minimale en nombre de cellules de dimension décroissante. Ainsi la carte topologique représente une partition de l'image en utilisant le minimum de faces, puis le minimum d'arêtes, puis enfin le minimum de sommets. Il s'agit par convention de la représentation standard des sphères dans le modèle des cartes topologiques.

### 2.1.7 Matrice des éléments intervoxels

La matrice des éléments intervoxels (ou matrice intervoxels) permet de représenter la géométrie des frontières de la partition. Elle contient tous les pointels, lignels et surfels appartenant à la subdivision de l'image dans le cadre de l'intervoxel. Chaque élément unitaire de l'intervoxel est dans l'un des deux états possibles : allumé ou éteint. Nous utilisons cet état afin de représenter la géométrie des cellules de la carte topologique dans la matrice intervoxel. L'état d'un élément unitaire dépend des éléments voisins :

– un surfel est allumé lorsqu'il sépare deux voxels appartenant à deux régions différentes ;



(a) (b) (c)

FIG. 2.9: Différentes cellules représentées par des éléments unitaires de l'intervoxel : (a) une face sphérique représentée par un ensemble de surfels, il n'y a pas de bord à cette face donc pas de lignels ; (b) une face représentée par un ensemble de surfels. Elle est bordée par une arête représentée par un ensemble de lignels. Comme l'arête n'a pas de bord, il n'y a pas de pointels ; (c) une arête représentée par un ensemble de lignels dont le bord (deux sommets) est représenté par deux pointels.

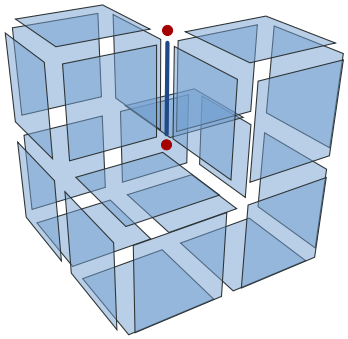


FIG. 2.10: Configuration où les deux pointels sont allumés alors qu'ils ne sont incidents qu'à un seul lignel. Les pointels représentent le plongement des sommets du bord de l'arête.

– un lignel est allumé lorsqu'il est incident à au moins trois surfels allumés ;
– un pointel est allumé lorsqu'il est incident à un ou à au moins trois lignels allumés.

Ainsi les éléments allumés de l'intervoxel décrivent les cellules appartenant aux frontières de la partition de l'image en régions. Nous définissons donc les faces, les arêtes et les sommets dans la matrice intervoxel en utilisant des ensembles d'éléments unitaires. Une face est représentée par un ensemble connexe de surfels qui ne sont pas séparés par des lignels. Une arête est un ensemble connexe de lignels qui ne sont pas séparés par des pointels et enfin un sommet est représenté par un pointel. La Figure 2.9 donne trois exemples de cellules représentées à l'aide d'une matrice intervoxel et la Figure 2.10 présente un exemple du cas particulier dans lequel un pointel incident à un seul lignel est allumé. Les cellules frontières de la carte combinatoire de la Figure 2.7 sont représentées par les éléments unitaires dessinés dans la Figure 2.11.

Les éléments allumés forment le plongement de la carte combinatoire minimale. Afin de faire le lien entre la topologie représentée par la carte combinatoire et la géométrie représentée par la matrice intervoxel, chaque brin de la carte est associé à un triplet. Un triplet désigne trois éléments unitaires incidents dans la matrice intervoxel : un pointel, un lignel et un surfel. La Figure 2.12 montre un exemple de triplet $t = (p, l, s)$. Les trois éléments définissent une orientation pour le surfel (par le

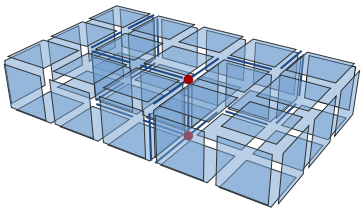
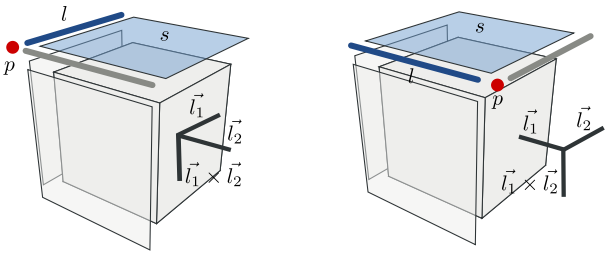


FIG. 2.11: Matrice intervoxel représentant le plongement de la carte de niveau 3 (Figure 2.7) avec des éléments unitaires.




(a) (b)

FIG. 2.12: Deux exemples de triplets $(p, l, s)$. Dans les deux cas, l'orientation du surfel est définie par le vecteur $\vec{l_1} \times \vec{l_2}$.

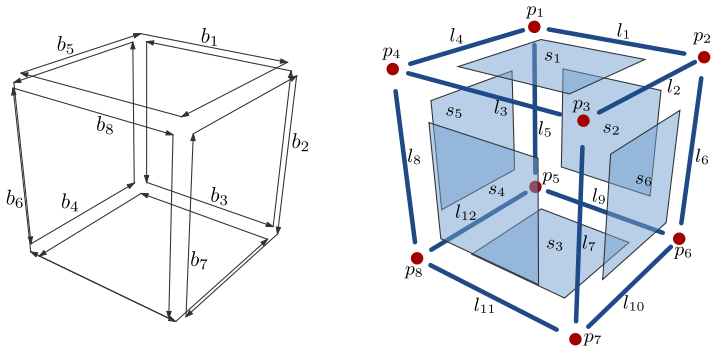
produit vectoriel $\vec{l_1} \times \vec{l_2}$ où $\vec{l_1}$ est le vecteur décrit par le pointel $p$ et le lignel $l$ et $\vec{l_2}$ est le vecteur décrit par le pointel $p$ et l'autre lignel incident à $p$ et $s$). Ainsi pour chaque brin $b$, le pointel désigné par le triplet est le plongement du sommet incident à $b$. Le lignel appartient au plongement de l'arête incidente à $b$, et le surfel appartient au plongement de la face incidente à $b$. Le voxel incident au surfel et situé dans la direction donnée par l'orientation du triplet appartient par convention à l'intérieur de la région incidente à $b$. Nous observons que les cellules de l'intervoxel associées à un brin de la carte combinatoire ne sont pas nécessairement allumées dans la matrice intervoxel : c'est par exemple le cas pour les surfaces composées d'arêtes fictives comme les sphères ou les tores (surfaces isolées). C'est également le cas pour les sommets incidents à une seule arête qui forme une boucle. Cette configuration se trouve illustrée par la 2.9b, le sommet qui forme le bord de l'arête qui entoure la face n'a pas de plongement allumé, tous les pointels sont éteints. Lorsque la géométrie n'est pas significative, elle n'est pas représentée comme dans l'exemple précédent où la position du sommet sur l'arête est quelconque.

La Figure 2.13 présente la carte combinatoire représentant un cube (Figure 2.13a) et la configuration des éléments intervoxels autour d'un voxel (Figure 2.13b). La Table 2.1 établit le lien entre les brins pris en exemple dans la Figure 2.13 et les triplets représentant leurs plongements. Les autres brins sont associés de manière similaire aux triplets correspondants.

### 2.1.8 Arbre des régions

L'arbre des régions représente la relation d'imbrication entre les régions de l'image. Les nœuds de l'arbre représentent les régions de l'image et les arêtes définissent la relation d'imbrication. En pratique, nous utilisons une structure arborescente pour représenter l'information d'imbrication.

Chaque région $r$ connaît sa région englobante, c'est-à-dire la région dans laquelle $r$ est directement imbriquée. Cette région est notée $pere(r)$.




(a) (b)

FIG. 2.13: Exemple donnant les triplets associés aux brins d'un cube. (a) Cube contenant un voxel représenté dans une carte combinatoire. (b) Éléments de l'intervoxel autour d'un voxel.

TAB. 2.1: Quelques triplets pour les brins représentés par la Figure 2.13.

| Brin | Triplet | Brin | Triplet |
|---|---|---|---|
| $b_1$ | $(p_1, l_1, s_1)$ | $b_5$ | $(p_4, l_4, s_1)$ |
| $b_2$ | $(p_2, l_6, s_6)$ | $b_6$ | $(p_8, l_8, s_4)$ |
| $b_3$ | $(p_5, l_9, s_2)$ | $b_7$ | $(p_7, l_7, s_6)$ |
| $b_4$ | $(p_8, l_{12}, s_5)$ | $b_8$ | $(p_4, l_3, s_4)$ |

Afin d'avoir la relation dans l'autre sens, une région $r$ connaît l'ensemble des régions qui sont directement imbriquées dans $r$. Ces régions sont regroupées en composantes connexes. En effet, toutes les régions imbriquées dans $r$ et appartenant à une même composante 18-connexe appartiennent à une même cavité de la région englobante. Dans l'arbre d'imbrication, nous regroupons toutes ces régions dans un ensemble que nous appelons une composante connexe ou $CC$. La Définition 15 introduit la notion de région représentante d'une composante connexe.

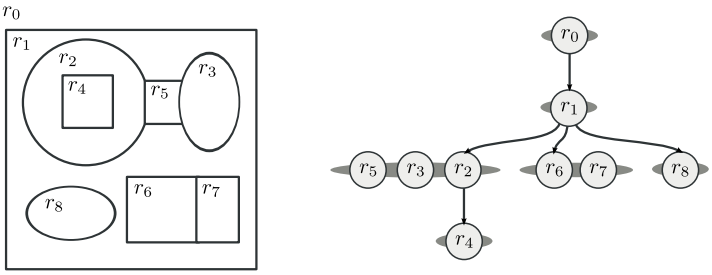
**Définition 15** (Région représentante d'une composante connexe)**.** *La région représentante d'une composante* 18*-connexe de régions* $CC$ *est notée* $\min(CC)$*. Il s'agit de la plus petite région de la composante connexe :* $\min(CC) = min(r, r \in CC)$.

À partir d'une région $r$, nous accédons à l'ensemble des régions appartenant à la même composante connexe. Nous connaissons également sa région englobante et nous possédons la liste des régions représentantes d'une composante connexe imbriquée. L'ensemble des régions représentantes imbriquées dans une région $r$ est noté *fils*$(r)$. La composante connexe à laquelle une région $r$ appartient est notée *CC*$(r)$.

La Figure 2.14 illustre l'arbre d'imbrication de la partition d'une image 2D. Nous observons comment les neuf régions sont regroupées en six composantes connexes représentées par les ellipses grises. Toutes les régions reliées par une même ellipse appartiennent à la même composante connexe de régions. Les flèches représentent la relation d'imbrication entre une région $r$ et les régions représentantes des composantes connexes imbriquées *fils*$(r)$.

Afin de lier les volumes de la carte combinatoire avec les régions qu'ils représentent, nous associons à chaque brin sa région d'appartenance. Si un brin $b$ appartient à une région $r$ alors *region*$(b) = r$. Tous les brins représentant les bords d'un même volume ont la même région d'appartenance, et deux brins représentant les bords de volumes différents ne peuvent pas avoir la même région d'appartenance.

Pour avoir l'information inverse, c'est-à-dire retrouver le volume dans la carte combinatoire à partir de l'arbre des régions, chaque région connaît un brin particulier que l'on appelle brin représentant. Le




(a) (b)

FIG. 2.14: Un arbre d'imbrication des régions d'une image 2D : (a) partition d'une image en neuf régions ; (b) arbre d'imbrication de la partition. Les flèches représentent les relations d'imbrication entre une région et les régions représentantes des composantes connexes de régions imbriquées. Les régions reliées par une même ellipse grise appartiennent à la même composante 18-connexe. Pour ne pas surcharger la représentation de l'arbre, nous ne représentons pas pour chaque région $r$ la relation *pere*$(r)$. Il est possible de retrouver la relation en utilisant les relations de fils et de composantes connexes.

brin représentant d'une région respecte la Définition 16 : il doit être incident à une face qui sépare sa région d'appartenance d'une région plus petite.

**Définition 16** (Brin représentant d'une région)**.** *Le brin représentant d'une région $r$, noté* $\text{rep}(r)$*, est un brin appartenant à la région $r$ tel que* $\text{region}(\beta_3(\text{rep}(r))) < r$.

D'après le Lemme 1, page 11, cela implique que le brin représentant d'une région est situé sur la surface externe de la région. Nous faisons ainsi le lien entre la surface externe d'une région représentée dans la carte combinatoire et la région dans l'arbre d'imbrication. Cependant cette étape n'est pas suffisante pour retrouver les informations perdues à cause des déconnexions de surfaces qui se produisent lors du passage de la carte de niveau 0 à la carte de niveau 1. Les surfaces internes correspondant aux cavités de la région ne sont pas atteignables. Nous utilisons l'arbre d'imbrication des régions afin de récupérer cette information.

**Proposition 1.** *Étant donnée une région $r$ qui entoure une composante connexe de régions $CC$, un brin de la surface interne entourant les régions de $CC$ est le brin* $\beta_3(\text{rep}(\min(CC)))$.

*Démonstration.* $\min CC$ est la région représentante de la composante connexe. D'après la Définition 15, il s'agit de la plus petite région dans la composante connexe. Or la Définition 16 précise que le brin représentant d'une région doit appartenir à une face qui sépare la région d'une région plus petite. La seule région plus petite que $\min CC$ est la région englobante de la composante connexe $CC$ d'après le Lemme 1. Ainsi *rep*$(\min(CC))$ appartient à une face qui sépare la région $\min CC$ de la région englobante $r$. L'opérateur $\beta_3(b)$ permet d'obtenir le brin appartenant à la même arête et la même face que $b$ mais pas au même volume. Ainsi le brin $\beta_3($*rep*$(\min(CC)))$ appartient au volume de $r$. Il s'agit donc d'un brin appartenant à la surface interne de $r$ qui entoure les régions de la composante connexe $CC$. □

Avec la Proposition 1, nous définissons l'Algorithme 5. Il prend en paramètres une carte topologique $M$ et une région $r$ et permet de parcourir l'ensemble des brins appartenant au bord d'une région, noté *brins*$(r)$ et défini par :

$$\textit{brins}(r) = \langle\beta_1, \beta_2\rangle(\textit{rep}(r)) \cup \bigcup_{r' \in \textit{fils}(r)} \langle\beta_1, \beta_2\rangle(\beta_3(\textit{rep}(r')))).$$

Nous retrouvons ainsi les informations sur l'imbrication des régions perdues lors du passage de la carte de niveau 0 à la carte de niveau 1.

**Algorithme 5** : Parcours des brins du bord d'une région

**Données** : Une carte combinatoire minimale $M$ ;
Une région $r$.
**Résultat** : Parcours des brins de la région $r$.

**pour chaque** $b \in \langle \beta_1, \beta_2 \rangle(\text{rep}(r))$ **faire**
  `// b est un brin de la surface externe`
**pour chaque** $r' \in \text{fils}(r)$ **faire**
  **pour chaque** $b \in \langle \beta_1, \beta_2 \rangle(\beta_3(\text{rep}(r')))$ **faire**
    `// b est un brin de la surface interne autour de CC(r')`

Afin de retrouver l'ensemble des voxels appartenant à une région $r$, noté *voxels*$(r)$, nous utilisons le mécanisme suivant. Soit $r$ une région dans une matrice $M$, nous obtenons un voxel de l'image en utilisant les coordonnées du voxel désigné par le triplet associé au brin représentant de la région $r$. En effet, d'après la Section 2.1.7, le voxel $v$ pointé par le triplet associé à un brin $b$ est situé à l'intérieur de la surface incidente au brin $b$. L'ensemble des voxels de la région $r$ est alors l'ensemble des voxels $v_i$ tel qu'il existe au moins un 6-chemin allant de $v$ à $v_i$ tels que deux voxels adjacents dans le chemin soient incidents à un même surfel éteint dans la matrice intervoxel. Nous pouvons retrouver tous ces voxels en utilisant un algorithme de croissance de région à partir de $v$.

Les cartes topologiques sont destinées à mettre en œuvre des opérations d'analyse d'images. Pour cela, nous relions le modèle avec les données de l'image : cette information est stockée dans les cellules par exemple au niveau des régions ou au niveau des faces. Les informations que nous stockons dans chaque cellule (régions, faces, etc.) sont des informations que nous pouvons calculer grâce à l'image ou au modèle des cartes topologiques, mais leur stockage permet d'y accéder en temps constant. Par exemple, dans une face, nous stockons le nombre de surfels appartenant au plongement de la face. Dans la suite, nous donnons quelques exemples de données qui sont stockées dans les régions.

#### Premier voxel

Comme nous souhaitons pouvoir comparer les régions d'après l'ordre de balayage de l'image, nous avons besoin des coordonnées du premier voxel de la région. Nous stockons alors cette information pour chaque région de la carte topologique.

#### Nombre de voxels

Le nombre de voxels d'une région permet d'évaluer sa taille, c'est un paramètre très courant dans les applications d'analyse d'images. Nous stockons donc la valeur de $|\textit{voxels}(r)|$ pour chaque région $r$.

#### Somme des intensités

Les images que nous utilisons sont généralement des images en niveaux de gris où le niveau de gris représente l'intensité d'un voxel. L'intensité d'un voxel $v$ dans une image $I = (I_d, I_f)$ est obtenue par la valeur $I_f(v)$.

Nous stockons ainsi pour chaque région $r$ la somme des intensités :

$$\sum_{v \in \mathit{voxels}(r)} I_f(v).$$

Nous pouvons ainsi calculer la moyenne des intensités que nous notons :

$$A(r) = \frac{1}{|\mathit{voxels}(r)|} \sum_{v \in \mathit{voxels}(r)} I_f(v).$$

Nous stockons également la somme des carrés des intensités :

$$\sum_{v \in \mathit{voxels}(r)} I_f(v)^2.$$

Nous pouvons ainsi calculer la variance des intensités de la région $r$ :

$$Var(r) = \sum_{v \in \mathit{voxels}(r)} I_f(v)^2 - (\sum_{v \in \mathit{voxels}(r)} I_f(v))^2.$$

#### Autres données

En fonction des besoins définis par l'utilisateur, nous pouvons ajouter des données que nous stockons afin d'être plus efficace lorsque nous réalisons des opérations de traitement d'images. Le modèle peut donc s'adapter facilement à de nouveaux algorithmes d'analyse d'images.

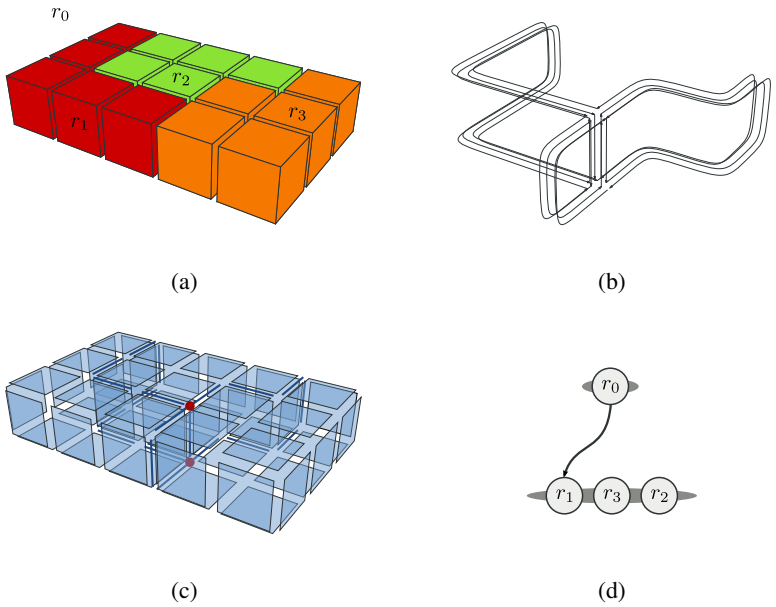
### 2.1.9 Convention de représentations

L'illustration d'une carte topologique 3D, par exemple pour expliquer les algorithmes, n'est pas une tâche aisée. Les limitations liées à la résolution, à la vue en perspective et au grand nombre de brins et éléments unitaires intervoxels à représenter entraînent une faible lisibilité des schémas.

Une première représentation consiste à visualiser séparément les trois structures de données (voir Figure 2.15). Nous utilisons un plongement géométrique pour dessiner la carte combinatoire afin de distinguer les différentes cellules représentées. Nous ne représentons généralement pas la surface de la région infinie pour ne pas surcharger le dessin. Dans ce type de représentation, nous ne faisons pas apparaître les relations $\beta_2$ et $\beta_3$ de manière explicite. Les éléments allumés dans la matrice intervoxel sont représentés en perspective en utilisant le même point de vue que pour la carte combinatoire. Les surfels sont représentés par des carrés, les lignels par des segments et les pointels par des points. Enfin l'arbre des régions est dessiné en utilisant la représentation détaillée utilisée dans la Section 2.1.8.

Cette représentation est difficile à utiliser lorsque le nombre de brins augmente ou lorsque les relations entre les régions sont plus complexes. Nous utilisons alors une représentation directe des cellules de la carte topologique sans détailler les brins et les relations entre ces brins. Cette représentation est utilisée pour expliquer des algorithmes et nous en trouvons deux exemples dans la Figure 2.16.

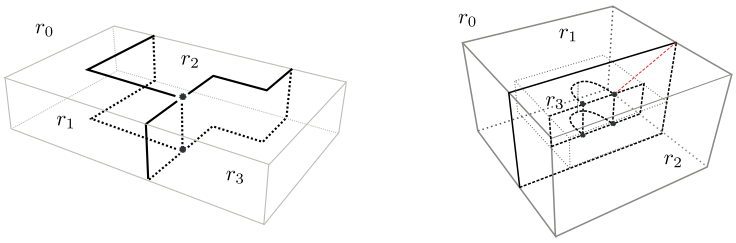
Pour montrer des résultats de segmentation, nous présentons classiquement une vue en coupe de l'image en utilisant une image étiquetée dans laquelle chaque étiquette est dessinée avec une couleur différente. Les vues en coupe peuvent également être représentées en superposant les cellules de l'intervoxel à la vue de l'image en coupe. La Figure 2.17 illustre les deux types de représentation d'une coupe d'une partition.




(a) (b)

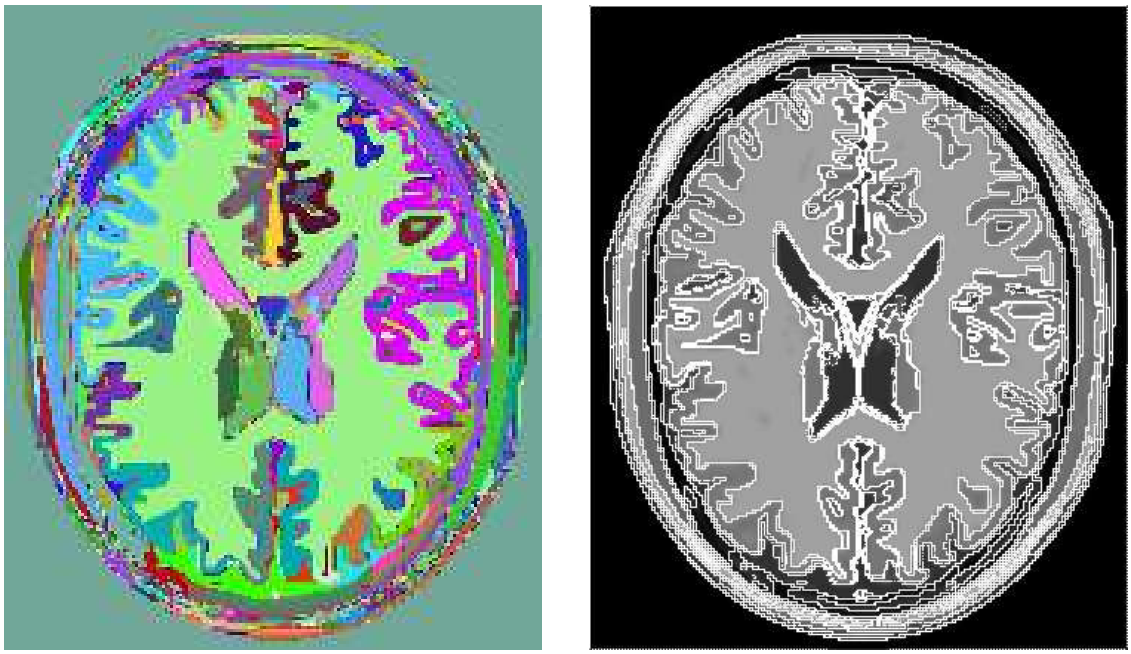
(c) (d)

FIG. 2.15: Illustration des 3 structures de données d'une carte topologique représentant une image : (a) image 3D avec quatre régions ; (b) carte combinatoire minimale représentant la partition ; (c) matrice intervoxel représentant les frontières des régions ; (d) arbre d'imbrication des régions.




(a) (b)

FIG. 2.16: Exemple de représentation en utilisant les cellules : (a) représentation de la partition donnée Figure 2.15 ; (b) représentation de trois régions : les deux régions $r_1$ et $r_2$ sont adjacentes et entourent la région $r_3$. La région $r_3$ est un tore.



(a) (b)

FIG. 2.17: Exemple de représentation d'une vue en coupe d'une partition d'une image : (a) par une image étiquetée ; (b) par une superposition des cellules de l'intervoxel (dessiné en trait blanc).

## 2.2 Extraction d'une carte topologique à partir d'une image

L'opération permettant de construire la carte topologique représentant une partition d'une image s'appelle l'extraction. Afin d'obtenir une carte topologique à partir d'une image, l'approche naïve consiste à utiliser la définition en niveaux de la carte topologique. Ainsi la première étape construit la carte de niveau 0 et par simplifications successives nous obtenons la carte de niveau 3 qui représente la partition. Cependant, cette approche nécessite la construction du premier niveau qui contient de très nombreux brins (24 par voxels) et cette approche n'est pas réalisable sur un ordinateur de bureau lorsque nous travaillons avec des images de taille importante.

Guillaume Damiand, après avoir proposé la définition des cartes topologiques, définit un algorithme incrémental d'extraction d'une carte topologique permettant d'obtenir la carte topologique d'une image étiquetée en régions [Dam01, Dam08b].

Dans la suite, nous présentons les opérations nécessaires pour réaliser l'extraction d'une carte topologique à partir d'une partition d'une image. Nous discutons notamment de l'opération de construction de l'arbre d'imbrication à l'aide d'une carte combinatoire et d'une liste de régions. Nous proposons ensuite une version de l'algorithme d'extraction qui permet d'extraire la carte topologique d'une partition d'une image quelconque. La partition de l'image est définie par un critère représenté sous forme d'un oracle $Oracle : I_d \times I_d \rightarrow \{vrai, faux\}$ qui détermine si, oui ou non, deux voxels se trouvent dans une même région. Il s'agit d'une généralisation de l'algorithme original proposé par Guillaume Damiand. Le résultat de l'algorithme original peut être obtenu en utilisant comme oracle une fonction qui répond vrai lorsque les deux voxels ont la même étiquette et faux sinon.

### 2.2.1 Construction de l'arbre d'imbrication

Dans la définition en niveaux de la carte combinatoire minimale, nous observons que le passage d'une carte de niveau 0 à une carte de niveau 1 entraîne des déconnexions de surfaces. Ces surfaces

sont reliées entre elles par un arbre des régions permettant de retrouver les relations d'imbrication entre les régions.

L'Algorithme 6 présente l'opération de construction d'un arbre d'imbrication des régions. Il prend en paramètre une carte combinatoire minimale $M$ dont les brins sont étiquetés par régions et la liste des régions $R$ triée par ordre croissant. L'algorithme construit alors l'arbre d'imbrication des régions prenant la région infinie $r_0$ comme racine.

**Algorithme 6** : Construction de l'arbre d'imbrication des régions

**Données** : Une carte combinatoire minimale $M$ ;
La liste triée des régions $R$.
**Résultat** : Construction de l'arbre d'imbrication des régions.

Initialiser les régions de $R$;
*pere*$(r_0) \leftarrow r_0$;
**pour chaque** *région* $r \in R$ *tel que* pere$(r) = nil$ **faire**
  $b \leftarrow$ *rep*$(r)$;
  *pere*$(r) \leftarrow$ *region*$(\beta_3(b))$;
  **pour chaque** *brin* $b' \in \langle \beta_1, \beta_2, \beta_3 \rangle (\text{rep}(r))$ **faire**
    $r' \leftarrow$ *region*$(b')$;
    **si** pere$(r') = nil$ **alors**
      *pere*$(r') \leftarrow$ *pere*$(r)$;
      Ajouter la région $r'$ comme imbriquée dans *pere*$(r')$ dans la même composante connexe que $r$;

L'initialisation consiste à définir chaque région $r \in R$ comme étant isolée : nous ne précisons aucune région englobante (*pere*$(r) = nil$) ni régions imbriquées (*fils*$(r) = \emptyset$).

Nous utilisons alors un parcours simultané de la carte topologique et de la liste des régions afin de construire l'arbre d'imbrication des régions. Le principe de l'algorithme est que la première région non traitée, $r \in R$, est la première région d'une composante connexe imbriquée dans une région englobante. Cette région englobante est obtenue par l'opération *pere*$(r) \leftarrow$ *region*$(\beta_3($*rep*$(r)))$. Les régions de la composante connexe appartiennent à l'ensemble $CC = \{$*region*$(b), \forall d \in \langle \beta_1, \beta_2, \beta_3 \rangle ($*rep*$(r))\} \setminus$ *pere*$(r)$. Il s'agit de toutes les régions dont un brin fait partie de la composante connexe du brin représentant de $r$. Comme une surface interne de la région englobante apparaît dans cette composante connexe, nous retirons la région englobante de la composante connexe imbriquée. Nous mettons à jour les relations d'imbrications entre les régions en définissant *pere*$(r') \leftarrow$ *pere*$(r)$, $\forall r' \in CC$ et en ajoutant toutes les régions de $CC$ dans une composante connexe imbriquée dans *pere*$(r)$. La définition du brin représentant et la définition de l'ordre des régions assure que *pere*$(r) \neq nil$. Nous construisons donc bien un arbre en ajoutant des feuilles à l'arbre d'imbrication de racine $r_0$.

La première région $r_1$ de la liste $R$ est la première région de l'image. Sa région englobante est la région infinie $r_0$. Ainsi toutes les régions de la même composante connexe sont imbriquées dans $r_0$. La prochaine région non traitée est identifiée par le fait que *pere*$(r) = nil$. Ainsi lorsque toutes les régions sont traitées, toutes les relations d'imbrications sont représentées dans l'arbre d'imbrication.

Nous ne parcourons qu'une seule fois chaque composante connexe de brins. D'autre part, le nombre de brins dépend du nombre de régions, il y a au moins un brin par région dans la carte topologique. Le traitement sur les régions est donc majoré par le traitement des brins de la carte. D'après ces constatations, la complexité de l'algorithme de construction d'un arbre d'imbrication est en $O(|$*brins*$(M)|)$. Une preuve de l'algorithme de construction de l'arbre d'imbrication ainsi qu'une étude plus détaillée de la complexité sont disponibles dans [Dam08b].

### 2.2.2 Algorithme incrémental d'extraction d'une image 3D

L'algorithme incrémental d'extraction, donné par l'Algorithme 7, consiste à construire la carte topologique en traitant successivement chaque voxel au modèle lors d'un unique balayage de l'image. Il prend en paramètre une image 3D et une fonction $Oracle : I_d \times I_d \rightarrow \{vrai, faux\}$ et retourne la carte topologique représentant la partition de l'image définie par l'oracle.

**Algorithme 7** : Extraction incrémentale d'une carte topologique

**Données** : Une image 3D $I = (I_d, I_v)$ ;
Une fonction $Oracle : I_d \times I_d \rightarrow \{vrai, faux\}$.
**Résultat** : La carte topologique de la partition de $I$ définie par l'$Oracle$.

$M \leftarrow$ carte topologique vide;
$R \leftarrow$ liste de régions vide;
**pour** $k \in [0, n_3[$ **faire**
  **pour** $j \in [0, n_2[$ **faire**
    **pour** $i \in [0, n_1[$ **faire**
      $v \leftarrow$ voxel de coordonnées $(i, j, k)$;
      $v_1 \leftarrow$ voxel de coordonnées $(i-1, j, k)$;
      $v_2 \leftarrow$ voxel de coordonnées $(i, j-1, k)$;
      $v_3 \leftarrow$ voxel de coordonnées $(i, j, k-1)$;
      ajouter dans $M$ le cube représentant $v$;
      allumer le plongement des différentes cellules du cube dans la matrice intervoxel;
      $r \leftarrow$ ajouter dans $R$ une nouvelle région pour le cube;
      coudre le cube par $\beta_3$ avec les faces des voxels $v_1$, $v_2$ et $v_3$;
      **si** $Oracle(v, v_1$ *(resp.* $v_2$, $v_3)$**)** **alors**
        éteindre le surfel qui est le plongement de la face;
        supprimer la face entre $v$ et $v_1$ (resp. $v_2$, $v_3$);
        fusionner $r$ avec la région de $v_1$ (resp. $v_2$, $v_3$);
      **si** *l'arête incidente à* $v$, $v_1$ *et* $v_2$ *est de degré 1* **alors**
        éteindre le lignel qui est le plongement de l'arête;
        **si** *la suppression de l'arête entraîne une déconnexion* **alors**
          rendre l'arête fictive;
        **sinon**
          supprimer l'arête;
      idem pour l'arête incidente à $v$, $v_1$ et $v_3$;
      idem pour l'arête incidente à $v$, $v_2$ et $v_3$;
      décaler les arêtes fictives si besoin;
      **si** *le sommet incident à* $v$, $v_1$, $v_2$ *et* $v_3$ *est de degré deux et si les deux arêtes ne sont pas des boucles* **alors**
        éteindre le pointel qui est le plongement du sommet;
        supprimer le sommet;
Construire l'arbre d'imbrication des régions à partir de la carte $M$ et de la liste $R$;
**retourner** $M$;

L'invariant de l'algorithme certifie que tous les voxels plus petits sont déjà traités dans la carte topologique. Dans une image $I$ de taille $n_1 \times n_2 \times n_3$, lorsque nous traitons le voxel $v = (i, j, k)$, les voxels $v_1 = (i-1, j, k)$, $v_2 = (i, j-1, k)$ et $v_3 = (i, j, k-1)$ sont déjà traités. Nous ajoutons à la carte topologique un cube représentant le voxel. Nous ajoutons également une région $r$ qui est associée

aux brins du cube et qui contient les informations de la région initialisée par le voxel $v$. Nous relions le cube à la carte topologique déjà existante en cousant par $\beta_3$ les faces qui séparent $v$ des trois voxels existants.

Ensuite nous interrogeons l'oracle pour savoir si le voxel fait partie de la même région qu'un de ses trois voisins. Si c'est le cas, la face qui sépare les deux voxels est intérieure : nous la supprimons. Nous fusionnons alors la nouvelle région avec la région existante pour propager les informations de la région.

Lorsque nous avons supprimé toutes les faces possibles, nous simplifions localement la carte en supprimant les arêtes qui ne sont pas minimales et en rendant fictives les arêtes entraînant des suppressions ou des déconnexions de faces parmi les arêtes incidentes au voxel $v$. Nous supprimons le sommet incident aux trois voxels plus petit du 6-voisinage de $v$ s'il n'est pas minimal.

Une fois que tous les voxels ont été traités, la carte combinatoire est minimale et représente la partition de l'image fournie par l'oracle. Le plongement est également à jour. De plus la liste des régions $R$ a été construite et $R$ est triée par ordre croissant d'apparition dans l'image. Nous utilisons alors l'opération de construction de l'arbre d'imbrication (Algorithme 6, page 44) afin d'obtenir la carte topologique complète représentant la partition de l'image $I$ décrite par le critère défini par l'oracle.

La complexité de l'Algorithme 7 est linéaire en fonction du nombre de voxels de l'image puisque nous ne traitons chaque voxel qu'une seule et unique fois. Le nombre de brins est linéaire par rapport au nombre de voxels, tout comme le nombre de régions. L'algorithme de construction de l'arbre d'imbrication étant linéaire par rapport au nombre de brins, la complexité de l'algorithme d'extraction incrémental est en $O(|\mathit{voxels}(I)|)$. De plus amples détails ainsi qu'une preuve de correction de l'algorithme d'extraction se trouvent dans [Dam08b].

Nous disposons donc d'un modèle topologique de représentation d'images complet qui représente toutes les cellules d'une partition. La géométrie de ces cellules et les relations entre elles sont conservées dans le modèle. Avec cette structure nous représentons toute la topologie de la partition de l'image. L'opération d'extraction permet de construire une carte initiale représentant une partition de l'image définie par un oracle.

## 2.3 Conclusion

Dans ce chapitre, nous avons donné la définition des cartes topologiques 3D. Une carte topologique représente la partition d'une image en région à l'aide de trois structures de données : une carte combinatoire pour représenter la topologie, une matrice intervoxel pour représenter la géométrie et un arbre de régions pour représenter la relation d'imbrication. Nous avons également introduit les opérations permettant de construire la carte topologique d'une partition d'une image ainsi que l'opération de simplification. Dans le Chapitre 3, nous discutons de différentes approches existantes pour segmenter des images et nous introduisons les notions de topologies utilisées dans le cadre de ce travail.

CHAPITRE 3

# SEGMENTATION D'IMAGES

La segmentation d'images est l'une des opérations les plus importantes dans le domaine de l'analyse d'images. L'objectif de cette opération est de diviser l'image en zones homogènes appelées régions. Ainsi, la segmentation d'une image permet de retrouver des formes ou des zones ayant un sens pour l'utilisateur ou pour une autre opération de traitement d'images.

Les problèmes de segmentation et de regroupement forment un grand défi en vision par ordinateur. Depuis l'apparition de la théorie du Gestalt en psychologie, il est reconnu que le regroupement perceptuel joue un rôle important dans la perception visuelle humaine. De nombreux problèmes peuvent tirer parti d'images segmentées lorsque ces segmentations sont fiables et efficaces à calculer. Par exemple, les problèmes de reconnaissance d'images ou d'indexation profitent des résultats de segmentation notamment pour séparer les objets du fond ou reconnaître un objet par morceaux.

Segmenter une image est une opération où l'expertise de l'utilisateur est importante. Une personne souhaitant segmenter une image cherche à mettre en valeur certains éléments de l'image. Il maîtrise les informations qui lui semblent pertinentes pour faire le choix de la partition. Cette connaissance fait que la personne est généralement appelée un expert, capable de dire si une segmentation est bonne ou pas. Cependant différents experts vont sans doute proposer des segmentations différentes et un même expert confronté à une même image à plusieurs reprises ne produit pas toujours la même partition. De plus, la segmentation manuelle par un expert est une opération très coûteuse en temps. Dans les images 3D, il faut généralement classifier les voxels coupe par coupe puis, fusionner les résultats dans l'image 3D pour obtenir la partition souhaitée. Des méthodes de segmentation automatique ou semi-automatique ont été définies afin de permettre des segmentations précises et rapides des images.

Dans ce chapitre, nous abordons la notion de segmentation et présentons les éléments importants pour situer notre travail. Nous commençons par donner la définition d'une segmentation, puis nous introduisons quatre approches de segmentation qui représentent les travaux que nous souhaitons réaliser avec le modèle des cartes combinatoires. Nous présentons ensuite les différents types de critères utilisés en segmentation d'images. Enfin, nous présentons ce qu'est la topologie et quelques-unes de ses applications à la segmentation d'images.

## 3.1 Définition de la segmentation

Une segmentation est paramétrée par un prédicat d'homogénéité $P : R \rightarrow \{vrai, faux\}$ indiquant si une région est homogène d'après le critère testé. Afin que le prédicat soit consistant, il est nécessaire que chaque sous-ensemble d'une région homogène soit homogène. Un algorithme de segmentation

est donc paramétré par un prédicat $P$ et une image. Il retourne une partition de l'image en régions homogènes. La Définition 17 définit la notion de segmentation d'une image en régions d'après [HP76].

**Définition 17** (Segmentation en régions homogènes [HP76])**.** *Une segmentation en régions homogènes d'une image $I$ pour un prédicat $P$ est une partition $R$ de $I$ telle que :*

1. $I = \bigcup_{r \in R} r$,
2. $r_i \cap r_j = \emptyset$ *pour tout* $(r_i, r_j) \in R^2$, $i \neq j$,
3. $P(r) = vrai$ *pour tout* $r \in R$,
4. $P(r_i \cup r_j) = faux$ *pour tout* $(r_i, r_j) \in R^2$, $i \neq j$, $r_i$ *et* $r_j$ *adjacents.*

La première condition assure que tous les éléments d'une image appartiennent à une région et donc que l'union des régions représente l'image entière. La deuxième condition indique que les régions ne se recouvrent pas : il n'y a pas d'intersection entre les régions. La troisième condition indique que chaque région doit être homogène d'après le critère défini par l'oracle. Enfin la dernière condition précise que la fusion de deux régions adjacentes ne doit pas être homogène.

À partir de cette définition, de très nombreuses méthodes de segmentation existent permettant de résoudre autant de problèmes différents. D'après [Ser06], on compte plus d'un millier d'approches distinctes. Dans ce travail, nous distinguons deux composantes dans l'opération de segmentation : d'une part les critères de segmentation et d'autre part les algorithmes réalisant ces segmentations.

## 3.2 Méthodes de segmentation

Une bibliographie de référence dans le domaine de la vision par ordinateur [Pri09] propose une classification des méthodes de segmentation. Nous nous en inspirons pour présenter quatre grandes familles de méthodes de segmentation. Ces quatre approches sont représentatives de ce que nous souhaitons pouvoir réaliser en utilisant la carte topologique pour représenter une partition.

### 3.2.1 Seuillage

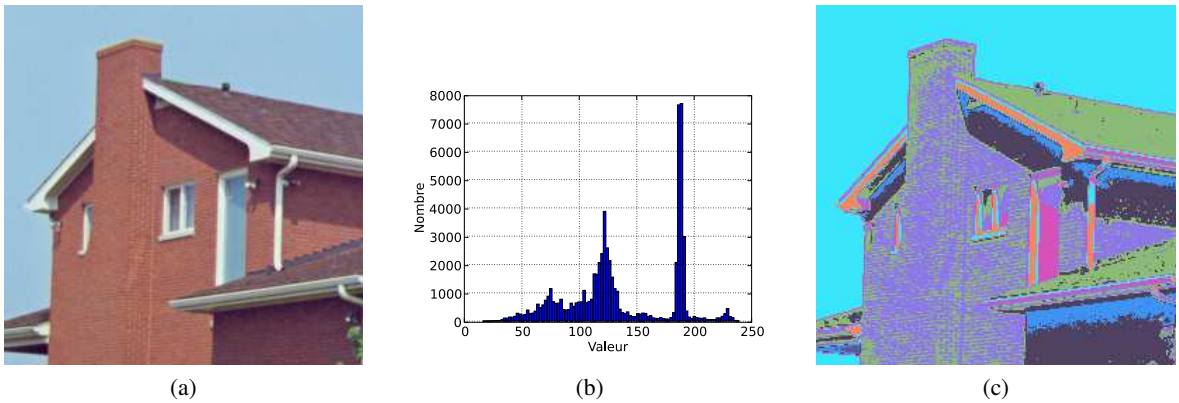
Historiquement, les méthodes par seuillage sont les premières approches utilisées en segmentation d'images. L'idée de ces méthodes est d'appliquer un filtre sur l'image et étant donné un seuil, de classifier l'ensemble des éléments de l'image en deux catégories : au-dessus du seuil et en-dessous du seuil [RK82]. Ces deux classes sont couramment appelées l'objet et le fond. La Figure 3.1 présente l'image originale (Figure 3.1a) et le résultat (Figure 3.1b) de la segmentation d'une image en niveau de gris par seuillage en utilisant la valeur médiane du niveau de gris comme valeur seuil. Nous définissons l'objet comme étant les pixels en-dessous du seuil, les pixels au-dessus du seuil sont les pixels de la classe du fond.

Les méthodes de segmentation par regroupement [RK82] (*clustering* en anglais) sont proches des méthodes de seuillage. Une approche classique utilise l'histogramme[1] de l'image afin de déterminer des classes de pixels. Chaque pixel est alors associé à une classe en fonction de sa valeur, puis les pixels sont regroupés en régions par leur position : des ensembles connexes de pixels partageant une même classe sont regroupés dans une même région. La Figure 3.2 montre un exemple de segmentation par regroupement de l'image Figure 3.2a. En utilisant l'histogramme de l'image, Figure 3.2b, l'utilisateur détermine qu'il y a 8 classes de pixels. Ensuite par regroupement de pixels connexes appartenant à la même classe, nous obtenons la partition présentée Figure 3.2c. Cette partition est présentée en fausses couleurs, c'est-à-dire que nous associons à chaque région une couleur différente afin de bien les distinguer.

[1] Un histogramme d'une image représente la distribution des valeurs de l'image sur le domaine des valeurs.

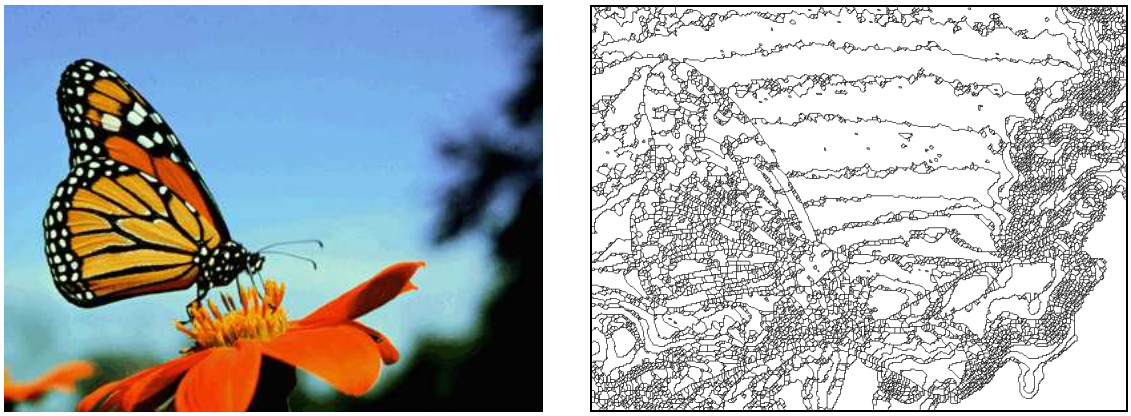
(a) (b)

FIG. 3.1: Exemple de seuillage des valeurs des pixels d'une image 2D : (a) image originale ; (b) image binaire correspondant au seuillage de l'image originale selon une intensité seuil de 150.




(a) (b) (c)

FIG. 3.2: Exemple de regroupement des valeurs des pixels d'une image 2D : (a) image originale en 256 niveaux de gris ; (b) histogramme de l'image ; (c) image étiquetée correspondant au regroupement des pixels d'après l'histogramme en 8 classes.



(a) (b)

FIG. 3.3: Exemple de segmentation par ligne de partage des eaux d'une image 2D : (a) image originale ; (b) image correspondant à la segmentation par ligne de partage des eaux de l'image originale.

Ces deux méthodes de segmentation ont un caractère global. Nous trouvons dans la littérature de nombreux autres algorithmes pouvant se classer dans cette catégorie, certains avec des critères et principes plus complexes qui permettent de résoudre des problèmes plus compliqués.

### 3.2.2 Croissance de régions

Les algorithmes par croissance de régions travaillent à partir de graines (*seeds* en anglais) et essaient d'agglomérer les éléments voisins de l'image de manière à respecter un critère : la région finale représentant un objet ou une partie d'un objet. Nous trouvons dans ce type d'approche de nombreux algorithmes de croissance de régions.

La segmentation par ligne de partage des eaux [VS91] (*watershed* en anglais) est une technique de segmentation par croissance de régions inspirée des bassins versants pour les rivières. L'algorithme travaille sur l'image de gradient représentant l'image. L'image de gradient associe à chaque élément de l'image une valeur numérique exprimant le gradient entre les valeurs des pixels de l'image. Les graines sont sélectionnées de manière automatique comme étant des minimums locaux de l'image de gradient (c'est-à-dire que tous les voisins d'une graine ont une valeur de gradient supérieure ou égale). Chaque graine détermine un nouveau bassin. Les éléments de l'image sont ensuite attribués à la graine de manière à remplir les bassins. Les frontières entre deux bassins sont appelées crêtes. Ce type de segmentation produit généralement une sursegmentation de l'image, c'est-à-dire qu'il y a beaucoup de régions et qu'il faut en fusionner afin d'obtenir des objets ayant un sens. La Figure 3.3 présente un exemple de segmentation par ligne de partage des eaux tiré de [Pet06].

Une généralisation des lignes de partages des eaux est proposée par la Transformée en Forêt des Images [FB04, FSdAL04] (*Image Foresting Transform ou IFT* en anglais). Cette méthode de segmentation est également une méthode par croissance de régions à partir de graines : [CBCN08] a étudié les relations entre l'opération de fusion de région et les méthodes de segmentation par ligne de partage des eaux. Dans cette approche, les graines sont associées à une étiquette, ce qui permet d'avoir plusieurs graines définissant une même région. L'algorithme se sert d'une représentation de l'image par un graphe (présenté dans la Section 1.1.5). Les nœuds du graphe représentent les éléments de l'image et les arêtes reliant les éléments adjacents sont valuées par une fonction de coût. Dans cette représentation, l'IFT cherche à associer chaque nœud à la graine de manière à ce que le chemin reliant le nœud à la graine soit le moins coûteux possible. En choisissant spécifiquement les nœuds et la fonction de coût, il est possible de mettre en œuvre une segmentation par ligne de partage des eaux à partir de l'IFT. L'intérêt

de cette approche est que l'algorithme permet de définir de nombreuses méthodes différentes. Il permet également de rajouter des graines rendant alors la segmentation semi-automatique et permettant de proposer rapidement de meilleures segmentations. [CBNC09] montre que les bassins versants d'une segmentation par ligne de partage des eaux ont une définition équivalente en terme d'optimum global (c'est-à-dire une forêt de poids maximum), si l'on se situe dans le cadre des arêtes d'un graphe.

Nous pouvons également noter que les algorithmes par croissance de régions sont couramment mis en œuvre par un algorithme consistant à fusionner des régions adjacentes en partant de région-graines.

### 3.2.3 Division-fusion

Une autre grande famille de méthodes est définie par une approche mixte mêlant un processus de fusion de régions avec un processus de division de régions.

Les approches par fusion de régions décrites comme de bas en haut (*bottom-up* en anglais) permettent d'obtenir une partition en fusionnant successivement des régions adjacentes répondant à un même critère. Le point de départ d'un algorithme de segmentation par fusion de régions est une image sursegmentée, c'est-à-dire qui comporte beaucoup de régions. Généralement la partition utilisée à cette fin comporte une région par élément de l'image. L'algorithme procède ensuite par fusions successives de régions adjacentes. Si deux régions respectent le critère de fusion alors elles sont effectivement fusionnées. La partition finale est atteinte lorsqu'il n'est plus possible de fusionner des régions, c'est-à-dire qu'il n'existe pas deux régions dont la fusion respecte le critère.

À l'opposé les approches par division de régions, de haut en bas (*top-down* en anglais) permettent d'obtenir la partition résultante en divisant les régions en régions de plus en plus petites jusqu'à ce que des critères particuliers soient satisfaits. Le point de départ de l'algorithme est une sous-partition de l'image. En général, la segmentation initiale est composée d'une seule région contenant l'image entière. Chaque région ne respectant pas un critère d'homogénéité est ensuite divisée de manière arbitraire en plus petites régions. Les premières approches de division consistent à partitionner une région en régions de taille identique. Ce mécanisme de division ressemble à un processus de construction d'un quadtree pour une image 2D. Lorsque toutes les régions respectent le critère de division, nous obtenons la partition finale.

L'approche mixte, proposée dans [HP76], revient à faire suivre une opération de division par une opération de fusion. Le gain montré par les auteurs tient au temps de calcul plus court pour une consommation mémoire semblable. Des variantes envisagent d'alterner les phases de division et les phases de fusion en modifiant dynamiquement les critères afin de produire une segmentation la plus précise possible tout en conservant un principe simple.

### 3.2.4 Modèle déformable

Les modèles déformables [TF88] forment une grande famille de méthodes de segmentation. En pratique, ils représentent la géométrie d'un objet. Cette géométrie est évolutive : des règles inspirées de la physique guident l'évolution du modèle. Ainsi un modèle déformable évolue de manière à mieux épouser les formes présentes dans les données.

Les contours actifs [KWT88] (*snakes* en anglais) forment une classe de modèles déformables très populaire. Ils représentent un contour déformable planaire. Ils sont utilisés afin d'approcher la position et la forme des bords des objets dans une image 2D sous l'hypothèse que les contours sont continus et lisses. Les contours actifs cherchent généralement à minimiser une énergie globale sur l'image. Cette énergie est alors le critère de segmentation.

Les isocontours [OS88] (*level sets* en anglais) forment une deuxième catégorie de modèles déformables. Dans le traitement d'images, un isocontour est un contour pour lequel une fonction d'énergie

(le critère) possède une valeur constante tout le long du contour. Les lignes de niveau sur les cartes topographiques en sont un exemple. En 3D les isocontours sont des isosurfaces, elles servent par exemple en imagerie médicale à représenter une surface entourant une région ayant une densité particulière dans les images de scanner et ainsi à visualiser les organes internes, les os ou d'autres structures.

## 3.3 Critères de segmentation

Les critères de segmentation forment la partie active d'un processus de segmentation. Ce sont eux qui définissent la partition à obtenir et sont donc liés à l'application. À l'instar des méthodes de segmentation, les critères sont nombreux et variés. Nous faisons ici une présentation des critères en quatre grandes classes déterminées par les informations utilisées par le critère pour guider la segmentation. En pratique, les critères sont souvent combinés afin de produire des résultats aussi pertinents que possible. Les trois premières classes sont développées dans la suite de cette section et la dernière classe, qui nous intéresse plus particulièrement, est décrite dans la Section 3.4.

### 3.3.1 Critères scalaires

Les critères basés sur les valeurs de l'image sont tous les critères qui utilisent directement l'information de valeur des éléments de l'image afin de déterminer l'homogénéité d'une région.

Parmi ces différents critères nous trouvons les critères basés sur l'intensité (pour les images en niveau de gris) ou la couleur. Un exemple de ces critères est un critère basé sur l'erreur quadratique des valeurs des pixels de la région que nous définissons comme homogène si l'erreur est inférieure à un seuil.

### 3.3.2 Critères fréquentiels

Les critères fréquentiels forment une deuxième classe de critères très utilisée. Dans ce type de critère, l'image est alors considérée comme un signal multidimensionnel et les opérations de traitement du signal sont utilisées pour définir des partitions.

Le traitement le plus connu pour l'utilisation de critères fréquentiels est la transformée de Fourier [GW90] qui permet de passer d'une représentation spatiale de l'image à une représentation fréquentielle. Une autre transformée proche est la transformée en cosinus discrète (*Discrete Cosinus Transform* en anglais) [RY90]. Les différentes transformées en ondelettes [RBB99] rentrent également dans cette catégorie.

### 3.3.3 Critères géométriques

Les critères géométriques sont très intéressants du point de vue de la segmentation d'images. En effet, la forme des objets est un paramètre généralement très important des objets que nous souhaitons retrouver. Voici une liste de paramètres qui peuvent entrer dans la définition d'un critère géométrique :

- l'aire ;
- le volume ;
- les dimensions (longueur, largeur, etc.) ;
- l'élongation (rapport longueur/largeur) ;
- la forme (courbure des bords, etc.).

Tous ces critères sont des valeurs mesurables qui dépendent de l'objet mais qui dépendent également du point de vue de l'image. Ainsi un même objet n'a pas les mêmes dimensions en fonction de son orientation et de son éloignement au point de vue du système d'acquisition. Les paramètres géométriques sont donc très dépendants des conditions d'acquisition.

## 3.4 Topologie dans la segmentation d'images

Les propriétés topologiques sont moins dépendantes de l'image puisqu'elles caractérisent des propriétés générales. À titre d'exemple, nous trouvons des informations comme le nombre de cavités ou le nombre d'anses qu'une région possède. D'autres informations comme l'adjacence ou bien l'imbrication entre régions sont des informations utiles qui se classent dans l'information topologique.

Le traitement d'images et la segmentation en particulier cherchent à capturer la forme des objets dans une image. Pour ce faire, nous cherchons à retrouver des propriétés intrinsèques des objets. Nous pouvons envisager d'effectuer une rotation ou un zoom sur l'image et observer les propriétés qui ne changent pas. Ces propriétés sont appelées les invariants de l'objet. La géométrie permet d'identifier certaines propriétés, par exemple les dimensions d'un objet sont des caractéristiques typiquement géométriques. La topologie s'intéresse à la définition d'invariants lorsque les déformations sont continues et continûment inversibles. Par exemple, un cube plein de pâte à modeler peut être étiré ou compacté sans changer ses caractéristiques topologiques. Par contre transformer le cube plein en *donut*[2] n'est pas une transformation qui préserve la topologie.

La topologie est une branche des mathématiques qui étudie la manière dont les espaces sont connectés. Des propriétés topologiques sont par exemple d'avoir deux tunnels, une cavité, d'être nouées ou d'avoir des composantes liées et ne pouvant être séparées. Dans le cadre de l'analyse d'images nous nous intéressons aux méthodes permettant d'une part de détecter ces propriétés dans les partitions et d'autre part de les utiliser dans des critères de segmentation.

### 3.4.1 Adjacence, incidence et imbrication

Les relations d'adjacence et d'incidence entre les cellules sont des informations topologiques importantes. En effet, par l'incidence entre les cellules nous sommes capables de déterminer des relations entre les régions et notamment les relations d'adjacence et d'imbrication entre les régions.

### 3.4.2 Caractéristiques topologiques

Dans cette partie, nous définissons quelques caractéristiques topologiques intéressantes dans le cadre du traitement d'images. Pour cela nous commençons par un rappel de topologie algébrique qui reprend quelques points détaillés dans [Mun84]. Nous pouvons trouver une étude de la topologie et les applications de celle-ci notamment dans le cadre du traitement d'images dans [Zom01].

La topologie algébrique applique les outils de l'algèbre à l'étude des espaces topologiques. Elle associe aux structures topologiques des objets algébriques (nombre, groupe, espace vectoriel), de sorte qu'à deux espaces homéomorphes[3] sont associés deux structures isomorphes[4]. De tels objets sont appelés des invariants algébriques.

Il existe de nombreux invariants mais parmi les plus connus, nous retrouvons :

- le groupe fondamental d'un espace topologique X en un point x : l'ensemble des classes d'homotopie des lacets de X de base x, la loi de composition interne étant la concaténation des lacets ;
- les groupes d'homotopie supérieure d'un espace topologique X en un point x ;
- les groupes d'homologie ou de cohomologie d'un espace topologique X.

Ces invariants décrivent les caractéristiques des espaces qu'ils représentent. Différents travaux cherchent à calculer des invariants topologiques pour des objets en utilisant différentes représentations.

[2] Beignet sucré en forme de tore.

[3] Un homéomorphisme est une application bijective continue entre deux espaces topologiques dont la réciproque est continue. Les deux espaces sont dits homéomorphes.

[4] Un isomorphisme est un morphisme admettant un inverse qui est lui-même un morphisme. Deux objets liés par un isomorphisme sont dits isomorphes.

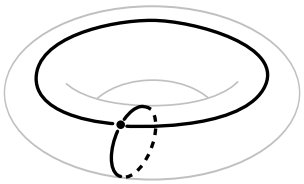


FIG. 3.4: Calcul de la caractéristique d'Euler pour un tore : $v = 1$, $f = 1$, $a = 2$ et donc $\chi = 1-2+1 = 0$.

Parmi ces travaux, [Pel06] calcule les groupes d'homologie sur des structures simpliciales, simploïdales et cellulaires. [DPF08] cherche à calculer les générateurs des groupes d'homologie sur des structures de cartes combinatoires généralisées. Cependant ces calculs sont complexes et même si l'information que ces invariants topologiques représentent est intéressante d'un point de vue théorique, nous cherchons à identifier des outils plus appliqués qui permettent d'obtenir des invariants topologiques plus simples pour les régions dans une image. Nous utilisons pour cela le fait que nous travaillons avec des images 2D ou 3D et que nous cherchons à identifier des objets orientables : il s'agit d'une configuration favorable pour le calcul des invariants topologiques [Mun84]. Nous étudions pour cela la caractéristique d'Euler (liée au genre), un invariant numérique très connu. Nous présentons ensuite les nombres de Betti qui forment d'autres invariants topologiques intéressants à calculer et utilisables pour caractériser des régions dans une image 3D.

### Caractéristique d'Euler

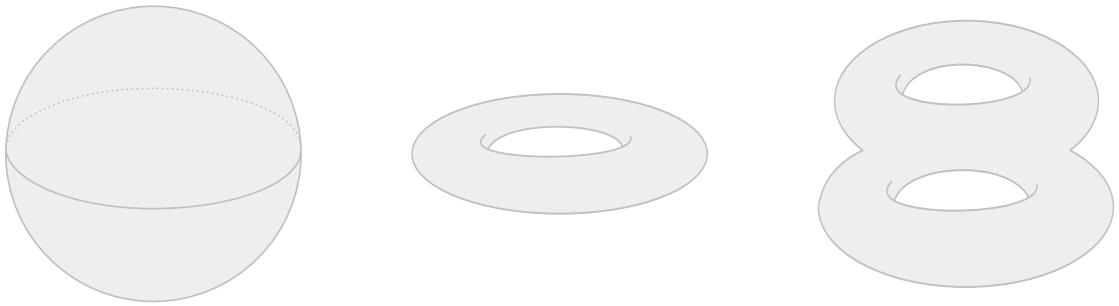
La caractéristique d'Euler, habituellement notée $\chi$, est un invariant numérique : un nombre qui décrit un aspect d'une forme de l'espace topologique. Pour les polyèdres de dimension 2, la caractéristique d'Euler se calcule en utilisant la formule donnée par la Définition 18 (voir exemple Figure 3.4).

**Définition 18** (Caractéristique d'Euler surfacique)**.** *La caractéristique d'Euler d'un polyèdre de dimension 2 est égale à $\chi = s - a + f$ où $s$ représente le nombre de sommets, $a$ représente le nombre d'arêtes et $f$ représente le nombre de faces du polyèdre.*

Pour les surfaces orientables sans bord, le genre est le nombre maximum de courbes fermées sans points communs pouvant être tracées à l'intérieur de la surface sans la déconnecter. Le genre est un invariant topologique pour les surfaces orientables sans bord : deux surfaces n'ayant pas le même genre ne sont pas homéomorphes. Le genre $g$ d'une surface est lié à la caractéristique d'Euler $\chi$ par la relation $\chi = 2 - 2g$ dans le cas des surfaces orientables fermées. Le genre représente le nombre d'anses de la surface (voir exemple Figure 3.5).

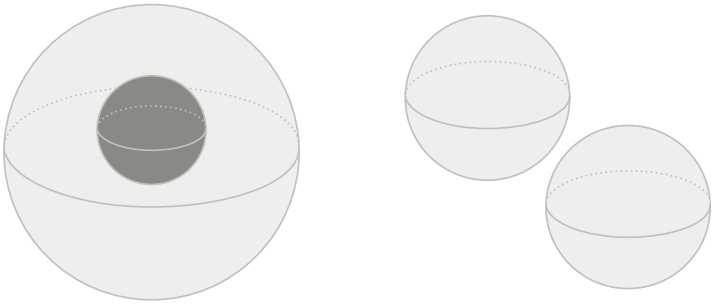
Cependant, lorsqu'un objet est composé de plusieurs composantes connexes ou lorsque l'objet est de dimension supérieure, la caractéristique d'Euler ne permet plus d'identifier les objets. Par exemple en utilisant la caractéristique d'Euler surfacique, un objet composé de deux surfaces sphériques imbriquées (Figure 3.6a) possède la même caractéristique d'Euler qu'un objet composé de deux sphères séparées (Figure 3.6b).

La Définition 19 donne la formule de calcul de la caractéristique d'Euler pour les complexes cellulaires en toutes dimensions à partir de la somme alternée du nombre de cellules de chaque dimension.



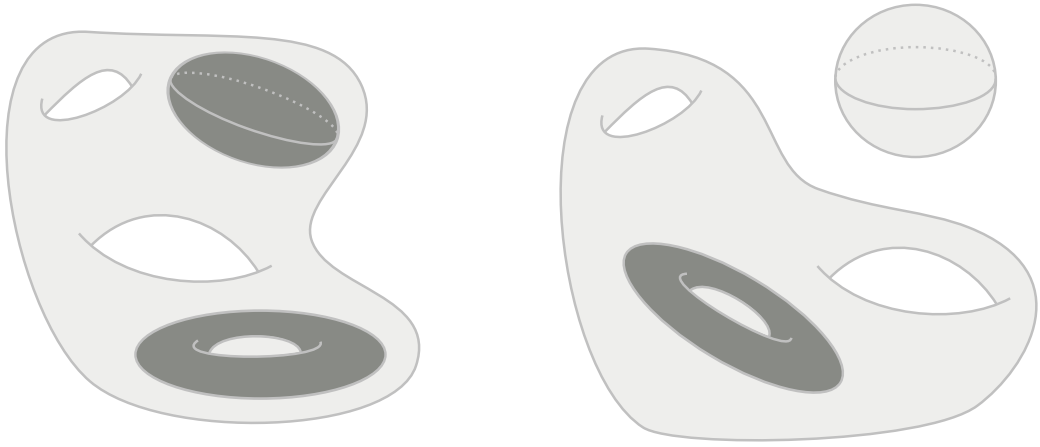
(a) (b) (c)

FIG. 3.5: Exemple de surfaces avec leur genre : (a) une sphère, $g = 0$ ; (b) un tore, $g = 1$ ; (c) un 2-tore, $g = 2$.



(a) (b)

FIG. 3.6: Exemple de deux objets ayant la même caractéristique d'Euler 2D (nous représentons les cavités en gris foncé) : (a) deux sphères (avec relation d'imbrication) $\chi = 4$ ; (b) deux sphères (sans relation d'imbrication) $\chi = 4$, la caractéristique d'Euler ne permet pas de différencier ces deux configurations.



(a) (b)

FIG. 3.7: Exemple d'objets avec leur caractéristique d'Euler 3D et leurs nombres de Betti associés (les cavités sont représentées en gris foncé). (a) Un objet avec une composante connexe, trois tunnels et deux cavités ($\mathfrak{b}_0 = 1$, $\mathfrak{b}_1 = 3$, $\mathfrak{b}_2 = 2$), $\chi = 0$. (b) Un objet avec deux composantes connexes, trois tunnels et une cavité ($\mathfrak{b}_0 = 2$, $\mathfrak{b}_1 = 3$, $\mathfrak{b}_2 = 1$), $\chi = 0$.

**Définition 19** (Caractéristique d'Euler)**.** *La caractéristique d'Euler d'un complexe cellulaire est définie par :*

$$\chi = \sum_{i=0}^{d} (-1)^i k_i$$

*où $d$ est la dimension de l'espace et $k_i$ est le nombre de cellules de dimension $i$.*

Pour identifier des objets en 3D, la caractéristique d'Euler n'est pas un outil suffisant. La Figure 3.7 présente un exemple avec deux objets topologiquement différents ayant la même caractéristique d'Euler. Nous étudions donc les nombres de Betti qui répondent à cette problématique.

### Nombres de Betti

D'un point de vue pratique pour les objets 3D, les nombres de Betti représentent pour chaque dimension le nombre de trous. Ainsi le premier nombre de Betti, noté $\mathfrak{b}_0$, compte le nombre de composantes connexes de l'objet. Le second nombre de Betti, noté $\mathfrak{b}_1$, représente le nombre de tunnels parfois appelés anses. Enfin le troisième nombre de Betti, noté $\mathfrak{b}_2$, compte le nombre de cavités de l'objet parfois appelées vides. Pour les objets 3D fermés et orientés les nombres de Betti $\mathfrak{b}_k$ avec $k > 2$ sont égaux à zéro.

Par exemple, l'objet de la Figure 3.7a est un objet 3D qui contient deux cavités, trois tunnels et qui est composé d'une composante connexe. Les trois premiers nombres de Betti correspondants sont donc $\mathfrak{b}_0 = 1$, $\mathfrak{b}_1 = 3$ et $\mathfrak{b}_2 = 2$. L'objet de la Figure 3.7b contient deux composantes connexes. Les trois premiers nombres de Betti valent donc $\mathfrak{b}_0 = 2$, $\mathfrak{b}_1 = 3$ et $\mathfrak{b}_2 = 1$.

La Définition 20 donne la définition mathématique des nombres de Betti. Ils représentent le rang des groupes d'homologie de l'objet sans torsion. Ainsi le $i^{\text{e}}$ nombre de Betti compte le nombre de générateurs du $i^{\text{e}}$ groupe d'homologie. Cela illustre la relation entre cet invariant numérique et les invariants algébriques étudiés en topologie.

**Définition 20** (Nombre de Betti [Mun84])**.** *Les nombres de Betti $\mathfrak{b}_k$ d'un espace $X$ sont définis pour tout entier naturel $k$ comme le rang du $k^e$ groupe d'homologie de $X$ pour les objets sans torsion.*

Les nombres de Betti sont liés à la caractéristique d'Euler par la Proposition 2.

**Proposition 2** (Caractéristique d'Euler et nombres de Betti (relation d'Euler-Poincaré) [Mun84])**.** *La caractéristique d'Euler $\chi$ d'un complexe cellulaire est égale à la somme alternée de ses nombres de Betti :*

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

*où $d$ est la dimension de l'espace et $\mathfrak{b}_i$ représente le $i^e$ nombre de Betti.*

Grâce à cette propriété nous obtenons un moyen alternatif pour calculer la caractéristique d'Euler d'un objet en utilisant les nombres de Betti de l'objet.

### 3.4.3 Topologie dans le contexte de l'image

Nous nous intéressons dans cette section à l'usage de la topologie dans le contexte du traitement d'images. Dans de nombreuses applications, les propriétés topologiques sont valorisées. Elles permettent de caractériser des propriétés intrinsèques des objets que nous cherchons à retrouver dans une image. Ainsi de nombreux travaux, notamment dans le milieu médical, portent sur la modification de partition en prenant en compte des informations topologiques.

Dans le contexte de la segmentation d'images, nous travaillons avec des images numériques. Or, l'acquisition et la discrétisation des objets entraînent du bruit et des artefacts produisant des incohérences topologiques. Ainsi la topologie d'un objet peut changer en fonction de la résolution d'une image.

Deux grandes catégories de procédés utilisent la topologie dans le cadre du traitement d'images. Il y a d'une part le contrôle de la topologie durant une segmentation et d'autre part la correction de la topologie d'une partition après segmentation. Nous présentons ici quelques travaux sur les méthodes de segmentation prenant en compte la topologie dans le contexte du traitement d'images. Le contrôle de la topologie durant la segmentation d'une image est un procédé très important pour garantir certaines propriétés au résultat final. En pratique, le contrôle de la topologie est souvent réalisé dans le cadre de segmentations par modèles déformables. En effet, il existe une caractérisation locale sur les éléments d'un objet permettant d'anticiper les changements topologiques : il s'agit des points simples [Ros70, Ber94, CB09].

Nous étudions dans un premier temps des opérations qui existent dans le cadre des modèles déformables avec des partitions binaires de l'espace. Nous décrivons ensuite une méthode utilisant un modèle déformable mais travaillant sur une partition multirégions. Enfin nous présentons une méthode qui utilise une partition multirégions de l'espace dans le cadre de la segmentation d'images par atlas.

#### Utilisation des nombres de Betti dans la segmentation d'images

Nous nous sommes intéressés à des travaux utilisant des contraintes topologiques basées sur les nombres de Betti [HYW08]. Ils proposent une application utilisant les contours actifs dans une image 2D où l'utilisateur définit des contraintes sur le nombre de composantes connexes (premier nombre de Betti) et le nombre de bords intérieurs (deuxième nombre de Betti) de l'objet à obtenir. L'évolution du contour actif est ensuite guidée par des critères basés sur les valeurs de l'image tout en étant contrainte par les paramètres de l'utilisateur.

La Figure 3.8 présente l'un des résultats publiés par les auteurs dans [HYW08]. Dans cette application, la segmentation d'une coupe d'une image de scanner est utilisée. La partition initiale proposée est

représentée par le cercle blanc dans l'image de gauche. Le résultat de l'algorithme présenté par les figures de droite varie en fonction des critères donnés par l'utilisateur. Dans le cas de la figure supérieure, le nombre de composantes connexes à obtenir est fixé à 1. Dans la figure inférieure, le nombre de composantes connexes est fixé à 2. Il n'y a pas de trou autorisé. L'algorithme déforme bien la partition afin de s'adapter à l'image tout en respectant la contrainte topologique.

Cette approche fonctionne sur des partitions binaires avec le fond et l'objet, qui peut lui-même avoir plusieurs composantes connexes avec des trous. Du point de vue de la topologie, l'utilisation des nombres de Betti (même si elle n'est pas illustrée comme cela dans la publication originale) est un principe intéressant. Il est également intéressant de se pencher sur l'application de cette approche au cas 3D.

#### Contrôle des tunnels durant la segmentation d'images

Dans [Ség08] l'auteur propose un nouveau type de contours actifs 3D permettant de contrôler la topologie durant la phase d'évolution de la partition dans une partition binaire de l'espace (fond/objet). Contrairement aux approches classiques qui, soit ne contrôlent pas la topologie, soit fixent une topologie donnée pour la partition, l'outil permet de contrôler précisément la topologie de chaque composante pour prévenir la formation de défauts topologiques. Ainsi les composantes peuvent fusionner, se diviser ou même disparaître mais ne peuvent jamais produire de tunnels ni fermer des tunnels existants. Cette approche conserve le genre des surfaces de chaque composante lors de l'évolution de la partition.

Pour cela, l'auteur utilise la notion de points multisimples qui dérive de la notion de points simples afin de permettre aux différentes composantes de son modèle de fusionner, de se diviser, d'apparaître ou de disparaître. Ainsi un point est multisimple si il peut être ajouté ou retiré à l'objet sans changer le nombre de tunnels dans l'objet ou le fond.

La Figure 3.9 présente un résultat extrait de [Ség08] dans lequel il applique son processus de déformation à une segmentation d'un cortex cérébral depuis une image anatomique obtenue par un IRM. La partition initiale est composée de 55 composantes connexes sans tunnel. La surface finale possède la même topologie qu'une sphère, son genre est zéro. L'auteur précise que la même évolution sans contrôle de la topologie par les points multisimples conduit à l'apparition de 18 tunnels.

La Figure 3.10 est un autre résultat extrait de [Ség08] où l'on observe le processus inverse : la segmentation de vaisseaux sanguins en faisant évoluer la boîte englobante qui, par division, donne la partition résultat.

Ces approches montrent des applications au contrôle de la topologie dans le cadre de l'imagerie médicale. Les contraintes topologiques sont des *a priori* donnés par les experts et qu'il est important de considérer dans le cadre d'une segmentation. Cependant, les applications considérées n'utilisent que des partitions binaires, or de manière générale la réalité des images et des objets que l'on souhaite repérer ne permet pas ce type de simplification. Nous souhaitons en effet contrôler la topologie d'une partition comportant de nombreuses régions.

#### Déformation à topologie constante multirégions

Une approche utilisant une extension des points simples à des partitions comportant plusieurs régions est proposée dans [CBG99, Coi99]. L'auteur définit la notion de déformation homotopique élémentaire. Il s'agit d'une déformation qui consiste à basculer un élément de l'image (en l'occurrence un voxel dans une image 3D) d'une région à une autre en garantissant la préservation de la topologie de chaque région de l'image.

L'idée de l'approche consiste à tester chaque élément avec l'ensemble des régions : si le point est simple pour une région $r_1$ en considérant que tout le reste fait partie du fond alors le point est $r_1$-simple. Au final, un élément est donc simple pour plusieurs régions, par exemple il peut être $(r_1, r_2)$-simple.

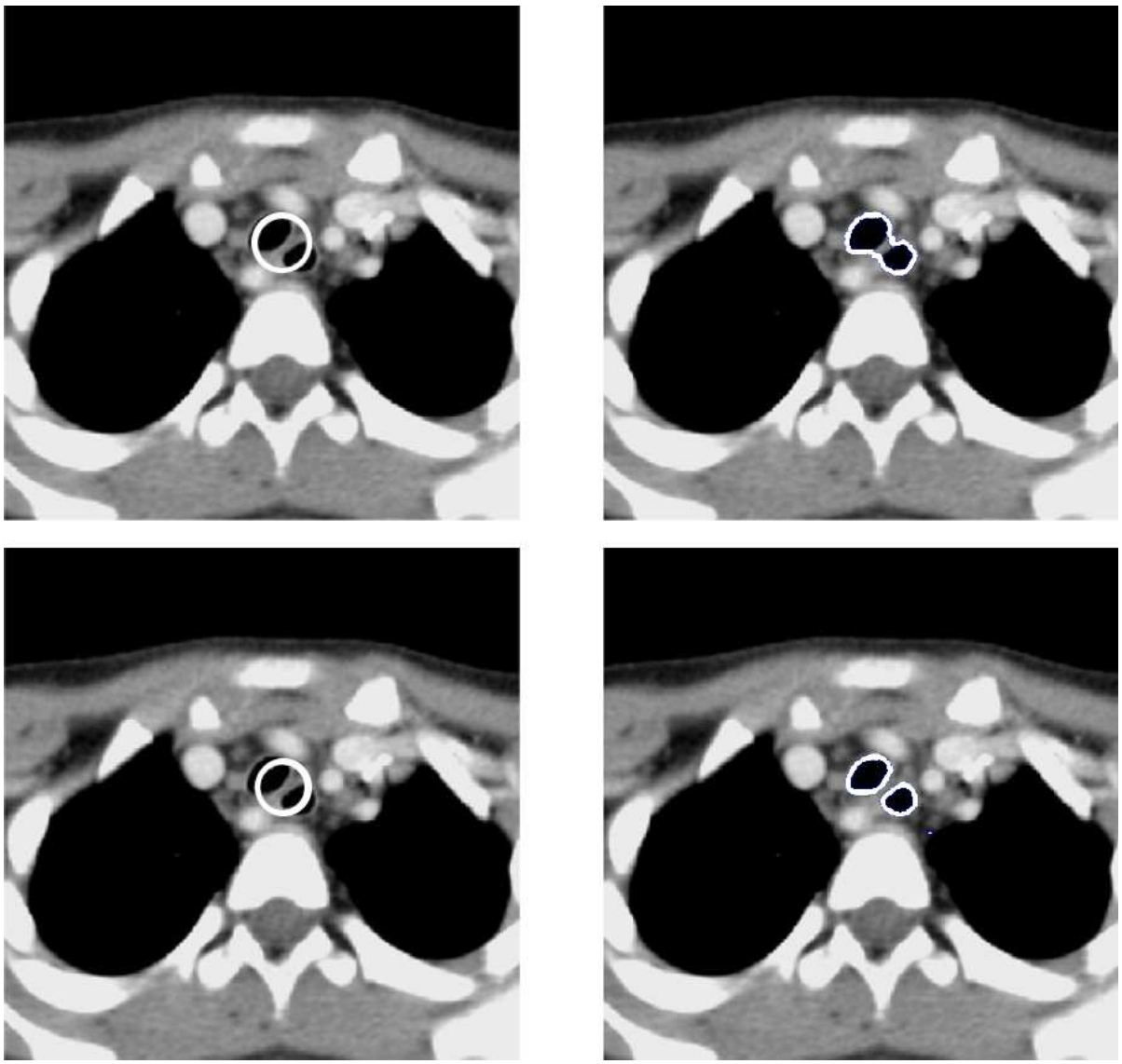


FIG. 3.8: Segmentation d'une image 2D. La partition initiale est représentée par les images de gauche et les partitions résultats par les images de droite. Les contraintes dans les images du haut sont d'avoir une composante connexe sans trou. Dans l'image du bas, les contraintes sont d'avoir deux composantes connexes sans trou. Les résultats montrent que les contraintes sont respectées.

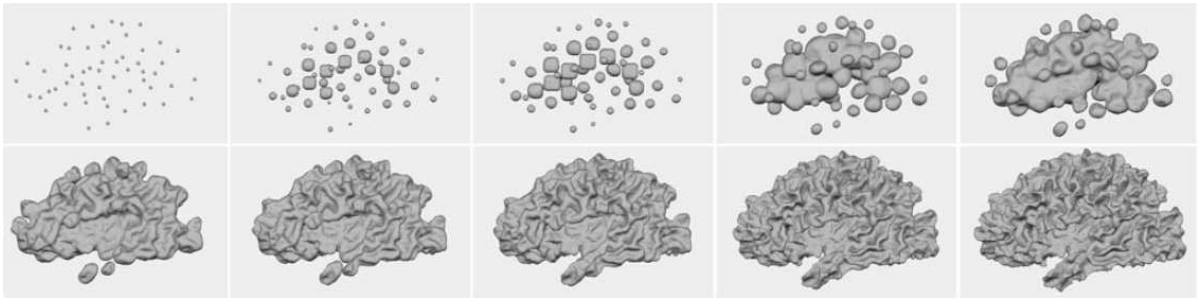


FIG. 3.9: Résultat de la segmentation (3D) d'un cortex cérébral dans une image IRM. La partition initiale est composée de 55 composantes connexes. La surface finale est de genre 0 ($\chi = 2$), elle ne contient pas d'anse.

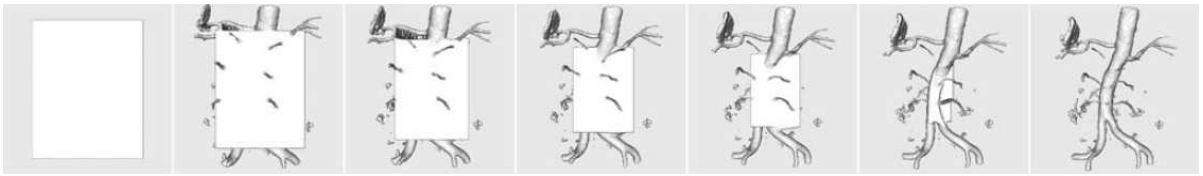


FIG. 3.10: Résultat de la segmentation (3D) de vaisseaux sanguins en faisant évoluer la boîte englobante par divisions successives. La partition résultat ne contient pas d'anses.

Cela signifie que le point peut être basculé de la région $r_1$ dans la région $r_2$ ou inversement sans changer la topologie de ces deux régions. La Figure 3.11 extraite de [Coi99] donne un exemple de classification des points dans une image 2D composée de trois régions.

Avec cette notion de points simples, l'auteur définit un modèle déformable permettant de réaliser la segmentation du cortex cérébral à partir d'IRM. Ce modèle répond donc au problème soulevé mais ne permet pas le contrôle complet de la topologie de la partition. En effet, cette approche de la déformation préserve la topologie de chaque région prise indépendamment. Cependant le processus de déformation ne préserve pas la topologie de la partition complète. Par exemple la déformation de la Figure 3.12 conserve la topologie de chaque région mais la topologie de la partition change : une adjacence entre deux régions disparaît.

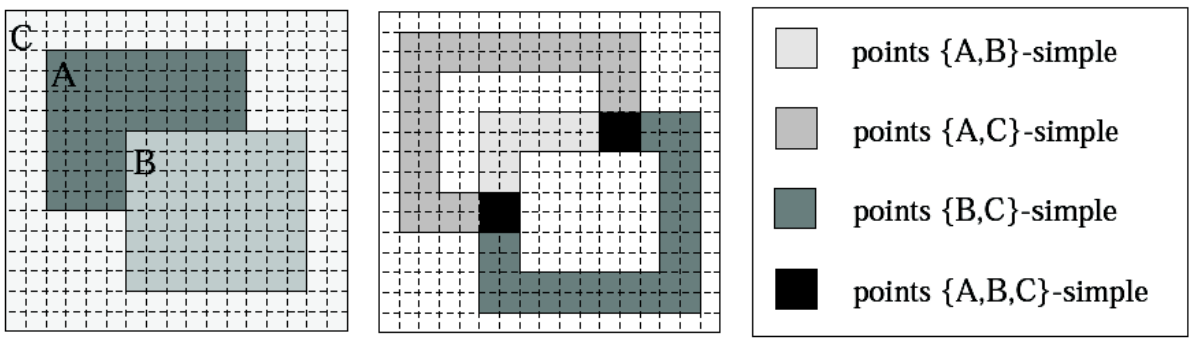
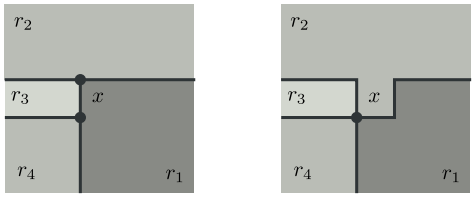



FIG. 3.11: Classification des points selon le fait qu'ils soient simples pour différentes régions dans une image 2D comportant trois régions. Les régions $A$ et $B$ sont considérées en 4-connexité et la région $C$ est considérée en 8-connexité.




(a) (b)

FIG. 3.12: Changement de la topologie de la partition sans changement de la topologie de chaque région. (a) Le basculement du pixel $x$ de la région $r_1$ dans la région $r_2$ ne change pas la topologie d'après la définition des points simples de [Coi99] ; (b) Un sommet et une arête disparaissent, faisant disparaître une relation d'adjacence entre la région $r_1$ et la région $r_3$ : il y a changement dans la topologie de la partition.

#### Segmentation par atlas

La segmentation par atlas [KSI$^+$96] consiste dans un premier temps à calculer une transformation qui effectue la correspondance entre une image 3D à traiter et l'atlas : une image de référence. Dans un second temps, le volume à traiter est projeté sur l'atlas et généralement une étape d'optimisation est réalisée afin de bien faire correspondre les données de l'image avec les régions identifiées dans l'atlas.

Il s'agit d'une méthode très utilisée dans la segmentation d'images médicales, notamment pour les images cérébrales. Le cerveau est décomposé en différentes structures qui jouent un rôle particulier au niveau physiologique. Ces structures sont identifiées par des experts dans une image de référence afin de définir l'atlas. La correspondance d'une nouvelle image avec les images de référence permet ainsi de définir une partition multirégions de l'image. Cependant si ces travaux donnent de bons résultats dans le cas de cerveaux sains, il n'est pas évident que ce type de segmentation fonctionne correctement dans le cas de pathologies particulières comme des tumeurs cérébrales.

Ce type de segmentation utilise une information topologique : l'adjacence et l'imbrication entre les régions font partie des informations contenues dans l'atlas.

## 3.5 Conclusion

Dans ce chapitre, nous avons rappelé la définition de la segmentation d'une image. Nous avons présenté rapidement quelques grandes familles de méthodes pour réaliser la segmentation. Ensuite, nous nous sommes plus particulièrement intéressé au segmentation utilisant des contraintes topologiques. Nous avons enfin étudié différents travaux existants qui utilise la topologie pour segmenter des images. Dans la partie suivante, nous commençons à présenter notre travail en introduisant les nouvelles opérations que nous avons définies pour permettre la réalisation d'opération de segmentation. Nous commençons au Chapitre 4 par définir deux approches de l'opération de fusion de régions, l'opération permettant de regrouper plusieurs régions initiales dans une seule région résultante.

## Deuxième partie

# Opérations de Modification

CHAPITRE 4

# FUSION DE RÉGIONS

La fusion de régions réduit une partition en agglomérant des régions connexes. Il s'agit d'une opération importante pour la classe de segmentation d'images par approche *bottom-up*. Suite à l'extraction, la carte topologique représentant une image est composée de nombreuses régions. Nous souhaitons fusionner des ensembles de régions afin de diminuer la complexité de la représentation et faire apparaître de grandes régions homogènes par rapport à un critère donné. Dans notre travail, nous avons développé deux approches qui résolvent ce problème. Dans un premier temps, nous présentons une approche dite *locale* qui permet la fusion d'un ensemble connexe de régions en minimisant les modifications sur la carte topologique. Cette approche s'applique lors de la fusion d'un petit nombre de régions ou lors de la fusion interactive de régions, les régions à fusionner étant désignées par un utilisateur. Dans un deuxième temps, nous étudions une approche dite *globale* qui modifie de manière plus importante la carte topologique pour permettre la fusion simultanée de plusieurs ensembles de régions connexes. Les applications de cette méthode se trouvent dans les traitements automatiques de l'image demandant beaucoup de modifications comme par exemple une opération de segmentation.

Afin de définir ces opérations, nous avons besoin des algorithmes de suppression de cellules d'une carte topologique. Dans la suite, nous commençons par introduire la suppression d'un sommet, d'une arête et d'une face, puis nous détaillons les deux approches de la fusion de régions. Nous définissons également une opération plus simple permettant dans certaines configurations de fusionner une région isolée avec sa région englobante. Enfin, nous comparons le temps d'exécution des opérations de fusion en fonction des configurations d'utilisation.

## 4.1 Suppression de cellules dans le modèle des cartes topologiques

Dans la Section 1.2.4, nous avons introduit les opérations de suppression de cellules dans une carte combinatoire. À l'aide de ces opérations, nous définissons les opérations de suppression de cellules dans la carte topologique. La nouveauté de ces opérations provient de la mise à jour du plongement des cellules supprimées dans la matrice intervoxel. Nous détaillons ainsi la suppression d'un sommet, d'une arête et d'une face. Ces opérations sont les opérations élémentaires pour la fusion de régions. Elles ne respectent pas les propriétés de la carte topologique, comme la minimalité. Ces contraintes sont prises en compte par les opérations de plus haut niveau utilisant les opérations de suppression de cellules.

### 4.1.1 Suppression d'un sommet

Le plongement associé à un sommet de la carte combinatoire est un pointel allumé dans la matrice intervoxel. Dans l'Algorithme 8, nous commençons par éteindre le pointel désigné par le triplet associé à $b$ puis nous utilisons l'opération de suppression d'un sommet (Algorithme 4) dans une carte combinatoire afin d'obtenir la suppression du sommet représenté par une carte topologique.

**Algorithme 8** : Suppression d'un sommet

**Données** : Une carte topologique $M$ ;
Un brin $b$.
**Résultat** : Suppression dans $M$ du sommet de degré inférieur ou égal à 2 incident à $b$.

éteindre le pointel désigné par *triplet*$(b)$;
suppression du sommet incident à $b$ dans la carte combinatoire;

### 4.1.2 Suppression d'une arête

Il existe dans le modèle des cartes topologiques deux types d'arêtes : les arêtes réelles et les arêtes fictives. Ces dernières n'ont pas de plongement ainsi la suppression d'une arête fictive utilise uniquement l'opération de 1-suppression de la carte combinatoire. Pour les arêtes réelles, il est nécessaire de prendre en compte le plongement qui est un ensemble connexe de lignels allumés n'étant pas séparés par un pointel allumé dans la matrice intervoxel.

Dans l'Algorithme 9, lorsque l'arête incidente à $b$ n'est pas fictive, nous éteignons dans la matrice l'ensemble des lignels représentant le plongement de l'arête. Il s'agit donc de parcourir tous les lignels successifs en partant du lignel désigné par le triplet associé à $b$ en terminant par le lignel désigné par le triplet associé à $\beta_2(b)$ (le cas des boucles n'est pas un cas particulier, le fait que le même pointel soit désigné par les deux triplets n'entraîne pas de différence dans le parcours des lignels). Enfin, que l'arête soit réelle ou fictive, nous utilisons l'opération de suppression d'une arête (définie par Algorithme 3) afin de supprimer de la carte combinatoire les brins correspondants.

**Algorithme 9** : Suppression d'une arête

**Données** : Une carte topologique $M$ ;
Un brin $b$.
**Résultat** : Suppression dans $M$ de l'arête de degré inférieur ou égal à 2 incidente à $b$.

**si** *l'arête incidente à b est réelle* **alors**
  éteindre les lignels du plongement de l'arête;
suppression de l'arête incidente à $b$ dans la carte combinatoire;

### 4.1.3 Suppression d'une face

La dernière opération est l'opération de suppression de face. Comme pour la 2-suppression, cette opération fusionne les volumes incidents. Dans l'opération présentée par l'Algorithme 10, nous réalisons la suppression des brins de la cellule et l'extinction du plongement géométrique de la face.

La face incidente à $b$ est plongée par un ensemble connexe de surfels allumés et qui sont séparés par des lignels éteints. Nous utilisons un parcours en largeur sur les surfels allumés à partir du surfel désigné par le triplet associé à $b$. Pour cela une pile de surfels, initialisée avec le surfel désigné par le surfel de *triplet*$(b)$, est utilisée. Nous traitons ensuite les surfels de la pile : pour chaque surfel nous

**Algorithme 10** : Suppression d'une face

**Données** : Une carte topologique $M$ ;
Un brin $b$.
**Résultat** : Suppression dans $M$ de la face incidente à $b$.

éteindre les surfels du plongement de la face incidente à $b$;
suppression de la face incidente à $b$ dans la carte combinatoire;

regardons ses quatre voisins qui appartiennent à la même face et nous les ajoutons à la pile. Puis nous éteignons le surfel courant. Une fois le plongement complètement éteint dans la matrice intervoxel, nous utilisons l'opération de 2-suppression (Algorithme 2) pour retirer les brins de la face dans la carte combinatoire.

## 4.2 Approche locale de la fusion de régions

L'approche locale de la fusion de régions fusionne un ensemble connexe de régions en une seule région que nous appelons région *résultante*. Le caractère local de cette approche implique que nous minimisions les modifications de la carte topologique et que nous ne dépendions pas de la taille des données représentant l'image pour effectuer la fusion. Ainsi nous nous interdisons de parcourir tous les brins de l'image et par conséquent nous devons par exemple modifier et non reconstruire l'arbre d'imbrication. Comme il s'agit d'une opération de modification d'une carte topologique, l'approche locale de la fusion doit également respecter la propriété de minimalité de la carte combinatoire et mettre à jour le plongement géométrique.

Nous commençons par détailler l'algorithme général utilisé pour la fusion de régions. Nous expliquons ensuite comment nous réalisons les trois étapes de l'algorithme. Enfin nous donnons la complexité de l'opération en fonction de la taille des données en entrée.

### 4.2.1 Algorithme

L'Algorithme 11 présente la fusion de régions par approche locale. Il prend en paramètres une carte topologique $M$ et un ensemble 6-connexe $S$ de régions à fusionner. Il modifie la carte topologique de sorte que toutes les régions de $S$ soient fusionnées. Il comporte trois étapes principales :

1. calcul de la région résultante et marquage des faces intérieures (c'est-à-dire les faces entre deux régions à fusionner) ;
2. mise à jour de l'arbre des régions pour prendre en compte les modifications possibles de l'arbre d'imbrication ;
3. mise à jour de la carte combinatoire et du plongement dans la matrice intervoxel en supprimant les faces intérieures et en simplifiant si besoin les arêtes et sommets incidents.

Pour éviter la création d'une nouvelle région qui serait le résultat de la fusion des régions sélectionnées, nous utilisons l'une des régions initiales qui contiendra l'ensemble des voxels. Cette région à la fin de l'algorithme représente la fusion des régions dans la carte topologique. Nous appelons cette région la *région résultante*.

Cette région n'est pas choisie au hasard : le brin représentant de cette région doit respecter les contraintes définies Section 2.1.8 pour être le brin représentant de la région finale. De manière générale, les brins représentants de plusieurs régions respectent cette condition. La Proposition 3 indique que la plus petite région de l'ensemble à fusionner respecte toujours cette condition. Ainsi, nous utilisons la plus petite région comme région résultante.

**Algorithme 11** : Approche locale de la fusion de régions

**Données** : Une carte topologique $M$ ;
Un ensemble 6-connexe de régions $S$.
**Résultat** : Fusionne ensemble les régions de $S$ dans $M$.

choisir la plus petite région de $S$ comme région résultante;
**pour chaque** *brin $b$ appartenant aux régions de $S$ dans la carte* **faire**
  **si** region$(\beta_3(b)) \in S$ **alors**
    marquer $b$ et $\beta_3(b)$;
mettre à jour l'arbre d'imbrication des régions;
supprimer toutes les faces intérieures (préalablement marquées);
simplifier les cellules incidentes aux faces supprimées;

**Proposition 3.** *La région la plus petite d'un ensemble de régions à fusionner est toujours un candidat possible pour devenir la région résultante.*

*Démonstration.* Pour être la région résultante, le brin représentant de la région doit être incident à une face qui sépare l'ensemble des régions à fusionner d'une région plus petite. Nous appelons $F$ l'ensemble des régions à fusionner et $r_{min} = min(F)$. Quelle que soit la région $r$ dans $F$, nous avons $r_{min} \leq r$. D'autre part, le brin $b$ représentant de $r_{min}$ respecte les contraintes définies dans la Section 2.1.8. $b$ est incident à une face qui sépare $r_{min}$ d'une région $r'$ et telle que $r' < r_{min}$. Nous concluons donc que $b$ est incident à une face qui sépare l'ensemble des régions de $F$ d'une région $r'$ plus petite : $r' < r_{min} \leq \{r \in F\}$. Le brin $b$ respecte donc la contrainte d'un brin représentant pour la région finale : $r_{min}$ est toujours un candidat possible pour la région résultante. □

### 4.2.2 Marquage des faces intérieures

Le marquage des faces intérieures consiste à marquer les brins appartenant aux faces situées entre deux régions qui fusionnent (c'est-à-dire deux régions appartenant à l'ensemble $S$).

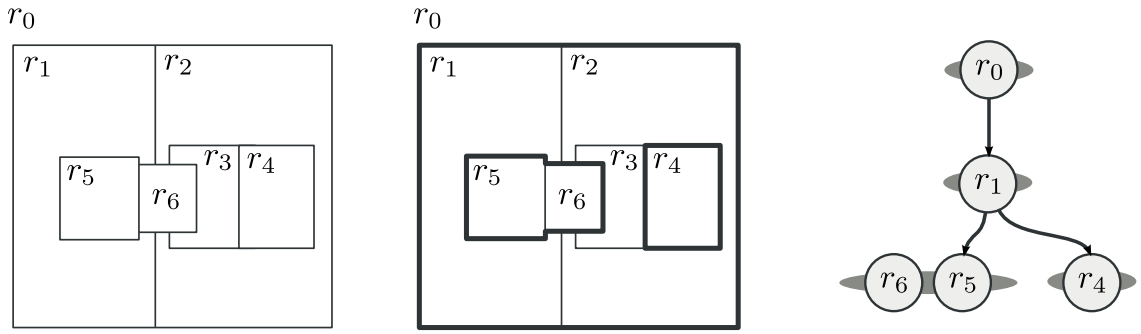
Afin de détecter si un brin appartient à une région à fusionner, nous marquons au préalable les régions de l'ensemble $S$. Avec cette marque sur les régions, nous détectons en temps constant si un brin $b$ appartient à une région sélectionnée en testant si *region*$(b)$ est marquée. En effet, tester une marque sur un brin ou sur une région est réalisé en temps constant et comme chaque brin connaît sa région d'appartenance, l'opération *region*$(b)$ est également réalisée en temps constant.

Pour réaliser le marquage des faces intérieures, nous parcourons tous les brins appartenant aux régions sélectionnées et nous marquons chaque brin $b$ incident à une face intérieure, c'est-à-dire tel que $region(b) \in S$ et $region(\beta_3(b)) \in S$.

### 4.2.3 Mise à jour de l'arbre d'imbrication

L'étape suivante met à jour l'arbre d'imbrication des régions afin de refléter la configuration finale des régions dans l'image après la fusion (voir Figure 4.1 pour un exemple de reconstruction de l'arbre d'imbrication illustré dans une image 2D pour simplifier les dessins, le fonctionnement est identique en 3D).

Dans un premier temps, nous retirons les régions qui fusionnent de l'arbre d'imbrication sauf la région résultante sélectionnée au début de l'algorithme. Celle-ci se trouve déjà bien placée dans l'arbre d'imbrication car la région résultante se trouve dans la bonne composante connexe de régions imbriquées.




(a) (b) (c)

FIG. 4.1: Illustration du processus de mise à jour de l'arbre d'imbrication. (a) Situation initiale avec 7 régions. (b) Nous fusionnons les régions $r_1$, $r_2$ et $r_3$. Nous identifions alors les surfaces frontières (dessinées en gras). (c) Arbre d'imbrication des régions mis à jour. Nous ajoutons les régions imbriquées par une même surface interne dans une composante connexe imbriquée dans la région résultante.

Ensuite, les régions imbriquées dans la région résultante sont détectées. Pour cela, nous identifions les différentes surfaces frontières de la région résultante. Chaque surface interne correspond à un ensemble de régions imbriquées. En parcourant les brins de la carte appartenant à la composante connexe de la surface interne, nous trouvons toutes les régions directement imbriquées et nous les ajoutons à l'arbre d'imbrication dans la même composante connexe.

Les relations d'imbrication des régions qui n'appartiennent pas à une cavité de la région résultante ne sont pas modifiées. Avec cette opération, nous avons bien mis à jour l'arbre d'imbrication des régions.

### 4.2.4 Suppression des faces intérieures et simplification

La dernière partie consiste à mettre à jour la carte combinatoire et son plongement dans la matrice intervoxel pour que la partition représentée soit une carte topologique valide dans laquelle les régions sélectionnées sont fusionnées ensemble.

Dans un premier temps, les faces intérieures de la carte topologique sont supprimées en utilisant l'algorithme de suppression d'une face présenté par l'Algorithme 10. La suppression d'une face change le degré des arêtes incidentes. Les arêtes peuvent donc devenir non minimales.

Dans un second temps, nous simplifions la carte topologique en utilisant l'opération de suppression d'arêtes de l'Algorithme 9. La suppression d'une arête change le degré des sommets incidents qui peuvent à leur tour devenir non minimaux. Nous utilisons l'opération de suppression de sommets présentée par l'Algorithme 8 afin de les supprimer.

À la fin de l'algorithme, la carte est minimale et ne contient pas de faces internes à une région. Nous avons une carte topologique valide qui représente la partition de l'image où les régions sélectionnées sont fusionnées.

### 4.2.5 Complexité

Nous étudions la complexité en temps de l'algorithme local de fusion de régions. La sélection de la région résultante est effectuée en $O(|S|)$ : il s'agit d'une recherche de la région minimale dans l'ensemble $S$, la comparaison entre deux régions étant réalisée en temps constant.

La boucle marquant les faces intérieures parcourt tous les brins appartenant aux régions de $S$ exactement une fois. Tester si $region(\beta_3(b)) \in S$ est réalisé en temps constant. Le marquage des faces

intérieures a une complexité en $O(|\textit{brins}(S)|)$ où $\textit{brins}(S)$ est l'ensemble des brins des régions sélectionnées dans l'ensemble $S$ : $\textit{brins}(S) = \bigcup_{r \in S} \textit{brins}(r)$.

La suppression dans l'arbre d'imbrication d'une région est réalisée en temps constant. Ainsi la complexité de la suppression de toutes les régions sélectionnées sauf la région résultante s'exprime en $O(|S|)$. Comme il y a au moins un brin par région de $S$, $|S| \leq |\textit{brins}(S)|$.

Pour la construction du nouvel arbre d'imbrication de la région résultante, nous parcourons tous les brins des régions sélectionnées afin de trouver les nouvelles surfaces internes : la complexité de cette partie est en $O(|\textit{brins}(S)|)$. S'il n'y a pas de nouvelle surface interne, il n'y a aucune région nouvellement imbriquée, cette étape est terminée. Si par contre il y a des nouvelles surfaces internes alors il existe des régions nouvellement imbriquées : nous notons l'ensemble de ces régions par $S_{imbrique}$. Afin de détecter toutes les régions appartenant à $S_{imbrique}$, un parcours des brins appartenant à chaque composante connexe de régions de $S_{imbrique}$ est réalisé : sa complexité est $O(|\textit{brins}(S_{imbrique})|$. La complexité de la construction du nouvel arbre d'imbrication est $O(|\textit{brins}(S)| + |\textit{brins}(S_{imbrique})|)$. Dans le pire des cas, toutes les régions de la carte topologique deviennent imbriquées dans la région résultante et ainsi tous les brins de la carte combinatoire sont parcourus.

La mise à jour de la carte combinatoire et de la matrice intervoxel utilise les opérations de suppression de cellules dans une carte topologique. La complexité de ces opérations dépend du nombre de brins utilisés pour représenter la cellule et du nombre d'éléments intervoxel appartenant au plongement de ces cellules. Les cellules supprimées appartiennent toutes aux bords des régions sélectionnées. Nous savons également que le nombre de brins utilisés pour représenter une cellule est contraint par le degré maximal des éléments intervoxels représentant ces cellules. Ainsi nous considérons que l'ensemble des opérations de suppression dans la carte combinatoire est linéaire en fonction du nombre de brins appartenant aux régions sélectionnées. Reste le problème du plongement. L'opération principale de suppression est la suppression de faces. Pour supprimer une face nous éteignons l'ensemble des surfels représentant son plongement. Ensuite nous supprimons les arêtes et sommets incidents. Si la surface comporte un seul surfel, il y a quatre arêtes et quatre sommets incidents : cette configuration nous permet d'anticiper sur le fait que le nombre d'éléments intervoxel éteints durant la suppression de cellules est proportionnel au nombre de surfels supprimés. Comme nous ne supprimons que des faces de l'ensemble des régions sélectionnées $S$, la mise à jour de la matrice intervoxel évolue linéairement par rapport au nombre de surfels appartenant au plongement des régions de $S$. Nous notons l'ensemble de ces surfels par $\textit{surfels}(S) = \bigcup_{r \in S} \textit{surfels}(r)$. La complexité de la mise à jour de la carte combinatoire et de la matrice intervoxel s'exprime en $O(|\textit{brins}(S)| + |\textit{surfels}(S)|)$. La complexité totale de l'opération de fusion locale s'exprime en $O(|\textit{brins}(S)| + |\textit{brins}(S_{imbrique})| + |\textit{surfels}(S)|)$. Cette valeur est majorée par le nombre de brins et le nombre de surfels de la carte topologique.

## 4.3 Approche globale de la fusion de régions

L'approche globale de la fusion de régions est plus générale que l'approche locale. La méthode globale fusionne simultanément plusieurs ensembles 6-connexes de régions alors que l'approche locale ne permet que la fusion d'un ensemble 6-connexe de régions. Le principe de l'approche globale est de séparer les modifications de la carte topologique du processus de fusion de régions proprement dit. Dans un premier temps, les régions sont manipulées avec un haut niveau d'abstraction. Puis, dans un second temps, les fusions de haut niveau sont retranscrites dans la partition représentée par une carte topologique en supprimant les cellules inutiles et en construisant le nouvel arbre d'imbrication des régions.

Nous commençons par détailler l'algorithme général utilisé pour la fusion globale de régions, puis nous expliquons comment nous réalisons les trois étapes de l'algorithme. Enfin nous donnons la complexité de l'opération en fonction de la taille des données en entrée.

### 4.3.1 Algorithme

L'Algorithme 12 présente le principe de la fusion de régions par approche globale. Il prend en paramètres une carte topologique $M$ et une fonction $Oracle : R \times R \rightarrow \{vrai, faux\}$ qui indique si deux régions doivent être fusionnées. L'algorithme modifie la carte topologique de sorte que tous les ensembles 6-connexes de régions désignés par l'oracle soient effectivement fusionnés dans la partition finale. Il comporte trois étapes principales :

- fusion des régions à haut niveau d'abstraction : c'est la fusion symbolique ;
- mise à jour de la carte combinatoire minimale et de la matrice intervoxel ;
- construction du nouvel arbre d'imbrication.

**Algorithme 12** : Approche globale de la fusion de régions

**Données** : Une carte topologique $M$ ;
Une fonction $Oracle : R \times R \rightarrow \{vrai, faux\}$.
**Résultat** : Fusionne toutes les régions par composante 6-connexe en fonction de l'oracle.

**pour chaque** *brin* $b \in M$ **faire**
  **si** $Oracle(\text{region}(b), \text{region}(\beta_3(b)))$ **alors**
    fusionner symboliquement *region*$(b)$ et *region*$(\beta_3(b))$;
supprimer toutes les faces intérieures;
simplifier la carte topologique;
construire le nouvel arbre d'imbrication des régions;

### 4.3.2 Fusion symbolique

La première partie de l'algorithme concerne la fusion des régions à un haut-niveau d'abstraction. Pour gérer la fusion et la représentation des ensembles de régions, nous utilisons une structure permettant de représenter des ensembles disjoints : les forêts d'ensembles disjoints. Au départ, chaque région appartient à un ensemble distinct. L'opération de fusion est définie sur les ensembles disjoints comme l'union des ensembles contenant les régions à fusionner. Nous propageons en même temps les caractéristiques internes des régions comme le nombre de voxels ou la couleur moyenne. Nous choisissons le représentant d'un ensemble disjoint de régions comme la région résultante de la fusion de ces régions. Lorsque toutes les fusions ont été effectuées, chaque racine d'un arbre est une région résultante.

Si l'oracle $O$ indique qu'il faut fusionner une région $r_1$ avec sa région adjacente $r_2$ alors nous fusionnons l'ensemble disjoint contenant $r_1$ et l'ensemble disjoint contenant $r_2$. À la fin de la fusion symbolique, comme seules des régions adjacentes ont été fusionnées, chaque ensemble disjoint représente une composante connexe de régions. Un exemple d'oracle peut être une fonction sur les labels qui demande la fusion des régions possédant un même label.

### 4.3.3 Suppression des faces intérieures et simplification

L'étape suivante de l'algorithme consiste à transcrire dans la carte topologique la partition représentée symboliquement par les ensembles disjoints de régions. Pour cela nous supprimons les faces intérieures de la partition symbolique, c'est-à-dire les faces qui sont entre deux régions appartenant au même ensemble disjoint.

Le processus est similaire à celui utilisé pour les faces intérieures de l'ensemble des régions sélectionnées dans l'approche locale (voir Section 4.2.4). La différence porte sur l'utilisation des ensembles disjoints : un brin $b$ appartient à une face intérieure si *trouver*(*region*$(b)$) $=$ *trouver*(*region*$(\beta_3(b))$)

(c'est-à-dire que les deux régions incidentes à la face appartiennent au même ensemble disjoint). Nous utilisons alors l'opération de suppression de faces (Algorithme 10) dans la carte topologique pour supprimer la face incidente à $b$.

Comme dans l'approche locale, la suppression des faces intérieures change le degré des arêtes incidentes. La carte combinatoire n'est plus nécessairement minimale. Comme nous nous permettons de modifier largement la carte topologique, nous utilisons une opération de simplification définie comme l'opération permettant de passer d'une carte au niveau 1 à une carte au niveau 3 (voir Section 2.2.2). Avec cette opération la carte combinatoire devient minimale. La carte et son plongement représentent la partition de l'image décrite par l'oracle.

### 4.3.4 Construction du nouvel arbre d'imbrication

L'étape finale consiste à construire le nouvel arbre des régions représentant la relation d'imbrication directe entre toutes les régions de l'image. Dans l'approche locale, Section 4.2.3, nous modifions l'arbre afin qu'il représente correctement l'imbrication. Deux solutions s'offrent à nous pour construire le nouvel arbre d'imbrication dans l'approche globale : soit nous utilisons sur chaque ensemble disjoint de régions la méthode utilisée dans l'approche locale, soit nous reconstruisons complètement l'arbre des régions. Nous avons choisi d'utiliser la seconde approche en réutilisant l'algorithme de construction de l'arbre d'imbrication défini pour l'extraction de la carte topologique d'une image. La complexité dans le pire des cas est la même mais le processus de construction est moins compliqué à mettre en œuvre et n'entraîne pas de surcoût en terme de complexité.

Dans un premier temps, nous parcourons tous les brins de la carte topologique afin d'attribuer leurs régions d'appartenance à la racine de l'arbre représentant l'ensemble disjoint. Ainsi pour chaque brin $b$ de la carte topologique, $\mathit{region}(b) \leftarrow \mathit{trouver}(\mathit{region}(b))$. Dans un deuxième temps, nous réinitialisons la liste triée des régions et nous supprimons en même temps toutes les régions qui ne sont pas racines d'un ensemble disjoint de régions. Enfin nous utilisons l'Algorithme 6, page 44, afin de construire le nouvel arbre d'imbrication des régions.

### 4.3.5 Complexité

Nous étudions la complexité en temps de l'algorithme global de fusion de régions. Les opérations sur les ensembles disjoints étant considérées comme réalisées en temps constant dans les cas pratiques, nous n'en tenons pas compte dans l'étude de la complexité de la fusion symbolique qui utilise cette structure pour représenter la partition des régions. Nous supposons que la complexité de l'oracle s'exprime à l'aide d'une fonction $f_{Oracle}$. La première étape consiste à tester à l'aide de l'oracle chaque couple de régions adjacentes pour les fusionner. Pour trouver l'ensemble des couples de régions adjacentes, nous parcourons l'ensemble des brins de la carte. L'union des deux ensembles étant réalisée en temps constant, la complexité de cette opération est $O(|\mathit{brins}(M)| \times f_{Oracle})$.

Nous exprimons la complexité de l'étape de simplification en fonction du nombre total de brins de la carte et en fonction du nombre total de surfels de la carte. La première partie se justifie par le fait que nous supprimons des cellules de l'image et toutes les cellules de l'image sont représentées par l'ensemble des brins de la carte. Pour la deuxième partie, nous utilisons le même argument que lors de l'approche locale (voir Section 4.2.5) pour dire que le nombre d'éléments intervoxels à éteindre lors de la phase de suppression des faces intérieures et de simplification est linéairement lié au nombre de surfels allumés dans la matrice intervoxel. Ainsi la complexité s'exprime en $O(|\mathit{brins}(M)|+|\mathit{surfels}(M)|)$.

Enfin la complexité de l'algorithme de construction de l'arbre d'imbrication dépend du nombre de brins de la carte topologique ainsi que du nombre de régions. Comme il y a au moins un brin par région nous exprimons la complexité de cette opération uniquement en fonction du nombre de brins. Nous assurons que la racine de l'arbre soit toujours bien triée dans la liste des régions. La carte combinatoire

étant à jour, nous avons les conditions nécessaires pour utiliser l'algorithme de construction de l'arbre d'imbrication dont la complexité est $O(|\mathit{brins}(M)|)$. La complexité de l'approche globale de la fusion de régions est donc finalement en $O(|\mathit{brins}(M)| + |\mathit{surfels}(M)|)$.

## 4.4 Suppression des régions isolées

Les deux approches de la fusion de régions sont des approches générales capables de fusionner les régions connexes dans toutes les configurations. Une configuration particulière a retenu notre attention : il s'agit de la suppression des régions isolées. Nous définissons une région isolée comme une région qui est imbriquée dans une autre région en remplissant totalement la cavité (c'est-à-dire qu'il n'y a aucune autre région imbriquée dans la même cavité), et ne contenant pas de région imbriquée.

Ces régions sont fréquentes lorsqu'il y a du bruit dans l'image (type poivre et sel) : il s'agit souvent de quelques voxels isolés dans une région plus grande. Afin de les faire disparaître, il est possible d'utiliser les opérations de fusion classique. Nous proposons cependant une approche dédiée qui tire parti de l'isolement de la région pour proposer un algorithme plus efficace permettant de fusionner ces régions avec leur région englobante. L'Algorithme 13 présente l'opération de suppression des régions isolées. Il prend en paramètre une carte topologique et un entier $s$ qui détermine la taille maximum d'une région qualifiée d'isolée. L'algorithme fusionne alors dans la carte topologique les régions isolées contenant moins de $s$ voxels dans leurs régions englobantes.

**Algorithme 13** : Approche globale de la fusion de régions

**Données** : Une carte topologique $M$ ;
Un seuil $s$.
**Résultat** : Fusionne les régions isolées dans leurs régions englobantes.

**pour chaque** *région* $r \in M$ **faire**
  **si** $\mathrm{CC}(r) = \{r\}$ *et si* $|\mathrm{voxels}(r)| \leq s$ **alors**
    `// r est une région isolée de taille` $\leq s$
    supprimer $r$ de l'arbre d'imbrication;
    éteindre le plongement de la surface de $r$;
    supprimer tous les brins de $\langle \beta_1, \beta_2, \beta_3 \rangle(\mathit{rep}(r))$;

Dans l'arbre d'imbrication, une région isolée est représentée par une feuille de l'arbre. Ces régions sont les seuls éléments de leur composante connexe : $\mathit{CC}(r) = \{r\}$. Nous supprimons alors la feuille de l'arbre d'imbrication sans avoir à reconstruire ou mettre à jour les relations entre les régions restantes. De plus, la région isolée est séparée de sa région englobante par une seule surface. La suppression de la région entraîne la disparition de la surface. Nous éteignons dans un premier temps tous les surfels de cette surface, puis nous supprimons donc tous les brins atteignables à partir du brin représentant de $r$. Comme la région est isolée, cela ne fait que supprimer les brins décrivant les cellules de la surface de la région.

La complexité dans le pire des cas est identique à celle de la fusion de régions lorsque toutes les régions sont considérées comme isolées, nous devons parcourir tous les brins de la carte combinatoire. En pratique cependant, la complexité dépend du nombre de surfels et du nombre de brins représentant les régions isolées ce qui rend l'opération attractive pour traiter ce type de régions.

## 4.5 Comparaison des approches locales et globales

Nous remarquons que les complexités des approches locale et globale de la fusion de régions sont identiques dans le pire cas. Cependant les différences entre les algorithmes font que les temps de traitements sont différents selon les configurations. Nous cherchons à comparer les temps d'exécution de la fusion locale et de la fusion globale en fonction des configurations de régions à fusionner.

Dans les expériences suivantes, nous utilisons une image 3D cubique de 64 voxels de côté comme image de départ. Nous utilisons une partition de cette image comportant une région par voxel et la région infinie soit $64^3 + 1 = 262145$ régions. Nous utilisons ici une image artificielle afin de comparer les résultats sans que le contenu de l'image intervienne dans les résultats mesurés. Dans la carte topologique représentant cette partition, nous sélectionnons des régions à fusionner de manière à former différentes configurations. Nous appliquons la fusion locale et la fusion globale à ces régions pour comparer le comportement des deux approches et mettre en lumière leurs points forts et leurs points faibles. Lors des différentes expériences suivantes nous détaillons les stratégies utilisées pour sélectionner les régions à fusionner.

Afin de valider les résultats expérimentaux sur les images artificielles, nous comparons également les deux approches dans des applications sur des images réelles provenant du domaine médical.

### 4.5.1 Fusion d'un ensemble connexe de régions

Cette première comparaison vise à différencier les deux approches lorsqu'une seule composante connexe de régions est fusionnée. Pour cela, nous introduisons deux protocoles d'expérimentation pour ce cas.

Dans un premier temps, Figure 4.2a, nous fusionnons un nombre croissant de régions en utilisant les deux approches sans créer d'imbrication. Nous observons que le temps de calcul de l'approche locale de la fusion augmente linéairement avec le nombre de régions à fusionner. Par contre, le temps de traitement de la fusion de régions par approche globale tend à décroître faiblement avec le nombre de régions à fusionner. En effet, lors de la fusion d'un grand nombre de régions, le nombre de brins total de la carte topologique décroît. De ce fait, la simplification et la reconstruction de l'arbre d'imbrication sont moins coûteuses que lorsqu'il y a peu de régions fusionnées. Pour conclure cette expérimentation, nous notons que l'approche locale est plus rapide que l'approche globale tant que le nombre de régions à fusionner reste petit. Au contraire, l'approche globale est plus rapide lorsque le nombre de régions fusionnées est grand par rapport à la taille de l'image.

La Table 4.1 présente le temps de traitement des trois différentes étapes des deux algorithmes. Les étapes sélectionnées sont celles utilisées lors de la présentation générale des algorithmes. Nous observons dans les résultats de l'approche locale que chaque étape prend un temps plus important lorsque le nombre de régions fusionnées augmente. L'approche globale se comporte différemment. L'étape de fusion symbolique prend un peu plus de temps lorsque le nombre de régions fusionnées augmente mais les opérations de suppression de faces, de simplification et de reconstruction de l'arbre d'imbrication prennent de moins en moins de temps lorsque le nombre de régions fusionnées augmente. Ces valeurs montrent la différence entre les deux approches et expliquent le comportement observé sur le temps de traitement total.

Dans un deuxième temps, nous nous intéressons à l'influence du nombre de régions imbriquées sur le temps de traitement des deux approches. Nous choisissons de fusionner un nombre constant de régions (5768), mais ces régions sont sélectionnées de manière à imbriquer de plus en plus de régions. Les mesures reportées Figure 4.2b montrent la fusion d'un petit nombre de régions par rapport à la taille de l'image. Cela explique pourquoi l'approche globale est plus lente que l'approche locale dans cette expérience. Le résultat qui nous intéresse dans cette expérience est que le temps de traitement de l'opération globale reste pratiquement constant malgré l'évolution du nombre de régions imbriquées,

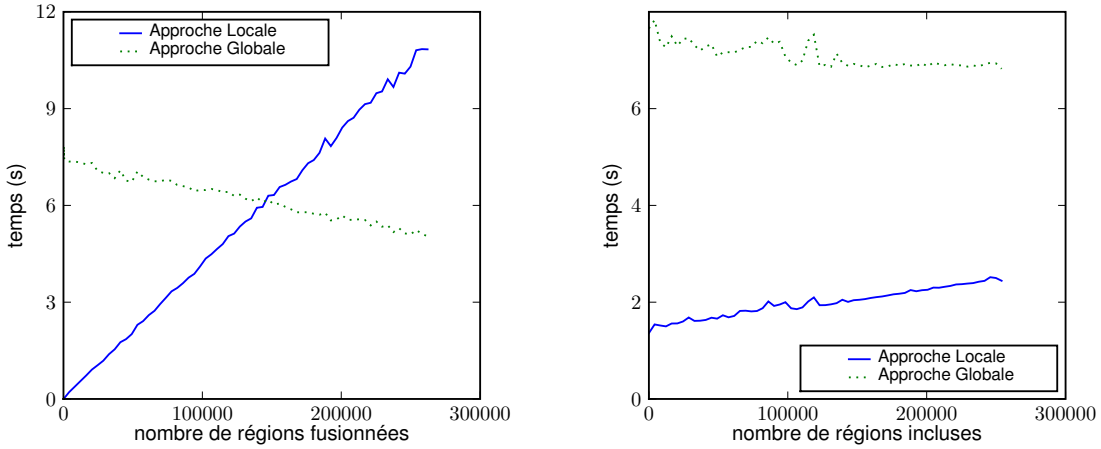



FIG. 4.2: Comparaison de l'approche locale et globale de la fusion en fonction (a) du nombre de régions à fusionner dans une seule région résultante. (b) du nombre de régions imbriquées dans la région résultante.

TAB. 4.1: Temps d'exécution (en secondes) des différentes étapes des deux approches en fonction du nombre de brins appartenant aux régions fusionnées.

| Brins fusionnés | 24576 | 196608 | 393216 | 589824 | 786432 |
|---|---|---|---|---|---|
| Approche locale | | | | | |
| Initialisation | 0,008 | 0,052 | 0,100 | 0,148 | 0,196 |
| Suppression de faces & simplification | 0,028 | 0,156 | 0,208 | 0,316 | 0,388 |
| Mise à jour de l'arbre d'imbrication | 0,012 | 0,120 | 0,304 | 0,480 | 0,616 |
| Total | 0,048 | 0,328 | 0,612 | 0,944 | 1,200 |
| Approche globale | | | | | |
| Initialisation | 0,060 | 0,064 | 0,064 | 0,072 | 0,072 |
| Suppression de faces & simplification | 0,520 | 0,476 | 0,424 | 0,380 | 0,336 |
| Construction de l'arbre d'imbrication | 0,288 | 0,228 | 0,152 | 0,084 | 0,008 |
| Total | 0,868 | 0,768 | 0,640 | 0,536 | 0,416 |

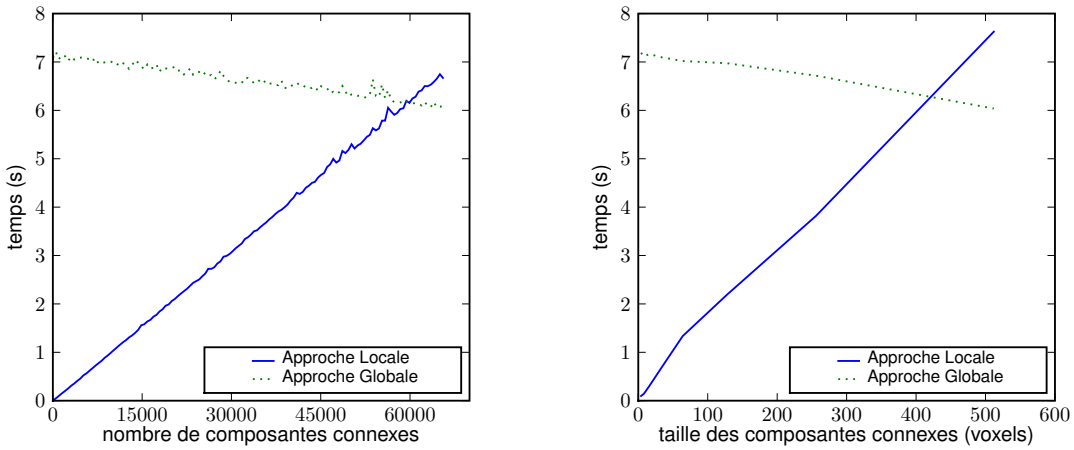



FIG. 4.3: Comparaison de l'approche locale et globale de la fusion en fonction : (a) du nombre d'ensemble de régions de cardinalité deux fusionnées. (b) de la taille des 256 ensembles fusionnés.

alors que le temps de traitement de l'approche locale augmente petit à petit. Ce comportement est dû à la complexité de la mise à jour de l'arbre d'imbrication des régions. Cette étape prend de plus en plus de temps lorsque le nombre de régions imbriquées augmente, car il s'agit de parcourir tous les brins appartenant à des régions nouvellement imbriquées. L'approche globale ne dépend pas de la configuration des régions imbriquées : son temps d'exécution varie peu sur cet exemple.

### 4.5.2 Fusion de plusieurs ensembles de régions connexes

La seconde comparaison vise à différencier les deux approches de la fusion de régions lorsque les régions sélectionnées forment plusieurs composantes de régions connexes. Nous étudions l'impact du nombre d'ensembles de régions sur les temps de traitement, ainsi que l'impact de leur taille. Pour fusionner plusieurs ensembles de régions connexes à l'aide de l'approche locale, nous utilisons l'algorithme itérativement sur chaque ensemble de régions connexes et nous additionnons le temps d'exécution de l'opération sur chaque ensemble. Le même résultat est obtenu en donnant à l'approche globale la totalité des ensembles en une seule fois.

Dans un premier temps, Figure 4.3a, nous fusionnons un nombre croissant d'ensembles de deux régions connexes. Nous observons que l'approche locale prend de plus en plus de temps avec l'augmentation du nombre d'ensembles de régions alors que l'approche globale tend à être de plus en plus rapide. Le comportement de l'approche locale est conforme à ce que nous attendons : l'utilisation répétée d'une opération ayant toujours le même coût (le nombre de régions fusionnées est toujours identique) implique une augmentation linéaire du temps d'exécution. Le comportement de l'approche globale s'explique comme précédemment par le fait que le nombre de brins restants diminue, et donc la simplification de la carte et la construction de l'arbre d'imbrication sont de plus en plus rapides. L'approche globale est plus efficace que l'approche locale lorsque le nombre d'ensembles est grand.

Dans un second temps, nous observons l'influence de la taille des ensembles de régions connexes sur les deux approches de la fusion. Dans cette expérience, Figure 4.3b, nous fusionnons toujours 256 ensembles de régions connexes, mais nous agissons sur la taille des ensembles en augmentant leurs nombres de 2 à 64 régions connexes. Comme dans les expériences précédentes, le temps d'exécution de l'approche globale diminue lorsque le nombre total de régions fusionnées augmente. Inversement l'augmentation de la taille des ensembles rend l'approche locale de plus en plus lente. Ainsi l'approche

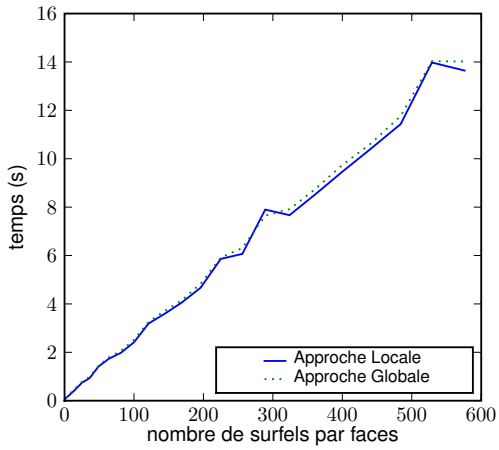



FIG. 4.4: Comparaison de l'approche locale et de l'approche globale de la fusion en fonction du nombre de surfels par face.

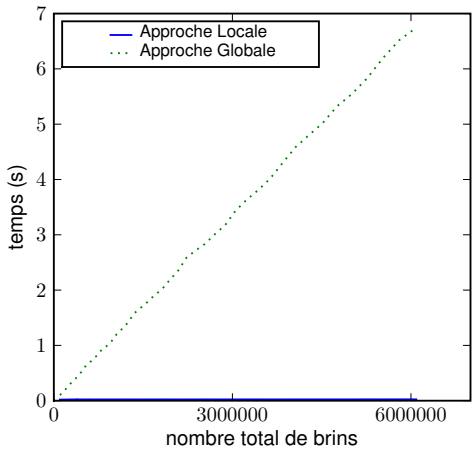



FIG. 4.5: Comparaison de l'approche locale et de l'approche globale de la fusion en fonction du nombre total de brins de la carte topologique. La courbe correspondante à l'approche locale est confondue avec l'axe des abscisses, le temps d'exécution moyen est de 0,02 secondes.

globale est plus rapide lorsqu'il s'agit de fusionner de nombreux ensembles comportant beaucoup de régions.

### 4.5.3 Influence du plongement

Dans l'expression de la complexité des opérations, nous remarquons que les deux approches de la fusion dépendent de la taille du plongement géométrique des faces supprimées lors du traitement. Dans cette expérience, nous étudions l'influence du nombre de surfels par face sur le temps d'exécution des opérations de fusion.

La Figure 4.4 présente la comparaison des temps d'exécution des deux approches pour la fusion de 2048 régions dans une image comportant 4096 régions. Nous faisons varier le nombre de surfels de chaque face de 1 à 576. Étant donnée la taille des deux images et le nombre de régions fusionnées, les temps d'exécution des deux approches sont quasiment identiques. Nous observons que les deux méthodes de fusion de régions dépendent linéairement du nombre de surfels des faces supprimées de la carte topologique.

### 4.5.4 Influence de la taille de la carte

Pour mettre en évidence les différences entre les deux approches, nous fusionnons un nombre constant de régions en faisant varier la taille de l'image et donc le nombre de régions. Nous fusionnons 512 régions, sans créer d'imbrication, dans chaque expérience en augmentant progressivement la taille de l'image de $16^3 + 1$ régions à $64^3 + 1$ régions. Ainsi nous comparons les temps d'exécution des deux approches de la fusion en fonction du nombre total de brins dans la carte topologique représentant la partition de l'image.

La Figure 4.5 présente les temps d'exécution des deux approches de la fusion en fonction du nombre total de brins de la carte. Nous constatons que les deux approches se comportent bien différemment. Le temps d'exécution de la fusion locale de 512 régions ne change pas quelle que soit la taille de l'image

TAB. 4.2: Caractéristiques des images réelles utilisées pour les mesures.

| | Image 1 | Image 2 | Image 3 |
|---|---|---|---|
| Nombre de voxels | 2883584 | 7274496 | 8126464 |
| Régions initiales | 147924 | 431576 | 310421 |

dans laquelle se trouvent ces régions. Par contre, nous remarquons comme prévu que l'approche globale de la fusion dépend linéairement de la taille de l'image.

Toutes les expériences comparant l'approche locale et l'approche globale de la fusion de régions ont une conclusion commune : l'approche locale est plus rapide lorsque le nombre de régions fusionnées est petit et le nombre de différentes composantes connexes reste faible. Cette propriété se retrouve généralement lorsque la fusion de régions est utilisée de manière interactive ou lorsque les régions sont fusionnées à l'aide d'un critère local, par exemple, pour fusionner toutes les régions adjacentes à une région donnée. Si le traitement est plus important, l'approche globale est plus efficace comme par exemple lors d'un processus de segmentation par fusion de régions. De plus, l'approche locale varie de manière très importante lorsque la fusion des régions crée des imbrications alors que l'approche globale possède un temps d'exécution plus stable.

### 4.5.5 Expériences pratiques

Afin de valider les résultats expérimentaux obtenus sur les images artificielles, nous comparons l'utilisation des deux approches dans différentes configurations en condition réelle. Pour chaque opération effectuée, nous mesurons successivement le temps d'exécution des deux approches de la fusion de régions. Nous utilisons trois images médicales différentes afin de faire une moyenne pour que les temps mesurés correspondent bien aux temps moyens des opérations et ne soient pas perturbés par une configuration locale spécifique. La Table 4.2 précise les caractéristiques des images utilisées.

La première expérience est une opération de segmentation qui consiste à fusionner un grand nombre de composantes connexes de régions. Les résultats sont présentés dans la Figure 4.6.

Nous observons que dans ce cas l'approche locale est nettement moins performante que l'approche globale. La différence s'accroît avec le nombre de régions fusionnées. En effet, l'approche globale ne dépend que faiblement du nombre de régions fusionnées et de leurs configurations relatives. De plus l'approche globale gère de manière directe la multiplicité des ensembles de régions connexes. Pour obtenir le même résultat avec l'approche locale nous devons regrouper les régions à fusionner par ensembles 6-connexes de régions en utilisant pour cela l'Algorithme 14.

Dans cet algorithme, étant donnée une région de départ non traitée, nous parcourons l'ensemble de ses brins de la carte topologique. Nous ajoutons chaque région de $F$ incidente à l'un des brins parcourus à la composante connexe $E$, si elle n'est pas déjà traitée. Nous marquons alors la région comme traitée. Nous procédons ensuite de la même façon pour toutes les régions de l'ensemble $E$ pour construire la composante 6-connexe maximale. La composante est 6-connexe car nous regardons uniquement les régions qui sont incidentes par faces. Nous ajoutons alors l'ensemble des régions découvertes dans la liste des différents ensembles et nous traitons la prochaine région non marquée qui débute un nouvel ensemble 6-connexe. Parmi les ensembles construits à l'aide de cette méthode nous appelons l'opération de fusion locale sur tous les ensembles comportant plus d'un élément (sinon, il n'y a pas de fusion à effectuer).

La seconde expérience utilise l'image résultante du traitement précédent. L'utilisateur de l'application cherche à fusionner manuellement un nombre restreint de régions. La Figure 4.7 compare les temps d'exécution des deux approches de fusions. Lorsque le nombre de régions reste faible, l'approche locale se révèle nettement plus avantageuse mais son avantage disparaît rapidement lorsque

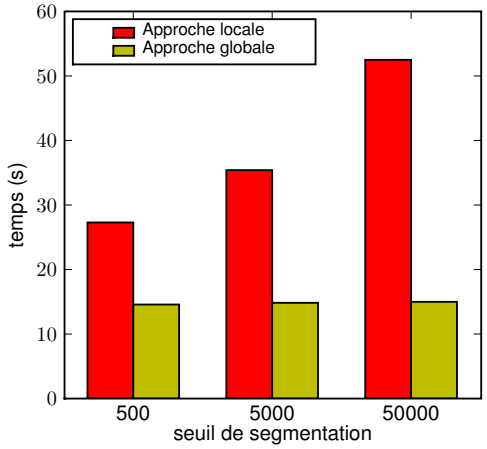



FIG. 4.6: Comparaison de l'approche locale et de l'approche globale lors d'un traitement automatisé impliquant un grand nombre de fusions de régions. Plus le seuil de segmentation est grand, plus le nombre de régions fusionnées est important.

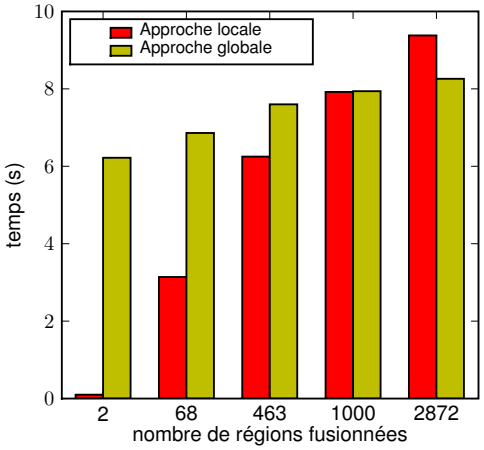



FIG. 4.7: Comparaison de l'approche locale et de l'approche globale lors d'un traitement manuel de fusion de régions. Un utilisateur utilise une interface afin de sélectionner des régions à fusionner.

**Algorithme 14** : Découpe en ensembles 6-connexes de régions

**Données** : Une carte topologique $M$ ;
Un ensemble $F$ quelconque de régions.
**Résultat** : Partition de $F$ en ensembles 6-connexes de régions.

$L \leftarrow$ liste de régions vide;
$m_{traite} \leftarrow$ marque sur les régions;
**pour chaque** *région* $r \in F$ **faire**
  **si** $r$ *n'est pas marqué par* $m_{traite}$ **alors**
    marquer $r$ avec $m_{traite}$;
    $E \leftarrow \{r\}$;
    `// parcours de la composante connexe qui contient` $r$
    **pour chaque** *région* $r_E \in E$ **faire**
      **pour chaque** *brin* $b \in$ brins$(r_E)$ **faire**
        **si** region$(\beta_3(b))$ *n'est pas marqué par* $m_{traite}$ *et si* region$(\beta_3(b)) \in F$ **alors**
          marquer *region*$(\beta_3(b))$ avec $m_{traite}$;
          ajouter *region*$(\beta_3(b))$ à $E$;
    enfiler $E$ à $L$;
**retourner** $L$;

nous essayons de fusionner beaucoup de régions. Nous observons également que le temps d'exécution de l'approche globale augmente avec le nombre de régions fusionnées. Le nombre de brins par faces et le nombre de surfels font que le temps moyen de traitement augmente. Ce n'est qu'en partie compensé par l'augmentation du nombre de régions fusionnées qui entraîne une diminution du temps d'exécution.

## 4.6 Conclusion

Dans ce chapitre, nous avons défini deux approches pour l'opération de fusion de régions. La première approche, qualifiée de locale, permet la fusion d'un ensemble connexe de régions. La seconde approche, qualifiée de globale, permet la fusion d'un nombre quelconque d'ensembles connexes de régions. Bien que les deux approches aient la même complexité, l'opération de fusion globale est plus rapide lorsqu'il s'agit de réaliser beaucoup de fusion et est ainsi plus adaptée pour une utilisation automatique dans le cadre d'un processus de segmentation. Dans le Chapitre 5, nous présentons l'éclatement et la division de régions qui sont les opérations inverses de la fusion de régions et permettent de mettre en œuvre des segmentations par division-fusion.

CHAPITRE 5

# ÉCLATEMENT ET DIVISION DE RÉGIONS

L'opération inverse de la fusion de régions est la division de régions. L'objectif est d'obtenir plusieurs régions en découpant une région initiale. De manière générale, la division de régions utilise un guide, par exemple une surface, afin de découper la région initiale. Le nombre de régions résultantes dépend alors de la géométrie de la région initiale et de la géométrie du guide utilisé pour la découpe. Une division particulière est l'éclatement d'une région. Il s'agit de la division d'une région de manière à ce que chaque région résultante ne contienne qu'un seul voxel.

Nous définissons la notion de cellule élémentaire comme étant une cellule qui ne peut pas être divisée géométriquement dans une carte topologique car il n'existe pas de plongement géométrique plus petit que celui d'une cellule élémentaire. Notons d'abord que tous les sommets sont élémentaires. Une arête élémentaire est une arête dont le plongement est composé d'un seul lignel. De la même façon, une face élémentaire est une face composée d'un seul surfel. Enfin, un volume élémentaire est un volume contenant un seul voxel.

Nous détaillons dans la suite les diverses opérations préalables à l'éclatement de régions et à la division de régions par un guide. Nous commençons donc par l'éclatement de cellules : éclatement d'arêtes, éclatement de faces et éclatement de volumes. Les algorithmes d'éclatement d'une cellule ne garantissent pas la propriété de minimalité de la carte topologique. En effet, ces opérations ajoutent des cellules non minimales à la carte topologique. Il s'agit d'opérations intermédiaires, le respect de la propriété de minimalité est garanti par les opérations principales : l'éclatement de régions et la division de régions par un guide, qui utilisent les opérations d'éclatement de cellules.

## 5.1 Éclatement d'une arête

L'éclatement d'une arête est l'opération qui consiste à remplacer une arête par une suite d'arêtes élémentaires. L'algorithme insère successivement des sommets sur l'arête à diviser jusqu'à obtenir uniquement des arêtes élémentaires. Nous étudions dans un premier temps l'insertion de sommets sur une arête, puis nous développons l'algorithme qui insère successivement tous les sommets sur l'arête à éclater.

### 5.1.1 Insertion de sommet

L'insertion d'un sommet sur une arête est l'opération de base de l'éclatement d'arêtes. L'opération se déroule en deux temps : d'abord au niveau topologique puis au niveau géométrique. Il s'agit d'insérer

dans la carte combinatoire les brins correspondants au nouveau sommet. Ensuite nous attribuons le plongement géométrique au sommet inséré.

L'Algorithme 15 présente l'insertion d'un sommet dans la carte topologique. Il prend en paramètres une carte topologique, un brin désignant l'arête à éclater ainsi que les coordonnées d'un pointel. L'algorithme insère un sommet sur l'arête désignée avec le plongement géométrique donné par le pointel.

**Algorithme 15** : Insertion d'un sommet sur une arête

**Données** : Une carte topologique $M$ ;
Un brin $b$ ;
Un pointel $p$.
**Résultat** : Insertion d'un sommet de plongement $p$ sur l'arête incidente à $b$ dans $M$

```
pour chaque brin b_0 ∈ ⟨β_2, β_3⟩(b) faire
    b_1 ← nouveau brin;
    β_1(b_1) ← β_1(b_0);
    β_1(b_0) ← d_1;
    si le brin β_2(b_0) a déjà été traité alors
        b_3 ← β_2(b_0);
        b_2 ← β_21(b_0);
        coudre par β_2 les brins b_1 et b_3;
        coudre par β_2 les brins b_0 et b_2;
    si le brin β_3(b_0) a déjà été traité alors
        b_2 ← β_3(b_0);
        b_3 ← β_31(b_0);
        coudre par β_3 les brins b_1 et b_2;
        coudre par β_3 les brins b_0 et b_3;
allumer le pointel p;
pour chaque brin b_i ∈ ⟨β_21, β_31⟩(β_1(b)) faire
    pointel(b_i) ← p;
```

Le principe de l'algorithme est illustré par la Figure 5.1. Pour chaque brin $b_0$ de l'orbite de l'arête incidente à $b$, nous ajoutons à la carte $M$ un nouveau brin appelé $b_1$ qui est incident au sommet inséré comme le montre la Figure 5.1b. Le brin $b_1$ est inséré entre $b_0$ et $\beta_1(b_0)$.

Ensuite nous modifions les relations $\beta_2$ et $\beta_3$ afin de donner la bonne topologie au sommet et aux deux arêtes que nous construisons. Pour cela nous regardons si :

1. le brin image par $\beta_2$ de $b_0$ a déjà été traité ;
2. le brin image par $\beta_3$ de $b_0$ a déjà été traité.

Dans le premier cas, illustré Figure 5.1c, le brin $b_2 = \beta_{21}(b_0)$ est un nouveau brin qui ne possède pas encore de relation $\beta_2$. Nous relions alors les quatre brins ($b_0$, $b_1$, $b_2$ et $\beta_2(b_0)$) par $\beta_2$ de manière à ce que les brins qui doivent être mis en correspondance dans l'orbite de l'arête soient convenablement reliés.

Dans le deuxième cas, illustré Figure 5.1d, le brin $b_3 = \beta_{31}(b_0)$ est un nouveau brin qui ne possède pas encore de relation $\beta_3$. Nous relions alors les quatre brins ($b_0$, $b_1$, $b_3$ et $\beta_3(b_0)$) par $\beta_3$ de manière à ce que les brins qui doivent être mis en correspondance dans l'orbite de l'arête soient convenablement reliés.

La Figure 5.1e montre le résultat après le traitement du brin suivant et la Figure 5.1f illustre le résultat final de l'opération.

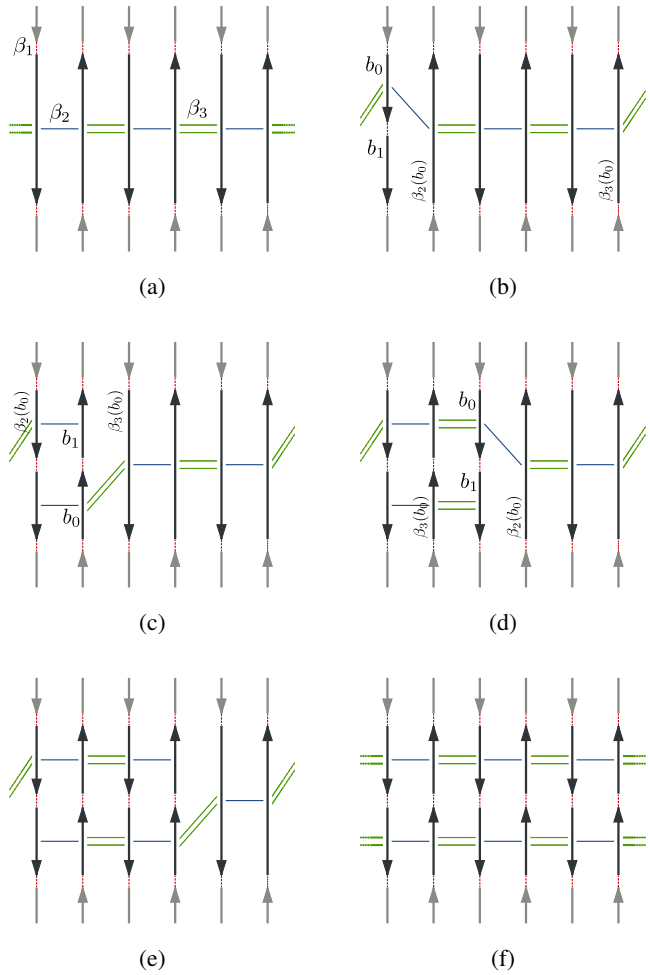
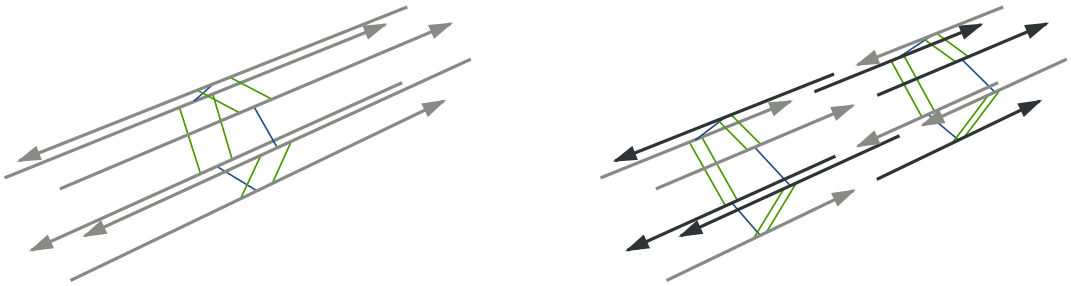



FIG. 5.1: Illustration de l'insertion d'un sommet sur une arête composée de six brins. (a) Configuration initiale avec six brins représentant l'arête incidente à $b$. La légende des relations entre les brins est donnée sur la figure. Les brins en gris clair sont les brins des arêtes incidentes. (b) Insertion de $b_1$ entre $b_0$ et $\beta_1(b_0)$. Les brins $\beta_2(b_0)$ et $\beta_3(b_0)$ n'ont pas encore été traités : nous ne modifions pas les relations $\beta_2$ ou $\beta_3$. (c) Lors du traitement du brin suivant, le brin $\beta_2(b_0)$ a été traité. Nous modifions donc les relations $\beta_2$ pour rendre conforme l'orbite des deux arêtes en cours de création. (d) Lors du traitement du brin suivant, le brin $\beta_3(b_0)$ a été traité. Nous modifions alors les relations $\beta_3$ pour être conforme aux orbites des deux nouvelles arêtes. (e) Traitement du brin suivant. (f) Résultat une fois que tous les brins ont été traités.



(a) (b)

FIG. 5.2: Insertion d'un sommet sur une arête illustrée en 3D. (a) Configuration initiale avec une arête représentée par six brins. (b) Résultat de l'insertion du sommet sur l'arête. Les brins dessinés en foncé sont les nouveaux brins et forment l'orbite sommet.

La Figure 5.2 illustre l'insertion d'un sommet sur une arête composée de six brins placés en 3D.

À ce stade, la topologie du sommet sur l'arête est définie. Nous mettons à jour la géométrie. Pour cela, nous allumons le pointel $p$ et nous attribuons à chaque brin de l'orbite sommet le pointel $p$. Le triplet effectif de chaque brin dépend de l'arête et de la face incidente à ce brin. Notons que l'opération d'insertion de sommets n'est pas définie pour les arêtes fictives car celles-ci n'ont pas de plongement.

### 5.1.2 Itération sur une arête

En utilisant l'opération d'insertion d'un sommet, nous définissons l'opération d'éclatement d'une arête. Le principe de l'algorithme est d'insérer successivement sur l'arête des sommets de sorte que, pour chaque sommet, nous divisons l'arête en deux : une arête élémentaire et une autre arête sur laquelle nous itérons le processus. La fin du processus est atteinte lorsque l'arête restant à diviser est élémentaire.

L'Algorithme 16 présente le processus d'éclatement d'une arête en arêtes élémentaires. Il prend en paramètres une carte topologique $M$ et un brin $b$ de la carte qui désigne l'arête à éclater. L'algorithme ajoute des sommets dans la carte topologique de manière à ce que l'arête soit éclatée en arêtes élémentaires.

**Algorithme 16** : Éclatement d'une arête

**Données** : Une carte topologique $M$ ;
Un brin $b$.
**Résultat** : Éclatement dans $M$ de l'arête incidente à $b$ en arêtes élémentaires

$p \leftarrow$ *pointel*$(b)$;
**si** $p$ *est éteint* **alors**
  allumer $p$;
$p' \leftarrow$ prochain sommet de l'arête tel que $pp'$ soit un lignel;
**tant que** $p'$ *est éteint* **faire**
  insérer sommet$(M, b, p')$;
  $b \leftarrow \beta_1(b)$;
  $p \leftarrow p'$;
  $p' \leftarrow$ prochain sommet de l'arête tel que $pp'$ soit un lignel;

Si l'arête à supprimer est une boucle isolée, le sommet dans la carte topologique n'a pas de plongement : le pointel associé à $b$ est éteint. Pour initialiser l'algorithme, nous commençons par fixer le plongement du sommet en allumant le pointel dans la matrice intervoxel.

Afin de déterminer si une arête est élémentaire, nous utilisons la matrice intervoxel. Étant donné un brin $b$ incident à une arête, nous obtenons le plongement du sommet de départ par les coordonnées du pointel et le premier lignel du plongement de l'arête. Ces éléments de l'intervoxel sont ceux désignés par le triplet du brin $b$. Nous calculons alors les coordonnées du pointel $p'$ qui est l'autre pointel incident à ce même lignel. Il s'agit d'un calcul simple en fonction des éléments désignés par le triplet de $b$. Soit $\mathit{triplet}(b) = (p, l, s)$, nous utilisons les coordonnées de $p$ comme point de départ pour les coordonnées de $p'$. En fonction de la direction donnée par $(p, l)$, nous ajoutons ou nous retirons un à la coordonnée associée. Par exemple si $(p, l)$ désigne la direction de l'axe $-\vec{z}$, nous retirons un à la troisième coordonnée de $p'$. Si $p'$ est déjà allumé dans la matrice intervoxel, alors l'arête ne comporte qu'un lignel : elle est élémentaire. Sinon, l'arête n'est pas élémentaire et nous l'éclatons en utilisant $p'$ comme plongement lors de l'insertion d'un sommet sur l'arête.

Le choix du plongement pour le nouveau sommet garantit que la nouvelle arête incidente à $b$ est élémentaire. Nous itérons alors le processus d'éclatement sur l'autre arête qui est désignée par $\beta_1(b)$, dont la longueur a été diminué de un. L'algorithme traite successivement cette arête jusqu'à ce que l'arête à diviser soit élémentaire. Notre algorithme réalise donc bien l'éclatement de l'arête.

La Figure 5.3 illustre le principe de l'éclatement d'une arête. Nous ne représentons pas tous les brins de l'orbite arête pour simplifier la lecture du schéma. La Figure 5.3a présente la situation initiale : l'arête $AB$, qui est composée de 3 lignels, va être éclatée en prenant $b_0$ comme brin de départ. La Figure 5.3b montre la situation après l'insertion du sommet $C$ sur l'arête. L'arête $AC$ est élémentaire. Nous itérons le processus sur l'arête $CB$ en utilisant le brin $b_1 = \beta_1(b_0)$ comme brin de départ. Dans la Figure 5.3c, le sommet $D$ est inséré dans l'arête $BC$. L'arête $CD$ est élémentaire. Comme l'arête incidente au brin $b_2 = \beta_1(b_1)$ est également élémentaire, l'algorithme est terminé : l'arête $AB$ est éclatée en arêtes élémentaires.

La Figure 5.4 présente un exemple d'éclatement de toutes les arêtes d'un volume dans une carte topologique 3D.

## 5.2 Éclatement d'une face

L'éclatement d'une face est l'opération qui consiste à remplacer une face par un ensemble de faces élémentaires. L'algorithme insère des arêtes élémentaires à l'intérieur de la face, puis coud ces arêtes aux bords de la face et ainsi crée des faces élémentaires. Nous étudions dans un premier temps l'insertion d'arêtes à l'intérieur d'une face, puis nous développons l'algorithme permettant de saturer une face par des arêtes pendantes. Enfin nous donnons la méthode permettant de fermer les faces et ainsi d'obtenir un ensemble de faces élémentaires à la place de la face initiale.

### 5.2.1 Insertion d'une arête pendante

L'insertion d'une arête élémentaire pendante est l'opération de base de l'éclatement de faces. Elle consiste à insérer dans la carte combinatoire les brins correspondant à l'arête insérée et mettre à jour la matrice intervoxel pour représenter sa géométrie. L'opération insère toujours une arête pendante, c'est-à-dire que topologiquement l'un des sommets incidents à l'arête n'est connecté avec aucune autre arête.

L'Algorithme 17 présente l'opération d'insertion d'une arête pendante. Il prend en paramètres une carte topologique, un brin désignant un sommet et les coordonnées d'un lignel incident au même som-

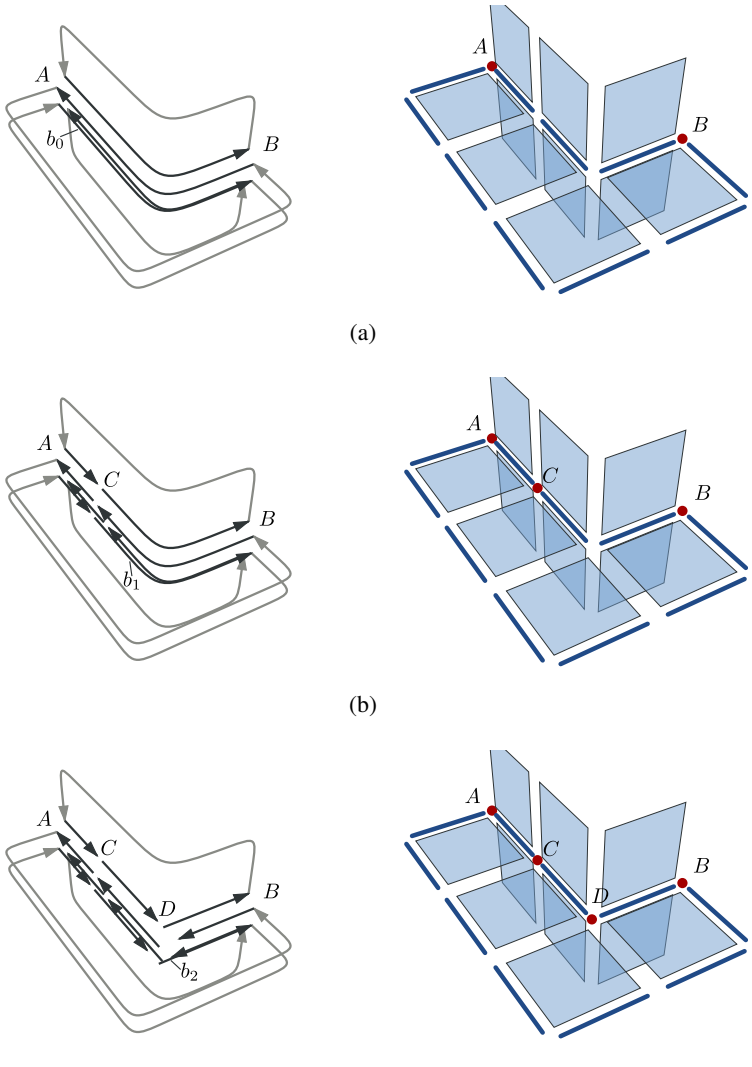
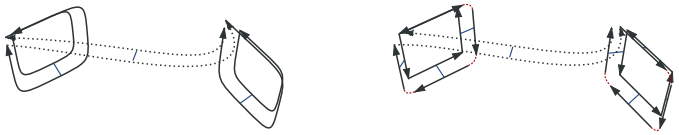



FIG. 5.3: Éclatement de l'arête $AB$ en arêtes élémentaires. (a) Configuration initiale. (b) Insertion du sommet $C$ dans l'arête $AB$ qui n'est pas élémentaire. (c) Insertion du sommet $D$ dans l'arête $CB$. L'arête $DB$ étant élémentaire, l'algorithme est terminé.



(a) (b)

FIG. 5.4: Exemple d'éclatement des arêtes en arêtes élémentaires. (a) Trois arêtes appartenant à la surface d'un volume : il y a deux arêtes réelles et une arête fictive (dont les brins sont dessinés en pointillés). (b) Résultat de l'opération d'éclatement des arêtes : les arêtes réelles sont remplacées par des arêtes élémentaires, l'arête fictive n'est pas modifiée.

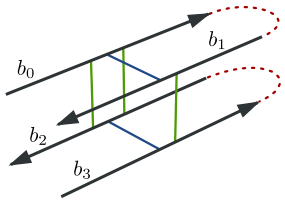



FIG. 5.5: Configuration de quatre brins $b_0$, $b_1$, $b_2$ et $b_3$ formant une arête pendante au sommet incident à $b_1$ et $b_2$. Les brins $b_0$ et $b_3$ ne sont pas cousus par $\beta_0$ et les brins $b_1$ et $b_2$ ne sont pas cousus par $\beta_1$.

met. L'algorithme insère alors les brins et les éléments intervoxels représentant une arête pendante attachée au sommet désigné par $b$.

**Algorithme 17** : Insertion d'une arête pendante

**Données** : Une carte topologique $M$ ;
Un brin $b$ ;
Un lignel $l$.

**Résultat** : Insertion dans $M$ d'une arête pendante de plongement $l$ attachée au sommet désigné par $b$ et dans la face incidente à $b$.

$b_0, b_1, b_2, b_3 \leftarrow$ 4 brins représentant une arête élémentaire de plongement $l$;
coudre par $\beta_1$ les brins $\beta_0(b)$ et $b_0$;
coudre par $\beta_1$ les brins $b_1$ et $b$;
coudre par $\beta_1$ les brins $b_2$ et $\beta_{21}(b)$;
coudre par $\beta_1$ les brins $\beta_2(b)$ et $b_3$;
allumer le lignel $l$;
allumer le pointel *pointel*$(b_1)$;

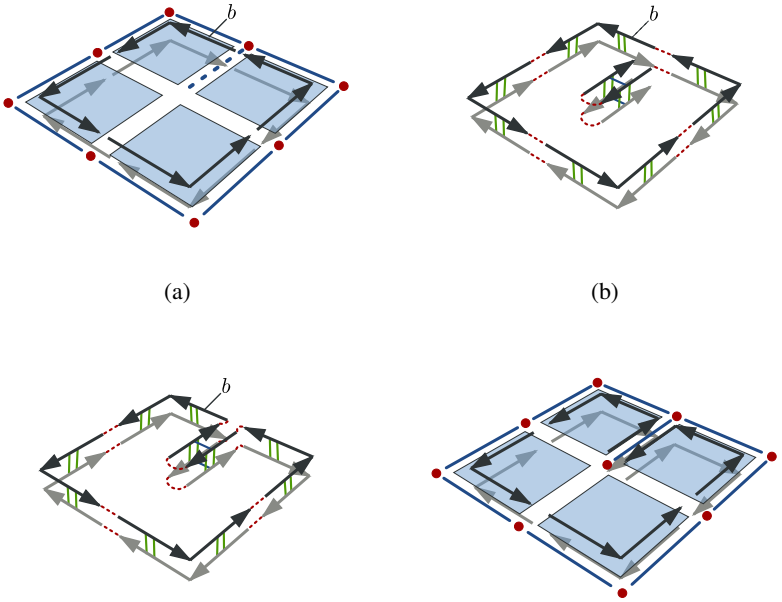
La première étape consiste à construire les quatre brins formant l'arête élémentaire pendante à insérer. Pour cela nous construisons la même configuration que les quatre brins de la Figure 5.5. L'arête est pendante autour du sommet incident à $b_1$.

Dans un second temps, nous attachons le sommet ouvert de l'arête (désigné par $b_0$) avec le sommet désigné par $b$ (et dans la face incidente à $b$). La dernière partie de l'algorithme consiste à allumer le plongement de l'arête et celui du sommet dans la matrice intervoxel.

La Figure 5.6 illustre le fonctionnement de l'algorithme. Dans la Figure 5.6a, nous observons la situation initiale avec une surface et un sommet désigné par $b$. Le lignel marqué en pointillés est le lignel $l$ qui va être utilisé comme plongement pour l'arête à insérer. Dans la Figure 5.6b, nous ajoutons à la carte topologique quatre brins que nous relions pour former une arête pendante au sommet incident à $b_1$. L'étape suivante qui consiste à accrocher cette arête sur le sommet incident à $b$ est réalisée Figure 5.6c. Enfin la géométrie est mise à jour dans la Figure 5.6d.

### 5.2.2 Saturation d'une face par des arêtes élémentaires

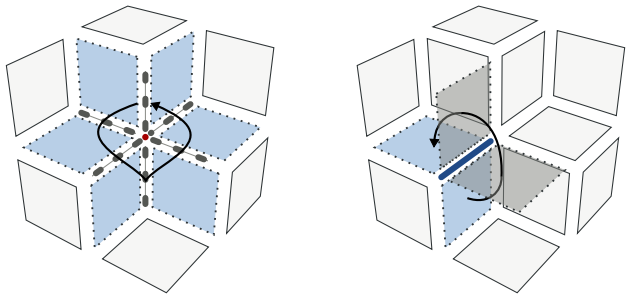
La seconde opération consiste à saturer une face à l'aide d'arêtes élémentaires de manière à ce que chaque lignel se trouvant entre deux surfels appartenant au plongement de la face soit le plongement d'une arête élémentaire. L'opération de saturation ne divise pas la face dans le modèle topologique, c'est l'opération suivante qui ferme les faces réalisant ainsi la division. Par contre, dans la




(a) (b)

(c) (d)

FIG. 5.6: Insertion d'une arête pendante dans une face. (a) Situation initiale de la face. (b) Quatre nouveaux brins sont ajoutés et cousus entre eux pour former une arête pendante. (c) Les brins sont reliés au sommet incident à $b$. (d) Le lignel et le pointel qui correspondent à l'arête et le sommet pendant sont allumés.



(a) (b)

FIG. 5.7: Ombrelles dans la matrice intervoxel. Les éléments intervoxels représentés en pointillés sont les éléments de l'ombrelle. Les flèches signifient la permutation. (a) Les six lignels et les six surfels appartenant à l'ombrelle du pointel dessiné. (b) Les quatre surfels appartenant à l'ombrelle du lignel dessiné.

géométrie, les lignels du plongement de l'arête que nous insérons sont allumés. Ainsi les faces sont divisées géométriquement.

Pour les explications de l'algorithme, nous avons besoin de définir la notion d'ombrelle d'un sommet plongé dans la matrice intervoxel par un pointel. Dans la Figure 5.7, il s'agit de tous les éléments intervoxels représentés en pointillés. L'ombrelle est la permutation des éléments intervoxels incidents au pointel, qui appartiennent à une même surface, de sorte que les éléments consécutifs soient incidents deux à deux. Nous définissons également la notion d'ombrelle autour d'un lignel comme la permutation des surfels incidents au lignel à l'intérieur d'un volume.

L'Algorithme 18 présente l'opération de saturation d'une face par des arêtes élémentaires. Il prend en paramètres une carte topologique et un brin désignant une face. Pour que l'algorithme fonctionne correctement, les bords de la face doivent être éclatés en arêtes élémentaires. L'algorithme insère alors des arêtes élémentaires de manière à ce que chaque lignel entre deux surfels de la face soit le plongement d'une des arêtes insérées.

**Algorithme 18** : Saturation d'une face par des arêtes élémentaires

**Données** : Une carte topologique $M$ ;
Un brin $b$ incident à une face dont les bords sont des arêtes élémentaires.
**Résultat** : Saturation par des arêtes élémentaires de la face incidente à $b$ dans $M$.

$F \leftarrow$ file de brins vide;
**pour chaque** *brin* $b_0 \in \langle \beta_1 \rangle(b)$ **faire**
  **si** *l'arête incidente à* $b_0$ *n'est pas fictive* **alors**
    enfiler $b_0$ sur $F$;
**tant que** $F$ *n'est pas vide* **faire**
  $b_0 \leftarrow$ tête de $F$;
  défiler $F$;
  $L \leftarrow$ lignel de l'ombrelle du sommet incident à $b_0$;
  **pour chaque** *lignel* $l \in L$ **faire**
    **si** $l$ *est éteint* **alors**
      ajouter une arête élémentaire pendante en $b_0$ et $l$;
      enfiler $\beta_0(b_0)$ sur $F$;
      $b_0 \leftarrow \beta_{02}(b_0)$;

L'algorithme utilise un parcours en largeur sur les lignels de la face. Ce parcours est réalisé à l'aide d'une file de brins. L'initialisation du processus consiste à insérer dans la file tous les brins de l'orbite $\langle\beta_1\rangle(b)$ qui n'appartiennent pas à des arêtes fictives. Les arêtes fictives servent à relier les différents bords entre eux de manière à ce que la face initiale soit homéomorphe à un disque topologique. L'opération de saturation ne connecte pas les bords entre eux. Ainsi les arêtes fictives sont toujours utiles. Dans cette opération, nous ne les traitons simplement pas.

Le processus itératif permettant de saturer la face traite chaque brin $b_0$ de la file de la manière suivante : pour chaque lignel éteint de l'ombrelle du sommet incident à $b_0$, nous insérons une arête élémentaire pendante avec l'opération décrite par l'Algorithme 17. Après avoir inséré une arête, nous ajoutons le brin $\beta_0(b_0)$ à la file pour ajouter, plus tard, les arêtes élémentaires qui lui sont incidentes. Afin que l'opération d'insertion d'une arête pendante soit correctement initialisée pour le prochain lignel éteint de l'ombrelle, nous utilisons le brin $\beta_{02}(b_0)$ comme le nouveau brin du sommet. En effet, le brin $\beta_{02}(b_0)$ est le brin de la nouvelle arête qui vient d'être inséré et qui est incident au même sommet que $b_0$. Avec ce mécanisme, nous insérons toutes les arêtes incidentes au sommet $b_0$ avant de passer à un autre sommet. Si l'un de lignels de l'ombrelle est allumé, l'arête élémentaire associée existe déjà. Nous ne nous préoccupons pas de cela, et nous poursuivons le traitement avec le prochain lignel éteint. La couture des arêtes autour du sommet n'est pas cohérente mais nous corrigerons cela dans l'opération de fermeture de faces. Si l'ombrelle du sommet ne comporte pas de lignel éteint, alors le sommet courant est géométriquement confondu avec un autre sommet du bord de la face. Nous n'avons pas d'arête à insérer et nous pouvons passer directement à un autre sommet.

Lorsque tous les brins de la file ont été traités, tous les lignels à l'intérieur de la face incidente à $b$ sont allumés et il existe une arête élémentaire qui possède ce lignel comme plongement. L'ensemble des lignels appartenant aux ombrelles des sommets de la face a été parcouru. Ainsi nous avons réalisé la saturation de la face par des arêtes.

La Figure 5.8 illustre le principe de l'algorithme de saturation de faces par des arêtes élémentaires. La face initiale dont le bord est éclaté est présentée par la Figure 5.8a. Étant donné un brin $b$ incident à une face dont le bord est composé d'arêtes élémentaires, nous insérons une arête élémentaire pendante pour chaque lignel éteint dans l'ombrelle du sommet incident à $b$ restreinte à la face courante. Ainsi nous insérons la première arête comme indiqué Figure 5.8b. Le nouveau brin $b$ est le brin $\beta_{02}(b)$. Il est également incident à un lignel éteint. Nous insérons donc une autre arête autour de ce sommet (Figure 5.8c). Il n'y a plus de lignel éteint dans l'ombrelle de ce sommet, nous passons donc à un autre sommet. Nous insérons la dernière arête pendante comme le montre la Figure 5.8d. Après cette insertion tous les lignels à l'intérieur de la face sont allumés. Il y a une arête pendante pour chacun d'entre eux. Nous avons donc saturé la face par des arêtes pendantes : l'algorithme est terminé.

### 5.2.3 Fermeture des faces élémentaires

Après la saturation de la face par des arêtes élémentaires, la géométrie de la face est correctement décomposée en faces élémentaires. Cependant la topologie n'est pas cohérente par rapport à cette géométrie. L'opération de fermeture des faces élémentaires connecte les arêtes autour de chaque sommet et ainsi permet la création des faces élémentaires qui remplacent la face initiale.

L'Algorithme 19 décrit l'opération de fermeture des faces élémentaires. Il prend en paramètres une carte topologique et un brin incident à une face. L'algorithme suppose que la face est saturée par des arêtes élémentaires pendantes. Il connecte les arêtes autour de chaque sommet de manière à construire les faces élémentaires qui remplacent la face initiale.

Afin de coudre correctement les brins autour des orbites sommets, nous utilisons la géométrie. $P$ est une structure associative qui pour un pointel $p$ désigne un ensemble de brins $D = \{b, \mathit{pointel}(b) = p\}$. En pratique, nous utilisons un tableau associatif implémenté sous forme d'un arbre binaire de recherche équilibré comme nous le trouvons dans le *map* de la librairie standard du C++ (*Standard Template*

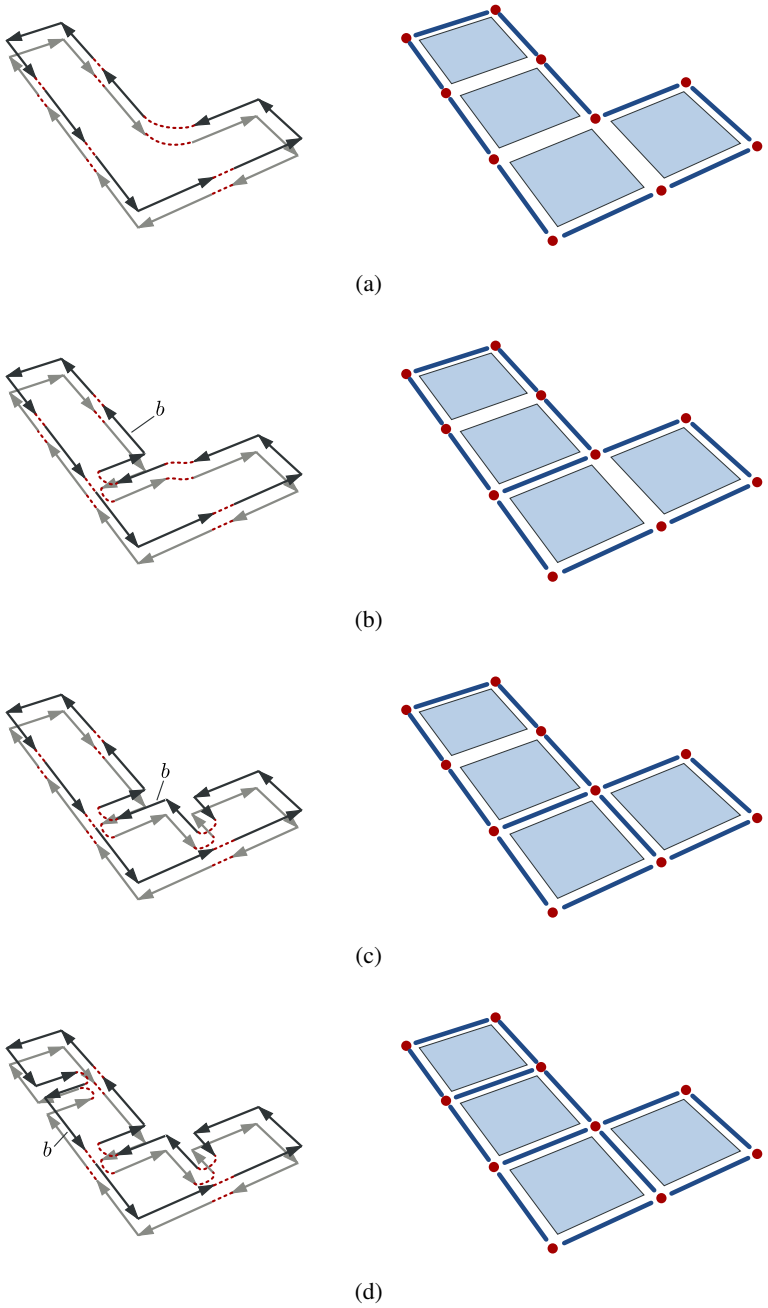



FIG. 5.8: Saturation d'une face par des arêtes élémentaires. (a) Configuration initiale. (b) Insertion de la première arête pendante. (c) Insertion de l'arête pendante suivante autour du sommet. (d) Insertion de la troisième arête pendante. Tous les lignels dans la face sont allumés, la face est saturée.

**Algorithme 19** : Fermeture des faces élémentaires

**Données** : Une carte topologique $M$ ;
Un brin $b$.
**Résultat** : Fermeture dans $M$ des faces élémentaires remplaçant la face incidente à $b$.

$P \leftarrow$ structure associant un pointel à un brin;
**pour chaque** *brin* $b_0 \in \langle\beta_1\rangle(b)$ **faire**
  **si** *arête incidente à* $b_0$ *n'est pas fictive* **alors**
    insérer $(pointel(b_0), d_0)$ dans $P$;
supprimer les arêtes fictives appartenant à la face incidente à $b$;
**pour chaque** *pointel* $p$ *dans* $P$ **faire**
  $D \leftarrow$ liste des brins associés à $p$ dans $P$;
  trier $D$ par ordre des lignels dans l'ombrelle de $p$;
  // $D = (b_0, \ldots, b_{|D|-1}$
  **pour** $i \in [1, |D|[$ **faire**
    **si** $\beta_0(b_i) = \beta_2(b_i)$ **alors**
      coudre par $\beta_1$ les brins $\beta_2(b_{i-1})$ et $b_i$;
      coudre par $\beta_1$ les brins $\beta_3(b_i)$ et $\beta_{23}(b_{i-1})$;

*Library ou STL* en anglais). Nous initialisons donc cette structure avec l'ensemble des brins appartenant à la même face et au même volume que $b$ : les brins de l'orbite $\langle\beta_1\rangle(b)$. Nous n'ajoutons pas les brins appartenant à des arêtes fictives.

En effet, les arêtes fictives permettent de conserver la connexité des bords de la face. Cependant, la fermeture des faces élémentaires connecte les différents bords de la face initiale par des arêtes réelles. Nous supprimons donc les arêtes fictives après avoir initialisé la structure de données car elles deviennent inutiles, les bords étant connectés par les faces élémentaires.

Pour fermer effectivement les faces, nous relions les arêtes pendantes autour de chaque sommet. Ainsi pour chaque pointel $p$ de la structure, nous ordonnons les brins dont le pointel est $p$ en fonction du plongement de leur arête. L'ordre des arêtes autour d'un sommet est donné par l'ombrelle du sommet $p$. Nous relions ensuite par $\beta_1$ les brins successifs de manière à construire l'orbite du sommet $p$, ce qui va connecter les arêtes pendantes.

À la fin de l'algorithme, tous les pointels de $P$ ont été traités. Ainsi chaque arête est cousue correctement autour de son sommet. Cela forme les faces élémentaires qui remplacent la face initiale.

La Figure 5.9 illustre l'algorithme de fermeture des faces élémentaires autour d'un sommet. Dans la Figure 5.9a, il y a quatre arêtes incidentes à un sommet. Deux de ces arêtes sont pendantes, nous les rattachons au sommet. Pour cela, tous les brins associés au pointel $p$ (dans la Figure 5.9d) sont ordonnés selon l'ordre des lignels dans l'ombrelle du sommet. Nous utilisons cet ordre pour coudre les arêtes en commençant par la première : Figure 5.9b. Nous relions ensuite la deuxième arête comme le montre la Figure 5.9c. Cette configuration est le résultat de l'opération de couture des arêtes pendantes autour d'un sommet.

La Figure 5.10 reprend l'exemple utilisé Figure 5.8 pour illustrer la saturation de face par des arêtes élémentaires. Nous avons étiqueté les sommets allant être fermés par $A$, $B$ et $C$.

### 5.2.4 Algorithme d'éclatement d'une face

L'Algorithme 20 prend en paramètres une carte topologique et un brin désignant une face. Le résultat de l'opération est l'éclatement de la face désignée par un ensemble de faces élémentaires.

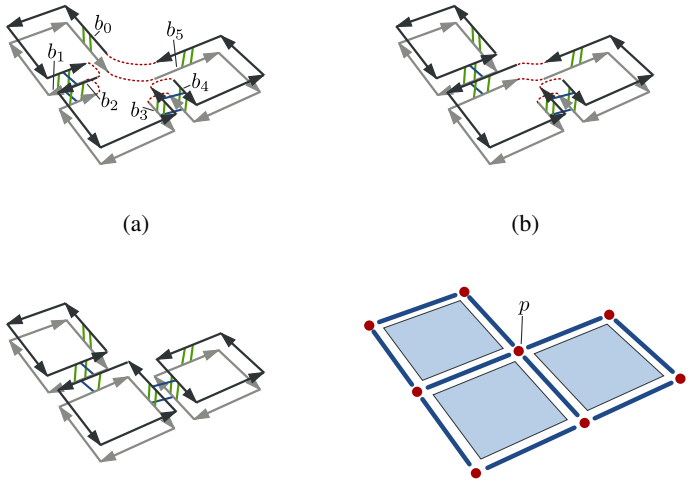



FIG. 5.9: Coutures des arêtes élémentaires autour d'un sommet. (a) Configuration initiale avec 4 arêtes incidentes à un sommet : deux arêtes appartiennent au bord de la face, les deux autres arêtes sont pendantes. Les brins sont numérotés dans l'ordre de l'ombrelle du sommet en partant de $b_0$. (b) Couture de la première arête (incidente à $b_1$ et $b_2$) au sommet. (c) Couture de la seconde arête (incidente à $b_3$ et $b_4$) autour du sommet. Il s'agit de la configuration finale. (d) Éléments intervoxels de la face.

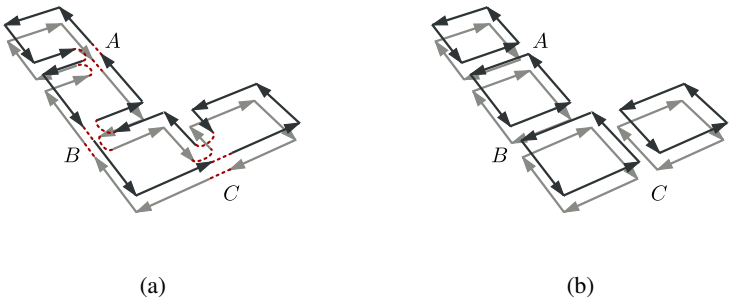



FIG. 5.10: Exemple de fermeture des faces élémentaires lors de l'éclatement d'une face. (a) Configuration initiale, les sommets étiquetés par $A$, $B$ et $C$ sont modifiés par l'Algorithme 19 afin de fermer les faces élémentaires. (b) Résultat de l'opération de fermeture des faces élémentaires. La face initiale est remplacée par quatre faces élémentaires.

**Algorithme 20** : Éclatement d'une face en faces élémentaires

**Données** : Une carte topologique $M$ ;
Un brin $b$.
**Résultat** : Éclatement dans $M$ de la face désignée par $b$ en faces élémentaires.

$fictif \leftarrow vrai$;
**pour chaque** *brin* $b_0 \in \langle\beta_1\rangle(b)$ **faire**
  **si** *arête incidente à* $b_0$ *n'est pas fictive* **alors**
    $fictif \leftarrow faux$;
    éclater l'arête désignée par $b_0$;
**si** $fictif$ **alors**
  créer une arête élémentaire réelle dans la face;
  $b \leftarrow$ brin de la même face et du même volume que $b$ sur la nouvelle arête;
  supprimer toutes les arêtes fictives de la face;
saturer la face incidente à $b$ par des arêtes élémentaires;
fermer les faces élémentaires dans la face incidente à $b$;

Pour réaliser l'éclatement d'une face en faces élémentaires, nous commençons par éclater, avec l'Algorithme 16, les arêtes réelles appartenant au bord de la face afin de respecter la précondition de l'algorithme de saturation de face par des arêtes élémentaires.

Si aucune arête réelle n'a été détectée, la face que nous éclatons est une sphère ou un $k$-tore (le nombre d'arêtes fictives donne le nombre de tunnels : une arête pour une sphère, deux arêtes pour un 1-tore, etc.). Dans cette configuration, il n'y a aucun plongement d'arête sur la face. L'algorithme de saturation ne peut pas être initialisé. Aussi nous construisons une arête réelle élémentaire sur la face (peu importe sa localisation) : cette arête est pendante au niveau de ses deux sommets incidents. Nous définissons que le brin $b$ incident à la face est maintenant l'un des brins de cette arête (nous assurons que le brin appartienne au même volume topologique). Enfin nous supprimons toutes les arêtes fictives, elles ne connectent aucune arête réelle du bord de la face et donc ne servent pas dans les algorithmes suivants.

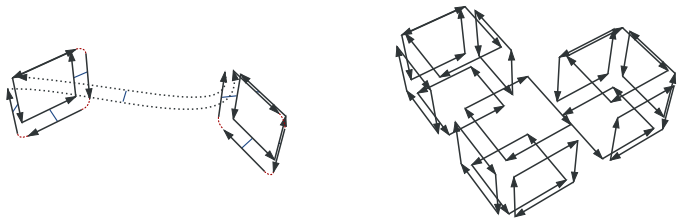
Nous utilisons ensuite l'Algorithme 18 pour obtenir une face saturée par des arêtes élémentaires. Enfin nous utilisons l'opération de fermeture de faces (Algorithme 19) pour terminer le traitement et obtenir un ensemble de faces élémentaires qui remplace la face initiale.

La Figure 5.11 présente un exemple d'éclatement de faces dans une carte topologique 3D en utilisant le même exemple que pour l'éclatement d'arêtes illustré par la Figure 5.3.

## 5.3 Éclatement d'un volume

L'éclatement d'un volume est l'opération qui consiste à remplacer un volume représenté dans la carte topologique par un ensemble de volumes élémentaires. L'algorithme fonctionne sur le même principe que celui d'éclatement d'une face. Il insère des faces élémentaires à l'intérieur du volume initial puis relie les faces avec les bords du volume construisant ainsi des volumes élémentaires. Bien que cette opération travaille sur les volumes et modifie donc la partition représentée, nous ne traitons pas ici la mise à jour des régions.

Nous étudions dans un premier temps l'insertion d'une face élémentaire pendante dans un volume. Nous développons ensuite l'algorithme permettant de saturer un volume par des faces élémentaires. Nous donnons l'opération permettant de relier les faces aux bords du volume pour construire les volumes élémentaires. Enfin, nous détaillons la méthode permettant l'éclatement d'un volume.



(a) (b)

FIG. 5.11: Exemple d'éclatement des faces d'un volume en faces élémentaires. Les bords de chaque face sont composés uniquement d'arêtes élémentaires ou fictives. Les brins des arêtes fictives sont dessinés en pointillés. (a) Une seule face n'est pas élémentaire : elle est bordée par dix brins dont deux appartiennent à l'arête fictive. Nous éclatons donc cette face. (b) Résultat de l'éclatement : l'arête fictive n'est plus nécessaire pour relier les deux bords de la face initiale, elle est donc supprimée.

### 5.3.1 Insertion d'une face élémentaire pendante

L'insertion d'une face élémentaire pendante est l'opération de base pour l'éclatement d'un volume. Elle consiste à insérer dans la carte combinatoire les brins correspondant à la face insérée et à mettre à jour la matrice intervoxel afin de représenter son plongement. Le bord de la face élémentaire est lui-même composé d'arêtes élémentaires. L'opération insère toujours une face pendante, c'est à dire que la face n'est pas rattachée aux éventuelles arêtes incidentes exceptée l'arête élémentaire servant de support à la face. Une face élémentaire ainsi insérée possède donc trois côtés pendants.

L'Algorithme 21 présente l'insertion d'une face élémentaire pendante. Il prend en paramètres une carte topologique, un brin désignant une arête élémentaire et les coordonnées d'un surfel incident à la même arête. L'algorithme insère alors les brins et les éléments intervoxels représentant une face pendante attachée à l'arête incidente au brin.

**Algorithme 21** : Insertion d'une face élémentaire pendante

**Données** : Une carte topologique $M$ ;
Un brin $b$ ;
Un surfel $s$.

**Résultat** : Insertion d'une face élémentaire pendante de plongement $s$ attachée à l'arête désignée par $b$ et dans le volume incident à $b$.

$D \leftarrow$ créer 8 brins formant une face de plongement $s$;
$b_0 \leftarrow$ brin de $D$ tel que $\mathit{lignel}(b_0) = \mathit{lignel}(b)$ et $\mathit{pointel}(b_0) = \mathit{pointel}(b)$;
coudre par $\beta_2$ les brins $\beta_2(b)$ et $b_0$;
coudre par $\beta_2$ les brins $b$ et $\beta_3(b_0)$;
allumer le surfel $s$;
allumer les 4 lignels incidents à $s$;
allumer les 4 pointels incidents à $s$;

Le début de l'algorithme consiste à construire les huit brins composant la face élémentaire de plongement $s$. Les arêtes du bord de cette face sont élémentaires. La topologie de la face étant toujours la même, nous utilisons la configuration présentée Figure 5.12 afin de coudre les brins ensemble.

L'un des brins ajouté, que nous appelons $b_0$, est plongé par le même lignel et le même pointel que le brin $b$ qui désigne l'arête. Les deux brins $b_0$ et $\beta_3(b_0)$ appartiennent en fait à l'arête incidente à $b$. Nous

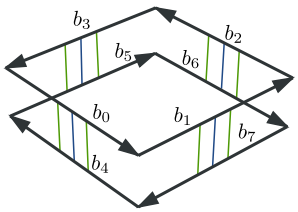



FIG. 5.12: Configuration de huit brins formant une face élémentaire pendante dont le bord est représenté par des cellules élémentaires.

relions les brins par $\beta_2$ conformément à l'orbite de l'arête pour attacher la face nouvellement créée à l'arête incidente à $b$.

Enfin nous actualisons le plongement de la face. Pour cela nous allumons le surfel dans la matrice intervoxel, mais également les quatre lignels qui correspondent aux arêtes élémentaires bordant la face et les quatre pointels qui forment le bord des arêtes. D'après les conditions initiales de l'algorithme et la configuration du voisinage de l'arête, certains de ces éléments intervoxels sont déjà allumés : nous ne modifions pas leur état.

La Figure 5.13 illustre le principe de l'insertion de face. La configuration initiale Figure 5.13a est une arête élémentaire désignée par un brin $b$ où un surfel incident à cette même arête et appartenant au même volume que $b$ n'est pas allumé. Dans une première étape, Figure 5.13b, nous ajoutons les brins associés formant une face élémentaire de plongement $s$. Ensuite nous relions les brins de l'arête incidente à $b$ comme le montre la Figure 5.13c. Enfin, Figure 5.13d, le plongement des cellules insérées est allumé dans la matrice intervoxel.

### 5.3.2 Saturation du volume par des faces élémentaires

L'opération de saturation consiste à remplir un volume de la carte topologique avec des faces élémentaires de manière à ce que chaque surfel se trouvant entre deux voxels appartenant au volume soit le plongement d'une face élémentaire. L'algorithme ne divise pas le volume initial dans le modèle topologique : c'est le rôle de l'opération suivante qui va fermer les volumes et réaliser ainsi la division. Cependant, dans la matrice intervoxel, les surfels qui forment le plongement des faces que nous insérons sont allumés de même que les lignels et les pointels incidents à cette face : le volume est divisé géométriquement.

L'Algorithme 22 présente l'opération de saturation d'un volume par des faces élémentaires. Il prend en paramètres une carte topologique et un ensemble de brins désignant chacun une surface appartenant au bord du volume. De plus l'algorithme requiert que les faces appartenant aux bords de la région soient éclatées en faces élémentaires. Le résultat de l'algorithme est le remplissage du volume par des faces élémentaires de manière à ce que chaque surfel situé entre deux voxels du volume soit le plongement d'une des faces insérées.

L'algorithme sature le volume avec des faces pendantes en réalisant un parcours en largeur des surfels qui se trouvent à l'intérieur du volume.

Nous utilisons une file de brins appelée $F$ et une marque sur les brins. L'initialisation du processus consiste à ajouter dans la file un brin par arête appartenant à la surface du volume. Nous choisissons ce brin de manière à ce qu'il soit également incident au volume à éclater. Nous parcourons pour cela chaque brin $b'$ appartenant à une surface du volume. Si $b'$ n'est pas marqué, l'arête élémentaire incidente

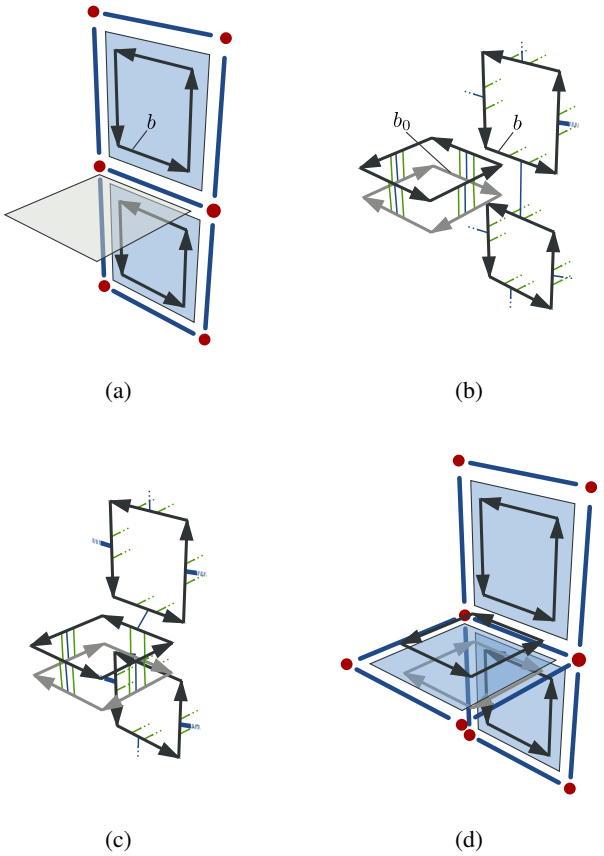



FIG. 5.13: Insertion d'une face élémentaire pendante dans un volume. (a) Configuration initiale avec un brin $b$ incident à une arête élémentaire et un surfel $s$ incident appartenant au même volume que $b$ et qui n'est pas allumé. (b) Ajout des huit brins représentant la face élémentaire pendante. Le brin $b_0$ est plongé avec le même lignel et le même pointel que le brin $b$. (c) Couture des brins $b$, $b_0$ et de leurs images par $\beta_2$ de manière à connecter la face insérée avec l'arête incidente à $b$. (d) Allumage du plongement des cellules insérées.

**Algorithme 22** : Saturation d'un volume par des faces élémentaires

**Données** : Une carte topologique $M$ ;
Un ensemble de brins $D$ contenant un brin par surface du bord du volume à éclater.
**Résultat** : Saturation dans $M$ par des faces élémentaires du volume incident aux brins de $D$.

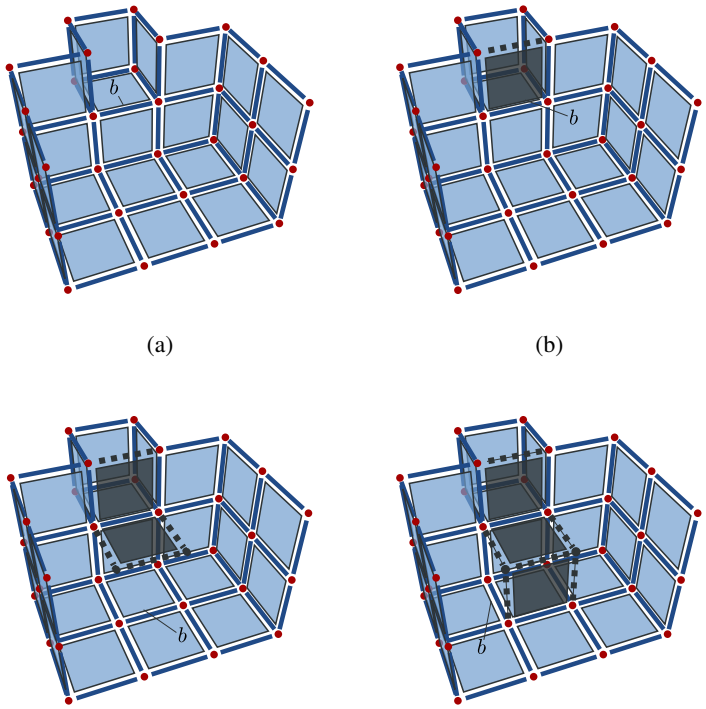
```
F ← file de brins vide;
m_traite ← marque sur les brins;
pour chaque brin b ∈ D faire
    pour chaque brin b' ∈ ⟨β1, β2⟩(b) faire
        si b' n'est pas marqué par m_traite alors
            marquer b' et β2(b') avec m_traite;
            enfiler b' sur F;
tant que F n'est pas vide faire
    b ← tête de F;
    défiler F;
    S ← ombrelle de l'arête incidente à b;
    pour chaque surfel s ∈ S faire
        si s est éteint alors
            ajouter une face élémentaire pendante en b et s;
            b0 ← β23(b);
            enfiler β1(b0), β11(b0) et β111(b0) sur F;
            b ← d0;
```

n'est pas encore représentée dans la file. Nous ajoutons donc $b'$ à $F$ et nous marquons $b'$ et $\beta_2(b')$, les deux brins du volume qui sont incidents à cette arête, pour ne plus les traiter.

Le déroulement de l'algorithme traite ensuite tous les brins de la file jusqu'à ce qu'elle soit vide. Nous appelons $b$ le brin courant. Il s'agit au départ du brin stocké dans la file. Nous récupérons dans une liste $S$ l'ensemble des surfels appartenant à l'ombrelle de l'arête dans le volume à éclater. Afin de traiter les surfels dans l'ordre, nous commençons par le surfel suivant *surfel*$(b)$ vers l'intérieur du volume.

Nous traitons alors chaque surfel $s$ de $S$. Si le surfel est allumé, la face correspondante existe déjà, nous n'ajoutons pas une nouvelle face. Nous passons donc au surfel suivant dans l'ombrelle. Sinon, nous insérons une arête pendante à l'aide de l'Algorithme 21. Nous récupérons le brin $b_0$ incident à la même arête que $b$ mais appartenant à la nouvelle face. Nous ajoutons un brin pour chacune des arêtes du bord de la face dans la file $F$ afin de poursuivre la saturation à partir de ces arêtes (sauf pour l'arête incidente à $b$ et $b_0$). Nous utilisons $b_0$ comme nouveau brin courant afin d'insérer la face correspondant au prochain surfel de $S$ de la bonne façon.

Lorsque $F$ est vide, nous avons ajouté une face pendante pour tous les surfels éteints incidents aux arêtes désignées par les brins de la file. Comme nous avons ajouté un brin pour chaque nouvelle arête appartenant aux bords des faces pendantes, toutes les arêtes ayant comme plongement un lignel à l'intérieur du volume à éclater ont été traitées. Ainsi nous sommes sûrs que tous les surfels sont maintenant allumés et correspondent à des faces élémentaires. La Figure 5.14 illustre les trois premières insertions de faces pendantes lors de l'éclatement d'un volume.




(a) (b)

(c) (d)

FIG. 5.14: Saturation d'un volume par des faces élémentaires. Pour illustrer le principe, nous utilisons uniquement une représentation par la géométrie. Les lignels en pointillés représentent les côtés pendants des faces insérées. (a) Situation initiale avec un volume dont le bord est éclaté en faces élémentaires. Nous ne représentons qu'une version partielle du volume afin de voir ce qui se passe à l'intérieur. $b$ désigne la première arête que nous traitons à l'aide de l'algorithme. (b) Insertion de la face pendante incidente à l'arête désignée par $b$. Il reste un lignel éteint dans le volume autour de l'orbite arête. $b$ désigne alors le brin de la même arête mais appartenant à la nouvelle face. (c) Insertion de la face pendante suivante. Tous les surfels autour de l'arête sont allumés, nous avons ajouté toutes les faces pendantes possibles. $b$ désigne alors le brin suivant appartenant à la surface du volume. (d) Situation après l'insertion de la troisième face. Nous itérons ainsi jusqu'à ce que tous les surfels à l'intérieur du volume soit allumés.

### 5.3.3 Fermeture des volumes élémentaires

Après la saturation du volume par des faces élémentaires, la géométrie du volume est correctement décomposée en volumes élémentaires. Cependant la topologie n'est pas cohérente par rapport à cette géométrie : des faces ne sont pas reliées entre elles autour d'un même lignel. L'opération de fermeture des volumes élémentaires connecte ces faces et ainsi engendre la création des volumes élémentaires qui remplacent le volume initial.

L'Algorithme 23 décrit l'opération de fermeture des volumes élémentaires. Il prend en paramètres une carte topologique et un ensemble de brins désignant chacun une surface du bord du volume. Nous supposons que le volume est saturé par des faces élémentaires. L'algorithme connecte les faces autour de chaque arête de manière à construire les volumes élémentaires qui remplacent le volume initial.

**Algorithme 23** : Fermeture des volumes élémentaires

**Données** : Une carte topologique $M$ ;
Un ensemble de brins $D$ désignant chacun un bord du volume.
**Résultat** : Fermeture dans $M$ des volumes élémentaires remplaçant le volume incident aux brins de $D$.

$L \leftarrow$ structure associant un lignel à un brin;
**pour chaque** *brin* $d' \in D$ **faire**
  **pour chaque** *brin* $d \in \langle \beta_1, \beta_2 \rangle (d')$ **faire**
    insérer $(\mathit{lignel}(d), d)$ dans $L$;
**pour chaque** *lignel* $l$ *dans* $L$ **faire**
  $B \leftarrow$ liste des brins associés à $l$ dans $L$;
  trier $B$ par ordre du plongement des surfels autour de l'arête désignée par $l$;
  // $B = (b_0, \ldots, b_{2k-1})$ avec $k = \frac{|B|}{2}$ le nombre de faces
  $i \leftarrow 1$;
  **tant que** $i < k$ **faire**
    coudre par $\beta_2$ les brins $b_{2i-1}$ et $b_{2i}$;
  coudre par $\beta_2$ les brins $b_0$ et $b_{2k-1}$;

Afin de coudre correctement les brins autour des orbites arêtes, nous utilisons la géométrie comme pour la fermeture des faces. $L$ est une structure associative qui pour un couple pointel, lignel $(p, l)$ désigne un ensemble de brins $B = \{b$, tel que $\mathit{lignel}(b) = l\}$. En pratique, nous utilisons un *map* de la STL comme dans l'opération d'éclatement de faces. Nous initialisons cette structure avec l'ensemble des brins appartenant aux arêtes du bord du volume. Ces brins appartiennent à l'orbite surface des brins de $D$ : $\langle \beta_1, \beta_2 \rangle (d'), \forall d' \in D$.

Ensuite, nous traitons chaque lignel $l$ appartenant à la structure $L$ où $l$ désigne une arête élémentaire appartenant au bord de la région. Nous trions alors les brins qui ont $l$ comme lignel selon leur orientation autour de l'arête de sorte que deux brins consécutifs appartiennent au même volume ou à la même face. Les brins appartenant à la même face sont par construction déjà cousus avec $\beta_3$. Nous ignorons les relations $\beta_2$ qui existent, nous assurons que nous allons les reconstruire correctement. Nous relions par $\beta_2$ les brins qui appartiennent à un même volume. Ce processus est illustré dans la Figure 5.15 par la couture de quatre faces autour d'un lignel.

Lorsque tous les lignels sont traités, toutes les faces élémentaires incidentes à des arêtes désignées par les lignels de $L$ sont cousues ensemble. Il ne reste donc plus de brins sans liaison par $\beta_2$.

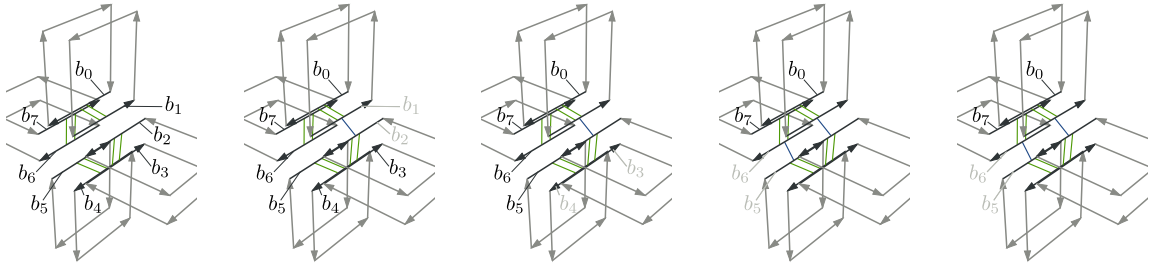



FIG. 5.15: Déroulement de l'opération de fermeture des faces pendantes. (a) Les brins sont numérotés dans l'ordre selon l'ombrelle du lignel. Dans l'algorithme, le nombre de faces est $k = 4$. (b) Première étape de couture, nous relions par $\beta_2$ les brins $b_1$ et $b_2$. (c) Deuxième étape de couture, nous relions par $\beta_2$ les brins $b_3$ et $b_4$. (d) Troisième étape de couture, nous relions par $\beta_2$ les brins $b_5$ et $b_6$. (e) Étape finale, nous relions le premier et le dernier brins par $\beta_2$.

### 5.3.4 Algorithme d'éclatement d'un volume

L'Algorithme 24 prend en paramètres une carte topologique et un ensemble de brins $D$ où chaque brin désigne un bord du volume à éclater. Le résultat de l'opération est l'éclatement du volume désigné par un ensemble de volumes élémentaires.

**Algorithme 24** : Éclatement d'un volume en volumes élémentaires

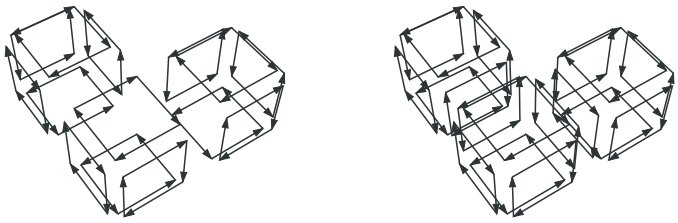
**Données** : Une carte topologique $M$ ;
Un ensemble de brins $D$ désignant chacun un bord du volume.
**Résultat** : Éclatement dans $M$ du volume désigné par les brins de $D$ en volumes élémentaires.

```
pour chaque brin b ∈ D faire
    pour chaque brin b' ∈ ⟨β1, β2⟩(b) faire
        si la face incidente à b' n'est pas élémentaire alors
            éclater la face désignée par b';
saturer le volume incident aux brins de D avec des faces élémentaires;
fermer les volumes élémentaires dans le volume incident aux brins de D;
```

Pour réaliser l'éclatement d'un volume en volumes élémentaires, nous commençons par éclater, avec l'Algorithme 20, les faces appartenant au bord du volume afin de respecter la précondition de l'algorithme de saturation d'un volume par des faces élémentaires. Chaque brin $b$ de $D$ désigne une surface formant un bord du volume. Nous parcourons tous les brins de la surface désignée par $b$ (il s'agit de l'orbite $\langle \beta_1, \beta_2 \rangle(b)$) et pour chacun de ces brins, nous éclatons la face incidente si nécessaire. Ainsi toutes les faces appartenant au bord du volume sont éclatées en faces élémentaires.

Nous utilisons ensuite l'Algorithme 22 pour obtenir un volume saturé par des faces élémentaires. Enfin nous appliquons l'Algorithme 23 de fermeture des volumes élémentaires pour terminer le traitement et obtenir l'ensemble des volumes élémentaires remplaçant le volume initial.

La Figure 5.16 présente un exemple d'éclatement d'un volume dans une carte topologique 3D : nous prenons le même exemple que celui illustrant l'éclatement de faces Figure 5.11.



(a) (b)

FIG. 5.16: Exemple d'éclatement d'un volume en volumes élémentaires. (a) Le bord du volume est éclaté en faces élémentaires. (b) Résultat de l'algorithme d'éclatement de volume, deux faces ont été insérées à l'intérieur du volume et reliées au bord du volume initial pour obtenir les trois volumes élémentaires issus de l'éclatement.

## 5.4 Éclatement d'une région

L'éclatement de régions consiste à éclater le volume représentant la région en se servant de l'opération d'éclatement en volume élémentaire, puis à construire et associer une nouvelle région à chaque nouveau volume. L'arbre d'imbrication est alors mis à jour pour refléter les changements d'imbrication consécutifs à l'éclatement de la région en régions élémentaires. Enfin la carte combinatoire est simplifiée pour obtenir la carte topologique représentant la nouvelle partition. Les bords des régions éclatés ne sont plus nécessairement minimaux à cause des opérations d'éclatement.

L'Algorithme 25 présente les différentes opérations réalisées pour éclater une région en régions élémentaires. Il prend comme paramètres une carte topologique et une région à éclater. L'algorithme modifie la carte topologique pour représenter l'éclatement de la région désignée en régions élémentaires.

**Algorithme 25** : Éclatement d'une région en régions élémentaires

**Données** : Une carte topologique $M$ ;
Une région $r$.
**Résultat** : Éclatement dans $M$ de la région $r$ en régions élémentaires.

$D \leftarrow$ ensemble de brins vide;
**pour chaque** *surface $S$ du bord de $r$* **faire**
  ajouter l'un des brins de $S$ à $D$ ;
éclater le volume incident aux brins de $D$ en volumes élémentaires;
construire une nouvelle région par volume et la stocker dans $R$;
mettre à jour l'arbre d'imbrication;
simplifier les bords des régions éclatées;

Dans un premier temps, l'algorithme construit un ensemble de brins $D$ où chaque brin désigne l'une des surfaces appartenant aux bords de $D$. Nous utilisons alors l'opération d'éclatement de volumes définie par l'Algorithme 24 afin d'éclater le volume de la région en volumes élémentaires.

Nous associons ensuite une nouvelle région pour chaque nouveau volume créé lors de l'éclatement de volumes. Nous utilisons pour cela un parcours sur les brins de la carte qui permet depuis son brin représentant de retrouver tous les volumes associés à la région $r$. Lorsque nous découvrons un nouveau volume, nous lui associons une nouvelle région en définissant les données images à l'aide du voxel contenu dans le volume. Nous ajoutons ensuite la région à l'ensemble $R$.

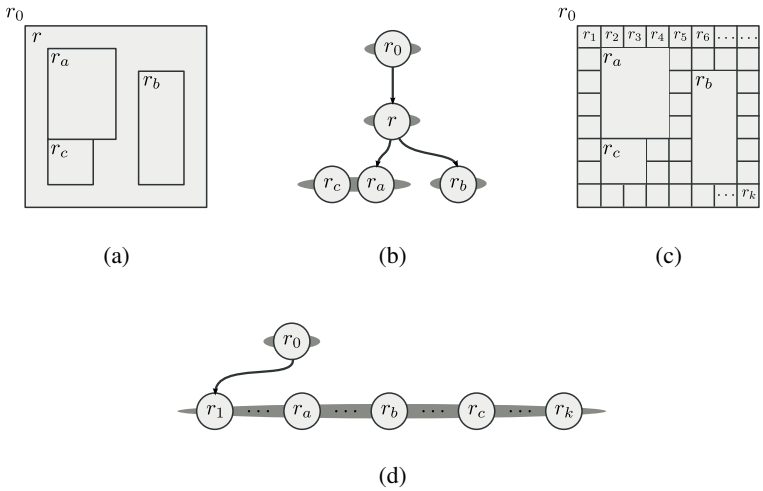



FIG. 5.17: Mise à jour de l'arbre d'imbrication des régions dans le cas de l'éclatement d'une région. (a) Partition initiale de l'image. (b) Arbre d'imbrication initial. (c) Résultat de l'éclatement de la région $r$ dans la partition. (d) Résultat de l'éclatement de la région $r$ dans l'arbre d'imbrication.

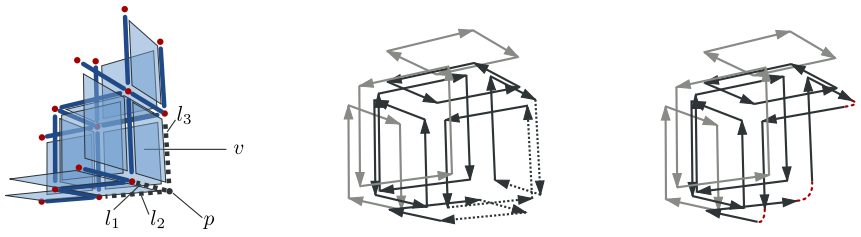
La mise à jour de l'arbre d'imbrication est réalisée en trois temps. D'une part, nous ajoutons toutes les régions imbriquées dans $r$ dans la même composante connexe que $r$. En effet, les régions imbriquées dans $r$ avant l'opération n'ont plus de relation d'imbrication avec $r$ après l'éclatement, mais sont maintenant imbriquées dans la région englobant $r$. Ensuite nous ajoutons toutes les régions nouvellement créées (stockées dans l'ensemble $R$) dans la même composante connexe que $r$. Les régions ont la même relation d'imbrication que $r$. Enfin nous supprimons $r$ de l'arbre d'imbrication puisque la région n'existe plus. La Figure 5.17 illustre en 2D le principe de la mise à jour de l'arbre d'imbrication. Dans la Figure 5.17a, se trouve une région $r$ imbriquant trois autres régions appelées $r_a$, $r_b$ et $r_c$. $r_a$ et $r_c$ font partie d'une seule cavité de la région $r$. L'arbre d'imbrication, Figure 5.17b, représente toutes ces informations. Dans la Figure 5.17c, la région $r$ est éclatée en régions élémentaires notées $r_i$ avec $i \in [1..n]$. Nous observons que les régions $r_a$, $r_b$ et $r_c$ sont maintenant au même niveau d'imbrication que les régions $r_i$ et que la région $r$. Cette configuration est représentée en ajoutant toutes ces régions dans la même composante connexe que $r$ dans l'arbre d'imbrication des régions (Figure 5.17d).

La dernière étape de l'algorithme simplifie la carte combinatoire pour obtenir la représentation minimale de la carte combinatoire. Parmi les nouvelles cellules créées lors des opérations d'éclatement, il existe des cellules qui ne sont pas minimales d'après la définition des cartes topologiques. C'est uniquement le cas au bord de la région éclatée. La Figure 5.18 illustre une configuration dans laquelle le volume dans le coin d'une région est incident à trois arêtes et un sommet non minimaux. Nous utilisons l'algorithme défini pour l'extraction d'une image afin de simplifier la carte topologique.

## 5.5 Division d'une région par un guide

L'éclatement d'une région construit une partition généralement sursegmentée de la région initiale. Une partition aussi détaillée n'est pas souhaitable dans la plupart des applications. D'une part le temps de calcul de cette opération est long, et d'autre part l'espace mémoire nécessaire pour représenter la partition éclatée est très grand et peut poser des problèmes dans les applications.

Aussi nous définissons une opération de division d'une région qui est plus adaptée à nos besoins. C'est l'opération de division de régions par un guide.




(a) (b) (c)

FIG. 5.18: Exemple pour la simplification de la carte topologique après l'éclatement d'une région. (a) Éléments intervoxels illustrant la situation. Le volume $v$ résulte de l'éclatement d'une région en régions élémentaires. Les lignels $l_1$, $l_2$, $l_3$ et le pointel $p$ sont les plongements de cellules non minimales à simplifier par la suite. (b) Carte représentant le volume $v$ éclaté (le volume est représenté par les brins en noir), les brins gris sont les brins des volumes adjacents et les brins en pointillés sont les brins appartenant à des cellules non minimales. (c) Carte obtenue après l'étape de simplification.

Classiquement dans les opérations de division, le guide servant à découper la région est une droite en 2D ou un plan en 3D. Ce type de guide divise alors une région en plusieurs morceaux (deux dans un cas simple). Dans le cadre des cartes topologiques, nous généralisons la notion de guide par l'utilisation d'un ensemble de faces pouvant s'intersecter et permettant de diviser la région en plusieurs régions résultantes. Ce type de guide permet alors de réaliser n'importe quelle division souhaitée par l'utilisateur. La Définition 21 spécifie la contrainte à respecter pour qu'un ensemble de surfels soit un guide de division valide.

**Définition 21** (Guide)**.** *Un ensemble de surfels $G$ définit un guide de division pour une région $r$ dans une carte topologique $M$ si et seulement si chaque surfel appartenant à $G$ sépare deux voxels de $r$ et est adjacent à au moins quatre surfels appartenant à $G \cup \text{surfels}(r)$ et que l'ensemble des surfels est séparant pour la région $r$.*

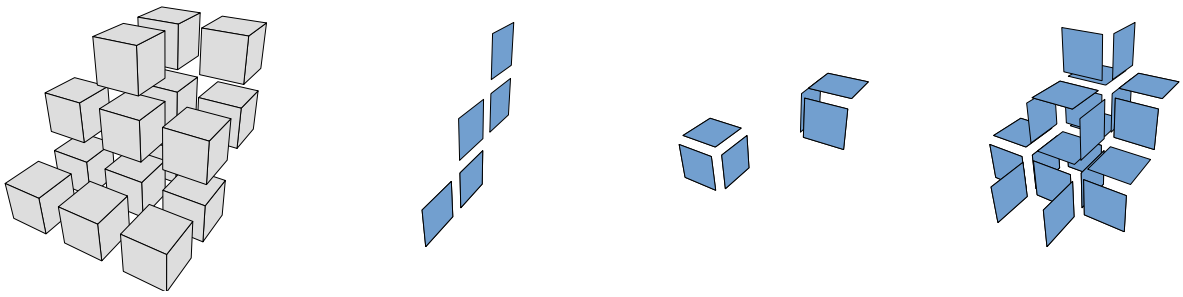
Cette définition permet de garantir que les faces produites par la division ne contiennent aucun trou. Nous assurons ainsi que la carte topologique représentant la partition soit bien formée.

Dans la suite, nous détaillons comment le guide est ajouté à la carte topologique puis nous expliquons comment nous relions le guide aux bords de la région et enfin nous donnons l'algorithme général permettant de réaliser la division d'une région par un guide.

### 5.5.1 Construction du guide de division

Le guide utilisé par la division de régions est un ensemble de surfels respectant la Définition 21. Cette définition assure que les surfels, une fois connectés au bord de la région, forment des faces n'ayant pas de trou et séparent la région initiale en au moins deux composantes distinctes.

La Figure 5.19 illustre la notion de guide pour un volume. Le volume d'exemple est représenté par ses voxels dans la Figure 5.19a. Chaque surfel se trouvant entre deux voxels du volume peut faire partie de l'ensemble définissant le guide. Nous avons ainsi trois exemples de guides. La Figure 5.19b représente un plan divisant la région en deux. La Figure 5.19c divise la région en trois composantes connexes, chaque ensemble de surfels connexes sépare un voxel du reste de la région. La Figure 5.19d illustre le guide maximal contenant tous les surfels intérieurs à la région : ce guide réalise l'éclatement de la région en régions élémentaires.



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

FIG. 5.19: Exemples de guides pour un volume. (a) Voxels du volume divisé. (b) Surface divisant le volume en deux. (c) Deux surfaces divisant le volume en trois. (d) Guide maximal éclatant le volume.

L'opération de construction du guide consiste à ajouter le guide dans la carte topologique. L'Algorithme 26 prend en paramètres une carte topologique et un ensemble de surfels $G$. Il intègre le guide dans le modèle et retourne l'ensemble des brins servant à représenter le guide.

**Algorithme 26** : Ajout d'un guide dans une carte topologique

**Données** : Une carte topologique $M$ ;
Un ensemble de surfels $G$.
**Résultat** : Insertion du guide décrit par $G$ dans la carte topologique $M$ et renvoie l'ensemble des brins ajoutés.

**pour chaque** *surfel* $s \in G$ **faire**
  créer 8 brins formant une face de plongement $s$;
  allumer le surfel $s$;
  allumer les 4 lignels incidents à $s$;
  allumer les 4 pointels incidents à $s$;
coudre les faces élémentaires du guide entre elles;
**retourner** *l'ensemble des brins que nous avons ajoutés*

Nous commençons par ajouter à la carte topologique les brins correspondant aux faces élémentaires décrites par les surfels de l'ensemble $G$. Pour cela nous ajoutons pour chaque surfel $s \in G$, huit brins, que nous relions selon le principe de la Figure 5.12, afin qu'ils représentent la face de plongement $s$. Nous allumons dans la matrice intervoxel le plongement de chacune des cellules élémentaires que nous avons ajoutées.

L'étape suivante consiste à coudre ensemble les différentes faces du guide autour des arêtes afin de créer les surfaces représentées par $G$. Cette opération utilise le même principe que l'opération de fermeture des volumes élémentaires dans l'éclatement de volumes (voir Algorithme 23). Il s'agit d'associer à chaque lignel l'ensemble des brins appartenant aux faces incidentes puis de coudre entre elles toutes les faces autour d'une arête.

Nous retournons alors l'ensemble des brins que nous avons ajoutés : ils décrivent maintenant le guide. Dans la suite nous pourrons également utiliser cet ensemble de brins pour décrire le guide en plus de l'ensemble de surfels.

### 5.5.2 Couture du guide avec les bords de la région

L'opération suivante consiste à coudre le guide aux bords de la région afin de diviser le volume de la région. Dans cette opération nous ne construisons pas de nouvelles régions et nous ne modifions pas l'arbre d'imbrication.

Cette opération est très similaire à la fermeture de volume donnée par l'Algorithme 23. La seule différence est qu'il faut insérer, dans la structure associant un pointel et un lignel à un brin, les brins de la région $r$ et les brins du guide. Le reste de l'algorithme est identique : les brins autour d'un même lignel sont cousus entre eux pour former les arêtes ce qui va progressivement intégrer le guide dans la région.

### 5.5.3 Construction de l'arbre d'imbrication des régions

L'opération de construction de l'arbre d'imbrication sert ici à mettre à jour l'arbre des régions pour ajouter les régions issues de la division, supprimer la région divisée et corriger les relations d'imbrication entre les régions.

L'Algorithme 27 présente le principe de cette opération. Il prend en paramètres une carte topologique, une région et un ensemble de brins contenant un brin par surface des volumes issus de la division. L'algorithme associe les brins de ces volumes à de nouvelles régions et modifie l'arbre d'imbrication pour refléter la nouvelle configuration de la partition.

Dans un premier temps nous attribuons correctement les brins de chaque volume à une nouvelle région. Pour cela nous traitons chacun des brins $b$ représentant les surfaces de manière itérative. Nous attribuons à l'ensemble des brins $b'$ de la surface désignée par $b$ une nouvelle région $r'$. Nous parcourons tous les voxels de cette région de manière à les marquer. Nous cherchons ensuite les surface internes à la région. Nous traitons donc tous les brins $b''$ désignant les surfaces. Si le voxel est à l'intérieur de la surface incidente à $b''$, alors tous les brins de l'orbite $\langle \beta_1, \beta_2 \rangle (b'')$ forment une surface interne de $r'$. Nous mettons de côté le brin $b''$ dans $S'$ afin de construire ensuite les relations d'imbrication.

Dans un deuxième temps, nous reconstruisons la relation d'imbrication. Pour cela, nous avons l'ensemble $S'$ qui contient un brin $b$ par surface interne d'une région. Nous parcourons la composante connexe de brins incidente à $b$ et pour chaque nouvelle région découverte, nous l'ajoutons à une même composante connexe imbriquée dans la région de $b$.

Le dernier temps consiste à ajouter les régions qui n'ont pas de relation de parenté comme fille de la région imbriquant $r$, puis à supprimer la région $r$ de l'arbre d'imbrication car plus aucun brin ne pointe dessus. La complexité de l'algorithme est linéaire suivant le nombre de brins issus de la division de la région $r$.

### 5.5.4 Algorithme de division d'une région par un guide

L'Algorithme 28 présente l'opération de division d'une région par un guide. Il prend en paramètres une carte topologique, une région et un ensemble de surfels qui est un guide de division pour la région sélectionnée.

La première étape consiste à éclater, à l'aide de l'Algorithme 20, le bord de la région à diviser en faces élémentaires.

L'étape suivante construit le guide de division dans la carte topologique. Nous réalisons cette opération avec l'Algorithme 26. Nous disposons alors d'un ensemble de brins représentant le guide. Chaque arête de ce guide est élémentaire. Les arêtes de ce guide qui sont des bords pendants doivent être reliées avec les bords de la région $b$.

L'étape de fusion du guide est réalisée par l'Algorithme 23 modifié comme le précise la Section 5.5.2. Il s'agit de coudre le guide et les bords de la région. Le résultat de cette opération est le

**Algorithme 27** : Construction de l'arbre d'imbrication des régions

**Données** : Une carte topologique $M$ ;
Une région $r$ ;
Un ensemble de brins $S$.

**Résultat** : Mise à jour de l'arbre d'imbrication de $M$ pour représenter la partition issue de la division de la région $r$ en volumes représentés par $S$.

$S' \leftarrow$ ensemble de brins vide;
**pour chaque** *brin* $b \in S$ **faire**
  **si** region$(b) = r$ **alors**
    retirer $b$ de $S$;
    $r' \leftarrow$ nouvelle région;
    **pour chaque** *brin* $b' \in \langle \beta_1, \beta_2 \rangle(b)$ **faire**
      *region*$(b') \leftarrow r'$;
    marquer les voxels imbriqué dans le volume désigné par $b$;
    **pour chaque** *brin* $b'' \in S$ **faire**
      **si** voxel$_1$(triplet$(b'')$) *est marqué* **alors**
        `// b'' appartient à une surface interne de r'`
        retirer $b''$ de $S$;
        ajouter $b''$ à $S'$;
        **pour chaque** *brin* $b_i \in \langle \beta_1, \beta_2 \rangle(b'')$ **faire**
          *region*$(b_i) \leftarrow r'$;
**pour chaque** *brin* $b \in S'$ **faire**
  $r_{pere} \leftarrow$ *region*$(b)$;
  **pour chaque** *brin* $b' \in \langle \beta_1, \beta_2, \beta_3 \rangle(b)$ **faire**
    $r' \leftarrow$ *region*$(b')$;
    **si** $r' \neq r_{pere}$ *et si* pere$(r') = nil$ **alors**
      *pere*$(r') \leftarrow r_{pere}$;
ajouter les régions qui n'ont pas encore de père à la place de la région $r$ dans l'arbre d'imbrication;
supprimer $r$ de l'arbre d'imbrication;

**Algorithme 28** : Division d'une région par un guide

**Données** : Une carte topologique $M$ ;
Une région $r$ ;
Un guide $G$.

**Résultat** : Division dans $M$ de la région $r$ par le guide $G$.

éclater les arêtes et les faces du bord de la région $r$;
$D \leftarrow$ ensemble de brins vide;
**pour chaque** *surface* $S$ *du bord de* $r$ **faire**
  ajouter à $D$ l'un des brins de $S$;
construire le guide de division défini par $G$;
coudre le guide aux bords de la région $r$;
construire l'arbre d'imbrication des nouvelles régions;
simplifier la carte topologique;

remplacement du volume associé à la région $r$ par l'ensemble des volumes décrits par le guide et les bords de la région.

Nous procédons ensuite à la mise à jour de l'arbre d'imbrication. Cette opération nécessite d'utiliser la géométrie afin de retrouver les relations d'imbrication entre les régions existantes et les nouvelles régions issues de la division.

Enfin une étape de simplification est nécessaire puisque le guide est construit à l'aide de faces élémentaires et puisque le bord de la région est éclaté en faces élémentaires. Nous retirons donc les arêtes non minimales afin d'obtenir la carte topologique représentant la nouvelle partition.

## 5.6 Comparaisons

Afin d'analyser les performances des deux approches de la division d'une région nous avons mis en place différentes expériences. À l'aide de la fusion de régions nous pouvons obtenir le même résultat en utilisant la division et l'éclatement de régions. Nous mettons ainsi en évidence les avantages et inconvénients des deux méthodes. La division de régions permet également de réaliser l'opération d'extraction de la carte topologique à partir d'une image. Nous proposons des variantes de l'algorithme d'extraction basées sur l'éclatement et la division d'une région initiale maximale afin de comparer les performances de nos algorithmes avec celles de l'algorithme d'extraction.

### 5.6.1 Éclatement par division maximale

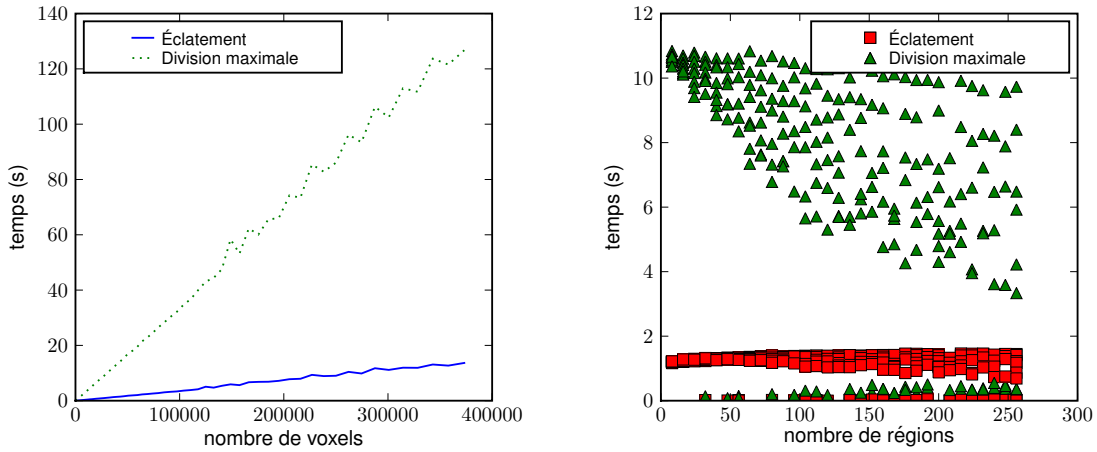
L'éclatement d'une région consiste à diviser la région de manière à obtenir un ensemble de régions de sorte que chaque voxel de la région initiale soit dans une région différente. Deux approches sont possibles pour obtenir ce résultat. La première approche est l'utilisation de l'opération d'éclatement qui effectue cette opération directement. La deuxième consiste à utiliser l'opération de division en utilisant un guide maximal, qui comporte tous les surfels possibles.

Afin de comparer les deux approches de l'éclatement d'une région, nous effectuons deux expériences. Dans la première, Figure 5.20a, nous éclatons la région contenant toute l'image en faisant varier la taille de l'image. Nous observons le comportement linéaire par rapport à la taille de l'image. L'approche utilisant l'éclatement est plus rapide que l'approche par division de régions. Dans la seconde expérience, Figure 5.20b, nous utilisons une partition faiblement segmentée initialisée aléatoirement. L'image comporte ainsi un nombre fixe de régions, leurs positions relatives étant très variables. Dans ces images, nous éclatons l'une des régions choisie au hasard : le temps d'exécution dépend de la taille de la région divisée. Les points de mesures montrent que l'approche utilisant l'éclatement est généralement plus rapide (sauf dans le cas des toutes petites régions). L'approche utilisant la division maximale est plus lente. Lorsque le nombre de régions augmente, la probabilité que la région à éclater soit de petite taille augmente. Ainsi nous observons une variation plus importante des temps lorsque le nombre de régions augmente. Les mesures très faibles correspondent à des configurations où la région à diviser est très petite.

### 5.6.2 Division par éclatement-fusion

Dans de nombreuses applications de la division de régions nous ne souhaitons pas éclater complètement une région en voxels mais simplement obtenir quelques régions à partir de la division initiale.

Pour obtenir le même résultat avec l'opération d'éclatement qu'avec l'opération de division par guide, nous utilisons successivement l'éclatement de régions puis l'opération de fusion de régions afin d'obtenir la partition souhaitée. Nous utilisons dans ce cas l'approche globale de la fusion de régions avec un oracle qui permet d'obtenir la partition finale. Afin de construire le guide de division, nous utilisons le même oracle pour déterminer les surfels appartenant au guide.




(a) (b)

FIG. 5.20: Comparaison de l'éclatement et de la division par un guide maximal : (a) en fonction du nombre total de voxels de la région dans une partition ne comportant qu'une seule région. (b) en fonction du nombre de voxels de la région dans des partitions aléatoires.

La Figure 5.21 présente une comparaison des temps d'exécution des deux opérations en fonction du nombre de régions de la partition résultante. Nous faisons varier le nombre de régions résultantes en utilisant une partition aléatoire générée à partir d'un nombre fixé de graines. En augmentant le nombre de graines, nous augmentons le nombre de régions. Nous observons que le temps d'exécution de l'opération d'éclatement suivie d'une fusion globale reste quasi constant car la fusion globale dépend peu du nombre de régions fusionnées. Lorsque le nombre de régions à obtenir par la division est faible, l'opération de division par guide est très performante. Par contre si le nombre de régions à produire augmente de façon importante alors la division par guide est moins performante.

### 5.6.3 Extraction par division de l'image

L'extraction d'une image utilise un balayage complet de l'image. Pour chaque voxel, un volume est créé dans la carte topologique. Puis les éléments qui ne respectent pas la définition de minimalité des cartes topologiques sont ensuite supprimés. Cette approche montre son efficacité dans les applications pratiques puisque c'est la méthode utilisée pour générer une carte topologique à partir d'une image. Nous souhaitons comparer son temps de calcul avec les opérations de division et d'éclatement puisque leurs buts sont similaires.

Dans une première expérience, nous construisons une opération d'extraction de carte topologique basée sur l'opération de division. Son principe consiste à parcourir les voxels afin de construire un guide de division permettant d'obtenir la partition initiale de l'image identique à celle produite par l'opération d'extraction. Ensuite nous utilisons ce guide pour l'opération de division de la région contenant toute l'image.

La Figure 5.22 présente les résultats de la comparaison entre la construction d'une carte topologique par division et par extraction. Nous utilisons des images initialisées aléatoirement avec un nombre de régions déterminé. Dans les expériences, nous observons que l'opération d'extraction est toujours plus rapide même quand le nombre de régions reste faible. L'écart augmente lorsque le nombre de régions de la partition augmente. Les écarts de mesure pour une même méthode avec un même nombre de régions s'expliquent par la configuration des régions qui est différente.

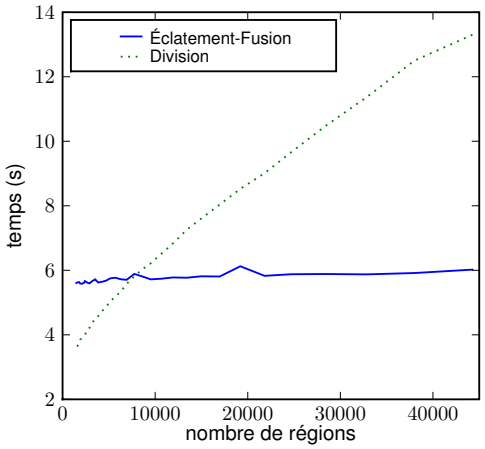



FIG. 5.21: Comparaison de la division avec l'éclatement suivie d'une fusion globale sur une image médicale en fonction de la taille de l'image.

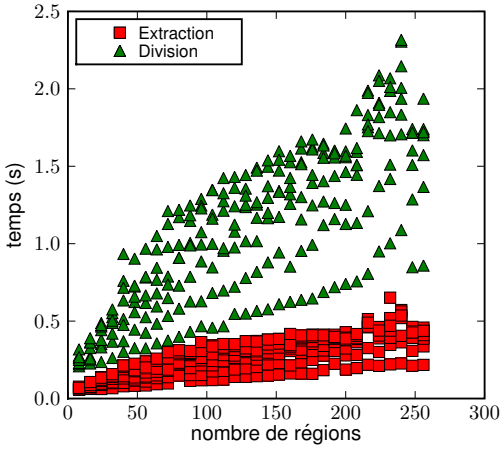



FIG. 5.22: Comparaison de l'extraction avec l'opération de division dans le cas d'images aléatoires simples.

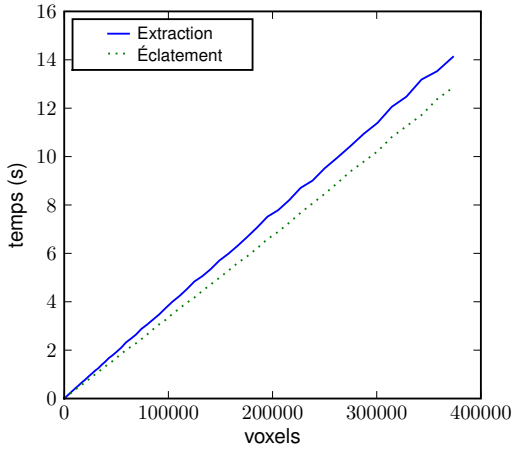



FIG. 5.23: Comparaison de l'extraction d'une image ayant pour chaque voxel une région différente avec l'opération d'éclatement de la région contenant toute l'image en fonction de la taille de l'image.

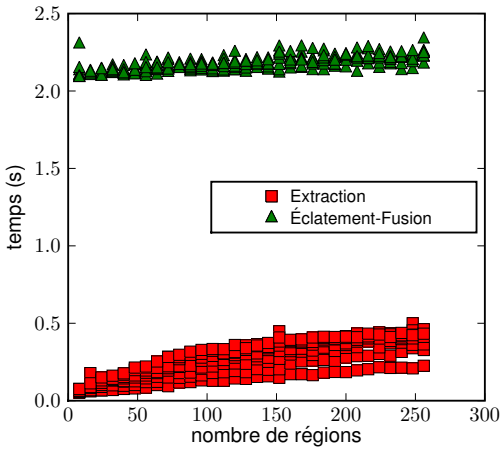



FIG. 5.24: Comparaison de l'extraction d'une image comportant plusieurs régions avec l'opération d'éclatement de la région suivie d'une fusion des régions ayant le même label.

Dans la seconde expérience, nous comparons le temps d'extraction d'une image dont la partition comporte une région par voxel entre l'algorithme d'extraction classique et l'algorithme d'éclatement de régions. Il s'agit du cas très favorable à l'éclatement de régions puisque l'extraction de l'image produit le même résultat que l'éclatement de la région maximale contenant toute l'image.

La Figure 5.23 montre que l'opération d'éclatement est plus performante dans cette configuration que l'opération d'extraction. En effet, l'opération d'extraction cherche à regrouper les voxels en composantes connexes de mêmes labels ce qui n'est jamais le cas dans la partition à obtenir. Nous remarquons que les deux approches sont linéaires par rapport à la taille de l'image.

Dans le cas plus général de l'extraction d'une carte topologique à partir d'une image, il est nécessaire de faire successivement une passe d'éclatement d'une région contenant tous les voxels de l'image, puis une fusion des régions ayant le même label. C'est l'objet de la troisième expérience. Nous mesurons les temps d'extraction sur des images aléatoires comportant un nombre fixé de régions et dont la configuration change.

La Figure 5.24 présente les temps d'exécution des deux approches en fonction du nombre de régions dans la partition résultante. Nous observons que le temps d'exécution de l'opération d'éclatement suivi de la fusion ne change pas en fonction de la complexité de la partition à obtenir. Pour l'opération d'extraction, le temps augmente mais faiblement lorsque le nombre de régions de la partition résultante augmente. D'après la seconde expérience et les mesures réalisées lors des expériences sur la fusion globale, nous observons que la différence de temps de traitement entre les deux approches est à peu près égale au temps pour réaliser la fusion par approche globale. Contrairement à l'extraction par division, le temps d'exécution de la méthode par éclatement-fusion globale ne dépend presque pas de la configuration des régions dans l'image.

## 5.7 Conclusion

Dans ce chapitre, nous avons présenté l'opération d'éclatement d'une région. Cette opération remplace une région par un ensemble de régions de sorte que chaque voxel de la région originale soit dans un voxel de la région résultat. Nous avons également défini l'opération de division d'une région à l'aide d'un guide. Les voxels de la région initiale sont répartis dans différentes régions en fonction des volumes définis par l'union du bord de la région avec le guide. Nous avons montré que l'opération

d'éclatement est beaucoup plus rapide que l'opération de division par un guide mais elle conduit à la création de très nombreuses régions ce qui peut poser des problèmes pour la représentation des données en mémoire. Dans le Chapitre 6, nous nous intéressons à une opération permettant de modifier la géométrie de la partition sans changer sa topologie : l'opération de déformation. Nous dérivons pour cela la notion classique de points simples pour l'utiliser dans le cadre d'images comportant plusieurs régions.

CHAPITRE 6

# DÉFORMATION DE PARTITIONS

La fusion et la division de régions sont deux opérations coûteuses en temps d'exécution, aussi nous souhaitons minimiser l'utilisation de ces opérations. De plus, les deux opérations précédentes permettent de modifier la topologie de la partition, or nous ne souhaitons pas toujours la modifier. Pour répondre à ces deux points, nous introduisons une nouvelle opération qui ne permet pas de modifier la topologie de la partition mais qui permet uniquement d'en changer la géométrie. Avec cette condition, la carte combinatoire et l'arbre d'imbrication ne changent pas. Aussi l'opération de déformation n'agit que dans le plongement dans la matrice intervoxel.

Nous proposons une approche générale, qui ne soit pas liée à la définition d'une carte topologique, pour déformer une partition comportant plusieurs régions sans en modifier la topologie. Dans le cadre des partitions binaires composées d'un objet et du fond, les outils principaux pour réaliser ces opérations sont basés sur la notion de points simples [Ros70]. À notre connaissance, il n'existe pas de définition de points simples préservant la topologie de chaque région et de la partition dans le cadre des images multirégions. Par exemple, les travaux de [CBG99, Coi99] présentés dans la Section 3.4.3 ne préservent pas les relations entre les régions mais seulement la topologie de chaque région. L'objectif de notre opération de déformation est de pouvoir basculer un voxel $v$ appartenant à une région $R$ dans une région adjacente sans changer la topologie des régions, ni les relations entre les faces appartenant aux bords des régions.

Nous proposons ici une nouvelle variante de points simples que nous appelons points simples multilabels dans des images comportant plusieurs régions. L'idée principale consiste à préserver le degré des lignels incidents au voxel à basculer de manière à éviter la suppression ou la création d'intersections de surfaces. Nous définissons de manière générale les points simples multilabels, puis nous précisons la manière dont l'algorithme de détection est mis en œuvre dans le cadre des cartes topologiques 3D. Nous définissons quelques énergies que nous utilisons dans le processus de déformation avec notamment une énergie basée sur une estimation discrète de l'aire des surfaces. Nous présentons finalement des expérimentations mettant en œuvre l'estimateur discret de l'aire en le comparant avec l'estimateur naïf qui utilise le nombre de surfels.

Nous nous intéressons ensuite à des méthodes permettant de faire évoluer la partition d'une image sans en modifier la topologie. Par exemple, après une opération de segmentation, nous obtenons une partition initiale ayant la même topologie que le résultat souhaité. Cependant, il est fréquent que la géométrie de la partition obtenue ne corresponde pas exactement aux données de l'image. En utilisant l'opération de déformation qui permet de faire évoluer géométriquement la partition sans en modifier la topologie, nous pouvons obtenir le résultat souhaité. Les énergies contrôlant la déformation sont choisies pour modifier la partition initiale afin qu'elle soit adaptée aux données de l'image. Nous présen-

tons le résultat de plusieurs opérations de déformation de partition qui préservent la topologie dans un contexte monorégion et multirégions en mettant en œuvre des algorithmes de la littérature.

Nous utilisons notre méthode au sein de différentes expérimentations de déformations. D'abord une approche classique de déformation d'images en n'utilisant que deux régions, l'objet et le fond. Ensuite nous déformons une partition contenant de multiples régions imbriquées les unes dans les autres. Ces deux premières approches sont des résultats déjà obtenus dans d'autres travaux en utilisant des modèles spécifiques. Nous montrons ainsi que la déformation d'une partition représentée par une carte topologique permet de réaliser le même type de traitement. Nous traitons enfin le cas plus général d'une partition contenant de multiples régions sans contraintes particulières : nous appelons cette opération le raffinement de la partition.

## 6.1 Points simples multilabels

Nous commençons par donner un bref rappel sur les points simples. Nous présentons ensuite la définition des points simples multilabels et nous donnons les éléments permettant de dire que le basculement d'un point ML-Simple préserve la topologie de la partition résultante. Pour finir nous présentons l'algorithme réalisant la détection des points ML-Simples.

### 6.1.1 Rappels sur les points simples

Nous rappelons à présent les notions permettant d'introduire la définition des points simples comme proposée dans [BM94].

L'ensemble des composantes $\alpha$-connectées d'un ensemble de voxels $R$ est appelé $C_\alpha(R)$. Le voisinage géodésique d'ordre $k$ d'un voxel $v$ dans $R$ est l'ensemble $N_\alpha^k(v,R)$ défini récursivement par : $N_\alpha^1(v,R) = N_\alpha^*(v,R) \cap R$ et $N_\alpha^k(v,R) = \bigcup\{N_\alpha(Y) \cap N_{26}^*(v) \cap R, Y \in N_\alpha^{k-1}(v,R)\}$. En d'autres termes, $N_\alpha^k(v,R)$ est constitué de l'ensemble des voxels $x$ de sorte que $x$ appartienne à $N_{26}^*(v) \cap R$ et tel qu'il existe un $\alpha$-chemin $\pi$ allant de $v$ à $x$ de longueur au plus $k$ avec tous les voxels de $\pi$ qui appartiennent à $N_{26}^*(v) \cap R$.

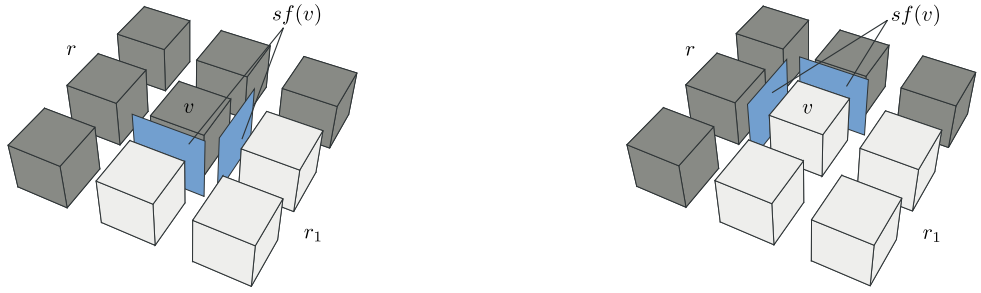
Dans nos travaux, nous utilisons la même notion de connexité que dans la carte topologique, ainsi nous utilisons le voisinage $(6,18)$, 6 étant pour l'objet et 18 pour le complémentaire. Dans ce cadre nous avons le 6-voisinage géodésique $G_6(v,R) = N_6^3(v,R)$ et le 18-voisinage géodésique $G_{18}(v,R) = N_{18}^2(v,R)$. À partir de ces notations, la Définition 22 explicite la notion de point simple dans les images binaires en considérant la connexité objet-fond en (6,18).

**Définition 22** (Caractérisation des Points Simples [BM94])**.** *Un voxel $v$ est simple pour une région $R$ si $\#C_6\left[G_6(v,R)\right] = \#C_{18}\left[G_{18}(v,\bar{R})\right] = 1$, où $\#C_k\left[Y\right]$ est le nombre de composantes $k$ connexes d'un ensemble $Y$.*

Un voxel d'une image binaire est définit comme simple si il peut être ajouté à l'objet ou supprimé de l'objet sans changer la topologie de l'objet et du fond, c'est-à-dire sans changer le nombre de composantes connexes, de cavités et de tunnels de l'objet $R$ et du fond $\bar{R}$. La préservation de la topologie de $R$ et $\bar{R}$ entraîne également, dans le cas binaire, la préservation de la topologie de la partition.

### 6.1.2 Points simples multilabels

Les points simples multilabels sont définis à l'aide des éléments intervoxels. Pour cette définition, nous avons besoin de quelques notations. Ainsi, nous appelons *surfels*$(v)$ l'ensemble des six surfels incidents à un voxel $v$. Pour un surfel $s$, nous appelons *lignels*$(s)$ les quatre lignels se trouvant entre $s$ et ses quatre surfels adjacents. Pour un lignel $l$, nous appelons *pointels*$(l)$ les deux pointels incidents à




(a) (b)

FIG. 6.1: Exemple d'un voxel ML-Simple et basculement du voxel. (a) Configuration initiale, le voxel $v$ appartenant à la région $r$ est ML-Simple. (b) Basculement de $v$ dans la région $r_1$. $v$ est toujours ML-Simple, nous pouvons le rebasculer dans la région $r$.

$l$. Nous utilisons également la notation *lignels*$(v)$ pour désigner les douze lignels incidents à un voxel $v$. Nous appelons $SF$ l'ensemble des surfels frontières d'une image, c'est-à-dire l'ensemble des surfels séparant deux voxels de labels différents. Il s'agit de l'ensemble des surfels allumés dans la matrice intervoxel. Étant donné un voxel $v$, *sf*$(v)$ $=$ *surfels*$(v) \cap SF$ est l'ensemble des surfels frontières incidents au voxel $v$ (voir les surfels désignés dans la Figure 6.1a).

Nous appelons *d*$(l)$ le degré du lignel $l$, c'est-à-dire le nombre de surfels frontières incidents à $l$. Nous remarquons que *d*$(l)$ prend les valeurs 0, 2, 3 ou 4 mais jamais 1, il n'y a pas de surfel frontière pendant. Nous notons *d*$(l, v)$ le degré du lignel $l$ restreint aux surfels frontières (*sf*$(v)$) incidents à $v$.

La Définition 23 introduit la notion de points simples multilabel (appelés aussi points ML-Simples) qui sont des points préservant à la fois la topologie des régions et les relations entre les surfaces.

**Définition 23** (Points ML-Simples)**.** *Un voxel $v$ est ML-Simple si :*

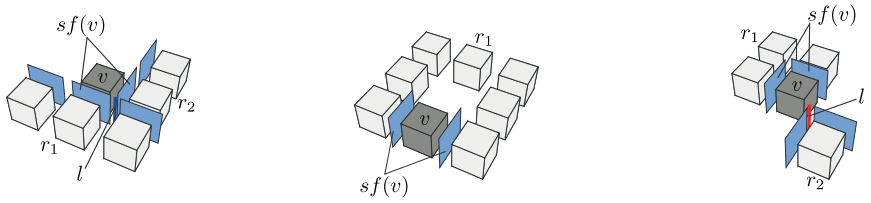
1. $\forall l \in \text{lignels}(v), \text{d}(l) \in \{0, 2\}$ *;*
2. $\text{sf}(v)$ *est homéomorphe à un 2-disque ;*
3. $\forall l \in \text{lignels}(v), \text{d}(l, v) = 0 \Rightarrow \text{d}(l) = 0$.

Intuitivement, les trois conditions de la Définition 23 permettent :

1. d'éviter les cas où plusieurs régions se trouvent autour du voxel $v$ : cette condition évite les lignels de degré supérieur à 2 ce qui est le cas lorsque plus de deux régions touchent le lignel $l$ (voir exemple Figure 6.2a) ;
2. de préserver la topologie des surfaces : si l'ensemble des surfels incidents à $v$ et séparant deux voxels ayant un label différent n'est pas homéomorphe à un disque, la suppression du voxel $v$ de sa région induit une modification topologique sur la surface (voir exemple Figure 6.2b) ;
3. de préserver les relations entre surfaces : si un lignel $l$ est tel que $d(l, v) = 0$ et $d(l) > 0$, la suppression du voxel $v$ de sa région force la surface à toucher une autre surface. Ceci crée un nouveau contact entre deux surfaces qui n'étaient précédemment pas adjacentes (voir exemple Figure 6.2c).

### 6.1.3 Les points simples multilabels sont des points simples

Afin de s'assurer que la topologie de chaque région est préservée lorsque un point simple multilabels est basculé, nous montrons que les points simples multilabels sont des points simples. Pour cela, nous commençons par montrer que les points ML-Simples ne se trouvent que dans un voisinage binaire.




(a) (b) (c)

FIG. 6.2: Exemple pour chaque condition de la Définition 23 d'une configuration rejetée. Dans chaque figure, $v$ appartient à une région $r$ et nous ne dessinons que les voxels appartenant à une autre région que $r$. (a) Le lignel $l$ est de degré 3. La condition 1 n'est pas vérifiée : nous ne basculons pas le voxel $v$ dans $r_1$, ni dans $r_2$. (b) *sf*$(v)$ ne contient que deux surfels qui ne sont pas adjacents : *sf*$(v)$ n'est pas homéomorphe à un disque. La condition 2 n'est pas vérifiée, nous ne basculons pas le voxel $v$ dans $r_1$. (c) $d(l) = 2$ mais $d(l, v) = 0$ : il n'y a pas de surfels de *sf*$(v)$ incident à $l$. La condition 3 n'est pas vérifiée, nous ne basculons pas le voxel $v$ dans $r_1$.

**Lemme 2.** *Soit $v$ un point ML-Simple, le nombre de labels distincts dans $N_{18}(v)$ est deux.*

L'idée générale de la preuve est d'étudier le voisinage de $v$ et de montrer que tous les voxels dans $N_{18}(v)$ ont soit le même label que $v$ (nous appelons $w = label(v)$), soit un même label $b$ avec $b \neq w$. La preuve se fait par l'absurde, en supposant que ce n'est pas le cas et en montrant que chaque configuration possible contredit l'une des conditions de la Définition 23.

*Démonstration du Lemme 2.* Soit $v$ un point ML-Simple, le nombre de labels dans $N_{18}(v)$ est plus grand que 1, sinon l'ensemble *sf*$(v)$ est vide (il n'y a pas de surfel frontière incident à $v$), ce qui contredit la condition (2) de la Définition 23.

Nous notons $b = label(v)$. Les voxels de $N_{18}(v)$ peuvent être partitionnés en deux ensembles $v_6 = N_6(v)$ et $v_{18} = N_{18}(v) \setminus N_6(v)$. L'ensemble $v_6$ est lui même divisé en deux ensembles $bv_6 = \{x \in v_6 | label(x) = b\}$ and $wv_6 = \{x \in v_6 | label(x) \neq b\}$.

Nous prouvons dans un premier temps que tous les voxels de $wv_6$ ont le même label (appelé $w$). Supposons que deux voxels $x_1$ et $x_2$ appartenant à $wv_6$ ont des labels différents. Comme l'ensemble de surfels *sf*$(v)$ est homéomorphe à un disque, il est connecté et ainsi il existe un chemin de surfels entre chaque couple de surfels de *sf*$(v)$. En transférant cela sur les voxels, il existe une chemin de voxels appartenant à $wv_6$ entre chaque couple de voxels de $wv_6$ tel que chaque paire de voxels consécutifs appartenant au chemin soit 18-adjacente. En considérant un tel chemin entre $x_1$ et $x_2$, nous pouvons trouver deux voxels $x'_1$ et $x'_2$ de $wv_6$ qui sont 18-adjacents et qui ont des labels différents (sinon $x_1$ et $x_2$ ont le même label). Ainsi, le voxel 6-adjacent à $x'_1$ et $x'_2$ dans $v_6$ est soit séparé de $x'_1$, soit séparé de $x'_2$ (ou des deux). Ainsi, il existe au moins un surfel incident à ce voxel et à $v$. Comme il existe un surfel entre $x'_1$ et $v$ et un surfel entre $x'_2$ et $v$, le lignel incident à ces surfels est incident à $v$ et $d(l) \geq 3$, ce qui contredit la condition (1) de la Définition 23.

Chaque voxel $x \in v_{18}$ est 6-adjacent à deux voxels de $v_6$, appelés $n_1(x)$ et $n_2(x)$. Comme nous l'avons montré précédemment, le label de chaque voxel de $v_6$ est soit $b$, soit $w$. Ainsi nous pouvons partitionner $v_{18}$ en trois ensembles : $bv_{18} = \{x \in v_{18} | label(n_1(x)) = label(n_2(x)) = b\}$, $wv_{18} = \{x \in v_{18} | label(n_1(x)) = label(n_2(x)) = w\}$, $mv_{18} = \{x \in v_{18} | label(n_1(x)) \neq label(n_2(x))\}$.

Le label de chaque voxel $x$ appartenant à $bv_{18}$ est $b$. Sinon, il existe deux surfels séparant $x$ de ses deux 6-voisins dans $v_6$. Ainsi, le lignel $l$ entre ces deux surfels est tel que $d(l, v) = 0$ (parce qu'il n'y a pas de surfel incident à $l$ et $v$ puisque le label de $v$, $n_1(x)$ et $n_2(x)$ est $b$) et $d(l) = 2$ (à cause des deux surfels), ce qui contredit la condition (3) de la Définition 23.

Le label de chaque voxel $x$ appartenant à $wv_{18}$ est $w$. Sinon, il existe deux surfels séparant $x$ de ses deux 6-voisins $n_1(x)$ et $n_2(x)$. Ainsi le lignel $l$ entre ces deux surfels est tel que $d(l) = 4$ (ces deux surfels plus les deux surfels entre $n_1(x)$ et $v$ et entre $n_2(x)$ et $v$), ce qui contredit la condition (1) de la Définition 23.

Le label de chaque voxel $x$ appartenant à $mv_{18}$ est $b$ ou $w$. Sinon, il existe deux surfels séparant $x$ de ces deux 6-voisins $n_1(x)$ et $n_2(x)$. Ainsi, le lignel $l$ entre ces deux surfels est tel que $d(l) = 3$ (ces deux surfels plus un surfel séparant le voxel ayant de label $l$ de $v$), ce qui contredit la condition (1) de la Définition 23.

Ceci prouve que le label de chaque voxel appartenant à $N_{18}(v)$ est soit $b$, soit $w$ et ainsi que le nombre de labels dans $N_{18}(v)$ est deux. □

Il faut noter que le fait que $v$ soit un point ML-Simple n'implique pas que le nombre de labels distincts dans $N_{26}(v)$ soit égal à deux. Cependant, le lemme assure que pour chaque point ML-Simple, il n'y a qu'un label permettant de retirer le voxel à sa région d'appartenance.

Maintenant, nous montrons que les topologies de $r$ contenant $v$ et de son complémentaire $\bar{r}$ sont préservées en prouvant la proposition suivante qui relie les points ML-Simples aux points simples.

**Proposition 4.** *Si $v \in r$ est un point ML-Simple, alors $v$ est un point simple pour $r$.*

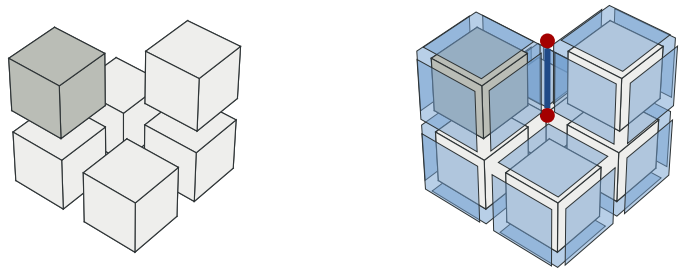
*Démonstration.* Nous prouvons la contraposée de la Proposition 4, c'est-à-dire si $v$ n'est pas un point simple pour $r$, alors $v$ n'est pas un point ML-Simple.

Nous définissons $n_1 = \#C_6\,[G_6(v, r)]$ et $n_2 = \#C_{18}\,[G_{18}(v, \bar{r})]$. Un voxel $v$ n'est pas simple dans les quatre cas suivants : (1) $n_1 = 0$, (2) $n_2 = 0$, (3) $n_1 \geq 2$, (4) $n_2 \geq 2$. Nous prouvons que, dans chacun des cas, le voxel $v$ n'est pas un point ML-Simple.

1. $n_1 = 0$. Il n'y a pas de composante 6-connexe de voxels appartenant à $r$ dans $G_6(v, r)$ : $v$ est un point isolé. Dans ce cas, *sf*$(v)$ contient tous les surfels incidents à $v$ et ainsi est homéomorphe à une sphère ce qui contredit la condition (2) de la Définition 23.
2. $n_2 = 0$. Il n'y a pas de composante 18-connexe de voxels appartenant à $\bar{r}$ dans $G_{18}(v, \bar{r})$ : $v$ est à l'intérieur de la région (c'est-à-dire tous les voxels 18-voisins de $v$ ont le même label que $v$. Dans ce cas *sf*$(v)$ est vide ce qui contredit la condition (2) de la Définition 23.
3. $n_1 \geq 2$ : il y a au moins deux composantes 6-connexes de voxels appartenant à $r$ dans $G_6(v, r)$. Si il y a deux voxels 18-adjacents $v_1$ et $v_2$ dans deux composantes connexes différentes, alors le voxel $v_3 \neq v$ 6-adjacent à $v_1$ et à $v_2$ appartient à $\bar{r}$ (sinon il n'y a qu'une composante connexe) et ainsi le lignel $l$ incident à $v$, $v_1$ et $v_2$ est tel que $d(l, v) = 0$ (parce que les deux voxels $v_1$ et $v_2$ appartiennent à $r$ ainsi il n'y a pas de surfels entre ces voxels et $v$) et $d(l) = 2$ (il y a deux surfels, un entre $v_3$ et $v_1$ et un entre $v_3$ et $v_2$), ce qui contredit la condition (3) de la Définition 23.

   Si il n'existe pas deux voxels $v_1$ et $v_2$, dans deux composantes connexes différentes, tels que $v_1$ et $v_2$ soient 18-adjacents, alors les composantes connexes sont séparées par $v$. Dans ce cas, *sf*$(v)$ n'est pas homéomorphe à un disque (c'est un anneau) ce qui contredit la condition (2) de la Définition 23.
4. $n_2 \geq 2$ : il y a au moins deux composantes 18-connexes de voxels appartenant à $\bar{r}$ dans $G_{18}(v, \bar{r})$. Si il y a deux voxels $v_1 \in N_6(v)$ et $v_2 \in N_6(v)$ dans deux composantes connexes différentes alors $v_1$ et $v_2$ ne sont pas 18-adjacents (sinon il n'y a qu'une composante connexe). Comme les deux surfels de *sf*$(v)$ entre $v_1$ et $v$ et entre $v_2$ et $v$ ne sont pas adjacents, *sf*$(v)$ n'est pas homéomorphe à un disque ce qui contredit la condition (2) de la Définition 23.

   Si il n'y a pas deux voxels $v_1$ et $v_2$ de $N_6(v)$ dans deux composantes connexes différentes cela signifie que l'un d'eux (disons $v_1$ par exemple) appartient à $N_{18}(v) \setminus N_6(v)$. Les deux voxels 6-voisins de $v_1$ dans $N_6(v)$ appartiennent à $r$ (sinon nous sommes dans le cas du point précédent).



(a) (b)

FIG. 6.3: Exemple de configuration montrant un point simple qui n'est pas ML-Simple. (a) Le voxel gris foncé est simple pour la région composé des voxels dessinés. (b) Le lignel de degré 4 incident au voxel gris foncé est interdit par la condition (1) de la Définition 23 : le voxel gris foncé n'est pas ML-Simple.

Comme le lignel $l$ incident à $v_1$ et à $v$ est tel que $d(l, v) = 0$ (parce que les deux voxels 6-voisins de $v_1$ dans $N_6(x)$ appartiennent à $r$ ainsi il n'y a pas de surfels entre ces voxels et $v$). Mais il y a deux surfels entre $v_1$ et ces deux voxels 6-voisins dans $N_6(v)$ donc $d(l) = 2$. Cela contredit la condition (2) de la Définition 23. □

La Figure 6.3 illustre une configuration où le voxel $v$ est un point simple mais pas un point ML-Simple. Le voxel $v$ est un point simple d'après la caractérisation donnée par la Définition 22. Cependant, la présence d'un lignel de degré 4 au voisinage du voxel ne permet pas de valider la condition (1) de la Définition 23. Cet exemple prouve que tous les points simples ne sont pas ML-Simples.

Maintenant, nous prouvons que la topologie de l'image est préservée (c'est-à-dire que la topologie de chaque région de l'image est préservée et les relations entre surfaces sont également préservées). Cette preuve est nécessaire puisque l'on travaille avec de multiples régions et que la Proposition 4 prouve seulement que la topologie est préservée dans le cas binaire.

**Proposition 5.** *Si $v$ est un point ML-Simple, la topologie de la partition de l'image n'est pas changée en basculant $v$.*

L'idée générale de la preuve est, dans un premier temps, de montrer que la topologie de chaque région est préservée en utilisant le lien avec les points simples (Proposition 4). En effet, comme le 18-voisinage est binaire, le fait de basculer $v$ change uniquement deux régions. Dans un second temps, nous démontrons que la topologie des surfaces est préservée (imbrication et adjacence entre surfaces). Avec ces deux parties, nous montrons que pour chaque région les nombres de Betti sont préservés.

*Démonstration de la Proposition 5.* Par la Proposition 4, nous savons que si $v$ est un point ML-Simple, alors $v$ est un point simple. Cela prouve que si $N_{26}(v)$ est binaire, la topologie de l'image est préservée lorsque l'on bascule $v$. En effet, nous sommes dans le cas binaire et par définition des points simples, les topologies de $r$ et de $\bar{r}$ sont préservées lorsque l'on bascule $v$ de $r$ dans $\bar{r}$. Les régions autres que $r$ et $\bar{r}$ ne sont pas modifiées par l'opération qui bascule $v$ et ainsi leur topologie reste inchangée.

Maintenant, nous considérons le cas où $N_{26}(v)$ n'est pas binaire. Dans ce cas, la Proposition 4 garantit que les topologies de $r$ et de $\bar{r}$ sont inchangées, ce qui prouve que la topologie globale est préservée. Cependant, $\bar{r}$ est composée par l'union de plusieurs régions $\{r_1, \ldots, r_k, q\}$ (avec $1 \leq k \leq 8$ et $q$ étant la deuxième région dans $N_{18}(v)$). Nous prouvons que la topologie de chaque région est préservée. D'abord, les régions $r_i$ et leurs surfaces ne sont pas modifiées lorsque nous basculons $v$ puisque ces régions ne sont pas 18-adjacentes à $v$ (par le Lemme 2) : leurs topologies sont inchangées.

Ensuite, nous étudions le cas de $q$. Pour prouver que la topologie de $q$ est préservée en ajoutant $v$, nous prouvons que $v$ est simple pour $q$. Ceci peut être fait en utilisant exactement la même preuve que pour la Proposition 4, en remplaçant $r$ par $q$. En effet, les rôles de $r$ et $q$ sont symétriques. Ainsi, $v$ est simple pour $q$ et la topologie de $q$ est inchangée en basculant $v$.

De plus, il n'y a aucune création ni suppression d'intersection de surfaces pour $r$ et pour $q$. La seule surface modifiée se trouve entre $q$ et $r$. Cette modification peut être soit la suppression d'une adjacence entre deux surfaces, soit la création d'une nouvelle adjacence entre deux surfaces. Le premier cas est évité grâce à la condition (1) de la Définition 23. En effet, une adjacence entre plusieurs surfaces implique que $d(l) > 2$. Comme la condition (1) garantit que $d(l) = 0$ ou $d(l) = 2$, il n'y a pas d'intersection de surface autour de $v$ et ainsi basculer $v$ ne peut pas supprimer une intersection. Le deuxième cas est évité par la condition (3) de la Définition 23. En effet, pour créer une nouvelle adjacence, il est nécessaire de toucher une surface existante avec des surfels basculés (c'est-à-dire des surfels qui ne séparent pas deux régions autour de $v$ avant le basculement).

Cela prouve que la topologie de chaque région entourant le voxel basculé reste inchangée que $N_{26}(v)$ soit binaire ou non. Ainsi le nombre de composantes connexes et le nombre de tunnels de chaque région sont préservés. Le nombre de cavités est préservé car la configuration des surfaces ne change pas. En effet, si les cavités d'une région ne sont pas préservées, il y a nécessairement des modifications pour les surfaces imbriquées dans $r$ (deux surfaces adjacentes deviennent non adjacentes ou réciproquement). Cela prouve que la topologie de la partition complète reste inchangée. □

### 6.1.4 Détection de points simples multilabels

Dans le modèle des cartes topologiques, les informations sur les cellules de l'intervoxel sont stockées dans la matrice intervoxel. Dans cette matrice, nous retrouvons notamment l'information sur le degré d'une cellule. Un surfel $s$ est allumé si et seulement si $s \in SF$ (c'est-à-dire $s$ est entre deux voxels ayant des labels différents). Un lignel $l$ est allumé si et seulement si $l$ est incident à plus de deux surfels allumés. Enfin un pointel est allumé s'il est incident à un ou à plus de deux lignels allumés.

La Figure 6.4 présente les cas typiques que nous avons à identifier. Ils sont représentatifs de toutes les configurations possibles par rotation et symétrie.

En utilisant la matrice intervoxel, l'Algorithme 29 détermine si un voxel $v$ est ML-Simple. Nous utilisons pour les explications les configurations de la Figure 6.4. Nous prouvons ensuite que notre algorithme retourne vrai si et seulement si $v$ est ML-Simple.

**Algorithme 29** : Détection de points simples multilabels

**Données** : Matrice intervoxel ;
Un voxel $v$.
**Résultat** : $vrai$ si et seulement si $v$ est un point ML-Simple.

```
pour chaque l ∈ lignels(v) faire
    si l est allumé alors retourner faux;
    si deux surfels incidents à l et à v sont éteints alors
        si au moins un surfel incident à l et pas à v est allumé alors
            retourner faux;
si configuration de surfels est A, D, H, J alors retourner faux;
retourner vrai;
```

*Démonstration.* Pour chaque lignel $l$ incident à $v$, $l$ est allumé implique $d(l) > 2$ ce qui contredit la condition (1) de la Définition 23 : l'algorithme retourne faux. Le deuxième test, si deux lignels

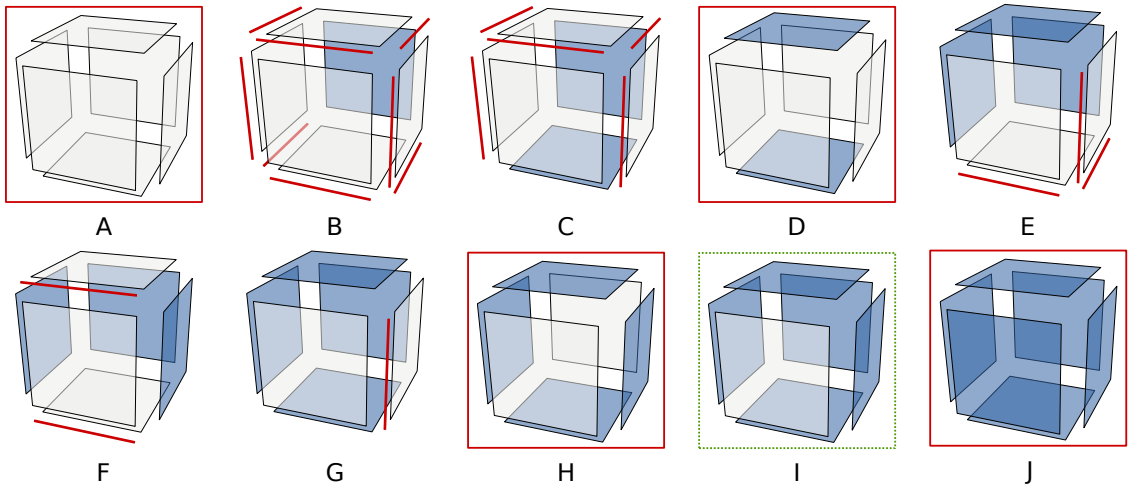



FIG. 6.4: Configurations de base en fonction du nombre de surfels allumés dans $sf(v)$. Les surfels éteints sont représentés en gris clair. Toutes les configurations peuvent être obtenues à partir de ces dix cas de base par rotations et symétries. (A) Cas d'un voxel intérieur avec zéro surfel frontière. (B) Un surfel. (C,D) Deux surfels. (E,F) Trois surfels. (G,H) Quatre surfels. (I) Cinq surfels. (J) Cas d'un voxel isolé avec six surfels frontières. Les lignels dessinés sont ceux pour lesquels $d(l, v) = 0$. Les cadres pleins mettent en évidence les configurations qui ne sont pas ML-Simples. Le cadre en pointillés montre la configuration qui est toujours ML-Simples.

incidents à $l$ et $v$ sont éteints, correspond au cas $d(l, v) = 0$. Dans cette situation, si au moins un surfel incident à $l$ mais pas incident à $v$ est allumé, nous avons $d(l) > 0$ ce qui contredit la condition (1) de la Définition 23 : l'algorithme retourne faux. Le dernier test vérifie la condition (2) de la Définition 23. Nous testons si $sf(v)$ est homéomorphe à un disque en testant tous les cas où cette condition n'est pas satisfaite et l'algorithme retourne alors faux dans ces cas. Après tous ces tests, les trois conditions de la Définition 23 sont satisfaites, ainsi $v$ est ML-Simple. L'algorithme retourne vrai. □

L'algorithme s'exécute en temps constant : sa complexité est $O(1)$. Il y a douze lignels incidents à $v$ et chaque test (si une cellule est allumée ou éteinte) est une opération atomique. Vérifier si nous sommes dans l'une des configurations interdites est effectué en temps constant pour chaque cas, en testant et comptant les surfels.

## 6.2 Déformation de surfaces

En utilisant l'algorithme de détection de points simples multilabels, nous proposons un outil de déformation de partition dans les cartes topologiques. Premièrement, nous détaillons l'opération de base permettant de basculer un voxel ML-Simple dans une partition représentée par une carte topologique. Deuxièmement, nous définissons un système d'énergie qui permet de contrôler la déformation et nous détaillons le calcul de ces énergies. Enfin nous réalisons l'algorithme de déformation d'une face.

### 6.2.1 Mise en œuvre du basculement d'un voxel simple multilabels

D'après la Proposition 5, le basculement d'un point ML-Simple ne change pas la topologie de la partition représentée. Il n'y a ni apparition ni disparition de cellules. De même les informations d'adjacence et d'incidence entre cellules sont conservées tout comme la relation d'imbrication entre les

régions de la partition. Cette propriété permet de dire que la carte combinatoire et l'arbre d'imbrication d'une carte topologique ne sont pas modifiés. Le basculement d'un voxel ML-Simple est une opération qui ne modifie que la géométrie et donc uniquement la matrice intervoxel.

La tâche élémentaire consistant à basculer un voxel $v$ ML-Simple d'une région dans l'autre dans la matrice intervoxel est réalisée en inversant l'état (entre allumé et éteint) de tous les surfels incidents à $v$. Par définition des points simples multilabels, il n'y a pas de lignel ni de pointel incident à un voxel ML-Simple.

La carte topologique stocke également d'autres informations concernant les régions comme par exemple la couleur moyenne d'une région ou le nombre de voxels des régions. Ces informations changent lorsqu'un voxel est basculé d'une région dans l'autre. Aussi nous mettons à jour les informations en utilisant la carte topologique pour connaître la région de départ qui perd un voxel et la région d'arrivée qui elle gagne un voxel.

### 6.2.2 Énergies

Les processus de déformation sont généralement guidés par un problème de minimisation de l'énergie de chaque surface représentée dans la matrice intervoxel. Nous allons définir les énergies utilisées dans notre problème. Nous utilisons trois énergies différentes : une utilisant les données de l'image, et deux utilisant l'aire des surfaces de la partition.

L'énergie $E_r$ est une énergie basée sur les données associées aux régions incidentes à la surface à déformer. Si $r_1$ et $r_2$ sont deux régions incidentes à la surface $S$, alors cette énergie est définie comme

$$E_r(S) = MSE(r_1) + MSE(r_2)$$

où $MSE(r)$ est l'erreur quadratique moyenne (*Mean Square Error* en anglais) de la région $r$. L'énergie de la région augmente avec l'erreur quadratique moyenne des deux régions incidentes.

L'erreur quadratique moyenne d'une région se calcule de la façon suivante : soit $M_0(r)$ le nombre de voxels d'une région $r$, soit $M_1(r)$ la somme des intensités d'une région $r$ et soit $M_2(r)$ la somme des intensités au carré d'une région $r$. La couleur moyenne $\nu(r)$ est définie par $\nu(r) = M_1(r)/M_0(r)$. La variance de la région $r$, notée $var(r)$, est donnée par $var(r) = M_2/M_0 - \nu^2(r)$. Enfin l'erreur quadratique moyenne $MSE(r)$ d'une région $r$ est définie par $MSE(r) = M_0 \times var(r)$.

L'énergie $E_s$ est une énergie basée sur le nombre de surfels de la surface en cours de déformation. Soit $|\textit{surfels}(S)|$ le nombre de surfels appartenant à la surface $S$, alors cette énergie est définie comme

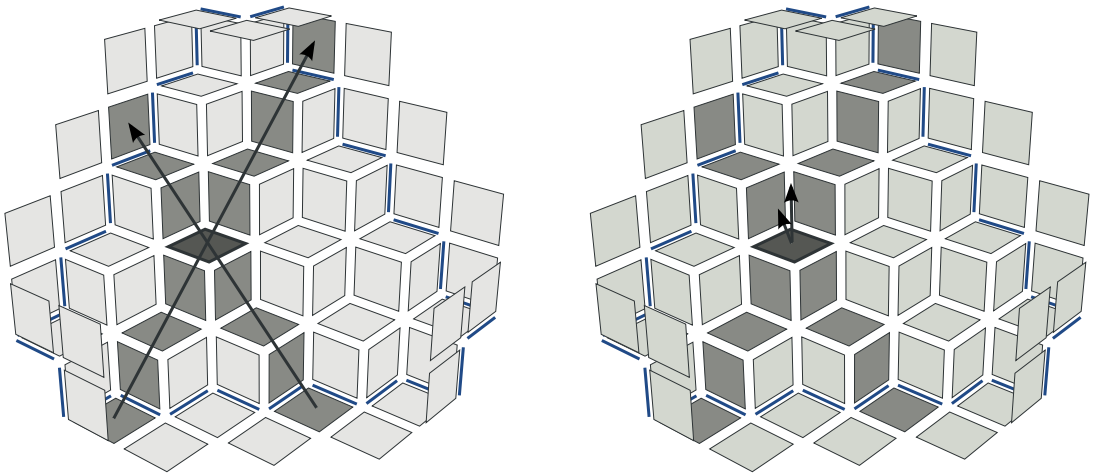
$$E_s(S) = |\textit{surfels}(S)|.$$

L'énergie $E_a$ est une énergie basée sur une approximation de l'aire de la surface à l'aide d'un estimateur discret. Celui-ci calcule la contribution de chaque surfel à l'aire totale. L'estimation de l'aire pour un surfel de la surface n'est pas identique en fonction du côté de la surface considérée. Nous prenons donc en compte les deux côtés de la surface dans notre calcul. Ainsi l'énergie $E_a$ est définie comme

$$E_a = \sum_{s \in \textit{surfels}(S)} \frac{aire(s, r_1) + aire(s, r_2)}{2}$$

où $aire(s, r)$ est l'estimateur d'aire du surfel $s$ en considérant la surface du côté de la région $r$. L'énergie de la surface diminue lorsque la surface diminue. Ainsi une surface plane possède l'énergie minimale.

L'estimation de l'aire d'une surface est calculée par la somme des contributions de chaque surfel à l'aire totale. La contribution d'aire d'un surfel est calculée par une estimation du plan discret



(a) (b)

FIG. 6.5: Calcul de la contribution d'un surfel (en gris foncé) à l'aire de sa surface. Les surfels appartenant aux segments discrets maximaux sont représentés en gris. (a) Les deux vecteurs directeurs des segments maximaux estimant le plan auquel le surfel appartient. (b) Normale naïve du surfel et normale au plan estimé : le produit scalaire de ces deux vecteurs donne la contribution du surfel à l'aire de la surface.

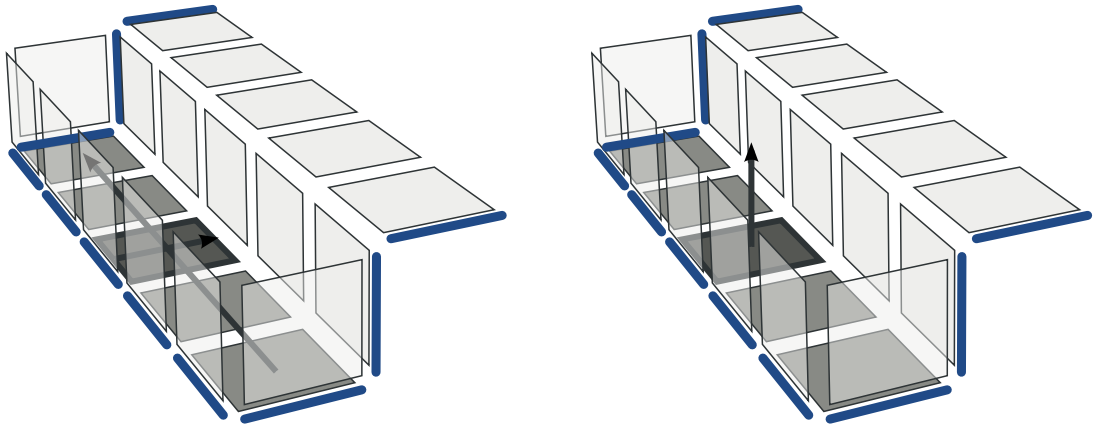
auquel appartient le surfel. Cette estimation du plan est réalisée à partir de la reconnaissance de deux segments discrets maximaux passant par le surfel. Ce processus est conduit à l'aide de la librairie ImaGene[Lac03]. Nous calculons les vecteurs directeurs de ces deux segments : ils estiment le morceau de plan dans lequel se trouve le surfel. En calculant le produit scalaire de la normale du plan avec la normale naïve du surfel, nous obtenons la contribution du surfel à l'aire de la surface. Cette valeur est toujours inférieure à un (elle vaut un lorsque le plan discret estimé est le même que celui du vecteur).

La Figure 6.5 illustre le fonctionnement du calcul de la contribution d'un surfel à l'aide de sa surface. Nous calculons deux segments discrets symétriques maximaux orthogonaux passant par le surfel. Les surfels de ces deux segments sont représentés en gris foncé. Les vecteurs directeurs de ces deux segments donne une estimation du plan contenant le surfel.

Le calcul est une approximation de l'aire qui dépend de la reconnaissance du plan discret autour du surfel. Sur le bord des surfaces ou dans le cas de surfaces très bruitées, le plan discret contenant le surfel n'est pas correctement identifié. La Figure 6.6 présente le cas d'un calcul de l'estimateur d'aire pour un surfel situé sur le bord d'une surface. Dans cette configuration, l'un des segments orthogonaux est réduit au surfel considéré : le plan reconnu est donc celui des 5 surfels coloriés en gris foncé. La contribution du surfel est donc égale à un, car la normale du plan est égale à la normale naïve. Cette contribution est fausse étant donné le plan (incliné à 45°) que nous avons. Ce constat est dépendant de l'algorithme de reconnaissance de segment maximaux et d'autres algorithmes peuvent améliorer ce résultat.

Le processus de déformation consiste à minimiser l'énergie de chaque surface représentée dans la matrice intervoxel. L'énergie totale $E(S)$ d'une surface $S$ est définie comme :

$$E(S) = \omega_r E_r(S) + \omega_s E_s(S) + \omega_a E_a(S)$$



(a) (b)

FIG. 6.6: Calcul de l'estimateur d'aire pour un surfel (colorié en gris foncé) sur le bord d'une surface. (a) Lors du calcul des deux segments orthogonaux, les surfels dessinés en gris représentent les segments maximaux reconnus. Le segment dans la direction de la pente est réduit au surfel considéré. (b) La normale au plan reconnu est égale à la normale estimée. L'aire estimée du surfel est égale à un, ce qui n'est pas correct vue l'inclinaison du plan.

où $E_r$, $E_s$ et $E_a$ représentent respectivement l'énergie associée aux régions incidentes à la surface, l'énergie associée au nombre de surfels et l'énergie associée à l'aire de la surface. Les paramètres $\omega_r$, $\omega_s$ et $\omega_a$ sont les poids définissant l'importance relative des termes.

À présent, nous avons vu quelles énergies sont utilisables dans le cadre de la déformation de surface. Nous avons également regardé comment ces énergies sont calculées. Il reste à nous intéresser à l'algorithme de déformation en lui-même.

### 6.2.3 Algorithme de déformation

L'algorithme de déformation d'une face est décrit par Algorithme 30. Cet algorithme prend une face frontière de l'image et la déforme de manière à diminuer l'énergie. L'algorithme présenté ne réalise qu'une étape de la déformation minimisant l'énergie : chaque surfel n'est traité qu'une fois et nous ne traitons pas les surfels qui sont créés lors du processus de déformation. Nous assurons que l'énergie associée à la surface résultante de l'algorithme de déformation est plus petite que l'énergie initiale de la surface.

L'énergie initiale est calculée et pour chaque surfel $s$ appartenant à la surface $S$, l'algorithme teste deux déformations : une pour chaque voxel $(v)$ incident à $s$. Si le voxel $v$ est ML-Simple, nous le basculons temporairement et calculons l'énergie de la surface associée à ce mouvement. Le mouvement d'énergie minimale est sélectionné. Si l'énergie du mouvement minimal est inférieure à l'énergie initiale, nous appliquons effectivement le basculement du voxel en question et définissons la nouvelle énergie.

L'algorithme peut être utilisé de différentes manières, par exemple pour déformer une face entre deux régions, pour déformer toutes les frontières d'une région ou pour définir une minimisation globale de l'énergie dans les faces frontières de la partition. Dans ce dernier cas, nous appliquons l'algorithme sur toutes les faces frontières représentées par la carte topologique et nous itérons jusqu'à ce qu'aucun voxel ne soit basculé (une énergie minimale est atteinte localement). Comme nous traitons les surfels associés aux faces et comme un basculement n'est fait que si l'énergie diminue strictement, nous arrêtons le processus lorsque l'énergie descend en-dessous d'une valeur seuil.

**Algorithme 30** : Déformation : une étape de la déformation de surface

**Données** : Une surface $S$ dans la matrice intervoxel.
**Résultat** : Une étape de déformation de $S$ pour diminuer l'énergie.

$E_{initial} \leftarrow E(S)$;
**pour chaque** $s \in$ surfels$(S)$ **faire**
  $(v_1, v_2) \leftarrow$ voxels incidents à $s$;
  $(E_{v_1}, E_{v_2}) \leftarrow (E_{initial}, E_{initial})$;
  **si** $v_1$ *est ML-Simple* **alors** $E_{v_1} \leftarrow E(S$ avec $v_1$ basculé$)$;
  **si** $v_2$ *est ML-Simple* **alors** $E_{v_2} \leftarrow E(S$ avec $v_2$ basculé$)$;
  **si** $E_{initial} > min(E_{v_1}, E_{v_2})$ **alors**
    basculer le voxel correspondant;
    $E_{initial} \leftarrow min(E_{v_1}, E_{v_2})$;

Il faut noter que l'ordre de parcours des surfels de la face influe sur le résultat final. En effet, en fonction de l'ordre dans lequel les déformations sont effectuées, nous changeons l'énergie calculée et le minimum global peut ne pas être atteint. Un minimum local est alors trouvé.

## 6.3 Expérimentations sur les énergies

Nous proposons une étude de l'énergie basée sur l'estimation discrète de l'aire d'une surface. Cette étude vise à comparer l'estimateur d'aire discret avec l'estimateur d'aire basé sur le nombre de surfels de la surface. Nous souhaitons ainsi mettre en évidence l'intérêt des deux approches. Dans un premier temps, nous regardons la précision de l'estimateur d'aire dans le cas de surfaces idéales, c'est-à-dire non bruitées. Puis, dans un second temps, nous étudions la pertinence de l'énergie basée sur l'estimateur discret pour débruiter une surface par rapport à l'énergie utilisant le nombre de surfels.

### 6.3.1 Précision de l'estimateur d'aire de surfaces

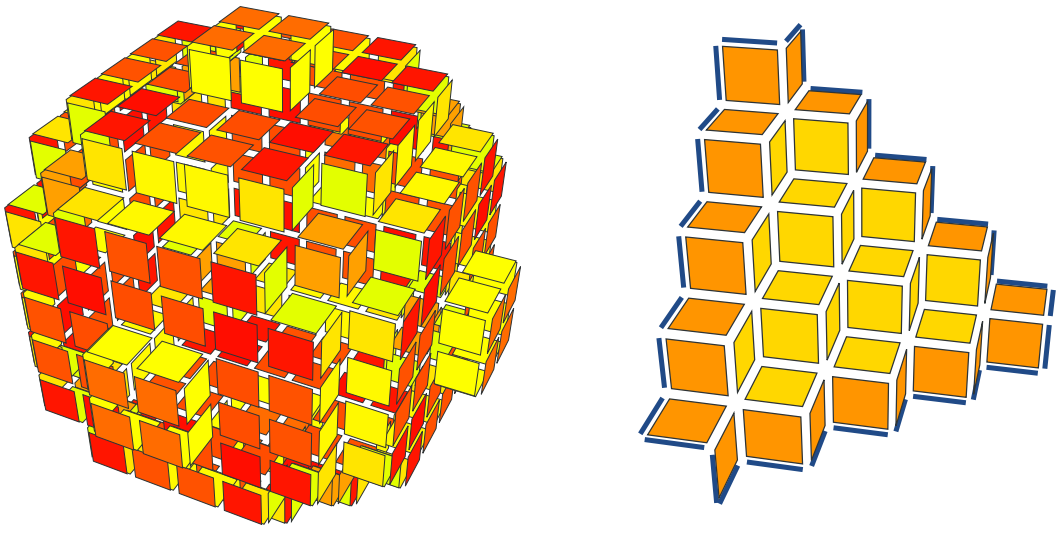
Nous commençons par étudier les valeurs calculées par l'estimateur discret d'aire sur certaines surfaces caractéristiques. L'objectif étant de s'assurer que les résultats proposés sont conformes aux résultats théoriques. Pour chaque surface, nous utilisons la formule du calcul de l'énergie basée sur l'estimateur d'aire discret qui donne une approximation de l'aire de la surface et nous la comparons avec l'aire théorique. Les aires dans tout le reste du manuscrit sont données en unité de surface où l'aire d'une surfel vaut une unité de surface.

La Table 6.1 présente les mesures réalisées avec l'estimateur d'aire pour des surfaces discrètes connues de tailles différentes. Nous comparons cette mesure avec le nombre de surfels et avec la mesure théorique calculée à l'aide des équations des surfaces. Nous mesurons l'aire d'une sphère de différents rayons, l'aire de morceaux de plans (un morceau horizontal, trois morceaux de plans obliques inclinés à 30, 45 et 60 degrés et un morceau d'un plan appelé R2 qui passe par les points (0,0,1), (0,1,0) et (1,0,0).

La Figure 6.7 présente des exemples de calcul de l'estimateur d'aire pour deux surfaces. L'échelle des couleurs indique la valeur de l'aire estimée pour chaque surfel. Elle va de vert (aire nulle : 0) à rouge (aire maximale : 1). Dans la Figure 6.7a l'estimateur est appliqué à une sphère discrète de rayon 5. L'estimation de la surface de cette sphère est de 281,29 alors que la valeur théorique est de 314,15. Dans la Figure 6.7b nous présentons l'estimation d'aire de chaque surfel d'un plan discret. L'estimateur donne une valeur de 25,50 alors que la valeur théorique est 21,65.

TAB. 6.1: Comparaison de l'estimateur d'aire avec la mesure théorique et le nombre de surfels pour des surfaces connues.

| Surface | Surfels | Aire estimées | Aire théorique |
|---|---|---|---|
| Sphère 10 | 1752 | 1254,69 | 1256,64 |
| Sphère 20 | 7248 | 4982,24 | 5026,55 |
| Sphère 50 | 46344 | 31282,60 | 31415,93 |
| Plan XY $10 \times 10$ | 100 | 98,67 | 100 |
| Plan XY $50 \times 50$ | 2500 | 2495,26 | 2500 |
| Plan Oblique $45^\circ$ $10 \times 10$ | 180 | 132,76 | 141,42 |
| Plan Oblique $45^\circ$ $50 \times 50$ | 4900 | 3493,75 | 3535,53 |
| Plan Oblique $30^\circ$ $20 \times 20$ | 678 | 390,30 | 447,21 |
| Plan Oblique $60^\circ$ $40 \times 40$ | 2568 | 1478,82 | 1788,85 |
| Plan R2 $10 \times 10$ | 162 | 100,54 | 86,60 |
| Plan R2 $50 \times 50$ | 3822 | 2199,87 | 2165,06 |



(a) (b)

FIG. 6.7: Exemples d'estimation d'aire pour deux surfaces discrètes. L'échelle des couleurs indique la valeur de l'aire estimée pour chaque surfel. Elle va de vert (aire nulle : 0) à rouge (aire maximale : 1). (a) Pour une sphère de rayon 5, la mesure estimée est de 281,29 alors que la mesure théorique est de 314,15. (b) Pour un plan R2 dans un cube de $5 \times 5 \times 5$ voxels, la mesure estimée est de 25,50 alors que la mesure théorique est de 21,65.

Nous n'obtenons pas la mesure exacte mais nous pouvons observer que plus la surface est grande, plus l'estimateur est précis. En effet, cela s'explique par les problèmes de mesure sur le bord des surfaces. Au milieu des surfaces l'estimateur discret est précis. Cependant sur les bords des surfaces, il manque d'information pour donner une mesure précise de la contribution des surfels à la surface totale. Ainsi si la surface est petite, l'erreur faite au bord des surfaces intervient plus dans le calcul complet. De manière générale, l'estimation d'aire par cette approche est très sensible au bruit et l'estimation manque alors de précision.

Considérons par exemple le plan présenté Figure 6.7b. Pour que ce plan corresponde à une face, nous avons créé deux régions adjacentes dont la face frontière forme un plan à 45 degrés. Le calcul de l'estimateur d'aire de chaque côté du plan n'est pas identique car nous considérons la surface complète de la région et pas simplement les surfels appartenant à la face entre les deux régions. Nous observons avec le code couleur que les pixels du bord ont une contribution plus forte à l'aire de la surface alors que selon nous, tous les surfels doivent avoir une contribution similaire. Cette erreur vient d'une part de la manière dont l'estimateur d'aire est calculé, mais vient également des différences de calcul selon le côté de la face considérée. En pratique nous calculons la moyenne des deux estimations pour calculer l'aire de la surface. L'erreur présentée ici n'est pas limitée au bord de l'image mais se retrouve également dans toutes les zones à fort bruit.

L'estimateur d'aire discret donne une bonne estimation de l'aire d'une surface et est en général meilleur que l'estimation naïve comptant uniquement le nombre de surfels comme nous pouvons le vérifier Table 6.1. Nous pouvons donc envisager de calculer cette énergie pour réaliser des déformations visant à diminuer l'aire d'une surface bruitée et ainsi retrouver une surface moins bruitée.

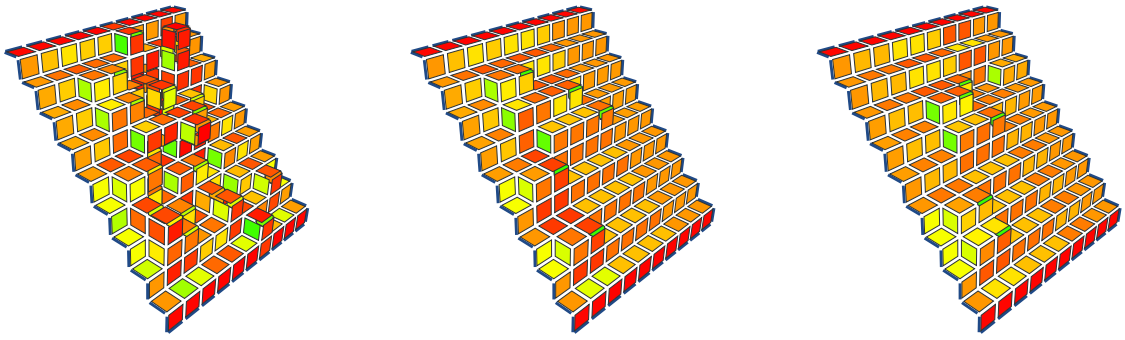
### 6.3.2 Lissage de surfaces bruitées par déformation

Nous utilisons ici une énergie basée sur l'estimateur d'aire discret pour débruiter des surfaces bruitées. Afin de mesurer l'intérêt de cette approche, nous comparons notre résultat avec une énergie cherchant à minimiser le nombre de surfels de la surface.

Dans la Figure 6.8, nous présentons un exemple de débruitage d'un morceau de surface discrète incliné à 45°. Le plan bruité original, Figure 6.8a, est dans un premier temps déformé en minimisant l'énergie basée sur le nombre de surfels de la surface : nous effectuons les basculements si le nombre de surfels diminue strictement. La Figure 6.8b montre les surfels du morceau de plan après cette déformation. Dans un second temps, nous déformons le plan original bruité en utilisant l'énergie basée sur l'estimateur d'aire : nous effectuons le basculement si l'aire estimée décroît strictement. Le plan résultant est présenté Figure 6.8c. Les mesures effectuées (voir Table 6.3) montrent que le nombre de surfels est plus faible pour la déformation minimisant le nombre de surfels et l'aire estimée est plus faible dans la déformation utilisant l'estimateur d'aire. Cependant la différence entre les deux n'est pas flagrante et aucune des deux approches ne permet d'atteindre le plan original. Les deux déformations arrivent à un minimum local.

Nous avons essayé d'utiliser le processus de déformation pour lisser des surfaces en utilisant différentes surfaces de taille variable afin de voir si l'estimateur d'aire propose parfois un meilleur résultat lorsque sa précision augmente. Les résultats obtenus pour les plans horizontaux sont présentés dans la Table 6.2. Nous avons également testé l'algorithme avec un plan incliné à 45° (Table 6.3). Dans ces deux expériences, nous observons que les deux approches (à l'aide de l'estimateur et à l'aide du nombre de surfels) donnent des résultats très proches.

Nous essayons le même traitement pour une surface sphérique de rayon variable. La Table 6.4 montre que l'estimateur d'aire donne des résultats plus proches des résultats théoriques que l'approche minimisant le nombre de surfels. Nous observons en pratique que la minimisation conduit à produire une surface plutôt sphérique mais de rayon plus important que la sphère initiale. En effet, la déformation aligne la surface de la sphère sur les sommets du bruit, car les déformations qui réduisent le plus



(a) (b) (c)

FIG. 6.8: Exemple de lissage d'une surface bruitée. (a) Surface bruitée originale. (b) Surface lissée en minimisant le nombre de surfels. (c) Surface lissée en minimisant l'estimateur d'aire.

TAB. 6.2: Comparaison du lissage d'une surface plane horizontale bruitée entre un processus de déformation minimisant le nombre de surfels et un processus minimisant l'aire estimée. Nous donnons les mesures effectuées en utilisant l'estimateur d'aire discret. Nous n'avons pas réalisé la déformation minimisant l'aire estimée pour le carré de 200 surfels car le temps de calcul n'est pas raisonnable (plus de deux heures).

| Côté de la surface | Aire théorique | Déformation Surfels | Déformation Estimateur |
|---|---|---|---|
| 5 | 25 | 24,24 | 24,24 |
| 10 | 100 | 98,66 | 98,66 |
| 15 | 225 | 223,26 | 223,26 |
| 20 | 400 | 397,80 | 397,80 |
| 30 | 900 | 897,15 | 897,94 |
| 40 | 1600 | 1596,10 | 1597,47 |
| 50 | 2500 | 2495,25 | 2500,70 |
| 60 | 3600 | 3595,06 | 3596,89 |
| 100 | 10000 | 9993,36 | 10003,84 |
| 200 | 40000 | 39987,45 | - |

TAB. 6.3: Comparaison du lissage d'une surface bruitée entre un processus de déformation minimisant le nombre de surfels et un processus minimisant l'aire estimée. Nous donnons les mesures effectuées en utilisant l'estimateur d'aire discret.

| Surface Oblique 45° | Aire théorique | Déformation Surfels | Déformation Estimateur |
|---|---|---|---|
| $10 \times 10$ | 141,42 | 126,73 | 127,42 |
| $15 \times 15$ | 318,20 | 296,13 | 296,80 |
| $20 \times 20$ | 565,69 | 536,25 | 536,25 |
| $25 \times 25$ | 883,88 | 847,07 | 848,36 |
| $30 \times 30$ | 1272,79 | 1228,88 | 1229,74 |

TAB. 6.4: Comparaison du lissage d'une surface sphérique bruitée entre un processus de déformation minimisant le nombre de surfels et un processus minimisant l'aire estimée. Nous donnons les mesures effectuées en utilisant l'estimateur d'aire discret.

| Rayon de la sphère | Aire théorique | Déformation Surfels | Déformation Estimateur |
|---|---|---|---|
| 5 | 314,15 | 456,56 | 456,56 |
| 8 | 804,24 | 788,77 | 737,42 |
| 10 | 1256,63 | 1800,43 | 1206,25 |
| 12 | 1809,55 | 2409,28 | 1945,09 |
| 15 | 2827,43 | 3805,52 | 2768,96 |

TAB. 6.5: Comparaison du temps d'exécution (en secondes) entre une déformation minimisant le nombre de surfels et une déformation minimisant l'aire estimée.

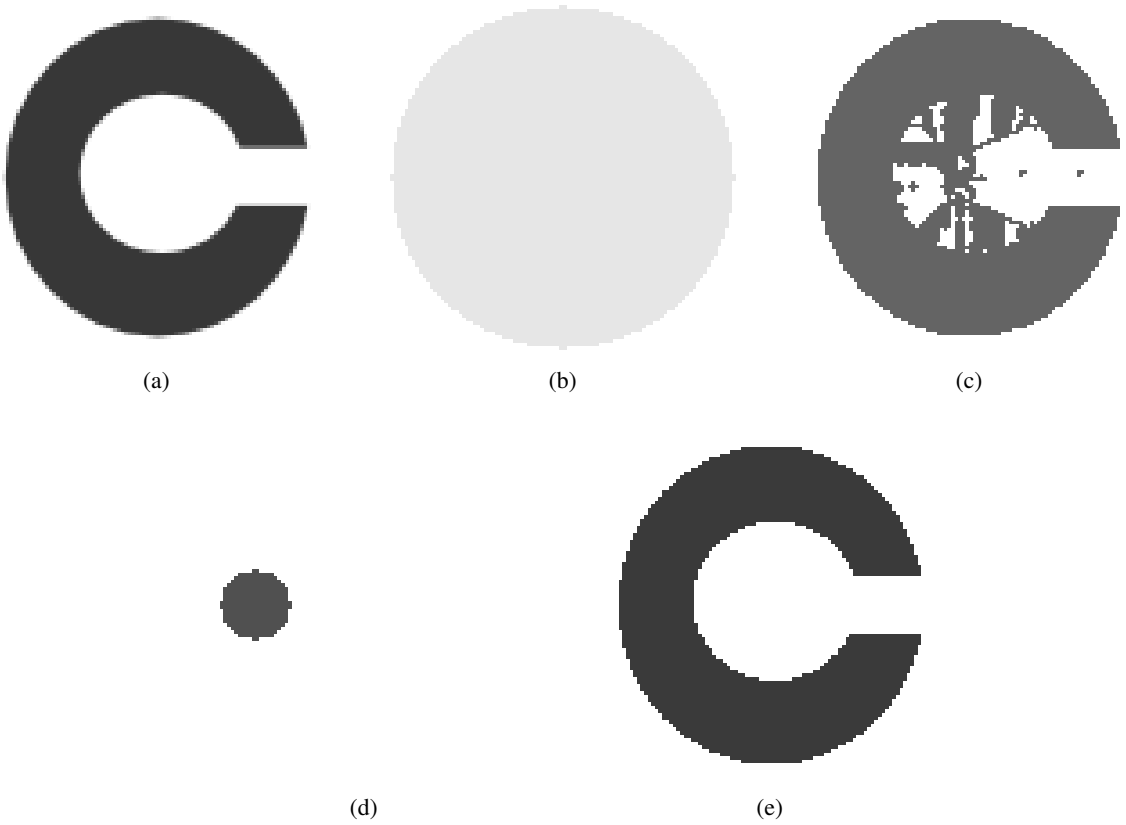
| Surface Oblique 45° | Déformation Surfels | Déformation Estimateur |
|---|---|---|
| $10 \times 10$ | 0,002 | 1,174 |
| $15 \times 15$ | 0,008 | 7,981 |
| $20 \times 20$ | 0,013 | 28,670 |
| $25 \times 25$ | 0,020 | 78,185 |
| $30 \times 30$ | 0,030 | 176,171 |

le nombre de surfels sont celles qui bouchent les petits trous à la surface de la sphère. À l'inverse, l'évaluation de l'estimateur d'aire diminue lorsque les bosses sont aplanies. Nous n'obtenons cependant pas un très bon résultat. En effet, les déformations pour "creuser" la sphère sont plus susceptibles de produire des configurations non ML-Simples ce qui empêche la déformation et empêche bien souvent de trouver la configuration optimale.

Étant donné la proximité des résultats obtenus pour les surfaces planes, nous souhaitons savoir quelle approche est à préférer dans nos applications. Nous avons donc mesuré le temps de calcul du processus de déformation pour les deux énergies pour le lissage d'un plan incliné. Les résultats des mesures sont présentés dans la Table 6.5. Nous observons que le temps de calcul de l'estimateur discret est très long comparé au temps de calcul du nombre de surfels. En effet, si le nombre de surfels d'une surface est rapide à retrouver (par un simple parcours des surfels de la surface) il est également possible de conserver incrémentalement sa valeur durant les opérations de déformations. L'algorithme de calcul de l'estimateur d'aire ne permet pas cette approche et nous remarquons même que cet algorithme a une complexité quadratique en fonction du nombre de surfels de la surface. La conclusion de cette expérience est que l'estimateur d'aire tel que nous le calculons est trop perturbé par le bruit et qu'il n'est donc pas un moyen plus fiable que le nombre de surfels lorsque nous l'utilisons comme énergie pour la déformation. Comme il est également beaucoup plus coûteux, nous utilisons par la suite le nombre de surfels lorsque nous faisons intervenir l'aire de la surface.

## 6.4 Déformation d'une partition binaire grossière

La première opération de déformation que nous avons expérimentée est la déformation sur une image binaire qui ne représente qu'un objet et le fond. Ce type d'approche est le plus courant dans le contexte des modèles déformables. Dans cette expérience, nous n'autorisons pas les régions à posséder plusieurs composantes connexes de par la définition des régions de la carte topologique. Nous revenons dans la Section 6.5 sur des méthodes permettant de simuler un objet ayant plusieurs composantes connexes en utilisant plusieurs régions initiales dans la partition.



(a) (b) (c)

(d) (e)

FIG. 6.9: Segmentation d'une image artificielle binaire en utilisant un modèle déformable. (a) Vue en coupe de l'image originale. (b) Partition initiale par une sphère englobant la forme. (c) Résultat après déformation de la sphère englobante, le résultat est trop imprécis. (d) Partition initiale par une petite sphère imbriquée dans la forme. (e) Résultat après déformation de la sphère, le résultat est beaucoup plus précis.

La première expérience consiste à déformer une sphère initiale pour la faire correspondre à une forme artificielle dans une image 3D. La Figure 6.9a présente une vue en coupe de la forme en 'C'. La Figure 6.9b présente une partition initiale représentant une sphère remplissant l'image. Nous appliquons le processus de déformation en utilisant l'erreur quadratique comme énergie à minimiser. Nous donnons un poids faible à l'énergie basée sur la surface favorisant ainsi l'énergie basée sur l'erreur quadratique. Nous favorisons du même coup la déformation de la région pour s'adapter à la forme artificielle représentée dans l'image. La Figure 6.9c présente la même coupe après la déformation. Nous observons que le résultat n'est pas conforme à ce que nous attendons, la forme en 'C' n'est pas correctement segmentée. Le problème vient de la mise en œuvre de l'algorithme de déformation par points ML-Simple. En effet en fonction de l'ordre dans lequel nous déformons une surface, nous pouvons produire une configuration dans laquelle aucun voxel n'est un point ML-Simple : le même genre de problème apparaît avec la notion usuelle de point simple [PCB08]. Une déformation utilisant la partition initiale présentée Figure 6.9d produit un résultat plus cohérent comme le montre la Figure 6.9e. Le comportement de l'algorithme de déformation est donc relatif à la fois à la position et la taille de la partition initiale. D'une manière générale, l'algorithme fonctionne mieux lorsque la surface est en expansion plutôt qu'en diminution. Cela provient du fait que le retrait de voxels d'une région provoque plus facilement l'apparition d'une configuration où tous les voxels ne sont plus ML-Simple alors que l'ajout de voxels n'en produit que rarement.

(a) (b) (c)

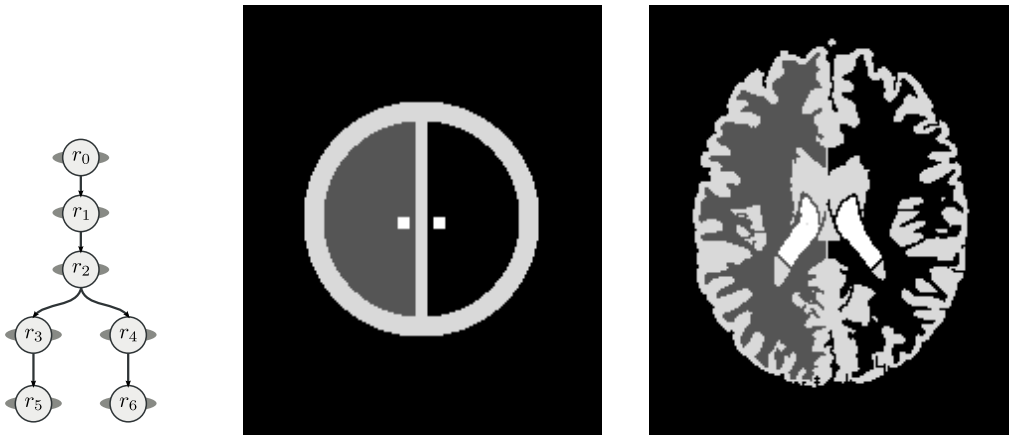
FIG. 6.10: Application de la déformation à topologie constante sur des sphères imbriquées en utilisant le principe de la méthode de [MPA08]. (a) Coupe de l'image IRM orginale d'un cerveau (image IRM simulée [KEP99]). (b) Partition, composée de trois sphères imbriquées, utilisée pour faire la segmentation. (c) Partition finale après la déformation. La topologie n'a pas changée mais la partition est maintenant un peu mieux adaptée aux données de l'image.

## 6.5 Déformation d'une partition avec plusieurs régions imbriquées

La seconde expérience utilise un principe établi par Miri et al. [MPA08] afin de permettre la segmentation multipartition en utilisant un algorithme de modèle déformable. Dans leurs travaux, la segmentation corticoïdale est réalisée en déformant à topologie constante une partition initiale selon des valeurs seuil de niveau de gris. La partition initiale est composée de sphères imbriquées les unes dans les autres. Nous intégrons à l'opération de déformation un critère basé sur l'erreur quadratique moyenne afin de reproduire le résultat en utilisant les points ML-Simples pour effectuer la déformation.

La Figure 6.10 présente les résultats obtenus dans nos travaux. La Figure 6.10a montre une coupe de l'image initiale utilisée. Dans la Figure 6.10b nous présentons une coupe de la partition initiale. Il s'agit de quatre sphères imbriquées. L'application de l'algorithme, en utilisant les notions d'énergie basée sur les données de l'image et sur la surface par nombre de surfels, produit une déformation similaire à celui proposé par les auteurs originaux (Figure 6.10c). Cependant l'utilisation d'un critère différent pour guider la déformation ne permet pas d'obtenir le même résultat. L'utilisation des cartes topologiques permet cependant d'envisager une partition initiale qui ne comprenne pas que des régions imbriquées. Afin de montrer un exemple pour ce type de partition, nous segmentons la même image mais en utilisant un masque différent qui contient une demi-sphère pour représenter le cortex (Figure 6.11). L'arbre d'imbrication montre la configuration d'imbrication des différentes régions de la partition déformée. Ainsi les cartes topologiques donnent une plus grande liberté puisque le même algorithme permet de traiter des configurations vraiment différentes.

Ces expériences montrent que la définition des points simples multilabels permet de mettre rapidement en œuvre des algorithmes de segmentation à base de modèle déformable sur des partitions représentées par des cartes topologiques. Ces modèles n'utilisent cependant pas encore toutes les possibilités liées au traitement multirégions, il n'y a pas d'adjacences multiples autour des arêtes. C'est pour illustrer ces possibilités que nous avons étudié l'opération de raffinement d'une partition.




(a) (b) (c)

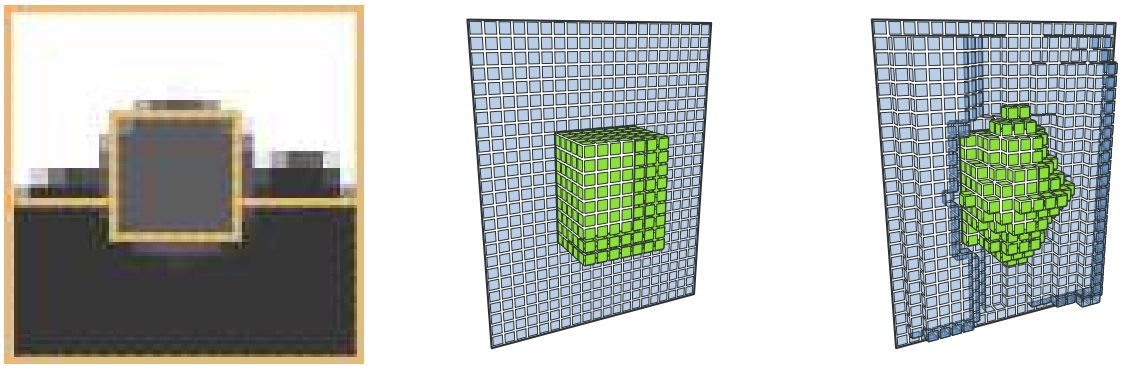
FIG. 6.11: Application de la déformation à topologie constante sur des surfaces avec différentes imbrications. (a) Arbre d'imbrication des régions de la partition. (b) Partition composée de six régions, plus la région infinie utilisée pour faire la segmentation. (c) Partition finale après la déformation. La topologie n'a pas changée mais la partition est maintenant adaptée aux données de l'image. Nous observons comment les deux hémisphères du cerveau ont bien été prises en compte.

## 6.6 Raffinement d'une partition

Le raffinement d'une partition d'une image à l'aide de l'opération de déformation consiste à prendre en entrée une image déjà segmentée et à appliquer une déformation sur les surfaces pour optimiser le critère énergétique. Nous supposons alors que la topologie de l'image représentée est la même que celle de la partition finale désirée, la différence étant seulement géométrique. Comme l'opération est définie à partir des points ML-Simples, nous savons qu'il n'est pas possible que les arêtes et les sommets de la partition changent de géométrie. Pour obtenir le résultat recherché, nous utilisons l'algorithme de déformation sur toutes les faces des régions de l'image en utilisant le système d'énergie présenté avec le nombre de surfels et l'évaluation de l'erreur quadratique permettant de mettre en correspondance les données de l'image et la partition.

Dans une première expérience, nous nous intéressons à l'optimisation d'une partition sur une image artificielle. La Figure 6.12 présente la déformation d'une partition simple représentant une image artificielle. L'image originale est composée de 3 régions, une région sphérique au milieu de l'image et deux régions qui remplissent le reste de l'image avec une frontière ondulante. La Figure 6.12a présente une coupe de l'image originale sur laquelle nous observons les frontières de la partition initiale. La partition initiale utilisée représente, par un morceau de plan, la surface entre les deux régions du fond et, par un cube, la région sphérique. Elle donne une géométrie grossière par rapport aux données de l'image. Deux des surfaces initiales sont dessinées Figure 6.12b. La déformation utilisant l'énergie image basée sur l'erreur quadratique moyenne modifie la géométrie des surfaces afin de mieux retrouver les régions représentées. La Figure 6.12c présente les deux mêmes surfaces après le processus de déformation. Nous observons que les formes originales sont mieux représentées. Cependant, les formes géométriques de la partition initiale sont encore visibles, notamment au niveau de la frontière entre les surfaces qui n'est pas modifiée par le processus de déformation. La non déformation des arêtes explique que la partition finale ne soit pas aussi précise que nous pourrions l'espérer.

Dans la seconde expérience de raffinement d'une partition, nous utilisons une segmentation d'une image médicale obtenue préalablement comme partition initiale. Dans cette image (voir Figure 6.13a), l'algorithme de segmentation a défini 14 régions. La partition totale comprend 75 faces. La région la



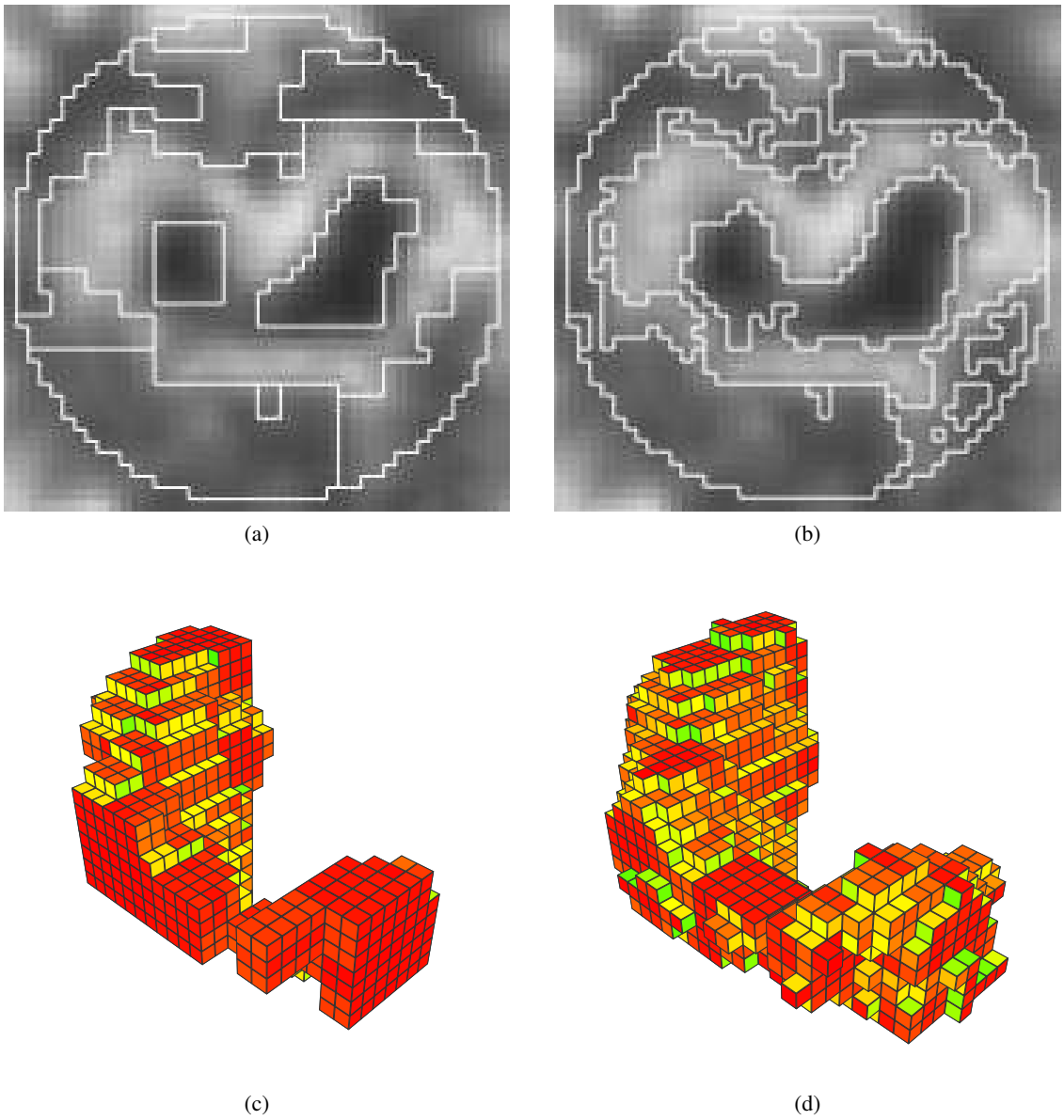
(a) (b) (c)

FIG. 6.12: Raffinement d'une partition sur une image artificielle comportant trois régions. (a) Coupe de l'image avec représentation de la partition initiale. (b) Deux surfaces dans la partition initiale. (c) Les mêmes surfaces après déformation de la partition. Nous observons que les deux surfaces semblent mieux correspondre aux données. Cependant, des artefacts liés à la partition initiale sont encore visibles (autour des arêtes entre les faces représentées par un trait noir).

plus sombre est bien composée d'une seule composante connexe de voxels lorsque son volume est considéré en 3D. Toutes les surfaces entre les régions sont déformées pour mieux correspondre aux données de l'image. Nous avons constaté que le résultat de la segmentation produisait des régions ayant un aspect bloc qui ne correspond pas aux données de l'image. Nous appliquons donc l'opération de déformation pour modifier la géométrie de la partition afin que la forme des régions corresponde mieux aux données de l'image. Les Figure 6.13c et Figure 6.13d montrent la surface de la région hachurée avant et après l'opération de déformation. L'algorithme de déformation a basculé 8594 voxels durant 321 passes de traitement. L'énergie de chaque face de la partition est alors minimale. Nous observons bien un lissage de la région qui correspond plus aux données de l'image comme dans l'expérience précédente. Il reste quelques incohérences au niveau des bords des surfaces puisqu'il n'y a pas de déformation autour des arêtes et autour des sommets dans l'image.

## 6.7 Conclusion

Dans ce chapitre, nous avons introduit la notion de points ML-Simples qui sont des voxels pouvant basculer d'une région à une autre sans changer la topologie de la partition. Les adjacences et imbrications entre les régions sont préservées. Nous utilisons les points ML-Simples pour déformer géométriquement la partition tout en préservant sa topologie. Afin de guider le processus de déformation, nous utilisons la minimisation d'une énergie basée sur les données de l'image et une autre énergie basée sur un estimateur d'aire discret pour les surfaces entre les régions. L'étude de cet estimateur d'aire montre des résultats encourageants qui sont à poursuivre dans des travaux ultérieurs. Nous avons montré quelques exemples de déformation de partitions d'une image médicale pour valider nos résultats. Dans la partie suivante, nous définissons des opérations de segmentation d'images 3D en prenant en compte des informations topologiques. Dans le Chapitre 7, nous adaptons un algorithme existant dans le contexte des cartes topologiques. Il s'agit de proposer une première approche pour la segmentation afin de montrer que les cartes topologiques permettent de réaliser des opérations de traitement d'images.



(a) (b)

(c) (d)

FIG. 6.13: Raffinement d'une partition sur une image médicale. (a) Coupe de l'image représentant la partition initiale. (b) Partition après déformation : nous observons que les formes sont globalement moins carrées. (c) Surface de la région hachurée avant la déformation : la forme est très cubique. Cette région contient 1293 voxels et sa surface est composée de 1192 surfels. (d) Surface de la région hachurée après la déformation : la forme est plus lisse. La région résultante est plus grosse, elle contient 1970 voxels et sa surface comprend 1716 surfels.

# Troisième partie

# Segmentation d'Images

CHAPITRE 7

# PREMIÈRE APPROCHE PAR CROISSANCE DE RÉGIONS

La première approche de segmentation que nous avons mise en œuvre est une adaptation d'un algorithme proposé par P. F. Felzenszwalb et D. P. Huttenlocher [FH98, FH04]. Elle est fondée sur une segmentation par croissance de régions qui utilise un critère basé sur la notion de contraste d'une région.

Le choix de cette méthode repose sur deux arguments. D'une part, il s'agit d'un algorithme fondé sur une représentation discrète, sous forme de graphe, de la partition de l'image et utilisant une information topologique : l'adjacence. La représentation par carte topologique donnant plus d'informations que les graphes, nous pouvons envisager de mettre en œuvre le même algorithme. D'autre part, l'adaptation de la méthode originale utilise seulement l'opération de fusion de régions, la seule opération de modification disponible lorsque nous avons commencé ce travail.

Dans cette section, nous commençons par présenter la méthode originale et le critère utilisé afin de définir la partition de l'image. Nous détaillons ensuite l'algorithme utilisé pour réaliser cette segmentation dans une partition représentée par une carte topologique. Nous justifions que notre méthode est bien équivalente à l'approche originale puis nous étudions la complexité de l'opération. Finalement nous évaluons au travers d'expériences les temps de calcul de l'opération de segmentation et les résultats que nous obtenons.

## 7.1 Segmentation par critère de contraste

La méthode décrite dans [FH98, FH04] donne une solution au problème de la segmentation d'une image en régions. Le critère utilisé mesure l'existence d'une frontière entre deux régions en utilisant une représentation de l'image par un graphe.

### 7.1.1 Représentation par graphe

Nous commençons par détailler la représentation par graphe d'une image que les auteurs utilisent dans leur méthode. Soit $G = (S, A)$ un graphe représentant une image, les sommets $s_i \in S$ sont les éléments (pixels) de l'image à segmenter, et les arêtes $(s_i, s_j) \in A$ correspondent aux paires de sommets voisins. À chaque arête $(s_i, s_j)$ nous associons un poids $w((s_i, s_j))$ qui est une mesure non négative de dissimilarité entre les éléments voisins $s_i$ et $s_j$.

Dans l'approche par graphe de la segmentation, une segmentation $R$ est une partition de $S$ en composantes de sorte que chaque composante (ou région) $r \in R$ correspond à une composante connexe dans un graphe $G' = (S, A')$ où $A' \subseteq A$. En d'autres termes, une segmentation est induite par un sous-ensemble des arêtes de $A$. De manière générale, une segmentation est bonne si les éléments dans une composante sont semblables et les éléments dans différentes composantes ne sont pas semblables. L'idée est donc que les arêtes d'une même composante connexe ont un poids relativement faible et les arêtes entre les composantes ont un poids plus élevé.

### 7.1.2 Critère de comparaison de régions

Nous définissons à présent la fonction déterminant si une frontière existe entre deux régions, c'est-à-dire si l'union des deux régions n'est pas homogène. Le prédicat utilisé mesure la dissimilarité entre les éléments qui sont de chaque côté d'une frontière entre deux régions. Le critère est basé sur la notion de contraste qui utilise plusieurs mesures sur et entre les régions.

Nous associons à chaque région $r$ une valeur, *Int*$(r)$, appelée *contraste interne* qui est le poids de l'arête de coût maximal dans l'arbre couvrant de poids minimal *MST*$(r, A)$ de la région $r$ (*Minimum Spanning Tree* en anglais) :

$$\textit{Int}(r) = \max_{a \in \textit{MST}(r,A)} w(a).$$

À chaque paire de régions $(r_1, r_2) \in S^2$ nous associons une valeur appelée *contraste externe*, notée *Ext*$(r_1, r_2)$, qui est le poids de l'arête de coût minimum qui relie les deux régions $r_1$ et $r_2$ :

$$\textit{Ext}(r_1, r_2) = \min_{s_i \in r_1, s_j \in r_2, (s_i, s_j) \in A} w((s_i, s_j)).$$

S'il n'y a pas d'arête qui relie $r_1$ et $r_2$, *Ext*$(r_1, r_2) = \infty$. À partir de ces deux valeurs, le prédicat qui vérifie l'existence d'une frontière entre deux régions $r_1$ et $r_2$ est défini en comparant les contrastes internes des deux régions avec le contraste externe entre les deux régions. Le prédicat est vrai lorsqu'il existe une frontière entre les régions, et faux sinon : si le plus petit des contrastes internes est supérieur au contraste externe entre les deux régions alors les régions peuvent fusionner :

$$\textit{Oracle}(r_1, r_2) = \begin{cases} vrai & \text{si } \textit{Ext}(r_1, r_2) > \textit{MInt}(r_1, r_2) \\ faux & \text{sinon} \end{cases}$$

*MInt*$(r_1, r_2)$ est défini comme le plus petit contraste interne des régions $r_1$ et $r_2$. Cette mesure est pondérée par une fonction $\tau$ :

$$\textit{MInt}(r_1, r_2) = \min(\textit{Int}(r_1) + \tau(r_1), \textit{Int}(r_2) + \tau(r_2)).$$

Le rôle de $\tau$ est de mitiger l'estimation du contraste interne lorsque celui-ci n'est pas représentatif comme par exemple pour les petites régions. Les auteurs suggèrent d'utiliser une fonction $\tau(r) = k/|r|$ avec $|r|$ le nombre de pixels de $r$ et $k$ une constante qui définit l'échelle de segmentation : plus $k$ est grand, plus les régions obtenues lors de la segmentation seront grandes. N'importe quelle fonction définie positive peut être utilisée pour $\tau$.

### 7.1.3 Algorithme original

Afin de proposer un algorithme qui segmente une image, les auteurs définissent deux propriétés importantes permettant de juger qu'une partition est bonne en fonction d'un critère. Ce sont les notions de sursegmentation (Définition 24) et de sous-segmentation (Définition 25).

**Définition 24** (Sursegmentation)**.** *Une segmentation $R$ est qualifiée de sursegmentation s'il existe une paire de régions $(r_1, r_2) \in R^2$ telle que* Oracle$(r_1, r_2)$ *soit faux.*

**Définition 25** (Sous-segmentation)**.** *Une segmentation $R$ est qualifiée de sous-segmentation s'il existe un raffinement de $R$ qui ne soit pas sursegmenté.*

Les auteurs établissent à partir de ces deux définitions le fait qu'il existe toujours une segmentation qui ne soit ni sursegmentée, ni sous-segmentée : une telle segmentation est qualifiée d'optimale. Ils proposent donc l'Algorithme 31 qui segmente une image de manière à obtenir une partition optimale. Cet algorithme met à jour le modèle de sorte que le contraste interne est actualisé de manière incrémentale lors des fusions de régions. Il prend en entrée un graphe et retourne une segmentation optimale de $S$.

**Algorithme 31** : Algorithme original de segmentation par contraste

**Données** : Un graphe $G = (S, A)$ avec $n$ sommets et $m$ arêtes.
**Résultat** : Segmentation de $S$ en composantes connexes $V = (C_1, \ldots, C_k)$.

trier $A$ par ordre croissant de poids dans $\pi = (a_1, \ldots, a_m)$;
$V_0 \leftarrow$ segmentation où chaque sommet $s_i$ est une composante distincte;
**pour** $q$=$1, \ldots, m$ **faire**
    $a_q = (s_i, s_j)$;
    **si** $s_i$ *et* $s_j$ *sont dans des composantes disjointes de* $V_{q-1}$ **alors**
        `// évaluation du critère de segmentation`
        **si** $w(a_q)$ *est petit comparé à la différence interne entre les deux composantes* **alors**
            fusionner les deux composantes;
**retourner** $V = V_m$;

Les arêtes du graphe sont tout d'abord triées par ordre de poids croissant. Puis la première partition est créée où chaque sommet est dans une composante séparée. Les arêtes sont ensuite traitées dans l'ordre croissant. Pour chaque arête $a_q$ entre deux composantes distinctes $r_1$ et $r_2$ l'inégalité $w(a_q) \leq$ *MInt*$(r_1, r_2)$ est vérifiée. Si ce critère est vrai, nous fusionnons les composantes et sinon nous ne faisons rien et nous passons à l'arête suivante.

En organisant les composantes de sommets à l'aide d'une forêt d'ensembles disjoints, la complexité de l'algorithme est donnée en $O(m)$.

## 7.2 Adaptation aux cartes topologiques

L'expression du critère dans le graphe utilise la notion d'arbre couvrant de poids minimum qui n'a pas d'équivalent direct dans les cartes topologiques. De plus, l'adjacence entre deux régions dans le graphe n'est toujours considérée que par une seule arête, l'arête de poids minimum entre deux voxels appartenant aux deux régions. Cette définition de l'adjacence ne prend donc pas en compte la multi-adjacence alors que la carte topologique le fait.

Aussi, afin d'adapter l'algorithme de segmentation dans le cadre des cartes topologiques, nous convertissons tout d'abord l'expression des valeurs de contraste et du critère de segmentation dans le cadre des cartes topologiques. Nous étudions ensuite l'algorithme et nous montrons l'équivalence de cette approche par rapport à la méthode originale.

### 7.2.1 Expression du contraste dans une carte topologique

L'absence de l'arbre couvrant de poids minimum dans les régions de la carte topologique interdit d'utiliser la même définition que dans le graphe pour le contraste interne. Cependant, les auteurs dans

la méthode des graphes proposent une définition incrémentale de cette valeur. En effet, la valeur du contraste interne vaut zéro lorsqu'il n'y a qu'un voxel. Lorsque deux régions fusionnent, la valeur du contraste interne de la région résultante est égale au contraste externe entre les deux régions. Nous débutons le processus avec une partition comprenant une région pour chaque voxel. Par fusions des régions, et mises à jour successives, nous obtenons une partition avec des valeurs de contraste interne équivalentes à la méthode utilisant les graphes.

Nous montrons maintenant comment la définition du contraste externe entre deux régions est transposée dans le modèle des cartes topologiques. Ainsi nous définissons la notion de contraste externe $Ext(f)$ pour une face $f$ d'une carte topologique comme la plus petite différence d'intensité entre deux voxels voisins se trouvant de part et d'autre de la face $f$ :

$$Ext(f) = \min_{s \in surfel(f)} w(voxel_1(s), voxel_2(s)).$$

Le contraste externe entre les régions se définit ensuite grâce au contraste externe sur les faces. Soient deux régions $r_i$ et $r_j$ adjacentes en $k$ faces $f_p$, $p \in [1, k]$ (les faces $f_p$ séparent la région $r_i$ de la région $r_j$). Le contraste externe $Ext(r_i, r_j)$ est égal au minimum des contrastes externes des faces $f_p$, $Ext(r_i, r_j) = min\{Ext(f_p), p \in [1, k]\}$.

Dans un souci de performance, nous stockons les contrastes internes et externes dans les cellules de la carte topologique. Le contraste externe est stocké sur chaque face de la carte et le contraste interne sur chaque région. Ces valeurs sont calculées durant l'extraction de la carte topologique et chaque opération appliquée sur la carte topologique met à jour les changements de contraste. Ainsi pour chaque région et chaque face de la carte topologique, nous obtenons les contrastes correspondants en temps constant (car chaque brin de la carte topologique connaît sa région incidente et sa face incidente).

### 7.2.2 Algorithme de segmentation

Le processus de segmentation consiste à fusionner les régions entre lesquelles il n'y a pas de frontière d'après le critère de contraste. Pour réaliser la fusion, nous disposons des deux approches présentées dans le Chapitre 4. La segmentation est un processus automatique qui réalise beaucoup de fusions de régions dans différentes composantes connexes. Nous utilisons donc l'approche globale de la fusion qui est la plus adaptée pour ces traitements (voir Section 4.3). Nous rappelons que cette approche est réalisée en deux étapes, avec d'une part la fusion symbolique des régions (à haut-niveau) puis la fusion effective (à bas-niveau).

La mise en œuvre de la segmentation par critère de contraste utilise exactement l'algorithme de fusion globale. Lors de la fusion symbolique, nous modifions le critère de fusion des régions pour utiliser le prédicat basé sur les valeurs de contraste. La fusion effective est ensuite utilisée afin de transcrire la partition définie sur les régions dans les différentes structures de la carte topologique.

L'Algorithme 32 présente la segmentation d'une image par critère de contraste à l'aide d'une carte topologique. Il prend en paramètre une carte topologique $M$ représentant une sursegmentation d'une image et modifie la carte afin que la partition représentée soit optimale.

Dans la première partie de l'algorithme, nous construisons la forêt d'ensembles disjoints représentant la partition de l'image conformément au prédicat. Comme pour la fusion globale, à l'initialisation de l'algorithme, chaque région est dans un ensemble distinct.

Nous commençons par créer une liste de brins $L$ composée d'un brin par face, et triée selon la valeur du contraste externe (Algorithme 33). Nous utilisons une marque sur les brins qui indique si la face incidente au brin est déjà dans la liste ou pas. Pour construire $L$, nous parcourons tous les brins de la carte topologique. Lorsqu'un brin $b$ non traité est découvert, nous marquons comme traités tous les brins appartenant à la même face (tous les brins de $\langle \beta_1, \beta_3 \rangle(b)$). Nous insérons le brin $b$ dans la liste. Elle contient donc un brin par face de la carte topologique. Nous utilisons alors un algorithme de tri

**Algorithme 32** : Segmentation par critère de contraste

**Données** : Une carte topologique $M$.
**Résultat** : $M$ représente la segmentation optimale de l'image.

$L \leftarrow$ liste triée des faces;
**tant que** $L$ *n'est pas vide* **faire**
  $b \leftarrow$ défiler $L$;
  $f \leftarrow$ *face*$(b)$;
  $r_1 \leftarrow$ *region*$(b)$ ; $r_2 \leftarrow$ *region*$(\beta_3(b))$;
  **si** $r_1 \neq r_2$ **alors**
    **si** Ext$(f) \leq$ MInt$(r_1, r_2)$ **alors**
      $r \leftarrow$ fusionner symboliquement $r_1$ et $r_2$;
      $Int(r) \leftarrow$ *Ext*$(f)$;
fusion effective des régions dans la carte topologique;

classique (tri rapide) afin d'ordonner de manière croissante la liste des brins selon le contraste externe de la face incidente à chaque brin $b$, donné par *Ext*(*face*$(b)$).

**Algorithme 33** : Construction de la liste triée des faces

**Données** : Une carte topologique $M$.
**Résultat** : Liste $L$ contenant un brin par face et triée dans l'ordre croissant de contraste.

$L \leftarrow$ liste de brins vide;
$m_{traite} \leftarrow$ marque sur les brins;
**pour chaque** *brin* $b \in M$ **faire**
  **si** $b$ *n'est pas marqué par* $m_{traite}$ **alors**
    ajouter $b$ à la liste $L$;
    marquer les brins de $\langle \beta_1, \beta_2 \rangle(b)$ avec $m_{traite}$;
trier $L$ par ordre croissant de valeur de *Ext*(*face*$(b)$);
**retourner** $L$;

Ensuite, nous traitons successivement tous les brins de la liste en commençant par celui dont le contraste externe de la face incidente est le plus petit. Ainsi pour chaque brin $b$, nous commençons par récupérer la face $f$ incidente ainsi que les deux régions adjacentes $r_1 =$ *region*$(b)$ et $r_2 =$ *region*$(\beta_3(b))$ séparées par $f$. En cas de multiadjacence, il est possible que ces deux régions soient déjà fusionnées dans le même ensemble disjoint (lorsque *trouver*$(r_1)$ est égal à *trouver*$(r_2)$). Dans ce cas, nous ne traitons pas la face correspondante qui est supprimée de toute manière lors de la fusion effective. Autrement, nous calculons la valeur de *MInt*$(r_1, r_2)$ et nous la comparons à la valeur du contraste externe *Ext*$(f)$ associée à la face $f$. Si la valeur du contraste est inférieure ou égale au minimum des contrastes internes des deux régions, il n'y a pas de frontière entre les deux régions : nous fusionnons les ensembles disjoints contenant les régions $r_1$ et $r_2$. La racine de l'ensemble disjoint résultant que nous appelons $r$ reçoit alors la nouvelle valeur du contraste interne pour l'union des deux régions. Il s'agit là de la mise à jour incrémentale de cette mesure qui est identique à celle utilisée par les auteurs dans la méthode originale.

Lorsque toutes les faces sont traitées, l'algorithme de segmentation utilise l'opération de fusion globale pour obtenir la carte topologique de la partition représentée par les ensembles disjoints.

### 7.2.3 Justification du résultat

La différence principale entre l'algorithme de segmentation original basé sur une représentation par graphe de l'image et l'algorithme adapté aux cartes topologiques tient à l'utilisation du contraste externe sur chaque face au lieu du contraste externe pour chaque couple de régions. Cette modification permet de prendre en compte la représentation de la multiadjacence par les cartes topologiques, une information qui n'est pas disponible dans l'approche originale. Nous montrons donc l'équivalence des deux approches afin de justifier notre algorithme.

La Proposition 6 assure que deux régions d'une carte topologique sont fusionnées si et seulement si elles respectent la condition établie sur le modèle à base de graphes représentant l'image. Pour cela, nous montrons que deux régions ne peuvent fusionner que lorsque nous étudions le contraste externe minimum entre les deux régions.

**Proposition 6.** *Si deux régions $r_i$ et $r_j$ ne fusionnent pas lorsque l'algorithme traite la première face qui sépare les deux régions, alors elles ne fusionnent jamais.*

*Démonstration.* Si $r_1$ et $r_2$ sont simplement adjacentes alors si elles ne fusionnent pas durant l'examen de la seule face les séparant, elles ne fusionnent jamais.

Considérons maintenant deux régions $r_1$ et $r_2$ d'une carte topologique telles que $r_1$ et $r_2$ soient multiadjacentes. Soit $f_p$, $p \in [1, k]$ et $k > 1$ les différentes faces séparant $r_1$ et $r_2$. Nous supposons que $\mathit{Ext}(f_i) \leq \mathit{Ext}(f_j) \forall i < j$. Nous savons que $\mathit{Ext}(r_1, r_2) = \min\{\mathit{Ext}(f_p), p \in [1, k]\}$, le contraste externe entre deux régions est égal au minimum des contrastes externes des faces $f_p$ : $\mathit{Ext}(r_1, r_2) = \mathit{Ext}(f_1)$.

L'Algorithme 32 traite les faces incidentes à $r_1$ et $r_2$ dans l'ordre croissant de variation externe. La première face considérée est donc $f_1$.

Supposons que les deux régions $r_1$ et $r_2$ ne fusionnent pas lors du traitement de la première face. Alors $\mathit{Ext}(f_1) > \mathit{MInt}(r_1, r_2)$. Avec cette hypothèse, nous montrons qu'il n'est pas possible que $r_1$ et $r_2$ fusionnent lorsque nous traitons les autres faces $f_q$, $q \in ]1, k]$. Il y a trois cas à étudier :

1. la valeur de $\mathit{MInt}(r_1, r_2)$ diminue. Ce cas est impossible car la valeur de $\mathit{MInt}(r_1, r_2)$ ne peut qu'augmenter dans l'Algorithme 32 : la fonction $Int$ est croissante et la fonction $\mathit{MInt}(r_1, r_2)$ dépend de $\mathit{Int}(r_1)$ et de $\mathit{Int}(r_2)$ et de $\tau$ qui est une fonction à valeur positive ;
2. la valeur de $\mathit{MInt}(r_1, r_2)$ n'est pas modifiée. Le traitement des autres faces $f_q$, $q \in ]1, k]$ ne permet pas de fusionner les deux régions puisque $\mathit{Ext}(f_q) > \mathit{Ext}(f_1) > \mathit{MInt}(r_1, r_2)$, $q \in ]1, k]$ ;
3. la valeur de $\mathit{MInt}(r_1, r_2)$ augmente entre le traitement de deux faces $f_i$ et $f_j$ ($i \geq 1$ et $i < j$). Nous montrons que cette modification ne change pas le résultat.

Supposons que $\mathit{Int}(r_1) + \tau(r_1) < \mathit{Int}(r_2) + \tau(r_2)$ et donc $\mathit{MInt}(r_1, r_2) = \mathit{Int}(r_1) + \tau(r_1)$ (cette supposition se fait sans perte de généralité, en renommant éventuellement les régions). Ainsi la valeur de $\mathit{MInt}(r_1, r_2)$ augmente si et seulement si la valeur de $\mathit{Int}(r_1) + \tau(r_1)$ augmente. Cette configuration n'est possible que si $r_1$ fusionne avec au moins une autre de ses régions voisines que nous nommons $r_3$ (autrement la valeur $\mathit{Int}(r_1) + \tau(r_1)$ reste constante).

Pour fusionner $r_1$ et $r_3$ après avoir traité $f_1$, nous avons besoin d'une face $g$ séparant $r_1$ et $r_3$ telle que $\mathit{Ext}(f_1) \leq \mathit{Ext}(g)$ et d'après la définition du minimum du contraste interne de deux régions, $\mathit{MInt}(r_1, r_3) \leq \mathit{Int}(r_1) + \tau(r_1)$. Notre hypothèse de départ est que $\mathit{Ext}(f_1) > \mathit{Int}(r_1) + \tau(r_1)$. En combinant les diverses comparaisons, nous obtenons $\mathit{Ext}(g) \geq \mathit{Ext}(f_1) > \mathit{Int}(r_1) + \tau(r_1) \geq \mathit{MInt}(r_1, r_3)$. Ainsi $\mathit{Ext}(g) > \mathit{MInt}(r_1, r_3)$ : cela contredit le fait que $r_1$ et $r_3$ fusionnent et donc que $\mathit{Int}(r_1) + \tau(r_1)$ augmente. Cette propriété montre que $r_1$ ne fusionne plus durant l'algorithme. Intuitivement, comme les faces sont triées et traitées dans l'ordre, si $r_1$ peut fusionner c'est uniquement avec la région $r_2$. Ainsi $\mathit{MInt}(r_1, r_2)$ n'augmente pas et chaque face $f_q$, $q \in [2, k]$, $\mathit{Ext}(f_q) > \mathit{MInt}(r_1, r_2)$. Les régions $r_1$ et $r_2$ ne fusionnent jamais. □

Ainsi la Proposition 6 est vérifiée, l'Algorithme 32 est équivalent à l'algorithme proposé dans la méthode originale : le fait de considérer toutes les faces et non pas uniquement celles de contraste minimum ne change pas le résultat final.

En cas d'égalité du contraste entre deux faces, c'est l'ordre de parcours qui détermine l'ordre des fusions possibles. C'est exactement le même comportement que pour l'algorithme original.

## 7.3 Complexité de l'algorithme

La complexité de l'algorithme dépend de la complexité de l'approche globale de la fusion et de la complexité d'évaluation du critère de segmentation. Nous étudions donc l'application de la méthode sur une carte topologique $M$ représentant une surpartition de l'image. Nous appelons $\mathit{faces}(M)$ l'ensemble des faces de la carte. Par construction $|\mathit{brins}(M)| \geq |\mathit{faces}(M)|$ car il y a au moins deux brins pour chaque face.

L'initialisation de la liste des faces nécessite le parcours de tous les brins. Les opérations de marquage et de vérification de marques sont des opérations réalisées en temps constant. Nous effectuons $|\mathit{faces}(M)|$ insertions dans la liste puis nous trions celle-ci. Cette partie a donc une complexité en $O(|\mathit{brins}(M)| + |\mathit{faces}(M)| \log(|\mathit{faces}(M)|))$. Les opérations sur la représentation des ensembles disjoints sont considérées comme des opérations réalisées en temps constant (voir Section 1.1.4).

Chaque face est traitée itérativement. L'extraction d'un élément d'une liste est réalisée en temps constant, et l'accès à la face (ou la région) d'un brin est également réalisé en temps constant. Avec l'hypothèse sur les ensembles disjoints, vérifier si deux régions sont déjà fusionnées se fait en temps constant. Enfin les calculs pour résoudre les contraintes de segmentation sont également réalisés en temps constant puisque les valeurs sont stockées et mises à jour incrémentalement. Ainsi cette partie est effectuée en $O(|\mathit{brins}(M)|)$.

La dernière étape est l'application de la mise à jour de la carte topologique par l'opération de fusion globale. Nous utilisons la complexité de l'opération donnée Section 4.3.5 : $O(|\mathit{brins}(M)| + |\mathit{surfels}(M)|)$.

La complexité totale de l'opération s'exprime donc

$$O(|\mathit{brins}(M)| + |\mathit{surfels}(M)| + |\mathit{faces}(M)| \log(|\mathit{faces}(M)|)).$$

L'opération dépend du nombre total de brins, du nombre de faces et du nombre de surfels de la partition.

## 7.4 Application et analyse

### 7.4.1 Analyse temporelle

Afin d'évaluer les performances de l'algorithme, nous avons mis en œuvre une application permettant de segmenter une image 3D représentée par une carte topologique. Nous avons mesuré le temps de calcul pour les différentes parties de l'algorithme sur trois images médicales provenant d'IRM et reporté les résultats dans la Table 7.1. Pour diminuer l'espace mémoire utilisé par la représentation initiale de l'image, nous présegmentons l'image durant l'extraction de la carte topologique en utilisant l'Algorithme 7 avec une fonction oracle qui autorise les régions à avoir une amplitude de niveaux de gris définie par un seuil $p$. Ainsi la différence entre l'intensité maximum $c_{max}$ et l'intensité minimum $c_{min}$ des voxels d'une région est donnée par $c_{max} - c_{min} \leq p$.

Les résultats sur ces trois images réelles montrent que la partie la plus rapide de l'opération de segmentation est l'opération de fusion symbolique. Cette opération est rapide car l'évaluation du critère et la fusion des régions dans la forêt d'ensembles disjoints sont des opérations dont le coût est faible.

TAB. 7.1: Informations sur l'image et temps d'exécution des trois parties de l'algorithme de segmentation sur trois images médicales présegmentées avec le seuil $p = 3$. Le paramètre de la fonction $\tau$ est $k = 5000$. Les images sont composées de coupes 2D de taille $256 \times 256$.

| Image | Img1 | Img2 | Img3 |
|---|---|---|---|
| Nombre de coupes | 44 | 111 | 124 |
| Nb de régions initiales | 147924 | 431486 | 310421 |
| Nb de régions restantes | 10121 | 30179 | 22523 |
| Initialisation liste (s) | 1,34 | 4,28 | 3,00 |
| (dont tri) | 0,76 | 2,68 | 1,77 |
| Fusion symbolique (s) | 0,30 | 0,97 | 0,67 |
| Fusion effective (s) | 5,10 | 13,44 | 10,00 |
| Total (s) | 6,75 | 18,70 | 13,68 |

TAB. 7.2: Informations sur la partition et temps d'exécution des trois parties de l'algorithme de segmentation en fonction du seuil de présegmentation $p$. L'image segmentée est l'image 2 de la Table 7.1 et le paramètre de la fonction $\tau$ est $k = 5000$. Nous donnons également l'espace mémoire (en Gigaoctets) occupé par l'image complète avant segmentation.

| $p$ | 0 | 1 | 2 | 3 | 4 | 5 | 10 |
|---|---|---|---|---|---|---|---|
| Espace mémoire (Go) | $> 4$ | 2,8 | 2,4 | 2,0 | 1,8 | 1,6 | 1,0 |
| Nb de régions initiales | - | 765841 | 555261 | 431486 | 350065 | 292342 | 151443 |
| Nb de régions restantes | - | 37136 | 32674 | 30179 | 27873 | 25537 | 18435 |
| Initialisation liste (s) | - | 6,60 | 5,31 | 4,28 | 3,61 | 3,16 | 1,72 |
| (dont tri ) | - | 4,38 | 3,48 | 2,68 | 2,23 | 1,91 | 0,91 |
| Fusion symbolique (s) | - | 1,52 | 1,18 | 0,97 | 0,81 | 0,69 | 0,36 |
| Fusion effective (s) | - | 16,35 | 14,40 | 13,44 | 11,99 | 11,14 | 8,42 |
| Total (s) | - | 24,48 | 20,90 | 18,70 | 16,42 | 15,00 | 10,49 |

L'initialisation de la liste des brins triée par ordre de contraste croissant est plus coûteuse puisqu'elle dépend du nombre de brins de la carte topologique. Plus il y a de régions, plus le nombre de brins est important et plus la création de la liste prend du temps. Nous observons par ailleurs que lors de la création de la liste, le tri des brins par ordre croissant de contraste externe est l'opération qui prend le plus de temps.

Enfin la partie la plus coûteuse de la segmentation est l'opération de fusion effective qui transforme la carte topologique afin qu'elle reflète la partition définie par la forêt d'ensembles disjoints. Les temps pour cette étape sont tout à fait semblables aux temps mesurés lors des expériences sur l'opération de fusion par approche globale.

Nous étudions maintenant l'impact du seuil de présegmentation $p$ sur le temps de calcul et sur le nombre de régions de la partition résultante. Dans la Table 7.2 nous reportons les mesures réalisées dans le cadre de cette expérience. Nous utilisons toujours le même paramètre $k = 5000$ pour la fonction $\tau$ mais nous faisons varier le seuil de présegmentation $p$ durant cette expérience.

L'intérêt de la présegmentation est avant tout de diminuer l'espace mémoire utilisé pour représenter la partition initiale de l'image. Nous observons que les petites valeurs de $p$ donnent un gain très impor-

(a) (b) (c)

FIG. 7.1: Exemple de segmentation d'une image 2D (Lena) à l'aide d'un critère de contraste. Le seuil de segmentation $k$ est fixé à 500 dans les deux applications. (a) Image originale ($256 \times 256$). (b) Segmentation à l'aide de l'algorithme original sur les graphes (4-connexité). (c) Segmentation à l'aide de l'algorithme adapté sur les cartes topologiques. Les deux partitions obtenues sont identiques.

tant en espace mémoire sans augmenter le temps d'extraction initial. Le temps de traitement diminue également mais pas de manière aussi importante que l'espace mémoire occupé. Enfin les nombres de régions initiales et finales sont plus petits au fur et à mesure que $p$ augmente mais la complexité de la partition reste dans les mêmes ordres de grandeur.

Nous utilisons donc la présegmentation afin de permettre la représentation dans la machine de la partition initiale. Mais étant donné le faible gain en terme de temps de traitement, nous essaierons dans la mesure du possible de conserver une valeur basse pour le paramètre $p$.

### 7.4.2 Comparaison avec le programme original

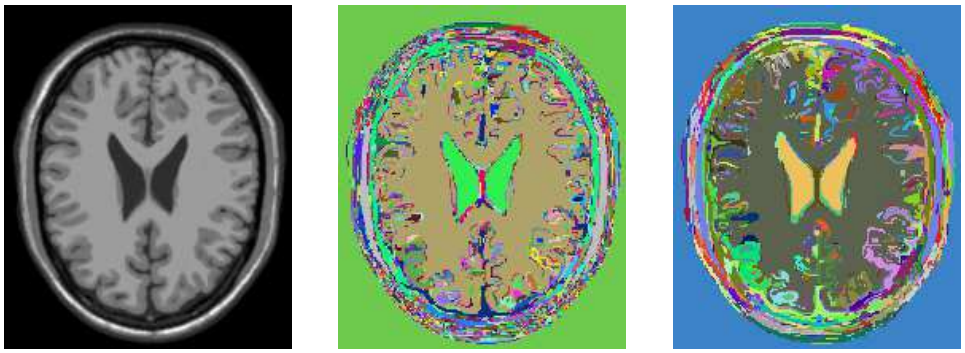
Nous avons à notre disposition le programme utilisé dans [FH98] pour réaliser la segmentation en régions d'images 2D en utilisant le critère de contraste. Nous avons légèrement modifié le code du programme afin que l'opération soit réalisée en considérant la $4$-connexité (au lieu de la $8$-connexité). Nous modifions également le critère utilisé afin que les deux processus utilisent bien la même évaluation du critère. Nous comparons alors les résultats entre l'application originale et notre algorithme dans les cartes combinatoires afin de montrer que les deux processus donnent des résultats égaux. Pour segmenter des images 2D à l'aide de notre programme, nous convertissons les images originales en images 3D ayant un voxel d'épaisseur.

La Figure 7.1 montre un exemple de segmentation obtenue en utilisant dans les deux algorithmes le même paramètre $k = 500$ pour la fonction $\tau$. Les deux segmentations produites sont identiques. Nous n'essayons pas de comparer les temps d'exécution des deux programmes. En effet, le programme sur les graphes utilise une structure *ad-hoc* adaptée à ce traitement et travaille uniquement sur les images 2D. Il est donc largement plus rapide que notre prototype qui lui traite des images 3D et est générique par rapport aux traitements que nous pouvons appliquer.

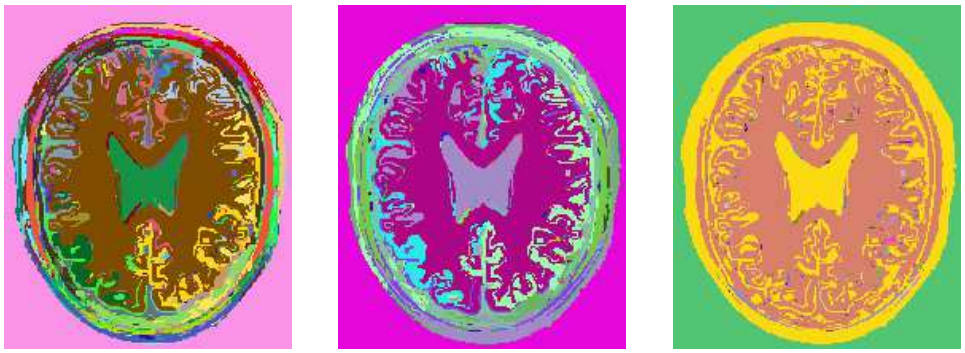
### 7.4.3 Segmentation d'images médicales

Nous avons ensuite testé notre algorithme pour segmenter des images médicales de cerveau acquises par l'imagerie à résonance magnétique. Ces images donnent la structure anatomique du cerveau. Nous présentons ici les résultats obtenus et discutons de leur qualité.

La Figure 7.2 présente une première série d'images représentant la même coupe d'une image 3D segmentée en faisant varier le paramètre $k$. La Table 7.3 donne les informations pour chacune de ces



(a) Originale (b) $k = 500$ (c) $k = 10000$



(d) $k = 50000$ (e) $k = 400000$ (f) $k = 2000000$

FIG. 7.2: Coupe de la segmentation d'une même image médicale de cerveau par IRM (de taille $181 \times 217 \times 181$) avec différentes valeurs pour le seuil $k$.

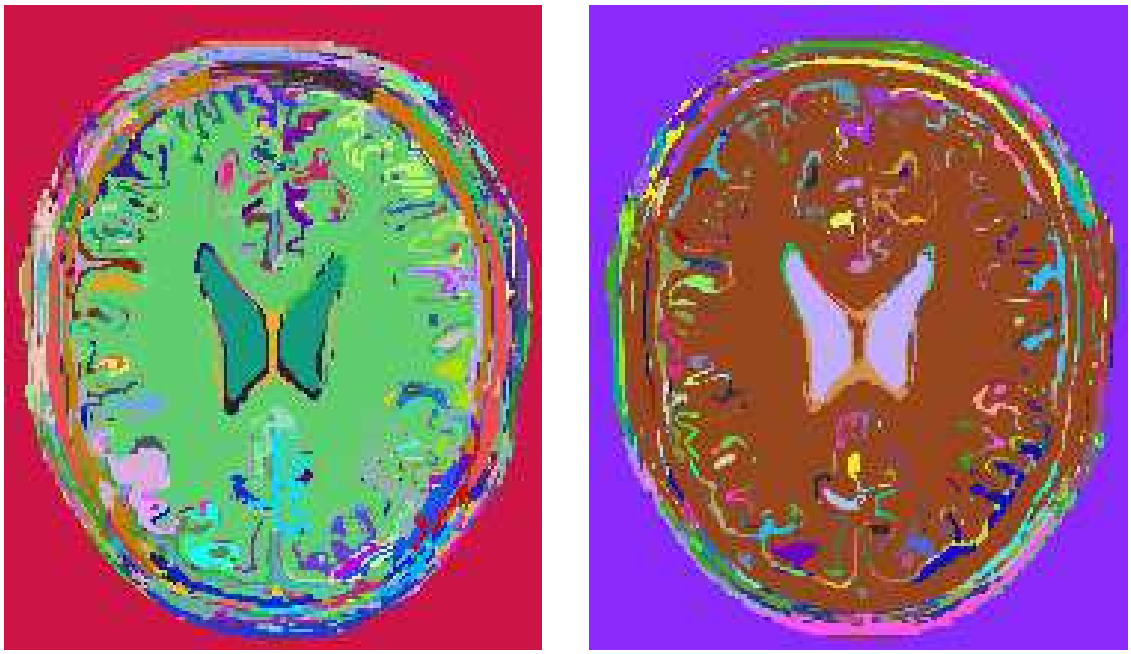
images. Nous observons la présence de nombreuses petites régions causée par le bruit de l'image même lorsque le seuil est défini avec des valeurs élevées. Nous observons également que lorsque le paramètre $k$ augmente, la taille moyenne des régions augmente également.

Afin de diminuer le bruit, les auteurs dans [FH98] utilisent un opérateur de lissage sur l'image afin de gommer les petites imperfections. La Figure 7.3 compare la segmentation produite par le critère sur la même coupe de l'image 3D originale et de l'image 3D filtrée par un filtre médian. Nous observons alors la disparition de toutes les petites régions pour laisser la place à des régions plus uniformes.

Ces expériences ont mis en évidence la capacité des cartes topologiques à être utilisées dans le contexte de l'analyse d'images. Cependant, le critère utilisé dans cette première approche de segmentation n'utilise pas beaucoup les informations portées par les cartes topologiques et se montre beaucoup plus lent que le modèle à base de graphe utilisé par les auteurs originaux. C'est pourquoi nous nous sommes intéressés à l'intégration d'un nouveau critère, prenant en compte des propriétés topologiques, qui montre alors pleinement le potentiel des cartes topologiques dans le traitement d'images.

TAB. 7.3: Informations sur les segmentation de l'image médicale avec différentes valeurs pour le seuil $k$.

| Valeur du seuil $k$ | 0 | 500 | 10000 | 50000 | 400000 | 2000000 |
|---|---|---|---|---|---|---|
| Nombre de régions | 6217089 | 226683 | 36448 | 31002 | 14942 | 10153 |
| Temps d'exécution (en secondes) | 0 | 20,89 | 19,72 | 19,90 | 19,99 | 20,97 |



(a) Sans lissage (b) Avec lissage

FIG. 7.3: Coupe de la segmentation d'une même image médicale de cerveau par IRM (de taille $181 \times 217 \times 181$) avec et sans lissage pour une même valeur de seuil $k = 10000$. (a) temps d'exécution 19,70 secondes, 241411 régions initiales, et 36448 régions résultantes. (b) temps d'exécution 17,06 secondes, 215359 régions initiales, et 29510 régions résultantes.

## 7.5 Conclusion

Dans ce chapitre, nous avons adapté un algorithme de segmentation existant au modèle des cartes topologiques. Le modèle original utilise comme critère une notion de contraste que nous avons modifié pour correspondre à notre modèle de représentation de la partition. À l'aide de cet algorithme, nous avons réalisé des segmentations d'images, d'abord en 2D afin de valider la méthode, puis en 3D sur des images médicales. Dans toutes ces opérations, l'adjacence, une information topologique, est utilisée pour déterminer les régions pouvant fusionner. Dans le Chapitre 8, nous proposons l'intégration d'un critère topologique basé sur les nombres de Betti d'une région. Ils permettent de caractériser les tunnels et les cavités de chaque région. Nous introduisons les algorithmes de calcul de ces invariants topologiques puis nous détaillons un algorithme de segmentation en prenant ces critères topologiques en considération.

CHAPITRE 8

# INTÉGRATION DE CRITÈRES TOPOLOGIQUES

La carte topologique permet de réaliser la segmentation d'images en utilisant un critère classique basé sur le critère de contraste. Cette première approche utilise déjà de nombreuses informations rendues disponibles par le modèle des cartes topologiques comme :

- les faces ;
- les régions ;
- l'adjacence entre les régions ;
- l'incidence entre les faces et les régions.

Notre but est maintenant d'étendre le critère de segmentation afin de prendre en compte de nouvelles informations, et proposer ainsi un nouveau critère utilisant la représentation complète de la partition par les cartes topologiques. Nous nous intéressons donc à la définition d'un critère topologique.

Nous cherchons à caractériser un invariant topologique sur les régions afin de nous en servir pour guider un processus de segmentation. Comme le rappelle la Section 3.4.2, la caractéristique d'Euler ne permet pas d'identifier les objets en dimension 3 comme les régions. Nous utilisons alors les nombres de Betti qui permettent de mettre en évidence certaines propriétés simples comme : est-ce qu'il y a des tunnels ou des cavités dans la région ?

Dans un premier temps, nous rappelons un travail existant permettant de calculer la caractéristique d'Euler du bord des régions dans une carte topologique. Dans un deuxième temps, en nous servant de ces résultats, nous détaillons le calcul des nombres de Betti d'une région représentée par une carte topologique. Dans cette partie, nous donnons les formules qui lient les nombres de Betti avec le nombre de cellules appartenant aux régions de la carte topologique. Dans un troisième temps, nous proposons une méthode incrémentale pour calculer les nombres de Betti au cours d'un processus de fusion de régions. Enfin nous détaillons la mise en œuvre d'un critère utilisant les nombres de Betti afin de segmenter une image. Nous étudions alors le temps de calcul nécessaire à cette opération et présentons quelques résultats.

## 8.1 Calcul de la caractéristique d'Euler du bord

Nous commençons par présenter l'algorithme de calcul de la caractéristique d'Euler pour les surfaces du bord des régions que nous trouvons dans [DPFL06].

La caractéristique d'Euler $\chi(r)$ de la région $r$ est pour nous la caractéristique d'Euler volumique. Elle se calcule avec le nombre de surfaces (bords du volume), de faces, d'arêtes et de sommets de la

région $r$. La caractéristique d'Euler du bord d'une région, que nous notons $\chi'(r)$, est une caractéristique d'Euler surfacique qui ne considère que les surfaces formant le bord de la région $r$ et se calcule ainsi avec le nombre de faces, d'arêtes et de sommets appartenant aux bords de la région. Les auteurs proposent de calculer $\chi'(r)$ pour chaque région $r$ en utilisant la Définition 18, page 54, sur chaque surface du bord.

Pour chaque surface $s$ appartenant au bord de la région, nous comptons le nombre de sommets, le nombre d'arêtes et le nombre de faces de cette surface. Nous utilisons ensuite la Définition 18, $\chi = \#s - \#a + \#f$, afin de calculer la caractéristique d'Euler de cette surface. La somme des caractéristiques d'Euler de toutes les surfaces d'une région est la caractéristique d'Euler du bord de la région :

$$\chi'(r) = \sum_{\mathfrak{s} \in \textit{surfaces}(r)} \chi(\mathfrak{s}).$$

Ainsi d'après la Définition 18, la caractéristique d'Euler du bord de la région $r$ est égale à

$$\begin{aligned}
\chi'(r) &= \sum_{\mathfrak{s} \in \textit{surfaces}(r)} \chi(\mathfrak{s}) \\
\chi'(r) &= \sum_{\mathfrak{s} \in \textit{surfaces}(r)} \#s(\mathfrak{s}) - \#a(\mathfrak{s}) + \#f(\mathfrak{s}) \\
\chi'(r) &= \#s(r) - \#a(r) + \#f(r),
\end{aligned}$$

où $\#s$ représente le nombre de sommets, $\#a$ le nombre d'arêtes et $\#f$ le nombre de faces de la surface ($\mathfrak{s}$) ou de la région ($r$). Pour compter les cellules, nous utilisons l'algorithme présenté dans la Section 8.1.1.

Les auteurs de la méthode proposent une méthode incrémentale de mise à jour lors de l'extraction de la carte topologique d'une image. Les nombres de sommets, d'arêtes et de faces sont stockés dans les régions. Ces nombres sont mis à jour lors des opérations d'insertion et de suppression de cellules pour toutes les régions incidentes.

Afin que cette mise à jour incrémentale fonctionne même après l'extraction, nous ajoutons à chaque opération de modification (suppression et insertion de cellules) définies Section 4.1 l'algorithme qui permet de tenir à jour le nombre de cellules appartenant au bord.

### 8.1.1 Compter les cellules

Afin de compter les différentes cellules incidentes à une région donnée, nous parcourons les brins de la carte appartenant à cette région en utilisant l'Algorithme 34. Afin de savoir si une cellule est déjà comptée, nous utilisons une marque sur les brins appartenant à cette cellule. Ainsi lorsque nous comptons les cellules appartenant au bord d'une région, nous utilisons 3 marques : une pour les sommets, une pour les arêtes et une pour les faces.

Nous parcourons l'ensemble des brins appartenant aux surfaces de la région en utilisant l'Algorithme 5, page 40, et pour chaque brin $b$, nous comptons les cellules auxquelles il appartient. Si le brin n'a pas la marque sur les sommets, alors le sommet incident n'est pas encore compté : nous marquons, avec la marque sommet, tous les brins de l'orbite $\langle \beta_{21}, \beta_{31} \rangle(b)$ et nous comptons un sommet de plus. Nous procédons de la même façon pour les arêtes et les faces en utilisant respectivement l'orbite $\langle \beta_2, \beta_3 \rangle(b)$ et l'orbite $\langle \beta_1, \beta_3 \rangle(b)$. Ainsi lorsque tous les brins sont traités, ils portent tous les trois marques : toutes les cellules du bord de la région sont comptées.

La complexité de l'algorithme est linéaire en fonction du nombre de brins, $|\textit{brins}(r)|$, de la région $r$. Chaque brin de la région n'est parcouru qu'une fois par le parcours de la surface de $r$ et une seule

**Algorithme 34** : Compter les cellules du bord d'une région

**Données** : Une carte topologique $M$ ;
Une région $r$.
**Résultat** : $(s, a, f)$ donnant respectivement le nombre de sommets, arêtes et faces du bord de la région $r$.

$m_{sommet}, m_{arete}, m_{face} \leftarrow$ marques sur les brins;
$(s, a, f) \leftarrow (0, 0, 0)$;
**pour chaque** *brin* $b \in$ brins$(r)$ **faire**
  **si** $b$ *n'est pas marqué par* $m_{sommet}$ **alors**
    marquer $\langle \beta_{21}, \beta_{31} \rangle(b)$ avec $m_{sommet}$;
    $s \leftarrow s + 1$;
  **si** $b$ *n'est pas marqué par* $m_{arete}$ **alors**
    marquer $\langle \beta_2, \beta_3 \rangle(b)$ avec $m_{arete}$;
    $a \leftarrow a + 1$;
  **si** $b$ *n'est pas marqué par* $m_{face}$ **alors**
    marquer $\langle \beta_1, \beta_3 \rangle(b)$ avec $m_{face}$;
    $f \leftarrow f + 1$;
**retourner** $(s, a, f)$;

fois également par chaque parcours comptant les cellules. De plus, l'orbite d'une cellule contient un nombre limité de brins : par la géométrie, il n'y a pas plus de quatre surfaces autour d'une arête, et six arêtes autour d'un sommet. Le nombre de brins parcourus dépend donc du nombre de cellules distinctes. Comme le nombre de cellules distinctes ne peut excéder le nombre de brins de la région $r$, la complexité de l'algorithme s'exprime en $O(|\mathit{brins}(r)|)$.

## 8.2 Calcul des nombres de Betti d'une région

Dans un premier temps, nous nous intéressons à une méthode permettant de calculer les nombres de Betti pour les régions de la carte topologique de manière non incrémentale.

Différents travaux proposent des méthodes de calcul pour obtenir les nombres de Betti dans des complexes cellulaires. Nous pouvons citer notamment [DE95] qui propose une méthode incrémentale pour les complexes simpliciaux. Nous n'utilisons pas les résultats de ces travaux car les cartes topologiques représentent des complexes cellulaires quelconques et pas simplement composés de simplexes. Les algorithmes proposés ne sont donc pas applicables tels quels. D'autres approches plus générales permettant de calculer les nombres de Betti sont détaillées dans [Mun84].

Une manière pour obtenir les nombres de Betti consiste à calculer les générateurs des groupes d'homologie afin de les compter et ainsi obtenir le rang des groupes d'homologie qui sont les nombres de Betti. Cependant, le calcul des générateurs est une tâche complexe et comme seul le rang de ces générateurs est nécessaire, nous proposons d'utiliser la définition intuitive des nombres de Betti afin de les calculer. Notre approche vise donc à compter le nombre de composantes connexes, de tunnels et de cavités pour les régions de la carte topologique.

Dans la suite nous présentons donc les différentes formules et méthodes permettant de retrouver les nombres de Betti à partir des informations contenues dans la carte topologique. Nous commençons par présenter le calcul des premier et troisième nombres de Betti. En effet, leurs valeurs sont ensuite utilisées dans le calcul du deuxième nombre de Betti.

### 8.2.1 Premier nombre de Betti

Le premier nombre de Betti, $\mathfrak{b}_0$, compte le nombre de composantes connexes d'un objet. D'après la définition des cartes topologiques, une région est un ensemble 6-connexe de voxels. Le premier nombre de Betti d'une région $r$ dans une carte topologique $M$, noté $\mathfrak{b}_0(r)$, est donc toujours égal à un : $\forall r \in M, \mathfrak{b}_0(r) = 1$.

### 8.2.2 Troisième nombre de Betti

Le troisième nombre de Betti, $\mathfrak{b}_2$, compte le nombre de cavités d'une région. Dans le modèle des cartes topologiques, l'arbre des régions représente la relation d'imbrication entre les régions. Pour chaque composante 18-connexe de régions imbriquées dans une région $r$, il existe une cavité. L'arbre d'imbrication de la carte topologique groupe toutes les régions d'une même composante 18-connexe, et $\mathit{fils}(r)$ est l'ensemble des plus petites régions de chaque composante, il contient donc une région par cavité : ainsi $\mathfrak{b}_2(r) = |\mathit{fils}(r)|$.

### 8.2.3 Deuxième nombre de Betti

Le deuxième nombre de Betti, $\mathfrak{b}_1$, compte le nombre de tunnels d'une région. Il n'existe pas de manière directe de compter les tunnels dans la carte topologique. Nous utilisons une méthode basée sur le calcul de la caractéristique d'Euler, puis nous utilisons le lien entre la caractéristique d'Euler et les nombres de Betti. Nous détaillons dans les sections suivantes la méthode employée.

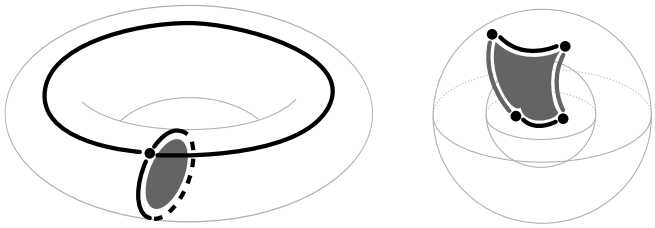
La Proposition 2 présente le lien entre la caractéristique d'Euler d'un objet 3D et les trois premiers nombres de Betti : $\chi = \mathfrak{b}_0 - \mathfrak{b}_1 + \mathfrak{b}_2$. Nous savons calculer $\mathfrak{b}_0$ et $\mathfrak{b}_2$ pour une région, nous cherchons donc à calculer la caractéristique d'Euler $\chi$ de cette région. Pour cela, nous utilisons les travaux sur le calcul de la caractéristique d'Euler du bord d'une région, rappelés Section 8.1 . Dans un premier temps nous montrons le lien entre $\chi$ et $\chi'$. Puis nous détaillons le calcul de $\mathfrak{b}_1(r)$ à partir de $\chi'(r)$, $\mathfrak{b}_0(r)$ et $\mathfrak{b}_2(r)$.

Le calcul de la caractéristique d'Euler suppose que l'objet soit constitué d'un ensemble de $i$-cellules homéomorphes à des $i$-boules. Cependant, la carte topologique représente les régions par leurs bords. Les sommets, les arêtes et les faces sont homéomorphes respectivement à des 0-, 1- et 2-boules. Cependant le volume de chaque région n'est pas forcément homéomorphe à des 3-boules. Nous définissons la notion de *cellules implicites* pour une région représentée par une carte topologique qui permet de modéliser une région sous forme d'une boule 3D. Le nombre de cellules implicites dépend du nombre de tunnels et du nombre de cavités comme le montre la Définition 26. Les cellules implicites permettent d'obtenir le nombre correct de cellules d'une région, utilisable pour le calcul de la caractéristique d'Euler.

**Définition 26.** *Les cellules implicites sont définies selon les deux règles suivantes pour obtenir un volume homéomorphe à une 3-boule :*

1. *pour chaque tunnel, une face implicite est ajoutée (voir Figure 8.1a) ;*
2. *pour chaque cavité, deux arêtes implicites et une face implicite sont ajoutées (voir Figure 8.1b). La face est bordée par quatre arêtes, deux sont incidentes à des surfaces de la région et les deux nouvelles arêtes forment alors le bord de la nouvelle face.*

La Figure 8.1 présente deux exemples montrant les cellules implicites. Dans la Figure 8.1a, la région contient un tunnel. La face implicite dessinée en gris rend l'objet homéomorphe à une 3-boule et ainsi complète le complexe cellulaire représentant la région. Dans la Figure 8.1b, la région contient une cavité. D'après la Définition 26, nous devons ajouter deux arêtes et une face pour que le volume de



(a) (b)

FIG. 8.1: Exemple de cellules implicites (dessinées en gris foncé) pour des régions qui ne sont pas homéomorphes à une 3-boule. (a) Une face implicite dans une région comportant un tunnel. La face permet de couper le tunnel en deux et ainsi rend la région homéomorphe à une 3-boule. (b) Deux arêtes implicites et une face implicite dans une région comportant une cavité. L'ajout de la face permet de connecter la cavité avec l'extérieur et ainsi rend la région homéomorphe à une 3-boule.

la région soit homéomorphe à une 3-boule. Les deux arêtes et la face dessinées en gris représentent les cellules implicites.

Nous nous intéressons à présent aux rapports entre $\chi$ et $\chi'$. La Proposition 7 établit le lien, pour une région $r$ représentée dans une carte topologique, entre la caractéristique d'Euler $\chi(r)$ et la caractéristique d'Euler de son bord $\chi'(r)$.

**Proposition 7.** *Pour une région $r$ représentée dans une carte topologique, $\chi(r) = \chi'(r)/2$.*

*Démonstration.* Soit $k_i$ le nombre de cellules de dimension $i$ dans le complexe cellulaire représentant une région $r$. Par la relation entre le nombre de cellules et la caractéristique d'Euler (Définition 19), nous avons $\chi(r) = k_0 - k_1 + k_2 - k_3$. Soient $\#s(r)$, $\#a(r)$ et $\#f(r)$ le nombre de sommets, arêtes et faces appartenant aux bords de la région $r$, par définition la caractéristique d'Euler du bord des régions s'exprime comme $\chi'(r) = \#s(r) - \#a(r) + \#f(r)$. Par définition des cellules implicites (Définition 26) le nombre $k_i$ de $i$-cellules est défini comme $k_0 = \#s(r)$, $k_1 = \#a(r) + 2\mathfrak{b}_2(r)$ (deux arêtes sont ajoutées pour chaque cavité) et $k_2 = \#f(r) + \mathfrak{b}_1(r) + \mathfrak{b}_2(r)$ (une face implicite est ajoutée pour chaque tunnel et une face implicite est ajoutée pour chaque cavité). Le nombre de volumes, $k_3$, est égal au nombre, $\mathfrak{b}_0(r)$, de composantes connexes de la région puisqu'il y a un volume pour chaque composante connexe : $k_3 = \mathfrak{b}_0(r) = 1$. La caractéristique d'Euler est donnée par :

$$\begin{aligned}
\chi(r) &= k_0 - k_1 + k_2 - k_3 \\
\chi(r) &= \#s(r) - (\#a(r) + 2\mathfrak{b}_2(r)) + (\#f(r) + \mathfrak{b}_1(r) + \mathfrak{b}_2(r)) - \mathfrak{b}_0(r) \\
\chi(r) &= \#s(r) - \#a(r) + \#f(r) - \mathfrak{b}_0(r) + \mathfrak{b}_1(r) - \mathfrak{b}_2(r) \\
\chi(r) &= \chi'(r) - \chi(r).
\end{aligned}$$

avec comme conséquence immédiate : $\chi(r) = \chi'(r)/2$. □

Maintenant que la Proposition 7 est introduite, $\mathfrak{b}_1(r)$ est calculé d'après la caractéristique d'Euler du bord de $r$ en utilisant la formule donnée par la Proposition 8.

**Proposition 8.** *Le deuxième nombre de Betti $\mathfrak{b}_1(r)$ d'une région $r$ est donné par $\mathfrak{b}_1(r) = \mathfrak{b}_0(r) + \mathfrak{b}_2(r) - \chi'(r)/2$.*

*Démonstration.* D'après la Proposition 7 et la Proposition 2, nous avons :

$$\begin{aligned}
\chi(r) &= \chi'(r)/2 \\
\mathfrak{b}_0(r) - \mathfrak{b}_1(r) + \mathfrak{b}_2(r) &= \chi'(r)/2 \\
\mathfrak{b}_1(r) &= \mathfrak{b}_0(r) + \mathfrak{b}_2(r) - \chi'(r)/2
\end{aligned}$$

□

La Section 8.1 nous donne l'algorithme de calcul de $\chi'(r)$, et les Section 8.2.1 et Section 8.2.2 nous permettent de calculer $\mathfrak{b}_0(r)$ et $\mathfrak{b}_2(r)$. Nous sommes donc capables de calculer $\mathfrak{b}_1(r)$ en utilisant la formule de la Proposition 8.

## 8.3 Méthodes incrémentales pour le calcul des nombres de Betti

Le calcul par la méthode classique, présentée dans la Section 8.2, des nombres de Betti est une opération coûteuse puisqu'il est nécessaire de parcourir l'ensemble des brins appartenant aux cellules du bord d'une région afin de les compter, notamment dans le cas du calcul de $\mathfrak{b}_1$. Dans un processus de segmentation par croissance de régions, lorsque nous réalisons différentes fusions entre les régions, la mise à jour des nombres de Betti devient vite trop coûteuse en temps. Nous proposons donc une approche incrémentale pour le calcul des nombres de Betti. L'idée est d'utiliser des valeurs (comme le nombre de cellules) calculées incrémentalement lors de la fusion de deux régions afin d'obtenir les nombres de Betti correspondants.

Le premier nombre de Betti $\mathfrak{b}_0(r)$ d'une région $r$ est constant quelle que soit la région $r$ car le nombre de composantes connexes ne change pas. Il n'y a pas besoin d'approche incrémentale pour $\mathfrak{b}_0$. Ensuite pour calculer le troisième nombre de Betti $\mathfrak{b}_2(r)$, la méthode normale utilise le nombre de cavités. Ainsi nous calculons incrémentalement le changement du nombre de cavités durant la fusion de deux régions. Enfin pour le calcul du deuxième nombre de Betti $\mathfrak{b}_1(r)$, nous utilisons le calcul incrémental de la caractéristique d'Euler du bord de la région [DPFL06] afin d'utiliser la Proposition 8 définie lors du calcul classique en n'utilisant que des valeurs calculées incrémentalement.

### 8.3.1 Calcul incrémental de $\mathfrak{b}_2$

Le calcul incrémental du deuxième nombre de Betti consiste à trouver les changements dans le nombre de cavités lors de la fusion de deux régions $r_1$ et $r_2$. Nous supposons dans cette partie que la région $r_1$ est plus petite que la région $r_2$ dans l'ordre des régions. D'après le Lemme 1, la région $r_1$ ne peut pas être imbriquée dans la région $r_2$. Ainsi trois configurations sont envisageables :

1. $r_2$ remplit une cavité de $r_1$ ;
2. la fusion de $r_1$ et $r_2$ entoure d'autres régions ou change le nombre de composantes connexes de régions imbriquées ;
3. la fusion de $r_1$ et $r_2$ ne change pas le nombre de cavités.

Dans le premier cas, la fusion des deux régions entraîne la suppression d'une cavité. Comme les régions n'ont qu'une composante connexe de voxels, la région $r_2$ ne peut pas remplir plus d'une cavité de $r_1$. Dans le second cas, il y a création d'au moins une cavité pour chaque composante 18-connexe de régions nouvellement imbriquées. Dans le troisième cas, il n'y a pas de création ni de suppression de cavités.

Le calcul incrémental utilise le lien entre le nombre de surfaces du bord d'une région et le nombre de cavités de cette région. La Proposition 9 donne la relation entre $\mathfrak{b}_2(r)$ et le nombre de surfaces de la région $r$. Le calcul incrémental de $\mathfrak{b}_2$ utilise le nouveau nombre de surfaces en considérant que les deux régions sont fusionnées.

**Proposition 9.** *Soit* $|\text{surfaces}(r)|$ *le nombre de surfaces de la région* $r$*. Le nombre de cavités de* $r$ *est donné par* $\mathfrak{b}_2(r) = |\text{surfaces}(r)| - 1$.

*Démonstration.* Il existe deux types de surfaces pour les régions, les surfaces externes qui forment les bords extérieurs de la région et les surfaces internes qui forment les bords intérieurs de la région correspondant aux cavités. D'après la définition de la carte topologique, les régions n'ont qu'une composante

connexe. Comme il y a une surface externe par composante connexe, le nombre de surfaces internes est donné par $|\mathit{surfaces}(r)| - 1$. Il y a une cavité par surface interne, ainsi nous obtenons bien le troisième nombre de Betti à l'aide de la formule $\mathfrak{b}_2(r) = |\mathit{surfaces}(r)| - 1$. □

Le problème du calcul incrémental de $\mathfrak{b}_2$ pour l'union de deux régions $r_1$ et $r_2$ est donc ramené au problème du calcul du nombre de surfaces de $r_1 \cup r_2$. La Proposition 10 introduit une relation entre le nombre de surfaces de $r_1$ et de $r_2$ avec le nombre de surfaces de $r_1 \cup r_2$.

**Proposition 10.** *Soient $r_1$ et $r_2$ deux régions 6-adjacentes telles que $r_1 < r_2$, nous appelons respectivement $|\text{surfaces}(r_1)|$ et $|\text{surfaces}(r_2)|$ le nombre de surfaces de $r_1$ et le nombre de surfaces de $r_2$. Le nombre de surfaces de l'union de $r_1$ et $r_2$ est donné par :*

$$|\text{surfaces}(r_1 \cup r_2)| = |\text{surfaces}(r_1)| + |\text{surfaces}(r_2)| + k - 2,$$

*où $k$ est le nombre de surfaces contenant un morceau de la surface externe de $r_2$ et qui ne se trouvent pas entre $r_1$ et $r_2$.*

*Démonstration.* Comme $r_1$ et $r_2$ sont adjacentes, il y a au moins une surface de $r_1$ qui est en contact avec une surface de $r_2$. De plus, $r_1 < r_2$, ainsi nous sommes dans l'une des trois configurations présentées Section 8.3.1. Nous savons ainsi que dans chaque configuration, la surface externe de $r_2$ est en contact avec une surface (la surface externe ou une surface interne) de $r_1$.

Nous calculons le nombre $k$ de surfaces disjointes contenant un morceau de la surface externe de $r_2$ sans traverser des faces intérieures (qui séparent $r_1$ et $r_2$). Si la région $r_2$ bouche une cavité, les faces séparant $r_1$ et $r_2$ sont intérieures, $k = 0$. Le nombre de surfaces de $r_1 \cup r_2$ diminue de 2, la surface externe de $r_2$ et la surface interne de $r_1$ disparaissant. S'il reste des surfaces ($k \neq 0$), alors le nombre de surfaces de l'union est la somme des nombres de surfaces de chaque région. Nous ajoutons à cette somme $k$, le nombre de nouvelles surfaces de l'union des deux régions. Cependant, $k$ compte également une face séparant $r_1$ et $r_2$, et comme cette face est déjà comptée, une fois dans le nombre de surfaces de $r_1$ et une fois dans le nombre de surfaces de $r_2$, nous retirons deux faces à la somme. □

Nous avons maintenant les formules reliant le troisième nombre de Betti avec le nombre de surfaces d'une région et reliant le nombre de surfaces de l'union de deux régions adjacentes avec le nombre initial de surfaces de chaque région et le nombre $k$. Nous cherchons donc à calculer $k$ en parcourant la surface extérieure de $r_2$ pour retrouver les composantes connexes de la surface délimitées par les faces intérieures (faces communes à $r_1$ et $r_2$). Nous commençons par définir une opération permettant de parcourir les régions en "sautant" les faces intérieures puis nous détaillons comment le calcul de $k$ est réalisé.

### 8.3.2 Parcours de surfaces en ignorant les faces intérieures

Pour compter les surfaces disjointes contenant un morceau de la surface externe de $r_2$, nous avons besoin d'une opération permettant de parcourir les brins d'une surface en "sautant" les faces intérieures à l'union de $r_1$ et de $r_2$.

L'Algorithme 35 présente la manière dont le parcours est réalisé. Il prend en paramètres une carte topologique, un brin de départ qui n'est pas intérieur et une fonction permettant de déterminer si un brin appartient à une face intérieure. Cette fonction doit respecter le fait que si un brin $b'$ appartient à une face intérieure, alors tous les brins de l'orbite $\langle \beta_1, \beta_3 \rangle(b')$ appartiennent également à une face intérieure.

Nous utilisons une pile de brins pour réaliser le parcours de la surface. La marque $m_{traite}$ indique si un brin a déjà été ajouté à la pile ou non.

**Algorithme 35** : Parcours de la surface en sautant les faces intérieures

**Données** : Une carte topologique $M$ ;
Un brin $b$ ;
Une fonction $estInterieur : D \rightarrow \{vrai, faux\}$.
**Résultat** : Parcours de la surface incidente à $b$ en ignorant les faces intérieures.

```
m_traite ← marque sur les brins;
P ← pile de brins vide;
marquer b avec m_traite;
ajouter b à P;
tant que P n'est pas vide faire
    b' ← tête de P;
    dépiler P;
    si β1(b') n'est pas marqué par m_traite alors
        marquer β1(b') avec m_traite;
        empiler β1(b') sur P;
    b2 ← β2(b');
    tant que estInterieur(b2) faire
        b2 ← β32(b2);
    si b2 n'est pas marqué par m_traite alors
        marquer b2 avec m_traite;
        empiler b2 sur P;
    // le brin b' appartient à la surface incidente à b
```

Nous commençons le traitement par le brin $b$ qui doit être tel que $estInterieur(b)$ soit $faux$. Nous le marquons et l'ajoutons à la pile pour initialiser le parcours.

Lorsqu'un brin $b'$ est traité, nous regardons son voisin par la relation $\beta_1$. Si celui-ci n'est pas marqué, nous le marquons et l'ajoutons à la pile $P$. Lorsque nous étudions le voisin $b_2$ de $b'$ par la relation $\beta_2$, il est possible que $b_2$ appartienne à une face intérieure. Nous sautons alors la face en utilisant $\beta_{32}(b_2)$ afin de passer sur la face suivante. Tant que $b_2$ appartient à une face intérieure, nous continuons de sauter la face. Lorsque nous arrivons sur un brin $b_2$ qui n'est pas intérieur (il y en a au moins un, $b'$), nous le marquons avec $m_{traite}$ et nous l'ajoutons à la pile si il n'est pas déjà marqué. Le traitement d'un brin $b'$ est illustré par la Figure 8.2.

Lorsque la pile est vide, tous les brins de la surface incidente à $b$ en ignorant les faces intérieures ont été parcourus.

### 8.3.3 Détection et comptage des faces de $r_1 \cup r_2$.

L'Algorithme 36 présente la méthode permettant de détecter et compter les surfaces en considérant l'union de deux régions comme une région unique. Il prend en paramètres une carte topologique $M$ et deux régions adjacentes $r_1$ et $r_2$ avec $r_1 < r_2$ et retourne le nombre de surfaces distinctes appartenant au bord de l'union de $r_1$ et $r_2$.

D'après l'hypothèse, $r_1 < r_2$ et $r_1$ est adjacente à $r_2$. Ainsi la surface externe de $r_2$ est la seule surface de $r_2$ en contact avec une surface de $r_1$. Comme nous ne pouvons savoir quelle surface de $r_1$ est en contact avec $r_2$, nous effectuons nos parcours à partir de la surface externe de $r_2$.

La première étape consiste à marquer les brins appartenant aux faces intérieures de $r_1 \cup r_2$. Pour cela, nous parcourons l'ensemble des brins de la surface externe de $r_2$ (ce sont tous les brins de l'orbite $\langle \beta_1, \beta_2 \rangle (rep(r_2))$) et pour chaque brin $b$ appartenant à une surface entre $r_1$ et $r_2$, nous marquons $b$

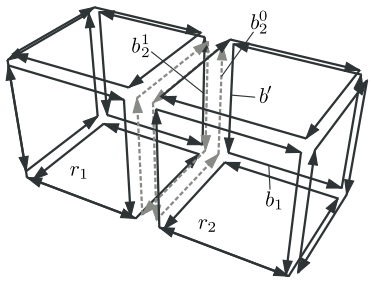



FIG. 8.2: Exemple du traitement d'un brin $b'$ lors du parcours d'une surface en ignorant les faces intérieures. Nous marquons et ajoutons le brin $b_1 = \beta_1(b')$ à la pile car il n'est pas encore marqué. En ce qui concerne le traitement de l'image par $\beta_2$ de $b'$, le premier brin $b_2^0 = \beta_2(b')$ est un brin appartenant à une face intérieure. Nous passons donc au brin $b_2^1 = \beta_{32}(b_2^0)$ qui n'appartient pas à une face intérieure. Il s'agit donc du brin que nous marquons et que nous ajoutons à la pile. Nous venons de "sauter" les brins de la face intérieure en passant dans l'autre volume.

**Algorithme 36** : Compte les surfaces de l'union de deux régions

**Données** : Une carte topologique $M$ ;
Deux régions $r_1$ et $r_2$ avec $r_1 < r_2$.
**Résultat** : Le nombre de surfaces de $r_1 \cup r_2$.

$k \leftarrow 0$;
$m_{inner}, m_{traite} \leftarrow$ marques sur les brins;
**pour chaque** *brin* $b \in \langle \beta_1, \beta_2 \rangle(\text{rep}(r_2))$ **faire**
  **si** region$(\beta_3(b)) = r_1$ **alors**
    marquer $b$ avec $m_{inner}$ et $m_{traite}$;
**pour chaque** *brin* $b \in \langle \beta_1, \beta_2 \rangle(\text{rep}(r2))$ **faire**
  **si** $b$ *n'est pas marqué par* $m_{traite}$ **alors**
    marquer avec $m_{traite}$ la surface incidente à $b$ en sautant les brins marqués par $m_{inner}$;
    $k \leftarrow k + 1$;
**retourner** $|\text{surfaces}(r_1)| + |\text{surfaces}(r_2)| + k - 2$;

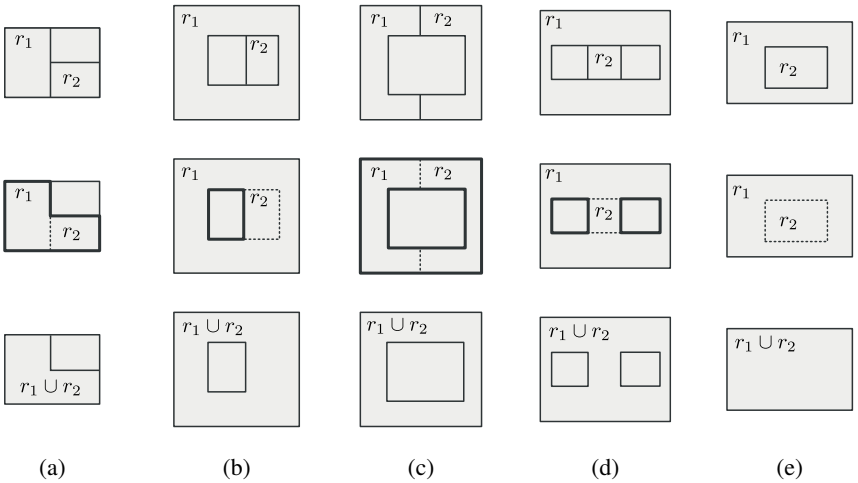



FIG. 8.3: Exemples de configurations entre deux régions $r_1$ et $r_2$. Sur la première ligne se trouve la configuration initiale. Sur la deuxième ligne nous représentons par des pointillés les surfaces intérieures à $r_1 \cup r_2$ et en gras les $k$ surfaces découvertes par l'Algorithme 36. La dernière ligne présente la configuration finale après la fusion de $r_1$ et de $r_2$. Les différents cas sont : (a) pas de changement d'imbrication ; (b) la cavité diminue mais ne disparaît pas ; (c) création d'une cavité ; (d) division d'une cavité existante en deux cavités ; (e) suppression d'une cavité.

comme intérieur avec la marque $m_{inner}$ et nous appliquons également la marque $m_{traite}$ pour ne plus s'en occuper durant le reste de l'algorithme.

La deuxième étape consiste à compter le nombre de surfaces distinctes contenant un morceau de la surface externe de $r_2$ en ignorant les faces intérieures. Ces surfaces sont parcourues en utilisant l'Algorithme 35 où le prédicat est intérieur est défini en utilisant la marque $m_{inner}$.

Enfin, nous pouvons retourner le nombre de surfaces de la région $r_1 \cup r_2$ qui est donné par le nombre de surfaces de $r_1$, le nombre de surfaces de $r_2$ et le nombre $k$ de surfaces détectées par l'algorithme, moins deux d'après la Proposition 10.

### 8.3.4 Calcul du nouveau nombre de cavités dans un processus incrémental.

Lors de l'initialisation, nous comptons à l'aide de l'arbre d'imbrication le nombre de cavités $\mathfrak{b}_2(r)$ pour chaque région $r$. Cela nous donne le nombre de surfaces $|\textit{surfaces}(r)| = \mathfrak{b}_2(r) + 1$ pour chaque région $r$ qui est stocké, et mis à jour de manière incrémentale durant l'opération de fusion.

Lorsque deux régions $r_1$ et $r_2$ sont fusionnées, avec $r_1 < r_2$, nous utilisons l'Algorithme 36 afin de compter le nombre de surfaces $|\textit{surfaces}(r_1 \cup r_2)|$ de $r_1 \cup r_2$. En utilisant la Proposition 9, $\mathfrak{b}_2(r_1 \cup r_2)$ est donné par $\mathfrak{b}_2(r_1 \cup r_2) = |\textit{surfaces}(r_1 \cup r_2)| - 1$.

La Figure 8.3 donne cinq exemples de cas envisageables lors de la fusion de deux régions $r_1$ et $r_2$. Nous illustrons ce fonctionnement en 2D pour des raisons de lisibilité, les cas étant identiques en 3D. Dans les deux premiers cas, le nombre de cavités ne change pas. Dans le troisième cas, il y a création d'une nouvelle cavité. Dans le quatrième exemple, une cavité existante est divisée en deux nouvelles cavités. Dans le cinquième exemple, une cavité disparaît.

### 8.3.5 Deuxième nombre de Betti $\mathfrak{b}_1$

D'après la Proposition 8, le deuxième nombre de Betti $\mathfrak{b}_1(r)$ se calcule en utilisant $\mathfrak{b}_0(r)$, $\mathfrak{b}_2(r)$ et la caractéristique d'Euler $\chi'(r)$ du bord de la région $r$. Ainsi pour calculer de manière incrémentale $\mathfrak{b}_1$,

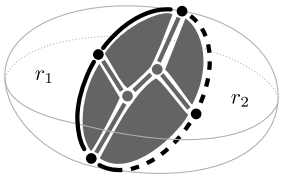



FIG. 8.4: Exemples de cellules intérieures et extérieures pour la région $r_1 \cup r_2$. Les cellules intérieures sont dessinées en gris foncé. Les cellules dessinées en noir forment le bord des faces intérieures et ne sont pas uniquement adjacentes à des cellules intérieures : ce sont des cellules extérieures.

nous calculons chaque partie de cette formule en utilisant une méthode incrémentale. Nous venons de donner l'algorithme incrémental permettant de calculer $\mathfrak{b}_2(r)$. $\mathfrak{b}_0(r)$ est constant pour toutes les régions représentées par une carte topologique. Il reste à étudier comment calculer de manière incrémentale la caractéristique d'Euler du bord $\chi'(r_1 \cup r_2)$ de l'union de deux régions $r_1$ et $r_2$.

Pour chaque couple $(r_1, r_2)$ de régions adjacentes, il existe au moins une face entre $r_1$ et $r_2$ : il y a autant de faces que d'adjacences entre deux régions. Nous appelons les faces présentes entre deux régions qui fusionnent des faces intérieures. Ces faces sont bordées de différentes cellules qui sont de deux types : les cellules intérieures et les cellules extérieures. Les cellules intérieures sont les faces intérieures et les cellules qui sont uniquement incidentes à des cellules intérieures. Dans une carte topologique, les cellules intérieures sont identifiées par le fait que tous les brins représentant la cellule (les brins de l'orbite) appartiennent à une face intérieure. Nous remarquons que les cellules qui forment le bord des faces intérieures ne sont généralement pas des cellules intérieures. La Figure 8.4 donne un exemple de cellules intérieures et extérieures.

Nous appelons *inner*$(r_1 \cup r_2)$ l'ensemble des cellules intérieures pour $r_1 \cup r_2$ et $\chi($*inner*$(r_1 \cup r_2))$ la caractéristique d'Euler des surfaces intérieures de $r_1 \cup r_2$, c'est-à-dire la somme du nombre de sommets intérieurs et de faces intérieures, soustrait au nombre d'arêtes intérieures. La Proposition 11 donne une définition incrémentale de $\chi'$ pour $r_1 \cup r_2$.

**Proposition 11.** $\chi'(r_1 \cup r_2) = \chi'(r_1) + \chi'(r_2) - 2\chi(\text{inner}(r_1 \cup r_2))$

*Démonstration.* Comme les cellules intérieures et extérieures sont définies pour l'union de deux régions, la notation $\#c_{out}(r_1 \cup r_2 | r_1)$ est introduite où $c$ est le type de la cellule ($s$ pour les sommets, $a$ pour les arêtes et $f$ pour les faces). Il s'agit du nombre de cellules extérieures de $r_1 \cup r_2$ restreint aux cellules appartenant à $r_1$. Avec cette notation, par décomposition du nombre de cellules entre les cellules intérieures ($in$) et cellules extérieures ($out$) en utilisant la formule donnée dans la Section 8.1, la valeur de $\chi'(r_1)$ est donnée par :

$$\begin{aligned}\chi'(r_1) =& \#s(r_1) - \#a(r_1) + \#f(r_1)\\ \chi'(r_1) =& + \#s_{in}(r_1 \cup r_2) + \#s_{out}(r_1 \cup r_2 | r_1) - \#a_{in}(r_1 \cup r_2)\\ & - \#a_{out}(r_1 \cup r_2 | r_1) + \#f_{in}(r_1 \cup r_2) + \#f_{out}(r_1 \cup r_2 | r_1).\end{aligned}$$

En utilisant le même procédé pour $r_2$, $\chi'(r_2)$ est défini par :

$$\begin{aligned}\chi'(r_2) =& \#s(r_2) - \#a(r_2) + \#f(r_2)\\ \chi'(r_2) =& + \#s_{in}(r_1 \cup r_2) + \#s_{out}(r_1 \cup r_2 | r_2) - \#a_{in}(r_1 \cup r_2)\\ & - \#a_{out}(r_1 \cup r_2 | r_2) + \#f_{in}(r_1 \cup r_2) + \#f_{out}(r_1 \cup r_2 | r_2).\end{aligned}$$

Le processus de fusion supprime les cellules intérieures pour obtenir la région résultante : $\chi'(r_1 \cup r_2)$ ne s'exprime qu'en fonction des cellules externes.

$$\begin{aligned}
\chi'(r_1 \cup r_2) =& + \#s_{out}(r_1 \cup r_2|r_1) + \#s_{out}(r_1 \cup r_2|r_2) - \#a_{out}(r_1 \cup r_2|r_1) \\
& - \#a_{out}(r_1 \cup r_2|r_2) + \#f_{out}(r_1 \cup r_2|r_1) + \#f_{out}(r_1 \cup r_2|r_2) \\
\chi'(r_1 \cup r_2) =& \chi'(r_1) + \chi'(r_2) - 2(\#s_{in}(r_1 \cup r_2) - \#a_{in}(r_1 \cup r_2) + \#f_{in}(r_1 \cup r_2)) \\
\chi'(r_1 \cup r_2) =& \chi'(r_1) + \chi'(r_2) - 2\chi(\textit{inner}(r_1 \cup r_2)).
\end{aligned}$$ □

Dans le calcul, nous comptons seulement les cellules qui sont incidentes aux faces intérieures. Cependant, lorsque nous faisons le calcul de $\chi'(r_1) + \chi'(r_2)$, les cellules externes incidentes aux faces intérieures (les cellules communes aux deux régions) sont comptées deux fois : une fois dans le calcul de $\chi'(r_1)$ et une autre fois dans le calcul de $\chi'(r_2)$. Cela ne pose pas de problèmes pour la justesse du calcul. En effet, les cellules incidentes aux faces intérieures sont les sommets et les arêtes appartenant au bord de ces faces. Il y a autant de sommets $s'$ que d'arêtes $a'$ parmi ces cellules. Comme nous utilisons la somme alternée du nombre de cellules pour calculer $\chi'$, $s' - a' = 0$, ces cellules n'apportent aucune contribution à la valeur de $\chi'(r_1)$ et $\chi'(r_2)$.

### 8.3.6 Calcul de la caractéristique d'Euler des cellules intérieures.

Pour calculer la caractéristique d'Euler des cellules intérieures à $r_1 \cup r_2$ en utilisant la formule donnée par la Définition 19, nous mettons en œuvre l'Algorithme 37 qui compte les cellules intérieures de l'union de deux régions. Il prend en paramètres une carte topologique $M$ et deux régions adjacentes $r_1$ et $r_2$ avec $r_1 < r_2$. Il compte le nombre de cellules intérieures et retourne la caractéristique d'Euler des cellules intérieures à $r_1 \cup r_2$.

**Algorithme 37** : Calcul de $\chi$ pour les cellules intérieures

**Données** : Une carte topologique $M$ ;
Deux régions $r_1$ et $r_2$ avec $r_1 < r_2$.
**Résultat** : La caractéristique d'Euler des cellules intérieures de $r_1 \cup r_2$.

$m_{inner}, m_{sommet}, m_{arete} \leftarrow$ marques sur les brins;
$(s, a, f) \leftarrow (0, 0, 0)$;
**pour chaque** *brin* $b \in \langle \beta_1, \beta_2 \rangle(\text{rep}(r_2))$ **faire**
  **si** region$(\beta_3(b)) = r_1$ *et* $b$ *n'est pas marqué par* $m_{inner}$ **alors**
    marquer $\langle \beta_1, \beta_3 \rangle(b)$ avec $m_{inner}$;
    $f \leftarrow f + 1$;
**pour chaque** *brin* $b \in \langle \beta_1, \beta_2 \rangle(\text{rep}(r_2))$ **faire**
  **si** $b$ *est marqué par* $m_{inner}$ **alors**
    **si** $b$ *n'est pas marqué par* $m_{sommet}$ **alors**
      marquer $\langle \beta_{21}, \beta_{31} \rangle(b)$ avec $m_{sommet}$;
      **si** *tous les brins de* $\langle \beta_{21}, \beta_{31} \rangle(b)$ *sont marqués avec* $m_{inner}$ **alors**
        $s \leftarrow s + 1$;
    **si** $b$ *n'est pas marqué par* $m_{arete}$ **alors**
      marquer $\langle \beta_2, \beta_3 \rangle(b)$ avec $m_{arete}$;
      **si** *tous les brins de* $\langle \beta_2, \beta_3 \rangle(b)$ *sont marqués avec* $m_{inner}$ **alors**
        $a \leftarrow a + 1$;
**retourner** $\chi(\text{inner}(r_1 \cup r_2)) = s - a + f$;

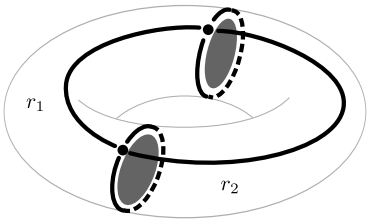



FIG. 8.5: Illustration du calcul du nouveau nombre de tunnels dans le cas de la fusion de deux régions. Dans l'exemple, la région $r_1$ et la région $r_2$ ne contiennent pas de tunnel : $\chi'(r_1) = 2$ et $\chi'(r_2) = 2$. L'algorithme de calcul de la caractéristique d'Euler des cellules intérieures détecte seulement deux faces (dessinées en gris), $\chi(\textit{inner}(r_1 \cup r_2)) = 2$. Ainsi la formule qui nous donne la nouvelle caractéristique d'Euler du bord de $r_1 \cup r_2$ s'exprime comme $\chi'(r_1 \cup r_2) = 2 + 2 - 4 = 0$. Il n'y a pas de cavité ainsi $\mathfrak{b}_1(r_1 \cup r_2) = 1$ : il y a un tunnel, la région résultante est un tore.

Dans un premier temps, nous identifions par la marque $m_{inner}$ les brins des faces intérieures de $r_1 \cup r_2$. Pour cela, nous parcourons l'ensemble des brins de la surface externe de $r_2$ et pour chaque brin qui se trouve entre $r_2$ et $r_1$, nous appliquons la marque $m_{inner}$ sur la face qui lui est incidente : comme il s'agit d'une face intérieure, nous la comptons. D'après l'ordre sur les régions, comme pour l'Algorithme 36, la configuration entre $r_1$ et $r_2$ implique que la surface externe de $r_2$ est en contact avec une surface (interne ou externe) de $r_1$. Les surfaces intérieures appartiennent donc à la surface externe de $r_2$.

Dans un deuxième temps, nous reprenons le parcours de la surface externe de $r_2$ afin de compter les sommets intérieurs et les arêtes intérieures. Lorsque nous traitons un brin $b$, si toute l'orbite de la cellule considérée est marquée par $m_{inner}$ alors nous comptons la cellule. Nous utilisons également une marque pour ne pas recompter la cellule qui contient le brin $b$. Le principe pour compter les différentes cellules est identique à celui utilisé dans l'Algorithme 34, la différence étant qu'il faut s'assurer que tous les brins de la cellule soient marqués comme intérieurs avant de compter la cellule.

L'algorithme ayant comptabilisé toutes les cellules intérieures ($s$, $a$ et $f$), le calcul $\chi(\textit{inner}(r_1 \cup r_2)) = s - a + f$ permet d'obtenir la caractéristique d'Euler associée à ses faces intérieures.

### 8.3.7 Calcul du nouveau nombre de tunnels dans un processus incrémental.

Lors de l'initialisation du processus incrémental, nous calculons pour chaque région le nombre de cellules appartenant à son bord (en utilisant l'Algorithme 34). Nous pouvons également utiliser l'algorithme proposé dans [DPFL06] qui permet de calculer incrémentalement le nombre de cellules durant l'extraction et que nous avons étendu aux opérations de suppression de cellules. Avec ces valeurs nous calculons la caractéristique d'Euler du bord $\chi'(r)$ de chaque région $r$. Cette valeur est alors stockée dans la région pour un accès en temps constant et nous mettons à jour cette valeur durant les fusions de régions.

Lorsque nous fusionnons deux régions $r_1$ et $r_2$, nous calculons à l'aide de l'Algorithme 37 la caractéristique d'Euler $\chi(\textit{inner}(r_1 \cup r_2))$ des cellules intérieures à $r_1 \cup r_2$. Avec cette valeur et les valeurs de $\chi'(r_1)$ et de $\chi'(r_2)$ stockées dans les régions, nous utilisons la Proposition 11 afin de calculer $\chi'(r_1 \cup r_2)$. Puis avec la valeur de $\mathfrak{b}_2$ qui est calculée incrémentalement, nous utilisons la formule de la Proposition 8 afin de calculer le nouveau nombre de tunnels $\mathfrak{b}_1(r_1 \cup r_2)$. La Figure 8.5 illustre un exemple de calcul du nombre de tunnels sur l'union de deux régions.

### 8.3.8 Synthèse des méthodes de calcul des nombres de Betti

Nous disposons à présent de deux approches pour calculer les nombres de Betti. D'une part, une approche classique qui utilise lorsque c'est possible la définition implicite (en nombre de composantes connexes et de cavités) des nombres de Betti ($\mathfrak{b}_0$ et $\mathfrak{b}_2$) pour les calculer. Nous utilisons alors la formule reliant la caractéristique d'Euler du bord d'une région aux nombres de Betti pour calculer $\mathfrak{b}_1$. D'autre part, nous disposons d'une approche incrémentale qui calcule les nombres de Betti pour l'union de deux régions adjacentes en utilisant des valeurs calculées sur chacune des régions.

Avec cette approche incrémentale, nous pouvons intégrer le calcul des nombres de Betti pour toutes les régions pendant l'opération de fusion de régions. Lorsque deux régions doivent fusionner, nous pouvons calculer les nombres de Betti de la région correspondante.

## 8.4 Utilisation dans le cadre de la segmentation

Afin de mettre en œuvre un critère basé sur les informations topologiques apportées par les nombres de Betti durant la segmentation, nous utilisons l'algorithme de calcul incrémental avant l'évaluation du critère décrit dans le Chapitre 7. L'idée de ce critère topologique est de contrôler le nombre de cavités et de tunnels que les régions peuvent contenir. Un exemple d'usage classique d'un tel critère se trouve dans la segmentation du cortex cérébral sur les images IRM. Un résultat connu en médecine dit que chaque hémisphère du cortex est homéomorphe à une sphère et ne contient donc ni tunnel ni cavité. Un critère basé sur les nombres de Betti permet d'obtenir ce résultat.

### 8.4.1 Initialisation du calcul incrémental

Afin d'initialiser le calcul incrémental dans le processus de segmentation, nous utilisons les algorithmes non incrémentaux de calcul de $\mathfrak{b}_1$ et $\mathfrak{b}_2$ afin de stocker les valeurs nécessaires aux calculs incrémentaux dans chaque région. Nous stockons également le nombre de sommets, d'arêtes et de faces pour chaque région. Ainsi les conditions de départ des algorithmes incrémentaux sont réunies, nous pouvons mettre en œuvre la segmentation.

### 8.4.2 Intégration du calcul à l'évaluation du critère

L'intégration du calcul des nombres de Betti dans le processus d'évaluation du critère de segmentation est réalisée de la façon suivante. Nous rappelons ici que l'évaluation du critère est réalisé durant la phase de fusion symbolique et c'est donc dans cette étape que nous intégrons le calcul des nombres de Betti et l'évaluation du critère topologique.

Lorsque nous étudions une face séparant une région $r_1$ d'une région $r_2$, la fusion symbolique est divisée en quatre étapes :

- évaluation d'un critère valeur (par exemple le critère de contraste) sur $r_1 \cup r_2$ ;
- si le critère valeur est vrai, calcul incrémental de $\mathfrak{b}_1(r_1 \cup r_2)$ et $\mathfrak{b}_2(r_1 \cup r_2)$ ;
- évaluation d'un critère dit topologique utilisant les valeurs de $\mathfrak{b}_1$ et de $\mathfrak{b}_2$ calculées ;
- si le critère topologique est vrai, fusion symbolique de $r_1$ avec $r_2$ et propagation des valeurs incrémentales nécessaires à l'algorithme de calcul des nombres de Betti.

Nous utilisons comme critère valeur le critère de contraste défini dans le Chapitre 7 afin de s'assurer que les régions soient fusionnées en zones homogènes. Nous réalisons ce test avant de calculer les nombres de Betti. En effet, l'évaluation du critère valeur est une opération très rapide alors que le calcul des nombres de Betti nécessite de nombreux parcours et est donc plus lent. Il n'est pas utile de calculer les nombres de Betti de l'union si les régions ne peuvent pas fusionner car leurs valeurs ne sont pas homogènes. Le prédicat définissant le critère topologique est basé sur les nombres de Betti. Il valide

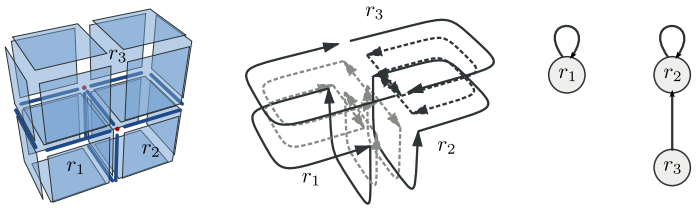



FIG. 8.6: Exemple du parcours de la surface externe de l'union de deux régions $r_1$ et $r_2$ où $r_2$ représente un ensemble disjoint de deux régions ($r_2$ et $r_3$). Les faces en pointillés gris sont les faces intérieures entre $r_1$ et $r_2$. Les faces en pointillés noirs sont les faces intérieures à l'ensemble disjoint représenté par $r_2$. Tous les brins en pointillés sont ignorés durant le parcours de surfaces : seuls les brins dessinés en trait plein sont traités.

la fusion si la topologie de la région résultante respecte les conditions données par l'utilisateur. Par exemple, nous pouvons refuser la fusion de deux régions si cela crée un nouveau tunnel ou si le nombre de cavités devient plus grand qu'un certain seuil. Lorsque la fusion est autorisée les ensembles disjoints représentant les deux régions sont fusionnés et les valeurs incrémentales des nombres de Betti et de la caractéristique d'Euler du bord sont mises à jour pour la région résultante.

### 8.4.3 Particularité de mise en œuvre

Dans le cadre de la fusion symbolique, les régions sont manipulées à l'aide d'une forêt d'ensembles disjoints. Au cours de l'algorithme, une région est en fait composée de l'ensemble des régions qui sont dans le même ensemble disjoint. Les faces intérieures aux ensembles disjoints, c'est-à-dire les faces qui séparent deux régions du même ensemble, n'ont pas encore été supprimées. Afin que les algorithmes de calcul présentés précédemment fonctionnent lorsqu'une région est représentée par un ensemble disjoint de régions, nous utilisons le parcours de surfaces présenté dans l'Algorithme 35. La fonction qui détermine si un brin $b$ appartient à une face intérieure est le prédicat qui est vrai lorsque les deux régions incidentes à la face contenant le brin $b$ sont dans le même ensemble disjoint :

$$\mathit{trouver}(\mathit{region}(b)) = \mathit{trouver}(\mathit{region}(\beta_3(b))).$$

Dans le calcul incrémental de $\mathfrak{b}_2$, nous utilisons déjà ce parcours en définissant une marque pour identifier les faces intérieures à l'union de deux régions $r_1$ et $r_2$. Lorsque $r_1$ et $r_2$ sont des ensembles disjoints, le test pour détecter les brins appartenant à une face intérieure est la disjonction du test sur la marque et du test sur l'égalité de la racine des arbres. La Figure 8.6 présente un exemple où $r_2$ est une région racine d'un ensemble disjoint de régions contenant deux régions.

## 8.5 Expérimentations

Nous expérimentons la segmentation utilisant un critère topologique basé sur les nombres de Betti sur des images réelles et artificielles afin d'étudier l'impact du prédicat d'homogénéité prenant en compte les informations topologiques en plus des informations de valeurs. Pour cela, nous réalisons tout d'abord des mesures sur le temps de calcul des nombres de Betti. Dans un premier temps, nous comparons la méthode classique avec la méthode incrémentale afin de montrer l'intérêt d'utiliser cette dernière dans une opération de segmentation. Dans un second temps, nous étudions les temps de calcul des nombres de Betti à l'intérieur de l'algorithme de segmentation engendré par le critère topologique. Dans cette expérience, nous regardons aussi l'impact du critère sur la topologie des régions obtenues.

Nous poursuivons cette expérience en travaillant avec des images artificielles pour vérifier que l'algorithme fonctionne et donne des résultats cohérents avec les paramètres que nous fournissons au critère.

Le critère utilisé dans ces expérimentations mélange le prédicat utilisant la notion de contraste avec un prédicat utilisant les nombres de Betti. Le prédicat choisi permet la fusion lorsque le nombre de tunnels et le nombre de cavités convergent vers des valeurs seuils données par l'utilisateur. Nous appelons $S_t$ (resp. $S_c$) le seuil donné pour le nombre de tunnels (resp. pour le nombre de cavités). Deux régions adjacentes peuvent fusionner d'après le prédicat topologique s'il n'y a ni création ni suppression de tunnels et de cavités ou si leurs nombres se rapprochent des valeurs de seuil. Par convention, un seuil négatif désactive le contrôle sur le nombre de Betti désigné. Nous travaillons ainsi de manière indépendante sur les tunnels ou sur les cavités, les valeurs des nombres de Betti étant tout le temps calculées incrémentalement.

Le critère topologique $C_{topo} : R \times R \rightarrow \{vrai, faux\}$ s'exprime de la façon suivante :

$$C_{topo}(r_1, r_2) = \begin{cases} vrai & \text{si } pred_t(r_1, r_2) \\ & \text{et } pred_c(r_1, r_2) \\ faux & \text{sinon} \end{cases}$$

avec

$$pred_t(r_1, r_2) : |\mathfrak{b}_1(r_1 \cup r_2) - S_t| \leq |\mathfrak{b}_1(r_1) + \mathfrak{b}_1(r_2) - S_t|$$
$$pred_c(r_1, r_2) : |\mathfrak{b}_2(r_1 \cup r_2) - S_c| \leq |\mathfrak{b}_2(r_1) + \mathfrak{b}_2(r_2) - S_c|.$$

Les prédicats $pred$ déterminent si les nombres de Betti ne divergent pas des seuils donnés par l'utilisateur.

### 8.5.1 Étude du temps d'exécution sur des images médicales

Dans cette section, les temps de calcul de l'opération de segmentation utilisant un critère topologique sont étudiés. Pour comparer les résultats, une image médicale de petite taille ($37 \times 44 \times 37$) est utilisée : elle représente une région d'intérêt dans une image médicale contenant 43198 régions. Sur cette image, une segmentation utilisant uniquement un critère couleur propose une partition en 1784 régions. Nous comparons les temps de calcul obtenus avec cette segmentation et la même segmentation avec un contrôle topologique sous la forme d'un critère de convergence vers des valeurs demandées par l'utilisateur.

Nous commençons par comparer l'approche classique du calcul des nombres de Betti avec l'approche incrémentale (Table 8.1). Dans cette comparaison, les nombres de Betti sont uniquement calculés, mais non pris en compte dans le critère de segmentation. Ensuite, deux exemples de segmentation avec différentes valeurs de convergence pour les critères sont utilisés pour illustrer l'augmentation du nombre de régions résultantes de la segmentation et l'augmentation du temps de calcul dans la partie décision de l'algorithme de segmentation.

Les deux premières lignes de la Table 8.1 donnent le nombre de régions avant et après la fusion. Les trois lignes suivantes indiquent le nombre de couples de régions adjacentes testées, le nombre de calcul des nombres de Betti et le nombre de fusions d'ensembles effectuées. Les quatre dernières lignes présentent les temps d'exécution des différentes parties de l'algorithme.

La colonne (1) présente les résultats sans calcul des nombres de Betti. Le nombre final de régions est donné par l'application de l'algorithme avec le critère valeur classique (critère de contraste). Le temps total d'exécution est d'à peu près deux secondes. La partie de fusion symbolique qui utilise le critère de segmentation est dans ce cas très rapide puisque le critère de contraste utilise des informations très rapides d'accès.

TAB. 8.1: Résultats des expériences : (1) sans calcul des nombres de Betti ; (2) calcul des nombres de Betti avec la méthode classique ; (3) calcul des nombres de Betti avec la méthode incrémentale ; (4) utilisation de seuils de convergence ($\mathfrak{b}_1 \rightarrow 0$ et $\mathfrak{b}_2 \rightarrow 0$) ; (5) utilisation de seuils de convergence ($\mathfrak{b}_1 \rightarrow 5$ and $\mathfrak{b}_2 \rightarrow 1$).

| Expérience | | (1) | (2) | (3) | (4) | (5) |
|---|---|---|---|---|---|---|
| nombre | Régions initiales | 43198 | | | | |
| | Régions finales | 1784 | | | 1938 | 2158 |
| | Fusion possibles | 80398 | | | 100886 | 91535 |
| | Calcul | 0 | 41414 | | 62429 | 54057 |
| | Fusion | 41414 | | | 41260 | 41040 |
| temps (s) | Initialisation | 1,05 | 1,05 | 1,05 | 1,05 | 1,05 |
| | Fusion symbolique | 0,07 | 2415,11 | 263,87 | 550,07 | 444,92 |
| | Fusion effective | 0,94 | 0,93 | 0,95 | 0,93 | 0,95 |
| | Total | 2,06 | 2417,09 | 265,88 | 552,05 | 446,93 |

Dans la colonne (2) se trouvent les résultats avec calcul des nombres de Betti en utilisant l'algorithme classique (non incrémental). Ainsi nous calculons $\mathfrak{b}_1$ et $\mathfrak{b}_2$ pendant chaque fusion symbolique de deux régions durant le processus de segmentation. Nous n'utilisons pas ces informations dans le critère. Ainsi la partition résultante est la même que lors de l'expérience proposée colonne (1). La division du temps total d'exécution par le nombre de "calcul" donne un temps moyen de 58 millisecondes pour chaque calcul des nombres de Betti.

La colonne (3) montre les résultats avec calcul des nombres de Betti en utilisant l'algorithme incrémental. Les valeurs de $\mathfrak{b}_1$ et $\mathfrak{b}_2$ sont calculées pendant chaque fusion symbolique de deux régions mais en utilisant les valeurs calculées précédemment. Les résultats comme le nombre de régions finales sont les mêmes que pour les expériences des colonnes (1) et (2) puisque nous ne prenons pas en compte les nombres de Betti comme critère. La partie de fusion symbolique est environ 10 fois plus rapide qu'avec l'algorithme non incrémental. Le temps moyen pour un calcul est de 6.3 millisecondes. Cela montre l'intérêt de l'approche incrémentale sur l'approche classique.

Dans la colonne (4) et la colonne (5), nous utilisons les nombres de Betti calculés de manière incrémentale dans le critère de fusion en utilisant des valeurs vers lesquelles l'algorithme essaie de faire évoluer la partition.

Dans la colonne (4), nous souhaitons obtenir une partition qui ne contient ni tunnel ni cavité. Cette configuration demande moins de fusions effectives ce qui provoque plus de régions résultantes mais aussi plus de tentatives de fusions car certaines fusions sont refusées par le critère topologique.

La colonne (5) présente les résultats lorsque nous demandons un résultat dans lequel le nombre de tunnels converge vers 5 et le nombre de cavités converge vers 1. Le temps moyen de calcul des nombres de Betti augmente encore pour la même raison que lors de l'étude de l'expérience présentée colonne (4). Le nombre de régions finales est plus grand puisque le critère de convergence est plus dur à atteindre.

La Table 8.2 présente la distribution des régions par nombres de Betti. Les deux premières lignes indiquent la distribution des régions obtenues sans contrainte topologique. Peu de régions ont des nombres de Betti supérieurs à zéro. Les deux lignes suivantes montrent la distribution des régions pour la contrainte topologique de convergence $\mathfrak{b}_1 \rightarrow 0$ et $\mathfrak{b}_2 \rightarrow 0$. Les résultats montrent une diminution du nombre de régions ayant des nombres de Betti supérieurs à zéro. Toutes les régions ne deviennent pas homéomorphes à des sphères car le critère de contraste empêche certaines fusions qui permettaient d'atteindre ce résultat. Les deux dernières lignes présentent la distribution des régions pour la contrainte $\mathfrak{b}_1 \rightarrow 5$ et $\mathfrak{b}_2 \rightarrow 1$. Dans ce cas, le nombre de régions ayant au moins un tunnel augmente, et le nombre de régions ayant plus de cinq tunnels diminue. Le nombre de régions ayant des cavités augmente.

TAB. 8.2: Distribution des régions par nombres de Betti : (1-3) sans contrainte topologique sur les nombres de Betti ; (4) en utilisant un critère de convergence pour les nombres de Betti ($\mathfrak{b}_1 \rightarrow 0$ et $\mathfrak{b}_2 \rightarrow 0$) ; (5) en utilisant un critère de convergence pour les nombres de Betti ($\mathfrak{b}_1 \rightarrow 5$ et $\mathfrak{b}_2 \rightarrow 1$).

| | Régions | Value | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | $> 7$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1-3 | 1784 | $\mathfrak{b}_1$ | 1768 | 6 | 1 | 2 | 1 | 1 | 0 | 0 | 5 |
| | | $\mathfrak{b}_2$ | 1781 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 4 | 1938 | $\mathfrak{b}_1$ | 1932 | 2 | 2 | 0 | 0 | 0 | 1 | 1 | 0 |
| | | $\mathfrak{b}_2$ | 1934 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 2 |
| 5 | 2158 | $\mathfrak{b}_1$ | 2099 | 24 | 6 | 9 | 11 | 5 | 0 | 2 | 2 |
| | | $\mathfrak{b}_2$ | 2147 | 4 | 2 | 0 | 1 | 0 | 1 | 0 | 3 |

Cependant, comme dans le cas précédent, la convergence du critère sur $\mathfrak{b}_1$ et $\mathfrak{b}_2$ n'est pas atteinte : le critère couleur mais aussi les interactions entre la contrainte sur $\mathfrak{b}_1$ et la contrainte sur $\mathfrak{b}_2$ font que toutes les régions ne respectent finalement pas le critère donné par l'utilisateur.

### 8.5.2 Influence du critère avec des images artificielles

Nous avons observé dans l'expérience précédente des changements dans les nombres de Betti des régions de la partition en fonction des seuils de convergence fournis par l'utilisateur. Nous montrons maintenant l'effet de ces seuils sur des images artificielles comportant des formes particulières engendrant des régions avec des tunnels et des cavités lors de la segmentation.

Nous utilisons différentes images pour faire nos expériences. Ces images sont des images 3D en niveaux de gris. Dans ces images, nous représentons les objets par des intensités claires. Ainsi nous dessinons des formes faisant apparaître différentes configurations :

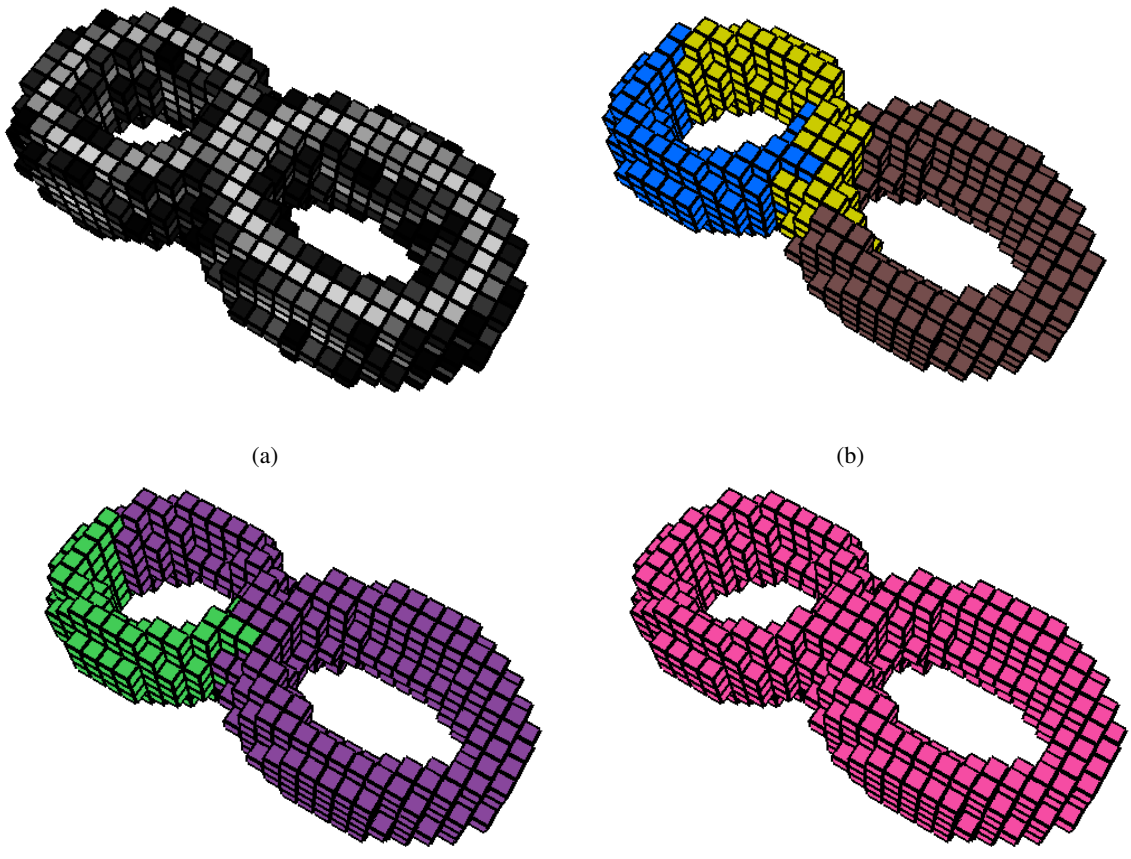
– une région comportant deux tunnels (2-tore) ;
– une région comportant cinq cavités en forme de tores et dix tunnels.

La Figure 8.7 montre quelques images des régions obtenues après la segmentation de la première image (2-tore) en faisant varier les seuils des nombres de Betti.

L'image originale, présentée dans la Figure 8.7a, montre les voxels d'une image 3D en niveau de gris. Dans cette image originale, il y a un voxel par région. La segmentation proposée fusionne les régions par rapport à leur niveau de gris. L'application de ce critère de segmentation produit deux régions (Figure 8.7d). L'une plutôt claire qui forme un 2-tore et l'autre plutôt foncée qui est le fond de l'image et qui entoure la région claire. L'application de contrainte sur les nombres de Betti va contraindre la topologie des deux régions. Dans la Figure 8.7b, le critère de convergence interdit l'apparition de tunnels et nous ne contrôlons pas le nombre de cavités. Nous observons une région pour le fond et 3 régions claires. L'union de ces régions claires est un 2-tore mais chacune des régions n'est homéomorphe à une sphère. La contrainte est bien respectée. La Figure 8.7c montre le résultat de la segmentation autorisant au maximum un tunnel. Il y a deux régions claires. L'une d'elle contient un tunnel et l'autre est homéomorphe à une sphère. En parcourant les régions dans un autre ordre, le résultat peut être différent mais aucune région ne peut avoir plus d'un tunnel.

Dans un second temps, nous étudions le résultat de la segmentation sur l'autre image de test. Cette image est définie de manière à ce qu'une segmentation produise le nombre demandé de tunnels et de cavités.

Nous utilisons alors des segmentations avec différents paramètres sur les nombres de Betti et nous étudions le résultat produit par ces différentes segmentations. La Table 8.3 présente les résultats de l'algorithme en fonction des nombres de Betti définis comme seuils lorsque nous appliquons la segmentation sur l'image.



(a) (b)

(c) (d)

FIG. 8.7: Expériences sur une image 3D artificielle. Le seuil de convergence sur le deuxième nombre de Betti qui est utilisé pour obtenir chaque image est indiqué pour chaque figure. Nous ne montrons pas la région de fond afin d'observer ce qui se passe sur la région claire. (a) Voxels d'un objet en forme de 2-tore. Chaque voxel de l'image appartient à une région différente. (b) 2-tore avec $\mathfrak{b}_1 \rightarrow 0$. L'objet est divisé en trois régions n'ayant aucun tunnel. (c) 2-tore avec $\mathfrak{b}_1 \rightarrow 1$. Une des deux régions a un tunnel, l'autre n'en a pas. Deux régions en forme de 1-tore ne sont pas obtenues à cause de l'ordre des fusions. (d) 2-tore obtenu sans utiliser de contraintes sur les nombres de Betti.

TAB. 8.3: Valeurs en fonction des seuils sélectionnés par l'utilisateur du nombre de régions, le nombre total de tunnels et de cavités ainsi que le nombre maximum de tunnels et de cavités pour une région. Les résultats pour l'image segmentée sans contrôle topologique sont donnés par la première ligne du tableau.

| | | | Somme | | Maximum | |
|---|---|---|---|---|---|---|
| $S_t$ | $S_c$ | Régions | Tunnels | Cavités | Tunnels | Cavités |
| -1 | -1 | 8 | 20 | 6 | 10 | 5 |
| 0 | 0 et plus | 82 | 5 | 1 | 5 | 1 |
| 10 | 0 | 164 | 241 | 2 | 10 | 1 |
| 10 | 10 | 16 | 25 | 13 | 8 | 4 |

Dans l'image initiale, la région de fond est déjà segmentée, il s'agit d'une région comportant cinq tunnels et une cavité. Cette cavité contient un voxel par région et c'est l'objet que nous allons segmenter.

La première ligne présente les résultats lorsqu'il n'y a pas de contrainte sur les nombres de Betti. L'algorithme de segmentation produit alors une partition de l'image en huit régions.

Lors de nos expériences, nous avons d'abord fait varier le nombre de cavités autorisées tout en refusant l'apparition de nouveaux tunnels. Cette contrainte empêche alors l'apparition de régions comportant d'autres cavités dans l'image utilisée pour cette expérience.

Dans les lignes suivantes, nous faisons varier le nombre de tunnels et le nombre de cavités autorisés. Nous observons que nous n'obtenons pas le résultat optimal autorisé par les seuils. Cela provient de l'ordre dans lequel sont réalisées les fusions de régions qui peuvent conduire à l'apparition de régions ayant beaucoup de cavités et qui ne peuvent plus évoluer. Le critère est ainsi bloquant pour obtenir un bon résultat pour la segmentation de l'image, cependant cette limitation provient de la manière dont le critère est exprimé.

## 8.6 Conclusion

Dans ce chapitre nous avons montré comment intégrer une information topologique comme critère pour guider la segmentation. Le modèle des cartes topologiques permet de calculer facilement et efficacement des invariants topologiques comme les nombres de Betti qui sont ensuite utilisés pour contraindre le nombre de tunnels et de cavités des régions produites par la segmentation. Dans le Chapitre 9 nous présentons un exemple montrant l'utilisation des différentes opérations pour réaliser la segmentation d'images médicales. Nous résolvons ainsi à un problème actuel de segmentation qui concerne la segmentation de tumeurs cérébrales dans des images TEP en utilisant des informations topologiques pour proposer un meilleur résultat.

CHAPITRE 9

# ALGORITHMES DE DIVISION-FUSION

Après avoir défini un premier outil utilisant une caractéristique topologique dans le cadre d'une segmentation de bas en haut, nous nous sommes intéressés à la mise en œuvre d'une chaîne complète de segmentation mêlant différentes opérations et algorithmes de segmentation. Le but ici est de travailler à la résolution d'un problème de segmentation sur des images médicales.

Pour cela, nous avons travaillé avec des images de tumeurs cérébrales obtenues en utilisant la tomographie à émission de positon ou TEP (*Positron Emission Tomography ou PET* en anglais). Dans un premier temps, nous présentons comment les images TEP sont acquises et ce qu'elles représentent. Ensuite nous listons les différents outils de segmentation, dérivés des outils de modification de la carte topologique, que nous mettons en œuvre afin de résoudre le problème. Nous expliquons par la suite comment les différents éléments de la chaîne que nous proposons sont articulés. Enfin nous présentons quelques premiers résultats obtenus en utilisant la chaîne de traitement.

## 9.1 Imagerie TEP

La tomographie à émission de positons est une technique d'imagerie médicale en 3D. Elle permet de mesurer l'activité métabolique d'un organe par la détection des émissions de positons produites lors de la désintégration d'un marqueur radioactif injecté au patient.

La Figure 9.1 présente un schéma de fonctionnement d'un scanner TEP (extrait de [Lan03]). La partie du corps du patient à diagnostiquer est placée dans le scanner (au milieu de l'anneau). Le marqueur émet un positon qui s'annihile avec un électron du milieu. Cette annihilation provoque l'émission de deux photons gamma dans des directions opposées. Le scanner détecte alors simultanément les deux rayons et peut ainsi déterminer sur quelle ligne le marqueur se trouve. Le volume 3D est reconstruit à partir des informations obtenues de chaque paire de rayons (à partir du sinogramme[1]).

L'imagerie TEP est une imagerie fonctionnelle par opposition aux techniques comme les rayons X ou les scanners qui donnent des images anatomiques. C'est donc un outil de diagnostic permettant de détecter des pathologies qui se traduisent par l'altération du métabolisme comme les cancers.

Après reconstruction, une image 3D est obtenue où les voxels représentent une petite zone du corps du patient. L'intensité de chaque voxel mesure alors la concentration du marqueur dans le volume représenté par le voxel. La Figure 9.2 présente trois coupes d'une image TEP de cerveau chez un patient sain.

Le marqueur le plus utilisé dans les images TEP est le $^{18}$F-FDG (Fluorodeoxyglucose marqué au Fluor 18). Il s'agit d'un sucre semblable au glucose rendu radioactif. Pour vivre, fonctionner et

[1] Un sinogramme contient l'ensemble des données directement issues de l'acquisition avant la reconstruction.

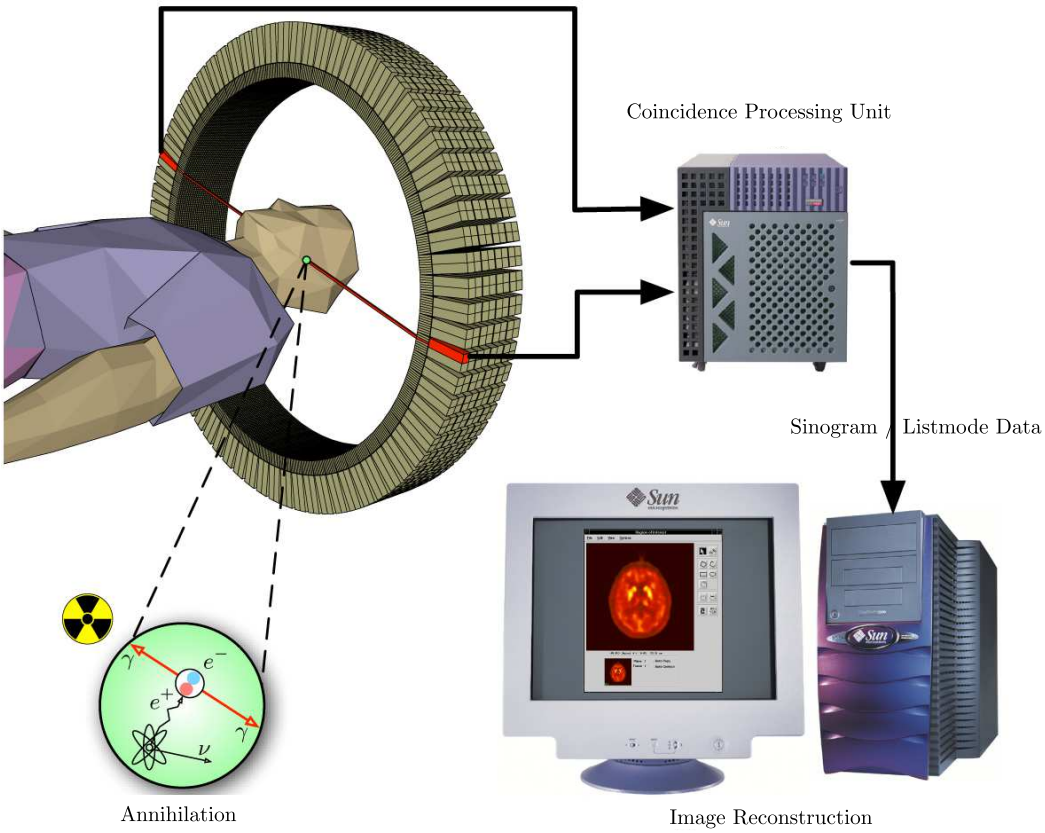



FIG. 9.1: Schéma de fonctionnement de l'imagerie TEP.

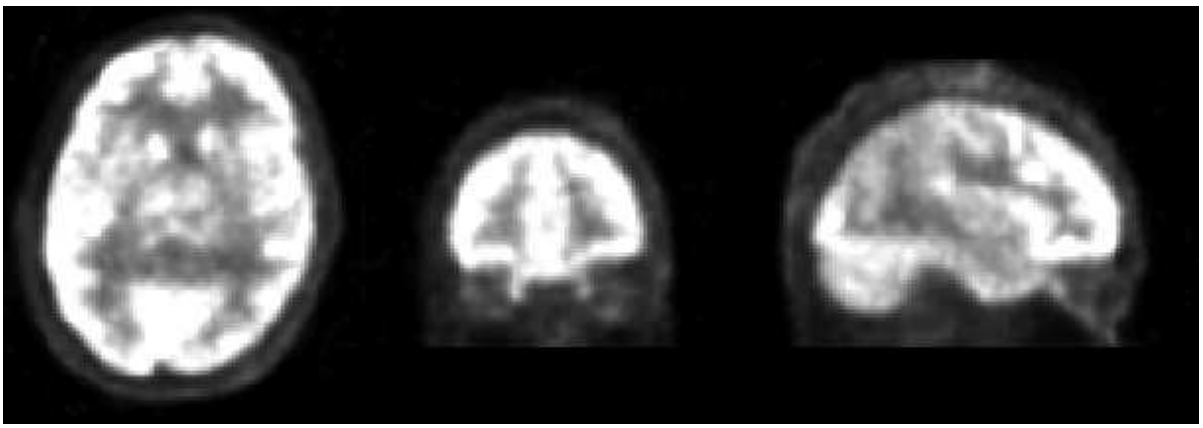


FIG. 9.2: Coupe axiale, coronale et sagitale d'un cerveau vu par imagerie TEP.

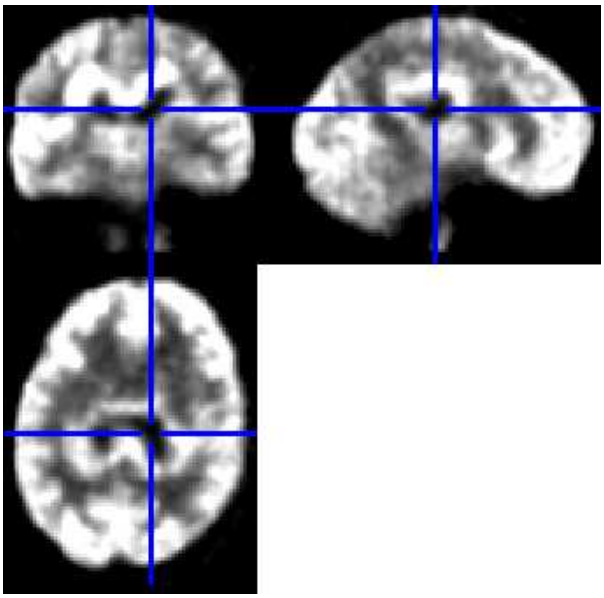


FIG. 9.3: Coupes axiale, coronale et sagitale d'un cerveau vues par imagerie TEP. Les lignes désignent le centre d'une nécrose, se traduisant par une tache sombre car peu métabolique et entourées par une zone de forte activité métabolique : la partie active de la tumeur.

se reproduire, les cellules ont besoin d'énergie sous forme de glucose. Plus l'activité des cellules est importante, plus la consommation de glucose augmente. Ainsi l'imagerie TEP permet de détecter les organes du corps humain qui utilisent beaucoup de glucose : ils ont une activité métabolique forte. Les cellules cancéreuses ont une activité métabolique forte, et par conséquent ont une consommation de glucose plus importante : l'imagerie TEP permet donc de détecter ces anomalies et permet ainsi la détection des tumeurs cancéreuses. Cependant, le glucose est également utilisé par les organes comme le cœur ou le cerveau. La détection des tumeurs dans ces organes pose donc de nombreux problèmes.

Nous disposons d'une soixantaine d'images de patients présentant des tumeurs cérébrales. La résolution des images est approximativement de $128 \times 128 \times 80$ voxels, le nombre de voxels de la hauteur variant légèrement selon les images. Le problème posé par le médecin est non pas de détecter la tumeur mais de segmenter avec précision son volume afin de permettre une étude quantitative de la tumeur. La résolution des images permet selon l'expert d'identifier deux zones distinctes dans les tumeurs. La première zone est la partie nécrosée au centre de la tumeur. Cette nécrose se traduit par une activité métabolique faible (donc de faible intensité dans l'image) et forme un volume sans tunnel ni cavité. La zone active de la tumeur est identifiée par une région qui entoure complètement la zone nécrosée et qui possède une forte activité métabolique. L'œdème qui se forme généralement autour d'une tumeur n'est pas visible du fait de la résolution des images. La Figure 9.3 présente l'exemple d'une tumeur chez un patient dans une image TEP d'un cerveau.

Parmi la base d'images dont nous disposons, les images forment des configurations très variées, notamment par le volume occupé par la tumeur par rapport à la boîte crânienne du patient. Nous nous sommes concentrés sur l'étude des algorithmes et critères de segmentation dans le cadre des cas les plus typiques formant visuellement la configuration proposée par le médecin (notre *a priori*). Ainsi nous avons sélectionné 10 patients afin d'expérimenter nos algorithmes de segmentation. Nous ne disposons pas de volumes de référence fournis par un expert ni d'outils existants permettant de réaliser ces expérimentations. Nous n'avons donc pas comparé nos résultats, ni évalué leur pertinence absolue.

Mais ces travaux forment une première application et de premiers résultats qui montrent l'intérêt de la carte topologique dans le traitement d'images.

## 9.2 Chaîne d'outils pour la segmentation

Pour répondre au problème posé, nous avons choisi de mettre en œuvre une segmentation semi-automatique où l'utilisateur fournit un minimum d'informations. Ces informations sont d'une part le centre approximatif de la tumeur et son diamètre. L'utilisateur de l'application détermine donc l'existence et la localisation de la tumeur dans le cerveau. D'après les explications de l'expert, ces informations sont pour le moment les seules à être vraiment utilisées dans le cadre de l'imagerie cérébrale : à l'œil, le médecin reconnaît la tumeur et détermine approximativement son rayon à l'aide d'un marqueur placé sur une coupe 2D.

Nous détaillons à présent les outils de segmentation développés à partir des outils de modification de la carte topologique, puis nous expliquons comment les éléments s'enchaînent afin de produire la partition résultat.

### 9.2.1 Outils de division

La première classe d'outils utilise l'opération de division de régions afin de raffiner la partition de l'image. Nous distinguons trois usages de cette opération permettant de réaliser diverses tâches.

Afin d'isoler une région d'intérêt (*region of interest ou ROI* en anglais) à l'intérieur d'une grosse région, nous divisons la région en utilisant un guide délimitant les contours de la région d'intérêt. Ainsi nous utilisons généralement deux types de formes pour ces guides : le cube ou la sphère. Afin de construire l'ensemble des surfels permettant de définir le guide, nous utilisons une image binaire dans laquelle nous dessinons avec un ensemble de voxels un objet discret correspondant au volume de la région d'intérêt. Dans cette image binaire, nous récupérons les surfels appartenant aux frontières de l'objet. Il est ainsi possible avec cette technique de construire des surfaces complexes permettant de diviser une région.

La division est également utilisée avec un critère basé sur la valeur des pixels. Ce type de division s'apparente à la division utilisée dans les modèles hiérarchiques à base d'octrees. Si une région n'est pas homogène selon un critère, nous divisons alors cette région à l'aide de trois plans orthogonaux qui s'intersectent au niveau du centre de la région. Cette division sur une région cubique provoque généralement l'apparition de huit régions résultantes : les huit octants de la structure des octrees. Cependant étant donné que les régions avec lesquelles nous travaillons sont quelconques, une division par ce type de plan peut provoquer toutes sortes de subdivisions. Il s'agit d'une solution simple qui permet de rapidement diviser un volume en sous-régions. Une autre approche envisageable consiste à mettre en place un parcours de la région qui réétiquette les voxels afin de définir une nouvelle partition de la région. De ce nouvel étiquetage, nous obtenons l'ensemble des surfels frontières et nous utilisons cette information pour diviser la région. En pratique, nous préférons utiliser la division par les trois plans car cela diminue de manière rapide la taille des régions (par rapport à la division avec un seul plan), et permet de n'utiliser que des informations relatives aux régions (le barycentre par lequel passe les trois plans) pour déterminer la division. De plus, nous ne reparcourons pas tous les voxels pour évaluer le critère.

Il est également possible d'utiliser un guide défini à l'intérieur du volume de l'image, puis de diviser l'ensemble des régions qui sont intersectées par ce guide. Pour cela, nous construisons le guide dans l'image et nous divisons chaque région avec les surfels du guide qui appartiennent à l'intérieur de la région. En retirant au fur et à mesure les surfels qui sont utilisés pour diviser chaque région, le processus s'arrête lorsque tous les surfels du guide ont été utilisés. Par construction, nous nous assurons que le

guide soit cohérent. Pour cela, nous utilisons une partition de l'image en étiquettes afin de construire l'ensemble des surfels. Avec cette technique, nous réalisons, par exemple, des divisions par surfaces concentriques sur toutes les régions de l'image. Cela revient à calculer une dilatation de la surface de la région originale et d'utiliser l'ensemble des surfels comme guide pour la division des régions voisines.

Nous n'utilisons cependant pas d'outils topologiques dans ce type d'opération. Il serait possible de les utiliser, par exemple, pour diviser ou éclater les régions comportant trop de tunnels et de cavités. Mais pour cela, nous avons besoin de mettre à jour les nombres de Betti des régions obtenues après la division, ce qui ne peut pas se faire de manière incrémentale. Il faudrait alors utiliser la méthode non incrémentale présentée Section 8.2, page 151, qui a l'inconvénient d'être coûteuse en temps.

### 9.2.2 Outils de regroupement

La deuxième classe d'outils que nous mettons en œuvre utilise la fusion de régions. Nous avons défini trois usages distincts pour les opérations de regroupement.

La première opération consiste à définir un algorithme de croissance de régions qui, étant donné un critère, cherche à agglomérer les régions voisines. Cet algorithme utilise la fusion de régions en changeant uniquement la manière dont les régions sont sélectionnées. Ainsi l'algorithme utilise l'information d'adjacence entre les régions afin de sélectionner les régions qui peuvent fusionner. Le critère porte généralement sur l'intensité de la région qui est en cours de construction. Étant donné un ensemble de régions $S_i$ qui correspond à l'ensemble des régions sélectionnées, nous passons à $S_{i+1}$ en ajoutant à $S_i$ une région $r$ adjacente à l'une des régions de $S_i$ telle que la région $r$ valide le critère couleur et apporte le moins de changement possible (par exemple que sa couleur soit la plus proche de la couleur moyenne de l'ensemble). Lorsqu'aucune région $r$ ne peut être ajoutée, le processus de croissance est terminé. Nous utilisons alors l'opération de fusion locale afin de construire la région résultante.

La seconde opération consiste à fusionner les régions à la manière d'une segmentation bottom-up. Cette opération est généralement consécutive à une opération de division de régions. Dans ce contexte, nous introduisons une version modifiée des algorithmes afin qu'ils ne prennent en compte que les régions issues de la division. Afin d'identifier ces régions (et pour être plus général), nous utilisons une marque sur les régions qui sont autorisées à fusionner. Le principe de la segmentation est alors le même que dans les approches présentées précédemment. Nous utilisons à la fois des critères topologiques, géométriques et basés sur les valeurs des régions afin de produire la partition résultante. Cette opération permet de construire de plus grosses régions étant donné un critère, et permet ainsi de fusionner des régions homogènes qui ont été divisées par la division en huit.

Nous utilisons enfin un troisième type d'opération. Il s'agit de la fusion des régions 26-adjacentes à une région $r$. Cette fusion utilise la carte topologique afin de retrouver toutes les régions formant une composante 6-connexe de régions englobant la région initiale $r$. Pour cela, nous utilisons l'opération de fusion locale en changeant la manière dont les régions sont sélectionnées. Pour sélectionner les régions, nous parcourons la surface externe de $r$ et pour chaque sommet, nous sélectionnons l'ensemble des régions autres que $r$ qui lui sont incidentes. Pour chaque brin des orbites des sommets, si la région d'appartenance du brin est différente de $r$, il s'agit d'une région 26-adjacente à $r$ que nous sélectionnons. Puis en utilisant l'opération de fusion locale, nous agglomérons toutes les régions afin de construire une enveloppe à la région $r$ et ainsi créer volontairement une région avec une cavité. Nous verrons l'usage de cette opération dans la Section 9.2.4.

Dans les opérations de regroupement, nous avons la possibilité de mettre en œuvre les critères topologiques afin d'aider à la résolution du problème. Cependant, devant le coût important du calcul des nombres de Betti, nous ne réalisons ce type de traitement que dans les opérations où il est nécessaire. Nous précisons donc lorsque nous utilisons le critère topologique dans la segmentation.

### 9.2.3 Outils de déformation

Dans la Section 6.6, nous avons noté l'intérêt de l'outil de déformation pour obtenir une partition dont la géométrie respecte les données de l'image et dont la topologie est inchangée par rapport à la partition initiale. Nous mettons dans notre chaîne de traitement deux outils basés sur la déformation de régions.

Le premier outil est le raffinement à topologie constante d'une partition. Cette déformation permet de changer la géométrie des faces mais ne touche pas à la géométrie des arêtes et des sommets de la partition pour ne pas engendrer de changements topologiques. Pour cette déformation, nous utilisons deux énergies, d'une part une énergie d'attache aux données qui cherche à minimiser l'erreur quadratique moyenne et d'autre part l'énergie de régularisation qui cherche à minimiser le nombre de surfels. En pratique, les poids respectifs des deux énergies font que l'énergie basée sur les régions prime sur l'énergie de surface. En effet, notre intérêt est dans ce type d'approche de rendre les faces (qui sont généralement très planes) plus complexes pour mieux coller aux données. Aussi faut-il laisser une marge de manœuvre pour que le résultat soit intéressant.

L'autre outil de déformation est l'opération de déformation de surfaces imbriquées dans une partition. Nous nous servons de cette opération pour améliorer la géométrie du résultat final en fonction des données de l'image. Nous utilisons dans cette application la même énergie basée sur l'erreur quadratique moyenne de l'intensité des régions et l'énergie minimisant le nombre de surfels. Dans cette opération par contre, nous définissons les poids respectifs des deux énergies de manière à ne permettre qu'une petite variation de la surface des régions. En effet, nous supposons que les données sont bien segmentées et qu'il n'y a qu'une petite différence entre la partition initiale et la partition résultat.

### 9.2.4 Chaîne mise en œuvre

Pour résoudre le problème de la segmentation des images TEP de cérébrales, nous avons mis les différentes opérations bout à bout dans une chaîne de traitement. Nous détaillons à présent comment s'enchaînent les opérations, quels sont les critères utilisés à chaque étape, et comment les paramètres topologiques sont utilisés. Les seuls paramètres donnés par l'utilisateur sont le centre approximatif $O$ de la tumeur et son rayon $r$.

Les différentes opérations de la chaîne de traitement s'enchaînent de la manière suivante :

1. extraction dans une région ;
2. division autour de la région d'intérêt (ROI) ;
3. division de la ROI par trois plans d'après un critère couleur pour diviser les zones sombres (appartenant à la nécrose) ;
4. croissance de la région nécrosée sous contrainte topologique ($\mathfrak{b}_1 \to 0$ et $\mathfrak{b}_2 \to 0$) ;
5. division par trois surfaces imbriquées des régions entourant la nécrose ;
6. fusion des régions adjacentes à la nécrose pour former la partie initiale de la partie active ;
7. croissance de la région formant la partie active de la tumeur sous contrainte topologique ($\mathfrak{b}_1 \to 0$ et $\mathfrak{b}_2 \to 1$) ;
8. fusion des régions autres que la zone active et la nécrose ;
9. déformation de surfaces imbriquées.

Nous détaillons à présent chaque étape de la chaîne de traitement. La première étape initialise le processus en procédant à l'extraction de l'image de sorte que toute l'image soit dans une seule région.

La deuxième étape construit un guide de division d'après les paramètres donnés par l'utilisateur. Le guide est composé des surfels formant la surface d'une sphère de centre $O$ et de rayon $1,5r$ permettant

d'entourer complètement la tumeur tout en réduisant le nombre de voxels à étudier à ceux contenus dans la sphère. Il y a maintenant deux régions : la région d'intérêt et le fond.

L'étape suivante est la division de la région d'intérêt en utilisant un critère sur la couleur. Nous commençons par calculer l'histogramme de la région d'intérêt et nous cherchons la valeur seuil $t_{necrose}$ qui fait qu'un tiers des voxels de la ROI est en-dessous du seuil. Cette proportion est fixée arbitrairement pour trouver une valeur de seuil qui permette de bien discriminer les voxels de la zone active avec les voxels de la nécrose. Nous utilisons alors ce seuil pour diviser les régions de manière récursive en utilisant un guide composé de trois plans tant que leurs valeurs minimum et maximum sont situées de part et d'autre du seuil $t_{necrose}$.

Nous procédons ensuite à la détection de la partie nécrosée de la tumeur. Pour cette partie, nous utilisons un processus de croissance de régions avec contrôle topologique. Pour ce faire, nous recherchons près du centre de la tumeur la région qui a la plus faible intensité. Cette région fait partie de la tumeur. Nous utilisons ensuite cette région comme graine pour un processus de croissance de régions qui utilise la valeur inférieure à $t_{necrose}$ comme critère de progression tout en assurant que la région ne possède aucun tunnel, ni cavité puisqu'il s'agit d'un *a priori* du problème.

Une fois l'estimation de la zone nécrosée réalisée, nous fixons la région associée afin qu'elle ne soit plus considérée dans les futures fusions et divisions de régions.

Nous nous intéressons maintenant à l'extraction de la partie active de la tumeur. Pour cela, nous utilisons l'*a priori* qui indique que la zone active entoure la tumeur. Nous réalisons donc dans un premier temps plusieurs divisions concentriques en utilisant des surfaces correspondantes à la dilatation de la nécrose. Nous construisons ainsi 3 surfaces (coquilles) correspondant à la dilatation de la nécrose par trois masques de plus en plus gros. Les masques utilisés sont trois sphères discrètes de rayon 3, 7 et 11. Par expérimentation, nous nous sommes rendus compte que trois surfaces étaient suffisantes pour diviser toutes les régions qui appartiennent à la partie active de la tumeur dans nos images. Grâce à cette opération, nous souhaitons obtenir une division des régions de manière à avoir différentes couches. Nous appliquons alors la division de toutes les régions par le guide déterminé par les trois surfaces.

Nous utilisons ensuite l'adjacence des régions afin de fusionner toutes les régions adjacentes à la nécrose de sorte que la nécrose devienne imbriquée dans la nouvelle région construite. Cette étape donne une graine de départ à la segmentation de la tumeur. Cela assure en même temps la topologie initiale pour la zone active comme étant une région n'ayant aucun tunnel et une unique cavité.

Nous utilisons ensuite un processus de croissance de régions contraint par la couleur minimum des régions qu'il est possible de fusionner et qui cherche à faire augmenter la couleur moyenne de la région. Nous utilisons une contrainte sur la compacité de l'objet. Cette contrainte géométrique est une hypothèse que nous avons émise et qui donne de bons résultats sur les images que nous avons à notre disposition.

Un processus de fusion cherche ensuite à agglomérer les régions du fond de l'image afin que la partition finale contienne quatre régions (la région infinie, la région de fond, la zone active de la tumeur et la nécrose) de sorte que toutes ces régions soient imbriquées les unes dans les autres.

Pour finir, nous utilisons l'opération de déformation afin d'améliorer la géométrie de la partition en fonction des données de l'images.

## 9.3 Expérimentations

Nous procédons à des expérimentations sur la chaîne de traitement que nous avons définie. En l'absence de segmentation de référence ou d'expert permettant de valider les résultats, nous ne jugeons pas de la qualité du résultat. Nous étudions ainsi, dans un premier temps, le temps de traitement de l'opération. Dans un deuxième temps, nous observons quelques exemples de segmentation sur différents

TAB. 9.1: Temps d'exécution (en secondes) des différentes parties de l'algorithme pour l'image traitée le plus rapidement (minimum), le plus lentement (maximum) et la moyenne sur les 20 images sélectionnées parmi celles à notre disposition.

| Opération | Minimum | Maximum | Moyenne |
|---|---|---|---|
| Extraction | 1,17 | 1,15 | 1,15 |
| Division (sphère) | 4,93 | 4,68 | 4,72 |
| Division (couleur $< t_{necrose}$) | 46,66 | 1025,19 | 334,7 |
| Croissance (nécrose) | 0,71 | 0,65 | 0,68 |
| Division (3 coquilles) | 52,97 | 296,69 | 293,52 |
| Fusion région adjacente | 0,72 | 0,92 | 0,85 |
| Croissance (zone active) | 1,27 | 1,28 | 1,32 |
| Fusion des régions du fond | 1,51 | 1,64 | 1,85 |
| Déformation | 3,89 | 5,74 | 6,14 |
| Total | 113,83 | 1337,96 | 644,92 |

patients. Nous présentons dans un troisième temps une expérience visant à comparer le volume d'une tumeur en utilisant deux images prises à quatre heures d'intervalle sur un même patient.

### 9.3.1 Temps de calcul

Notre première expérience consiste à mesurer le temps pris par chaque étape afin d'évaluer le coût de chaque opération. La Table 9.1 donne les temps d'exécution pour les différentes étapes de la segmentation pour 3 images TEP prises dans les images disponibles.

Nous observons que les étapes les plus coûteuses sont les étapes mettant en œuvre l'opération de division. En effet, les traitements relatifs à la division d'une région demandent de nombreuses opérations (comme l'éclatement des surfaces et la simplification de la carte) qui en pratique rendent l'opération lente. D'un autre côté, les algorithmes de fusion de régions même avec contrôle topologique sont plus rapides et ne constituent pas un frein au traitement. Ceci provient du fait que les algorithmes de fusion sont utilisés dans un mode de croissance de régions où l'on ne considère que les fusions entre la région à faire croître et les régions qui lui sont adjacentes. Cela réduit beaucoup le nombre de combinaisons possibles et permet du même coup de traiter les images rapidement. Cela montre que notre modèle est plus efficace pour les approches de type *bottom-up* que pour celles de type *top-down* où il n'est pas possible d'utiliser des mises à jour incrémentales des caractéristiques des régions.

Cependant, suite aux échanges avec les experts, les temps de traitement sont encore trop longs pour envisager une utilisation effective de ce type d'algorithme. Nous expliquons la lenteur du processus par le manque d'optimisation de l'opération de division de régions par un guide. En effet, les outils que nous avons développés permettent des traitements génériques et sont à l'état de prototypes de recherche.

Nous observons notamment que les opérations de division cherchent à raffiner la partition pour proposer la partition finale souhaitée. Nous disposons également de l'opération d'éclatement qui permet d'obtenir le même résultat et nous avons noté dans la Section 5.6.2 que l'opération d'éclatement était largement plus rapide que l'opération de division par un guide. Nous avons donc remplacé l'étape trois et six par un éclatement de la ROI à la place de l'étape trois. Nous obtenons alors des temps de traitement beaucoup plus rapides (en moyenne 6 secondes au lieu de 628 secondes) au prix d'un coût en mémoire plus important (120Mo au lieu de 80Mo). Étant donnée la taille des objets à segmenter, l'utilisation de la division ne se justifie pas.

### 9.3.2 Exemples de segmentation

La Figure 9.4 montre la partition se construisant lors des étapes clefs dans la chaîne de traitement et la Figure 9.5a donne un exemple de résultat obtenu. Nous présentons dans un premier temps le volume correspondant à la partie nécrosée d'une tumeur (Figure 9.5a) puis le volume correspondant à la partie active de la tumeur (Figure 9.5b).

La Figure 9.6 présente la partition obtenue sur deux coupes de deux images TEP distinctes. Nous distinguons bien les régions attendues représentant les différentes parties que nous cherchons à retrouver dans le cerveau. Nous notons cependant que les résultats ne sont pas très précis, notamment pour la Figure 9.6b.

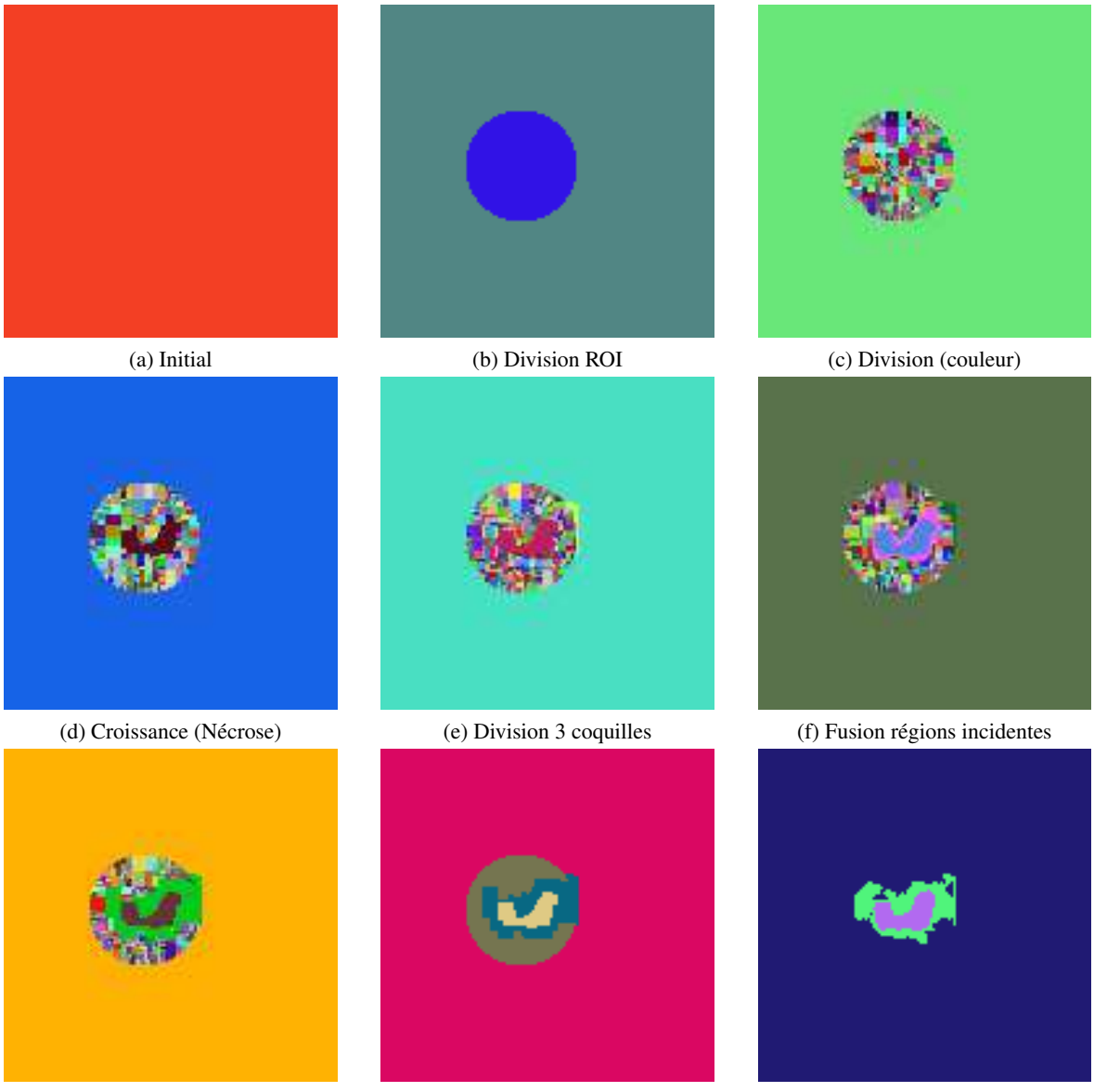
### 9.3.3 Comparaison de volumes

Nous disposons de deux images TEP pour chaque patient, prises à 4 heures d'intervalle avec la même injection de marqueur. Dans cette expérience, nous cherchons à comparer les résultats de la segmentation des deux images d'un même patient afin d'observer si le processus donne un résultat régulier. La Figure 9.7 présente la même coupe dans deux images segmentées par notre algorithme pour un même patient. Comme l'image est décalée, à cause des conditions d'acquisition différentes, l'identification des coupes est faite à la main.

Comme le résultat précédent nous l'a montré, la segmentation en l'état actuel n'est pas assez précise. Les volumes obtenus sont également différents : la nécrose mesure 525 voxels dans la première image et 429 dans la deuxième. La zone active mesure 4450 voxels dans la première image et 3932 voxels dans la deuxième. Les experts nous ont confirmé qu'il peut y avoir un changement d'échelle entre deux images mais même en comparant les rapports (1,22 pour la nécrose contre 1,13 pour la zone active), nous n'obtenons pas de mesure fiable et répétée. Le critère est pour le moment *ad-hoc* et ne prend pas en compte suffisamment d'informations. Pour améliorer ces résultats, il est nécessaire de travailler les critères, ce qui n'étaient pas l'objectif de cette thèse. Pour finir, la capacité des cartes à prendre en compte diverses informations tant sur l'intensité, sur la géométrie que sur la topologie des régions nous permet d'espérer de bons résultats dans le futur.

## 9.4 Conclusion

Dans ce chapitre, nous avons montré comment l'ensemble des opérations définies autour des cartes topologiques peuvent être utilisées pour résoudre un problème actuel de segmentation. Nous présentons ainsi l'intérêt de l'utilisation des cartes topologiques qui rendent facile l'intégration de critères topologiques pour la segmentation d'images 3D. Ces travaux constituent un premier résultat sur la segmentation d'images médicales par les cartes topologiques. Il est maintenant nécessaire de poursuivre ces expériences afin de proposer un meilleur résultat, c'est-à-dire plus proche des besoins clinique, et continuer à démontrer l'intérêt des cartes topologiques dans le cadre du traitement d'images.

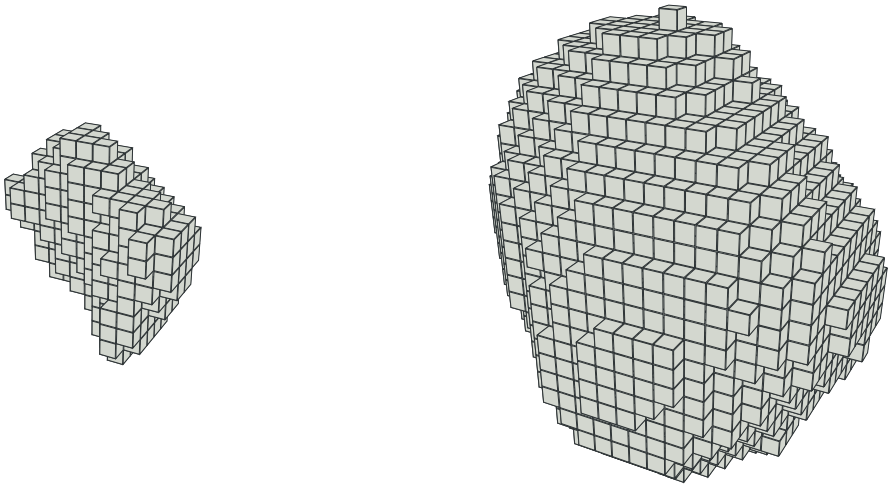


(a) Initial (b) Division ROI (c) Division (couleur)

(d) Croissance (Nécrose) (e) Division 3 coquilles (f) Fusion régions incidentes

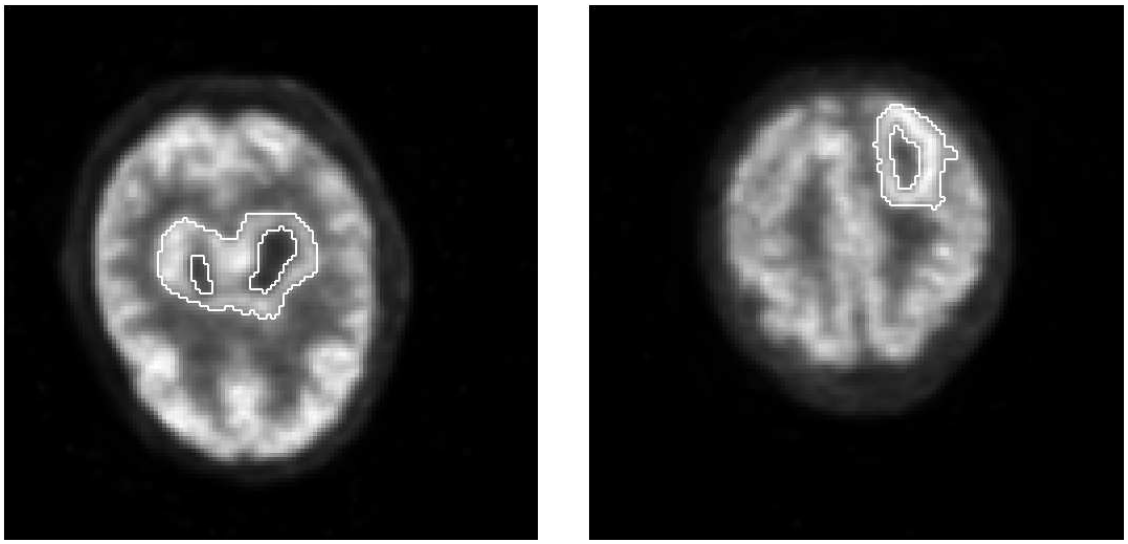
(g) Croissance (Active) (h) Fusion régions fond (i) Déformation et fusion finale

FIG. 9.4: Coupe de la partition de l'image aux différentes étapes de la chaîne de traitement.



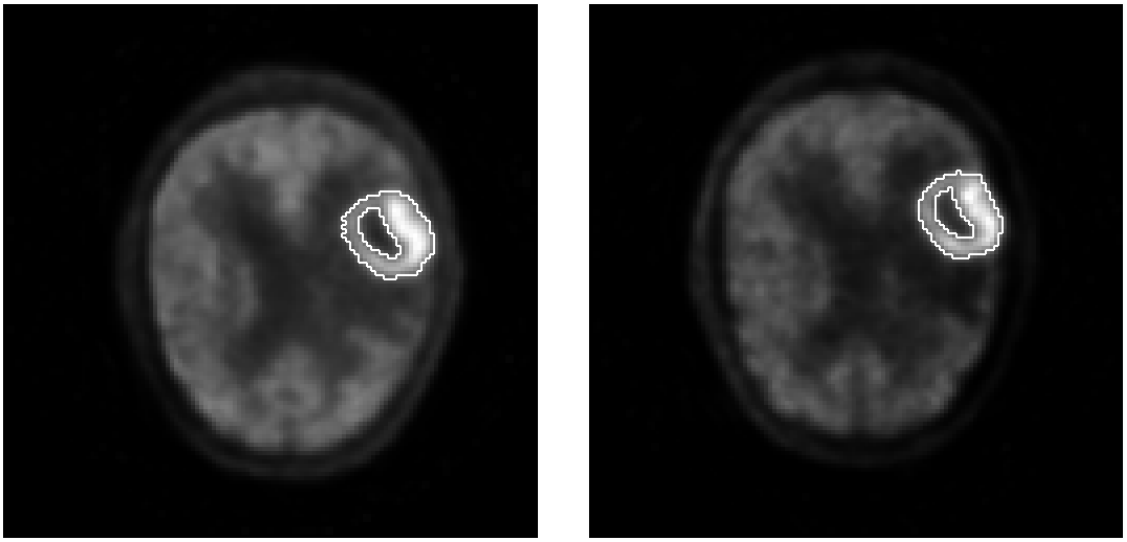
(a) (b)

FIG. 9.5: Exemple présentant le volume de la nécrose et le volume de la zone active qui forment le résultat de la segmentation de l'image par notre algorithme.



(a) (b)

FIG. 9.6: Exemple présentant des coupes de deux images de patients différents segmentées par notre algorithme. (a) Coupe de l'image du premier patient. La nécrose semble être composée de deux cavités dans cette image mais il n'y a bien qu'une seule zone nécrosée, les deux parties se rejoignant dans une autre coupe. (b) Coupe de l'image du second patient.



(a) (b)

FIG. 9.7: Exemple présentant approximativement la même coupe dans deux images du même patient prises à quatre heures d'intervalle. L'intensité des valeurs de l'image est normalisée sur l'intervalle [0,255] pour permettre une bonne visualisation. (a) Première image. (b) Deuxième image. Nous observons la forme très similaire des deux partitions.

# CONCLUSION

Dans ce travail de thèse, nous avons développé des outils de manipulation d'une carte topologique permettant de réaliser des opérations de segmentation. Nous avons travaillé sur le calcul et l'expression de propriétés topologiques à l'aide du modèle de représentation d'images afin de mettre en place des critères permettant le contrôle de la topologie de la partition durant le processus de segmentation. Nous nous sommes également intéressés à la définition d'un nouveau type de points simples : les points ML-Simples qui permettent d'envisager la déformation à topologie constante d'une partition de l'image.

Notre contribution en terme d'outils pour la carte topologique est centrée sur la définition de trois nouvelles opérations de manipulation d'une partition d'une image.

La fusion de régions [DD08a], opération qui consiste à rassembler des ensembles de régions connexes en diminuant le nombre de régions, est définie de deux manières différentes : par une approche locale adaptée à la fusion ponctuelle de régions, et par une approche globale adaptée à de nombreuses fusions de régions. La comparaison des deux approches confirme les points forts de chaque méthode. D'un point de vue pratique, la fusion de régions est une opération rapide sur le modèle des cartes topologiques. Mais nous avons également remarqué qu'une configuration particulière de régions, les régions isolées, peut bénéficier d'un algorithme plus optimisé. Si l'approche locale semble peu intéressante pour mettre en place la segmentation d'une image, l'approche globale semble au contraire toute indiquée. Le fonctionnement de cette dernière en deux passes (d'une part une fusion à haut-niveau, puis d'autre part la fusion effective des régions dans le modèle) permet de mettre en œuvre des opérations de traitement manipulant les régions de manière abstraite avant d'appliquer la fusion.

La division d'une région est la deuxième opération de modification d'une carte topologique que nous avons étudiée. Là encore, nous avons proposé deux approches distinctes, d'une part l'éclatement de régions et d'autre part la division par un guide. La première méthode consiste à remplacer une région à éclater en un ensemble de régions de sorte que chaque voxel de la région initiale soit dans une nouvelle région distincte. La division par guide est une version généralisée de la division par une surface. Le guide est défini par l'ensemble des surfels qui vont former les nouvelles frontières de la partition. Nous avons comparé les approches de la segmentation en remarquant que lorsque la partition à obtenir est très fine, il y a peu de voxels par région et dans ce cas, l'éclatement suivi d'une opération de fusion est plus efficace que la division directe par un guide.

Nous avons proposé une définition des points simples multilabels (appelés ML-Simples) [DDL09] qui permet la mise en œuvre d'une troisième opération de manipulation d'une carte topologique : la déformation à topologie constante. Les points ML-Simples ne sont pas à proprement parler spécifiques au modèle des cartes combinatoires : ils sont définis dans le cadre des éléments intervoxels. La définition actuelle des points ML-Simples indique qu'un voxel est ML-Simple si il peut être basculé dans une région adjacente sans changer la topologie de la partition, c'est-à-dire ne pas changer le degré d'un poin-

tel ou d'un lignel dans la représentation en intervoxel des bords de la région. Cette définition s'adapte bien à la carte topologique, la déformation devenant alors une opération purement géométrique, la carte combinatoire comme l'arbre d'imbrication des régions restent donc inchangés. À partir de la définition des points ML-Simples, nous avons développé une opération de déformation d'une partition. Grâce à celle-ci, nous avons adapté plusieurs algorithmes définis dans la littérature en généralisant les procédés de déformation avec le modèle des cartes topologiques.

Nous avons utilisé l'ensemble de ces opérations dans le cadre d'opérations de segmentation. Notre contribution repose sur l'illustration à travers quelques prototypes de la capacité du modèle des cartes topologiques pour faire du traitement d'images.

Dans un premier temps, nous nous sommes intéressés à la mise en œuvre d'un algorithme, trouvé dans la littérature, dans le modèle des cartes topologiques [DD08b]. Cette étape permet de montrer que des opérations de traitement d'images peuvent être réalisées avec le modèle. Cette approche utilise la fusion de régions afin de réaliser la segmentation d'une image et met en œuvre un critère basé sur le contraste qui est équivalent à celui utilisé dans la méthode initiale.

Nous avons ensuite étudié l'intégration d'un critère topologique sur les caractéristiques des régions à obtenir dans la partition [DD09]. Ce critère utilise les nombres de Betti de chaque région pour caractériser leurs nombres de tunnels et de cavités. Dans un premier temps, nous avons proposé des algorithmes permettant de calculer les trois premiers nombres de Betti pour une région dans une carte topologique. Nous nous sommes ensuite intéressés à la définition d'une méthode incrémentale permettant de calculer les nombres de Betti pour l'union de deux régions adjacentes. Enfin nous avons intégré ce calcul dans le processus de fusion de régions afin de pouvoir contrôler les nombres de Betti des régions à obtenir.

Nous avons finalement regroupé toutes les opérations et méthodes de segmentation définies dans cette thèse dans une chaîne de traitements afin de résoudre un problème de segmentation d'images médicales. Il s'agit de la segmentation du volume d'une tumeur cérébrale dans des images TEP. La chaîne mise en place est un exemple concret des différentes méthodes et de leur enchaînement afin de résoudre un problème de segmentation complexe.

Dans le futur nous souhaitons explorer différentes idées pour améliorer nos résultats et donner des suites à ces travaux.

Les pistes de travail autour des méthodes de fusion de régions passent sans doute par l'étude du calcul incrémental d'autres propriétés comme par exemple l'aire des différentes surfaces. Les méthodes incrémentales ont déjà montré leurs performances (par exemple lors de l'extraction d'une carte topologique) pour le calcul de la caractéristique d'Euler et des nombres de Betti et il semble intéressant de travailler avec d'autres propriétés. Du point de vue des outils, il peut être intéressant de disposer d'opérations de fusion adaptées à des configurations particulières. Nous pouvons notamment envisager une opération de fusion simplifiée dans le cas où les relations d'imbrication entre les régions n'évoluent pas. La détection *a priori* de ces configurations permettrait d'utiliser automatiquement la meilleure méthode pour réaliser l'opération.

En ce qui concerne les opérations de division de régions, nous envisageons d'étudier l'insertion de cellules uniquement lorsque cela est nécessaire. Actuellement, même dans le cas de la division par un guide, l'ensemble de la surface de la région à diviser est éclaté, c'est-à-dire que chaque cellule du bord de la région est rendue élémentaire. Généralement, beaucoup de ces cellules sont supprimées lors de la phase de simplification qui suit la division. Nous envisageons d'insérer uniquement les cellules qui permettent d'obtenir la division de la région selon le guide. De plus, la possibilité de ne pas faire la simplification quand nous enchaînons les divisions sur un même volume permettrait d'améliorer les temps de traitement.

La première définition des points ML-Simples ouvre de nombreuses perspectives pour étendre ces travaux. Nous sommes notamment intéressés par une définition plus permissive permettant de déplacer

des arêtes et des sommets de la partition toujours sans changer sa topologie. L'étude d'une définition des points ML-Simples en 2D s'avère intéressante d'un point de vue pratique par exemple pour des applications de déformation dans les cartes topologiques 2D. La définition et l'étude de nouvelles énergies adaptées à un modèle déformable basé sur les cartes (utilisant donc les informations apportées par le modèle) sont également des pistes que nous souhaitons explorer dans le futur.

En ce qui concerne la segmentation, d'une manière générale, il faut nous pencher sur la mise en œuvre de nouveaux critères dans les cartes topologiques. Pour poursuivre les travaux sur les critères topologiques basés sur les nombres de Betti, une étude des algorithmes pourrait être réalisée afin d'accélérer les traitements. Par exemple certains algorithmes, comme les opérations de comptage des cellules, peuvent être réalisés en parallèle afin de diminuer les temps d'exécution. À plus long terme, il est intéressant d'étudier les algorithmes de calcul pour d'autres invariants topologiques dans le modèle des cartes topologiques. Nous pensons notamment au calcul des générateurs des groupes d'homologie qui fournirait une indication sur la manière de découper une région pour résoudre des défauts topologiques.

Pour achever les travaux sur la chaîne de segmentation, il est maintenant nécessaire de compléter l'étude en s'appuyant sur l'expertise de médecins spécialistes afin d'améliorer les méthodes de segmentation et les critères utilisés. Il est également important de faire valider les résultats obtenus. À plus long terme, l'usage de la carte topologique pour résoudre d'autres problèmes similaires semble intéressant, notamment pour étudier les images que nous avons écartées au début de l'étude.

Nous sommes également intéressés par des travaux en cours sur l'utilisation du modèle des cartes topologiques dans le cadre d'un processus ponctuel. Des pistes sont déjà lancées dans ces directions. Les cartes topologiques apportent des avancées considérables dans le calcul de certaines propriétés, notamment sur la prise en compte du nombre de cavités ou de l'adjacence (et de la surface d'adjacence) entre les régions. Ces travaux ouvrent de nouvelles perspectives pour l'intégration des cartes topologiques pour d'autres méthodes de traitement d'images.

# RÉFÉRENCES BIBLIOGRAPHIQUES


[AFG99] Ehoud AHRONOVITZ, Christophe FIORIO et Sylvain GLAIZE : Topological operators on the topological graph of frontiers. *In* Gilles BERTRAND, Michel COUPRIE et Laurent PERROTON, éditeurs : *DGCI*, volume 1568 de *Lecture Notes in Computer Science*, pages 207–217. Springer, 1999.

[BBH08] Valentin E. BRIMKOV, Reneta P. BARNEVA et Herbert A. HAUPTMAN, éditeurs. *Combinatorial Image Analysis, 12th International Workshop, IWCIA 2008, Buffalo, NY, USA, April 7-9, 2008. Proceedings*, volume 4958 de *Lecture Notes in Computer Science*. Springer, 2008.

[BCKO08] Mark de BERG, Otfried CHEONG, Marc van KREVELD et Mark OVERMARS : *Computational Geometry : Algorithms and Applications. (Third edition).* Springer-Verlag, Heidelberg, Germany, third edition édition, 2008.

[BD96] Jean-Pierre BRAQUELAIRE et Jean-Philipe DOMENGER : Representation of region segmented images with discrete maps. Rapport technique, LaBRI, Université de Bordeaux I, 351, cours de la Libération x 33405, Talence Cedex – France, 1996.

[BD97] Luc BRUN et Jean-Philipe DOMENGER : A new split and merge algorithm with topological maps and inter-pixel boundaries. *In The fifth International Conference in Central Europe on Computer Graphics and Visualization*, february 1997.

[BDD01] Achille BRAQUELAIRE, Pascal DESBARATS et Jean-Philipe DOMENGER : 3d split and merge with 3-maps. *In Workshop on Graph-Based Representations in Pattern Recognition*, pages 32–43, Ischia, Italy, may 2001. IAPR-TC15.

[Ber94] Gilles BERTRAND : Simple points, topological numbers and geodesic neighborhoods in cubic grids. *Pattern Recognition Letters*, 15(10):1003–1011, 1994.

[BM94] Gilles BERTRAND et Grégoire MALANDAIN : A new characterization of three-dimensional simple points. *Pattern Recognition Letters*, 15(2):169–175, 1994.

[CB09] Michel COUPRIE et Gilles BERTRAND : New characterizations of simple points in 2D, 3D and 4D discrete spaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(4):637–648, avril 2009.

[CBCN08] Jean COUSTY, Gilles BERTRAND, Michel COUPRIE et Laurent NAJMAN : Fusion graphs : merging properties and watersheds. *Journal of Mathematical Imaging and Vision*, 30(1): 87–104, janvier 2008.

[CBG99] Yann COINTEPAS, Isabelle BLOCH et Line GARNERO : Joined segmentation of cortical surface and brain volume in mri using a homotopic deformable cellular model. *In 3DIM*, pages 240–251. IEEE Computer Society, 1999.

[CBNC09] Jean COUSTY, Gilles BERTRAND, Laurent NAJMAN et Michel COUPRIE : Watershed Cuts : Minimum Spanning Forests and the Drop of Water Principle. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(8):1362–1374, 2009.

[CLR89] Thomas H. CORMEN, Charles E. LEISERSON et Ronald L. RIVEST : *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company, 1989.

[Coi99] Yann COINTEPAS : *Modélisation homotopique et segmentation 3D du cortex cérébral à partir d'IRM pour la résolution des problèmes directs et inverses en EEG et en MEG*. Thèse de doctorat, Ecole Nationale Supérieure des Telecommunications, 1999.

[Dam01] Guillaume DAMIAND : *Définition et étude d'un modèle topologique minimal de représentation d'images 2d et 3d*. Thèse de doctorat, Université Montpellier II, décembre 2001.

[Dam08a] G. DAMIAND : Topological model for 3d image representation : Definition and incremental extraction algorithm. *Computer Vision and Image Understanding*, 109(3):260–289, March 2008.

[Dam08b] Guillaume DAMIAND : Topological model for 3d image representation : Definition and incremental extraction algorithm. *Computer Vision and Image Understanding*, 109(3): 260–289, 2008.

[DD08a] Alexandre DUPAS et Guillaume DAMIAND : Comparison of local and global region merging in the topological map. *In* BRIMKOV *et al.* [BBH08], pages 420–431.

[DD08b] Alexandre DUPAS et Guillaume DAMIAND : First results for 3d image segmentation with topological map. *In* David COEURJOLLY, Isabelle SIVIGNON, Laure TOUGNE et Florent DUPONT, éditeurs : *DGCI*, volume 4992 de *Lecture Notes in Computer Science*, pages 507–518. Springer, 2008.

[DD09] Alexandre DUPAS et Guillaume DAMIAND : Region merging with topological control. *Discrete Applied Mathematics*, 157(16):3435–3446, 2009.

[DDL09] Alexandre DUPAS, Guillaume DAMIAND et Jacques-Olivier LACHAUD : Multi-label simple points definition for 3d images digital deformable model. *In* Srecko BRLEK, Christophe REUTENAUER et Xavier PROVENÇAL, éditeurs : *DGCI*, volume 5810 de *Lecture Notes in Computer Science*, pages 156–167. Springer, 2009.

[DE95] Cecil Jose A. DELFINADO et Herbert EDELSBRUNNER : An incremental algorithm for betti numbers of simplicial complexes on the 3-sphere. *Computer Aided Geometric Design*, 12(7):771–784, 1995.

[Des01] Pascal DESBARATS : *Structuration des images segmentées 3D discrètes*. Thèse de doctorat, Université Bordeaux I, décembre 2001.

[DL03] Guillaume DAMIAND et Pascal LIENHARDT : Removal and contraction for n-dimensional generalized maps. *In* Ingela NYSTRÖM, Gabriella Sanniti di BAJA et Stina SVENSSON, éditeurs : *DGCI*, volume 2886 de *Lecture Notes in Computer Science*, pages 408–419. Springer, 2003.

[DPF08] Guillaume DAMIAND, Samuel PELTIER et Laurent FUCHS : Computing homology generators for volumes using minimal generalized maps. *In* BRIMKOV *et al.* [BBH08], pages 63–74.

[DPFL06] Guillaume DAMIAND, Samuel PELTIER, Laurent FUCHS et Pascal LIENHARDT : Topological map : An efficient tool to compute incrementally topological features on 3d images. *In*

Ralf REULKE, Ulrich ECKARDT, Boris FLACH, Uwe KNAUER et Konrad POLTHIER, éditeurs : *IWCIA*, volume 4040 de *Lecture Notes in Computer Science*, pages 1–15. Springer, 2006.

[DRS80] Charles R. DYER, Azriel ROSENFELD et Hanan SAMET : Region representation : boundary codes from quadtree. *Communications of the ACM*, 23(3):171–179, march 1980.

[Edm60] Jack EDMONDS : A combinatorial representation for polyhedral surfaces. *Notices of the American Mathematical Society*, 7:1, 1960.

[FB74] Raphael A. FINKEL et Jon Louis BENTLEY : Quad trees : A data structure for retrieval on composite keys. *Acta Inf.*, 4:1–9, 1974.

[FB04] Alexandre X. FALCÃO et Felipe P. G. BERGO : Interactive volume segmentation with differential image foresting transforms. *IEEE Trans. Med. Imaging*, 23(9):1100–1108, 2004.

[FH98] Pedro F. FELZENSZWALB et Daniel P. HUTTENLOCHER : Image segmentation using local variation. *In CVPR*, pages 98–104. IEEE Computer Society, 1998.

[FH04] Pedro F. FELZENSZWALB et Daniel P. HUTTENLOCHER : Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[Fio95] Christophe FIORIO : *Approche interpixel en analyse d'images : une topologie et des algorithmes de segmentation*. Thèse de doctorat, Université Montpellier 2, novembre 1995.

[Fio96] Christophe FIORIO : A topologically consistent representation for image analysis : the topological graph of frontiers. *In* Serge MIGUET, Annick MONTANVERT et Stéphane UBÉDA, éditeurs : *DGCI*, volume 1176 de *Lecture Notes in Computer Science*, pages 151–162. Springer, 1996.

[Fio08] Christophe FIORIO : *Modélisation, Analyse, Représentation des Images Numériques – Approche combinatoire de l'imagerie*. Habilitation à diriger des recherches, Université Montpellier 2, novembre 2008.

[FKN80] Henry FUCHS, Zvi M. KEDEM et Bruce F. NAYLOR : On visible surface generation by a priori tree structures. *In Proceedings of the 7th annual conference on Computer graphics and interactive techniques*, pages 124–133. ACM New York, NY, USA, 1980.

[FSdAL04] Alexandre X. FALCÃO, Jorge STOLFI et Roberto de ALENCAR LOTUFO : The image foresting transform : Theory, algorithms, and applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(1):19–29, 2004.

[GW90] C. GASQUET et P. WITOMSKI : *Analyse de Fourier et applications : filtrage, calcul numérique, ondelettes*. Masson, 1990.

[HP76] Steven L. HOROWITZ et Theodosios PAVLIDIS : Picture segmentation by a tree traversal algorithm. *J. ACM*, 23(2):368–388, 1976.

[HYW08] Chih-Yu HSU, Chih-Hung YANG et Hui-Ching WANG : Topological control of level set method depending on topology constraints. *Pattern Recognition Letters*, 29(4):537–546, 2008.

[KEP99] Remi K.-S. KWAN, Alan C. EVANS et G. Bruce PIKE : Mri simulation based evaluation and classifications methods. *IEEE Trans. Med. Imaging*, 18(11):1085–1097, 1999.

[KM95] Walter G. KROPATSCH et Herwig MACHO : Finding the structure of connected components using dual irregular pyramids. *In Discrete Geometry for Computer Imagery*, pages 147–158, *invited lecture*, september 1995.

[Köt02] Ullrich KÖTHE : Xpmaps and topological segmentation - a unified approach to finite topologies in the plane. *In* Achille J.-P. BRAQUELAIRE, Jacques-Olivier LACHAUD et Anne VIALARD, éditeurs : *DGCI*, volume 2301 de *Lecture Notes in Computer Science*, pages 22–33. Springer, 2002.

[Kov89] Vladimir A. KOVALEVSKY : Finite topology as applied to image analysis. *Computer Vision, Graphics, and Image Processing*, 46:141–161, 1989.

[Kov08] Vladimir A. KOVALEVSKY : *Geometry of Locally Finite Spaces*. Publishing House Dr. Baerbel Kovalevski, Berlin, Germany, 2008.

[KSI+96] Ron KIKINIS, Martha Elizabeth SHENTON, Dan V. IOSIFESCU, Robert W. MCCARLEY, Pairash SAIVIROONPORN, Hiroto H. HOKAMA, Andre ROBATINO, David METCALF, Cindy WIBLE, Chiara M. PORTAS, Robert M. DONNINO et Ferenc A. JOLESZ : A digital brain atlas for surgical planning, model-driven segmentation, and teaching. *IEEE Trans. Vis. Comput. Graph.*, 2(3):232–241, 1996.

[KWT88] Michael KASS, Andrew P. WITKIN et Demetri TERZOPOULOS : Snakes : Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[Lac03] Jacques-Olivier LACHAUD : Coding cells of digital spaces : a framework to write generic digital topology algorithms. *Electronic Notes in Discrete Mathematics*, 12:337–348, 2003.

[Lan03] Jens LANGNER : Development of a Parallel Computing Optimized Head Movement Correction Method in Positron Emission Tomography. Master of computer science thesis, University of Applied Sciences Dresden and Research Center Dresden-Rossendorf, 2003.

[MK05] Hans MEINE et Ullrich KÖTHE : The geomap : A unified representation for topology and geometry. *In* Luc BRUN et Mario VENTO, éditeurs : *GbRPR*, volume 3434 de *Lecture Notes in Computer Science*, pages 132–141. Springer, 2005.

[MK06] Hans MEINE et Ullrich KÖTHE : A new sub-pixel map for image analysis. *In* Ralf REULKE, Ulrich ECKARDT, Boris FLACH, Uwe KNAUER et Konrad POLTHIER, éditeurs : *IWCIA*, volume 4040 de *Lecture Notes in Computer Science*, pages 116–130. Springer, 2006.

[MPA08] Sanae MIRI, Nicolas PASSAT et Jean-Paul ARMSPACH : Topology-preserving discrete deformable model : Application to multi-segmentation of brain mri. *In ICISP*, pages 67–75, 2008.

[Mun84] James R. MUNKRES : *Elements of Algebraic Topology*. Addison-Wesley, 1984.

[OS88] Stanley OSHER et James A. SETHIAN : Fronts propagating with curvature dependent speed : algorithms based on Hamilton-Jacobi formulations. *Journal of computational physics*, 1988.

[PCB08] Nicolas PASSAT, Michel COUPRIE et Gilles BERTRAND : Minimal simple pairs in the 3D cubic grid. *Journal of Mathematical Imaging and Vision*, 32(3):239–249, novembre 2008.

[Pel06] Samuel PELTIER : *Calcul de groupes d'homologie sur des structures simpliciales, simploïdales et cellulaires.* Thèse de doctorat, Université de Poitiers, June 2006.

[Pet06] Frédéric PETIT : Segmentation hiérarchique d'images couleurs. Mémoire de master recherche, Université de Poitiers, 2006.

[Pri09] Keith PRICE : Annotated computer vision bibliography, septembre 2009. http ://www.visionbib.com/bibliography/.

[RBB99] R.M. RAO, A.S. BOPARDIKAR et T. BOROS : Wavelet transforms : Introduction to theory and applications. *Journal of Electronic Imaging*, 8:478, 1999.

[RK82] Azriel ROSENFELD et Avinash C. KAK : *Digital Picture Processing*. Academic Press, 1982.

[Ros70] Azriel ROSENFELD : Connectivity in digital pictures. *J. ACM*, 17(1):146–160, 1970.

[Ros74] Azriel ROSENFELD : Adjacency in digital pictures. *Information and Control*, 26(1):24–33, 1974.

[RY90] KR RAO et P. YIP : *Discrete cosine transform*. Academic Press San Diego, 1990.

[Sam84] Hanan SAMET : The quadtree and related hierarchical data structures. *Computing surveys*, 16(2):187–260, 1984.

[Ség08] Florent SÉGONNE : Active contours under topology control - genus preserving level sets. *International Journal of Computer Vision*, 79(2):107–117, 2008.

[Ser83] Jean SERRA : *Image Analysis and Mathematical Morphology*. Academic Press, Inc., Orlando, FL, USA, 1983.

[Ser06] Jean SERRA : A lattice approach to image segmentation. *Journal of Mathematical Imaging and Vision*, 24(1):83–130, 2006.

[Tar75] Robert Endre TARJAN : Efficiency of a good but not linear set union algorithm. *J. ACM*, 22(2):215–225, 1975.

[TF88] Demetri TERZOPOULOS et Kurt W. FLEISCHER : Deformable models. *The Visual Computer*, 4(6):306–331, 1988.

[VS91] Luc VINCENT et Pierre SOILLE : Watersheds in digital spaces : An efficient algorithm based on immersion simulations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(6):583–598, 1991.

[Zom01] Afra ZOMORODIAN : *Computing and Comprehending Topology : Persistence and Hierarchical Morse Complexes*. Phd thesis, University of Illinois, 2001.

## Opérations et Algorithmes pour la Segmentation Topologique d'Images 3D


### Résumé

Une carte topologique 3D est un modèle servant à représenter la partition en régions d'une image 3D pour le traitement d'images. Dans ce travail, nous développons des outils permettant de modifier la partition représentée par une carte topologique, puis nous utilisons ces outils afin de proposer des algorithmes de segmentation intégrant des critères topologiques. Dans une première partie, nous proposons trois opérations. La fusion de régions est définie avec une approche locale adaptée à une utilisation interactive et une approche globale pour une utilisation automatisée comme lors d'une segmentation. La division de régions est proposée avec une méthode d'éclatement en voxels et la division à l'aide d'un guide. Enfin, la déformation de la partition est basée sur la définition de points ML-Simples : des voxels pouvant changer de région sans modifier la topologie de la partition. À l'aide de ces opérations, nous mettons en œuvre dans une seconde partie des algorithmes de segmentation d'images utilisant les cartes topologiques. Notre première approche adapte au modèle des cartes topologiques un algorithme existant qui utilise un critère basé sur la notion de contraste. Nous proposons ensuite des méthodes de calcul d'invariants topologiques sur les régions : les nombres de Betti. Grâce à eux, nous développons un critère topologique de segmentation permettant de contrôler le nombre de tunnels et de cavités des régions. Enfin, nous illustrons les possibilités de tous nos outils en mettant en place une chaîne de traitement pour la segmentation de tumeurs cérébrales dans des images médicales.



**Mots-clefs :** modèles topologiques, cartes combinatoires, traitement d'images, imagerie médicale, nombres de Betti, segmentation, points simples, modèles déformables


## Operations and Algorithms for Topological Segmentation of 3D Images


### Abstract

A 3D topological map is a model used in image processing which represents the partition of a 3D image into regions. In this work, we introduce some tools that allow to modify a partition presented by a topological map, and we use these tools to propose segmentation algorithms implementing topological criteria. In a first part, we propose three operations. The region merging is defined with a local approach suited for interactive use, and a global approach suited for automatic processing like image segmentation. The region splitting is introduced with a burst into voxel approach, and the split with a guide. Last, a deformation of the partition based on the definition of ML-Simple points: voxels that can be flipped of region without changing the topology of the partition. With these operations, we implement in a second part image segmentation processes using topological maps. First we adapt to our model an existing algorithm using a criterion based on the notion of contrast. Then, we propose methods to compute topological invariants of regions: the Betti numbers. Using these methods we implement a topological criterion that controls the number of tunnels and cavities of the regions. Last, we give an overview of the possibilities of our tools by creating a toolchain to segment brain tumors in medical images.



**Keywords:** topological model, combinatorial map, image processing, 3D medical imaging, Betti numbers, segmentation, simple points, deformable model