Chapitre 5

La carte topologique en dimension 3

Nous définissons dans un premier temps la carte topologique en dimension 3, en étendant notre définition progressive donnée en dimension 2. Cette extension ne pose pas de problème majeur, à l'exception des problèmes de déconnexion. Ce type de problème se produit déjà en dimension 2, mais uniquement entre des régions, et est résolu par l'adjonction d'un arbre d'inclusion. Pour la dimension 3, ce problème se pose également pour les régions et la solution est toujours l'adjonction d'un arbre d'inclusion. Mais un autre problème de déconnexion apparaît pour les faces, et n'a pas d'équivalent en dimension 2. La résolution de ce problème consiste à conserver des éléments fictifs, ici des *arêtes fictives*, permettant que chaque face reste homéomorphe à un disque topologique. Nous étudions précisément ces arêtes fictives, afin de déterminer leur rôle et leurs propriétés. Elles jouent en effet un rôle crucial dans l'obtention d'une représentation minimale.

Nous présentons ensuite des algorithmes d'extraction. L'algorithme naïf ne pose pas de problème particulier, comme en dimension 2. L'algorithme optimal est également simple à définir en dimension 3, étant donné qu'il se résume à un balayage de l'image en exécutant le code correspondant au précode courant. Mais comme pour la dimension 2, la difficulté réside dans la définition des précodes à traiter pour chaque niveau de carte. Nous avons effectué une étude précise de ces précodes afin de définir exactement ceux à traiter pour extraire chaque niveau de carte, mais également afin de factoriser ceux pouvant se traiter de manière similaire.

Le plan de ce chapitre est volontairement très proche de celui du chapitre précédent présentant la carte topologique en dimension 2 pour mieux mettre en évidence les similarités ou les différences entre ces deux dimensions. Nous commençons section 5.1 par un rappel sur les images 3d segmentées en régions, et sur les frontières dans ces images, qui peuvent être définies à l'aide de la notion d'intervoxel. Mais nous avons vu en dimension 2 que la définition des frontières interpixel est inutile pour définir la carte topologique. Nous ne définissons donc pas formellement cette notion. Section 5.2, nous définissons la carte topologique en dimension 3. Puis section 5.3 nous traitons les problèmes de déconnexion et étudions en détail les éléments fictifs. Section 5.4 nous présentons quelques plongements possibles pour la carte topologique. Section 5.5 nous étudions l'algorithme d'extraction naïf qui découle, comme pour la dimension 2, directement de la définition de la carte topologique. Puis section 5.6 nous présentons l'algorithme optimal d'extraction, ainsi que les précodes à traiter pour extraire chaque niveau de carte. Section 5.7, nous donnons un troisième algorithme d'extraction, qui est une solution intermédiaire intéressante entre les deux premiers algorithmes. Enfin section 5.8 nous analysons quelques résultats et étudions l'évolution de quelques caractéristiques sur les différents niveaux de carte, la section 5.9 conclut ce chapitre.

5.1 Images, segmentation en régions et intervoxel

Il s'agit de représenter des images 3d segmentées en régions. Nous rappelons d'abord cette notion, puis présentons la notion d'intervoxel et discutons de la notion de frontière en dimension 3, en montrant en quoi la définition de ces frontières est plus complexe qu'en dimension 2.

5.1.1 La segmentation en régions

Un *voxel* est un point coloré de l'espace discret $\mathbb{Z}^3$, et une *image* est un ensemble fini de voxels. Nous utilisons en 3d les notions de 6-*voisinage* et de 6-*connexité*. La notion de *segmentation en régions* est similaire à celle présentée en dimension 2, sauf que chaque élément de la partition, c'est-à-dire chaque région, doit maintenant être un ensemble 6-connexe. Comme pour la dimension 2, nous considérons que l'image est incluse dans une *région infinie* R_0 contenant l'ensemble des voxels n'appartenant pas à l'image. La notion d'inclusion s'étend également immédiatement en dimension 3.

Définition 22 (inclusion) *Une région R_i est* incluse *dans une région R_j si et seulement si tout chemin 6-connexe allant d'un voxel de R_i vers un voxel de R_0 (la région infinie) possède au moins un voxel appartenant à la région R_j.*

Nous ne revenons pas sur les propriétés de cette relation, qui sont les mêmes que pour la dimension 2. Il en va de même pour la relation *d'inclusion directe* présentée définition 13, définie à partir de la relation d'inclusion, et qui reste donc valable ici.

5.1.2 Intervoxel et frontières

L'*intervoxel* est simplement l'extension en dimension 3 de l'interpixel défini en dimension 2. L'espace de dimension 3 est composé de voxels, et nous considérons la subdivision de cet espace en cellules de dimension inférieure ou égale. Cette subdivision représente des *voxels*, cellules de dimension 3, des *surfels*, cellules de dimension 2 « entre » deux voxels, des *lignels*, cellules de dimension 1 « entre » deux surfels et des *pointels*, cellules de dimension 0 « entre » deux lignels. Nous parlons, comme pour la dimension 2, de *i-cellule* pour une cellule de dimension i. Ces différentes cellules sont présentées figure 5.1.

Les notions d'*adjacence* et d'*incidence*, présentées en dimension 2 section 4.1.2, restent valides en dimension 3. Informellement, une *frontière* entre deux régions est l'ensemble des

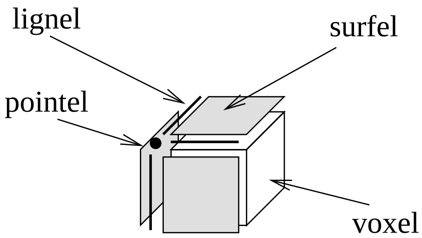


Fig. 5.1 – Les divers éléments intervoxel en dimension 3.

cellules se trouvant entre ces deux régions. En dimension 2, une frontière est une courbe 1d. En effet, comme chaque région est 4-connexe, de par la définition de la segmentation en régions, une frontière ne peut pas être composée seulement d'un pointel. En dimension 3, une frontière est maintenant une surface, du fait de la définition de la segmentation en régions qui implique que chaque région soit 6-connexe. C'est ici qu'apparaît une difficulté supplémentaire. En effet, il faut étudier les bords de ces surfaces frontières, car ce sont eux qui sont représentés par les cartes combinatoires.

Le problème de définition de surface en dimension 3 est difficile et relève du domaine de la topologie discrète. Il a été étudié par de nombreux travaux qui utilisent la topologie discrète afin de définir les surfaces comme des ensembles de 2-cellules, ou surfels, qui sont les bords des voxels. Des algorithmes de détection de telles surfaces par suivi de contour ont tout d'abord été proposés par [AFH81, GU89]. Les preuves de la validité de ces algorithmes ont été publiées seulement quelques années plus tard dans [HW83, KU92] de par les difficultés de celles-ci. Mais ces deux approches ne s'intéressent pas à la définition théorique de ces surfaces.

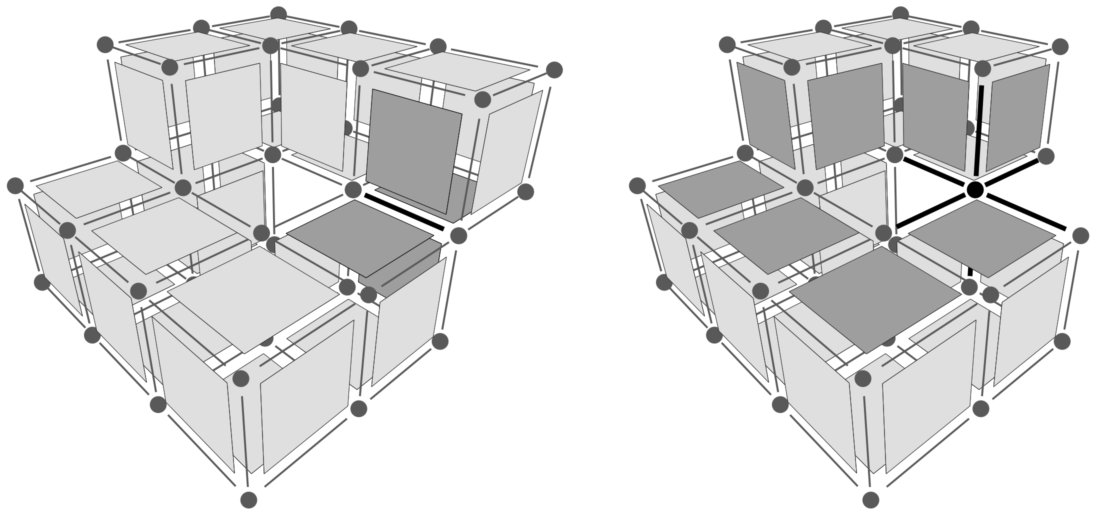
D'autres travaux plus théoriques se sont penchés sur la définition des surfaces dans le cadre de la topologie interpixel. [KF81] prouve qu'un ensemble de faces est une 2-variété si elle satisfait aux trois conditions suivantes :

- les faces s'intersectent seulement par des arêtes ou des sommets ;
- chaque arête appartient seulement à deux faces ;
- les faces autour d'un même sommet peuvent être ordonnées de sorte que deux faces consécutives pour cet ordre s'intersectent par une arête incidente à ce sommet.

Ces surfaces ont été étudiées également par [RKW91]. Mais elles ne sont pas utilisables pour notre travail, car elles sont définies uniquement pour des objets noir et blanc, afin d'utiliser une connexité différente pour l'objet et pour le fond.

Nous présentons ici les propriétés que nous aimerions voir satisfaites par les surfaces que nous traitons et montrons les problèmes que cela soulève. Cela nous permet lors de la présentation de la carte topologique section 5.2, de montrer comment ces problèmes sont résolus avec les cartes combinatoires alors que ce n'est pas le cas en intervoxel.

Il faut tout d'abord que la définition des surfaces en dimension 3 soit « cohérente » avec les objets en dimension 2, tout simplement car une surface en dimension 3 est un objet 2d. Mais cette propriété pose un premier problème. En effet, en dimension 2 chaque lignel d'un objet 2d est incident à un ou deux surfels de cet objet. Mais cela n'est plus vrai



a. Un lignel incident à 4 surfels. b. Un pointel incident à 6 lignels.

Fig. 5.2 – Deux exemples de problèmes pour la définition des surfaces.

pour les surfaces en dimension 3, comme nous pouvons voir sur l'exemple présenté figure 5.2.*a*. Cet exemple montre un objet 3d, représenté par sa surface. Nous pouvons voir qu'elle possède un lignel incident à 4 surfels, ce qui est impossible en dimension 2. Cela pose des problèmes dans la définition des surfaces, car nous devons différencier ce type de cas de ceux non valides de surfaces repliées sur elles-mêmes. Une solution, utilisée par exemple dans [AAF95], consiste dans ce cas à ne pas considérer ce lignel comme faisant partie de la surface. Mais cette solution n'est pas très satisfaisante, car des trous apparaissent localement, ce qui n'est pas cohérent avec les définitions existantes en dimension 2.

Il faut également que les bords des surfaces soient des courbes de dimension 1. En effet, les cartes combinatoires représentent le bord des faces. Cette propriété pose un autre problème, comme nous pouvons le voir figure 5.2.*b*. Dans ce cas, un pointel est incident à 6 lignels, et la courbe 1d représentant le bord de la surface en gris foncé[1] passe par 4 de ces lignels, et deux fois par le pointel. Il faut donc redéfinir la notion de courbe 1d afin d'autoriser ce type de courbe, tout en interdisant certaines configurations impossibles. Un problème similaire se pose pour l'objet de la figure 5.2.*a*, où un lignel appartient deux fois dans la même courbe 1d.

Définir les surfaces en dimension 3 est donc un problème difficile. Mais nous avons vu au chapitre précédent, lors de la définition de la carte topologique en dimension 2, que la définition des frontières interpixel n'est pas nécessaire pour définir cette carte. En effet, cette définition s'effectue uniquement à partir de la carte complète, qui représente toutes les cellules de l'espace sans utiliser la notion de frontière, puis par application de plusieurs opérations de fusions qui n'ont plus aucun lien avec les frontières interpixel. Étant donné que nous étendons cette définition de la carte topologique en dimension 3, nous n'avons

[1] Cette surface pourrait être une frontière entre l'objet 3d représenté sur cette figure et un autre objet posé dessus.

donc pas besoin de la définition précise des frontières intervoxel car nous conservons le même principe de définition progressive basé sur nos différents niveaux de simplification.

Nous utilisons ces frontières afin de prouver que la carte topologique représente bien toutes les propriétés topologiques de l'image. Pour cela, nous définissons partiellement ces frontières, uniquement pour leurs surfels. Une *frontière* intervoxel entre deux régions R_i et R_j est un ensemble de surfaces, tel que chaque surfel de celles-ci est incident exactement à un voxel de R_i et à un voxel de R_j. Nous appelons *surface frontière* une surface appartenant à une des frontières de l'image. Ces surfaces sont maximales : tout surfel de l'image incident à un voxel de R_i et à un voxel de R_j appartient forcément à une de ces surfaces, et deux surfels adjacents appartenant à une frontière appartiennent forcément à la même surface frontière.

Cette définition est incomplète, tout d'abord car elle s'appuie sur la définition de surface que nous n'avons pas donnée. De ce fait, nous n'avons pas de contraintes sur les lignels et pointels, et ne pouvons donc pas définir le bord de ces frontières en termes de courbes 1d. Cette définition serait nécessaire afin de définir la carte topologique de manière directe. Nous conservons les mêmes notations déjà présentées en dimension 2. Nous notons $frontière(R_i, R_j) = \{s_1, \ldots, s_l\}$ la frontière entre les régions R_i et R_j composée des surfaces $s_1 \ldots s_l$, posons $frontière(R_i, R_i) = \emptyset$ et notons $Frontières(R_i) = \cup_{j=1}^{k} frontière(R_i, R_j)$. Nous parlons de *surfel frontière* pour un surfel appartenant à une frontière de l'image.

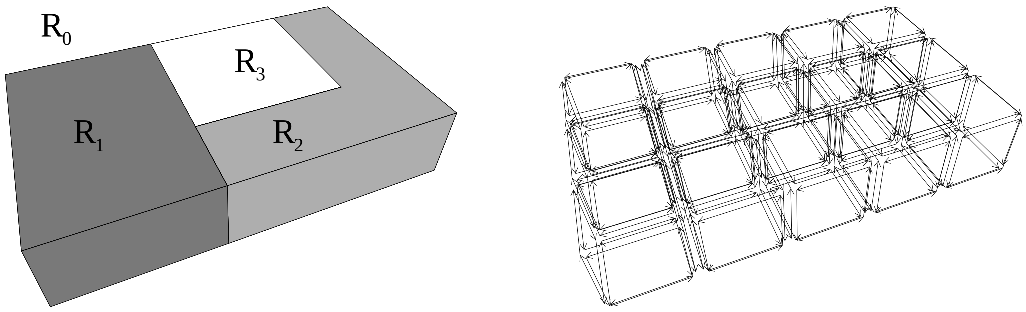
5.2 Les niveaux de simplification

Afin de définir la carte topologique en dimension 3, nous reprenons la définition progressive donnée en dimension 2, et la modifions afin de tenir compte des différences avec la dimension 3. Nous partons de la carte combinatoire complète représentant chaque cellule de l'espace intervoxel, que nous appelons comme en dimension 2 la carte de niveau 0. Puis nous la simplifions progressivement au moyen de l'opération de fusion, en construisant des niveaux de simplification intermédiaires, jusqu'à obtenir la carte minimale qui ne peut plus être simplifiée sans perte d'information topologique.

Nous présentons dans un premier temps, comme en dimension 2, uniquement l'aspect topologique, afin de bien le séparer des aspects géométriques. Le lien avec des éventuels modèles de plongements est l'objet de la section 5.4. Par rapport à la dimension 2, un problème de déconnexion supplémentaire apparaît. Il faut le résoudre afin de définir correctement la carte topologique. Comme pour le plongement, nous commençons par ne pas traiter ces problèmes de déconnexion, ce qui permet de définir simplement nos différents niveaux de carte. Puis, section 5.3, nous résolvons ces problèmes.

5.2.1 Le niveau 0 : la carte complète

Cette première carte est, comme en 2d, le point de départ de notre processus, et ne code pas les frontières intervoxel de l'image.



a. Une image 3d segmentée en régions. b. La carte de niveau 0 correspondante.

FIG. 5.3 – Le niveau 0 d'une image 3d.

Définition 23 (carte de niveau 0) *La* carte de niveau 0 *correspondant à une image de $n_1 \times n_2 \times n_3$ voxels, est la carte combinatoire ayant $n_1 \times n_2 \times n_3$ cubes, β_3-cousus entre eux lorsqu'ils sont adjacents. Chacun de ces cubes représente un voxel de l'image, et un volume « entourant » ces cubes, représente la région infinie.*

Cette carte code tous les éléments intervoxel de l'image. La figure 4.4.*a* représente une image 3d segmentée en régions, et la figure 4.4.*b* la carte de niveau 0 correspondante.

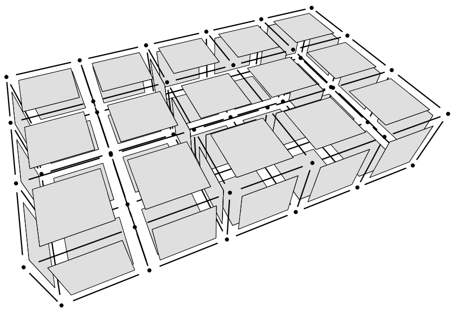
Cette carte est composée, pour une image de taille $n_1 \times n_2 \times n_3$, de $(n_1 \times n_2 \times n_3) + 1$. $n_1 \times n_2 \times n_3$ cubes composés chacun de 6 faces carrées, chacune composée par 4 brins, représentant l'ensemble des voxels de l'image. Le volume supplémentaire représente la région infinie, et est composé de $2 \times (n_1 \times n_2 + n_1 \times n_3 + n_2 \times n_3)$ faces carrées. Ces faces sont cousues par β_3 aux faces extérieures des cubes correspondants. Nous ne représentons pas ce volume infini sur toutes les figures comme par exemple sur la figure 5.3.*b*, afin de ne pas surcharger les schémas. Pour cette même raison, nous avons volontairement choisi une image de petite taille pour cet exemple, de $5 \times 3 \times 1$ voxels.

Cette carte de niveau 0 code donc tous les éléments intervoxel de l'image ainsi que toutes les relations d'adjacence et d'incidence. Comme en dimension 2, nous la simplifions progressivement afin de définir la carte topologique.

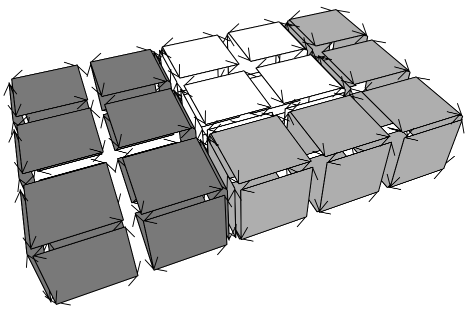
5.2.2 Le niveau 1 : la carte lignel

Pour définir cette première carte représentant les frontières intervoxel de l'image, nous « supprimons » les faces de la carte de niveau 0 représentant des surfels n'appartenant pas à une surface frontière de l'image. Comme pour la dimension 2, nous utilisons l'opération de fusion, détaillée section 5.5.

Définition 24 (carte de niveau 1) *La* carte de niveau 1 *est la carte obtenue à partir de la carte de niveau* 0 *en fusionnant chaque couple de volumes adjacents appartenant à la même région.*



a. L'image et ses frontières intervoxel.



b. La carte de niveau 1 correspondante.

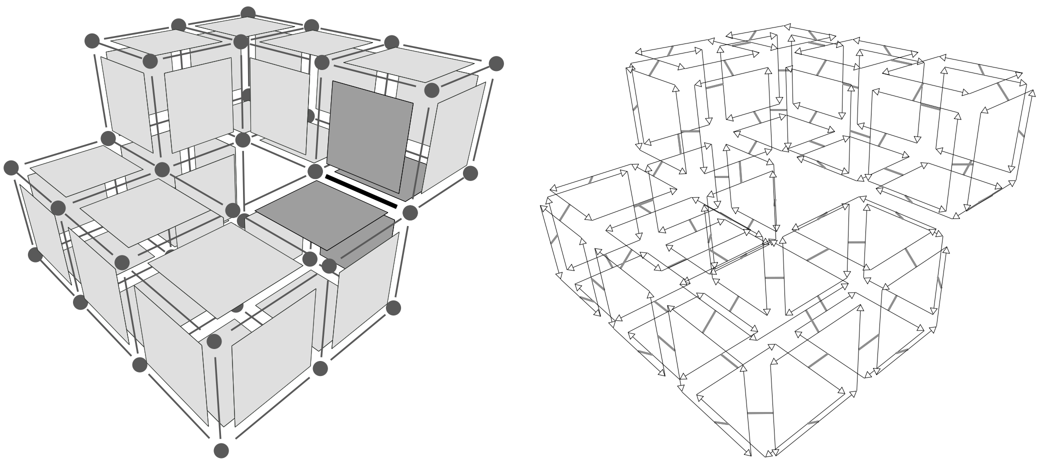
Fig. 5.4 – La carte de niveau 1.

Nous pouvons voir figure 5.4.a notre image d'exemple, présentée avec ses éléments intervoxel, et figure 5.4.b la carte de niveau 1 correspondante.

Nous vérifions facilement que chaque face de cette carte représente bien un surfel frontière de l'image. En effet, pour la carte de niveau 0, chaque face représente un surfel de l'image. Pour obtenir la carte de niveau 1, nous avons fusionné chaque couple de volumes adjacents et de même région. Cette fusion a donc entraîné la disparition des faces entre deux voxels de même région. Il ne reste donc plus, dans la carte de niveau 1, que les faces entre deux voxels de régions différentes qui représentent bien toutes des surfels frontière. De plus, chaque arête de cette carte est de longueur unitaire, car cette propriété est vérifiée par construction de la carte de niveau 0, et nous n'avons pas modifié la longueur des arêtes entre ces deux niveaux. De ce fait, chaque arête de la carte de niveau 1 représente un lignel de l'image. C'est pour cette raison que nous appelons cette carte la *carte lignel*.

Comme pour la dimension 2, cette carte de niveau 1 représente chaque région de l'image par une surface extérieure, et éventuellement plusieurs surfaces intérieures. Pour les surfaces extérieures, cette remarque découle simplement du fait que chaque région de l'image est finie et possède donc une surface extérieure, à l'exception de la région infinie R_0. Les surfaces intérieures apparaissent seulement lorsqu'une région est totalement incluse dans une autre. Dans ce cas, la surface extérieure est déconnectée des éventuelles surfaces intérieures. Il existe alors plusieurs composantes connexes dans la carte, et nous avons perdu les informations permettant de les positionner les unes par rapport aux autres. Cette information d'inclusion est une information topologique importante qu'il faut conserver. Nous verrons section 5.3 que ce problème se résout, comme pour la dimension 2, par l'adjonction d'un arbre d'inclusion des régions.

Cette carte lignel dotée d'un arbre d'inclusion représente bien toutes les propriétés topologiques contenues dans l'image. En effet, chaque surfel frontière est représenté, et toutes les relations d'incidence et d'adjacence sont correctement représentées par la carte combinatoire. Les problèmes évoqués lors de la définition de surface en intervoxel n'en sont plus ici, comme nous pouvons le vérifier figure 5.5.b où nous présentons la carte de niveau 1 de l'objet 3d présenté figure 5.5.a qui posait problème pour la définition de surface.



a. Un objet 3d. b. La carte de niveau 1 correspondante.

FIG. 5.5 – La carte de niveau 1 d'un objet 3d ayant un lignel incident à 4 surfels.

Sur cet exemple, nous considérons que l'objet 3d présenté figure 5.5.a est totalement inclus dans une autre région. De ce fait, il ne possède qu'une seule face frontière avec cette région. Afin de ne pas surcharger la figure, nous ne représentons pas les faces de cette région, qui seraient normalement des copies de toutes les faces de l'objet représenté, cousues entre elles par β_3. Nous avons représenté la relation β_2 par de petits segments gris.

Nous voyons sur la carte de niveau 1 représentant cet objet qu'il n'y a pas de problème relatif au lignel incident à 4 surfels. En effet, la surface frontière de l'objet est obtenue par un parcours d'orbite $< \beta_1, \beta_2 >$ d'origine un brin de cette surface. Nous parcourons donc correctement cette surface frontière. De plus, étant donné que nous ne considérons pas β_3, deux arêtes distinctes représentent ce lignel. Nous obtenons donc bien une surface, qui est une carte combinatoire 2d, où chaque arête est toujours incidente à deux faces. Du point de vue de l'objet, tout se passe au niveau de ce lignel comme s'il y avait deux arêtes topologiques différentes : nous ne représentons pas la 18-adjacence. Ce n'est plus vrai si nous considérons les brins de la région infinie, et donc cette arête non plus uniquement pour l'objet mais pour l'objet et son complémentaire. Dans ce cas, nous obtenons une seule arête topologique qui représente le lignel correspondant dans la subdivision de l'espace.

Remarquons que la construction de cette carte de niveau 1 à partir de la carte de niveau 0 n'utilise pas la notion de degré de cellule, mais plus simplement l'information de région. En effet, une face est toujours incidente à un ou deux volumes, et nous devons effectuer la fusion de deux volumes uniquement lorsqu'ils appartiennent à la même région. De plus, c'est uniquement pour ce niveau de carte que nous effectuons un lien entre la carte et l'image segmentée en régions. Les autres niveaux vont désormais être construits uniquement à partir de ce premier niveau de carte, en effectuant certaines fusions suivant le degré de cellules, sans utiliser l'image segmentée en régions.

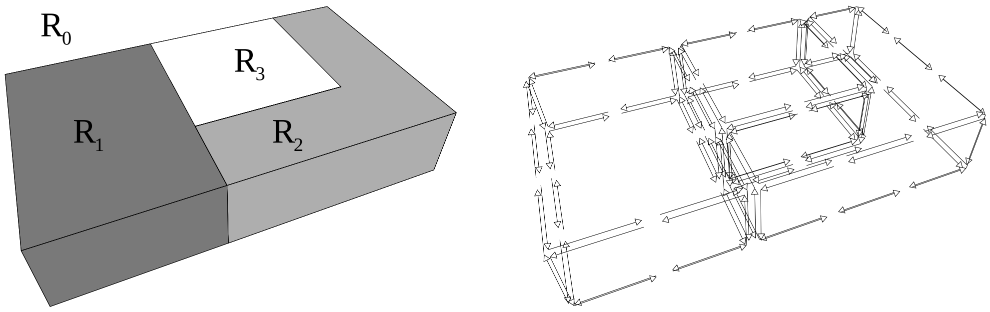


Fig. 5.6 – La carte de niveau 2.

5.2.3 Le niveau 2

Afin de définir la carte topologique, nous suivons le même principe que celui utilisé en dimension 2. Nous avons tout d'abord défini la carte de niveau 0 qui représente tous les éléments intervoxel de l'image, puis la carte de niveau 1 en fusionnant les volumes appartenant à la même région. Nous effectuons maintenant la fusion des cellules de degré deux, qui comme en dimension 2, représentent la même information topologique. Mais par rapport à la dimension 2, il existe un type de fusion supplémentaire. En effet, il existe la fusion de volumes, déjà utilisée pour définir la carte de niveau 1, la fusion de faces et la fusion d'arêtes. Il faut appliquer la fusion de faces avant la fusion d'arêtes, car sinon il n'existe pas d'arêtes incidentes à des sommets de degré deux étant donné que les faces n'ont pas encore été fusionnées. De manière générale, nous devons effectuer les fusions par dimension décroissante, en commençant par la fusion de même dimension que l'espace.

Définition 25 (carte de niveau 2) *La* carte de niveau 2 *est la carte obtenue à partir de la carte de niveau* 1*, en fusionnant chaque couple de faces adjacentes, coplanaires, et incidentes à une arête de degré un ou deux.*

Comme pour la dimension 2, nous commençons dans un premier temps par fusionner uniquement les cellules « alignées », i.e. les faces coplanaires, afin d'obtenir la définition de la carte des frontières comme l'un des niveaux de simplification. Rappelons que cette carte des frontières présente l'avantage de pouvoir être plongée simplement, en associant à chacun de ses sommets topologiques un point géométrique de l'espace, étant donné que chacune de ses arêtes représente un segment de droite. De plus, le fait de simplifier la carte petit à petit nous permet ensuite de décomposer plus finement l'étude sur les précodes.

La carte de niveau 2 de notre image d'exemple est présentée figure 5.6. Nous voyons sur cet exemple que les faces coplanaires incidentes à des arêtes de degré deux ont été fusionnées. Par exemple la face supérieure de la région R_1 est désormais représentée par une seule face composée de 10 brins.

Mais contrairement aux différents niveaux définis en dimension 2, nous fusionnons, pour cette définition, les faces adjacentes, coplanaires et incidentes à une arête de degré *un ou deux*. En effet, au fur et à mesure des fusions, certaines arêtes qui étaient de degré deux,

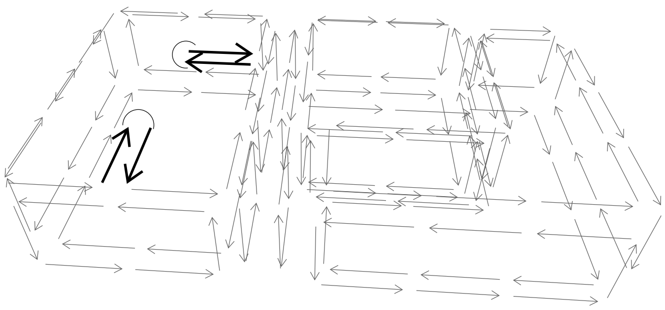


FIG. 5.7 – La carte de niveau 2 en cours de construction.

peuvent devenir de degré un, c'est-à-dire incidentes à une seule face, comme le montre la figure 5.7. Cette figure présente la carte de niveau 2 en cours de construction. Nous n'avons pas effectué toutes les fusions de faces coplanaires nécessaires. Il reste en effet les deux arêtes noires, incidentes à une seule et même face. Mais ces deux arêtes étaient de degré deux dans la carte de niveau 1 initiale, et sont devenues de degré un au fur et à mesure des fusions. Nous devons supprimer ces deux arêtes par deux fusions de faces, afin de ne pas dépendre de l'ordre dans lequel sont effectuées les fusions. De plus, ces arêtes n'ont aucune raison d'être conservées, car elles ne représentent pas une partie d'un bord d'une face frontière. Nous verrons au chapitre 6 qu'une autre solution consiste à considérer que l'arête est incidente localement à deux faces différentes.

Ces fusions de faces sont similaires aux fusions de faces effectuées en dimension 2 pour définir la carte de niveau 1. De ce fait, nous pouvons avoir le même problème de déconnexion que celui rencontré en dimension 2. Nous verrons section 5.3 comment il peut se produire et comment le résoudre.

5.2.4 Le niveau 3 : la carte des frontières

Après avoir fusionné les faces adjacentes, coplanaires, et incidentes à des arêtes de degré un ou deux, nous fusionnons maintenant les arêtes alignées incidentes à des sommets de degré deux. Nous obtenons alors une carte où chaque arête représente un segment de droite et peut donc être plongée uniquement par ses sommets. C'est cette carte qui est définie dans [BFP99] et qui est appelée *carte des frontières*.

Définition 26 (carte de niveau 3) *La* carte de niveau 3 *est la carte obtenue à partir de la carte de niveau* 2*, en fusionnant chaque couple d'arêtes adjacentes, alignées, et incidentes à un sommet de degré deux.*

La carte de niveau 3 de notre exemple est présentée figure 5.8. Comme pour la dimension 2, chaque arête d'une carte des frontières correspond à un segment de droite maximal du bord d'une frontière de l'image initiale. Contrairement à la définition de la carte de niveau 2, il ne faut pas ici fusionner les arêtes incidentes à un sommet de degré un. En effet, dans ce cas cette arête forme une boucle autour de ce sommet, et nous ne pouvons ni

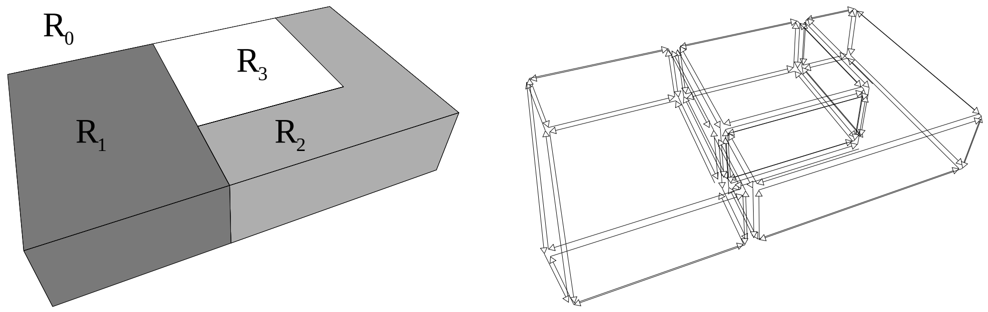


FIG. 5.8 – La carte de niveau 3.

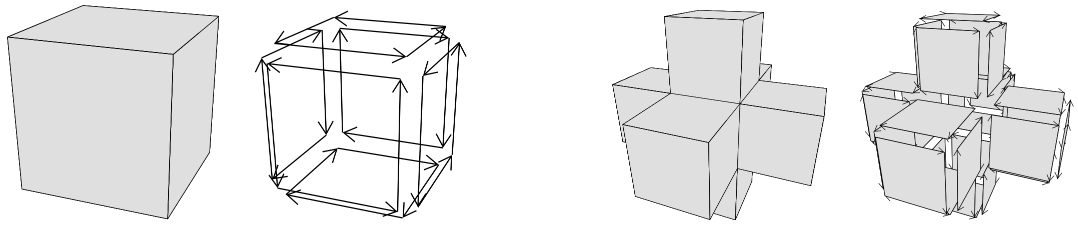


FIG. 5.9 – Deux objets topologiquement équivalents et leurs cartes des frontières.

ne devons effectuer la fusion de cette arête avec elle-même. Remarquons enfin que, comme pour la dimension 2, cette fusion d'arêtes ne peut pas entraîner de déconnexion étant donné qu'elle revient à étirer les arêtes, tout en conservant exactement les mêmes relations topologiques.

Ce niveau de carte présente les mêmes inconvénients qu'en dimension 2 : un même objet topologique sera représenté par des cartes totalement différentes suivant sa géométrie. Ce niveau n'est pas stable par rotation, translation et homothétie, comme nous pouvons le vérifier sur la figure 5.9. Cette figure montre deux objets 3d topologiquement équivalents, et les cartes des frontières les représentants. Ces deux cartes sont totalement différentes. De plus, la représentation de ces objets n'est pas minimale, et les cartes de ce niveau pourront avoir beaucoup de brins si la géométrie des objets contenus dans l'image est complexe.

5.2.5 Le niveau 4

Avec la carte des frontières, nous avons simplifié au maximum les faces coplanaires et les arêtes alignées. Afin de continuer notre processus, nous allons maintenant refaire les mêmes opérations de fusion que précédemment, pour les faces non coplanaires et les arêtes non alignées, qui sont les seuls éléments restants à simplifier.

Définition 27 (carte de niveau 4) *La* carte de niveau 4 *est la carte obtenue à partir de la carte de niveau* 3*, en fusionnant chaque couple de faces adjacentes et incidentes à une arête de degré un ou deux.*

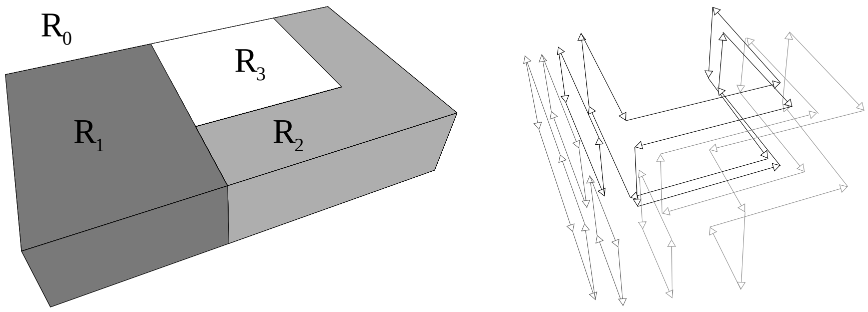


Fig. 5.10 – La carte de niveau 4.

Nous ne précisons pas dans cette définition que les faces adjacentes et incidentes à une arête de degré un ou deux doivent être non coplanaires, car les faces coplanaires vérifiant ces contraintes ont déjà été fusionnées lors du calcul de la carte de niveau 2.

La carte de niveau 4 de notre image d'exemple est présentée figure 5.10. Nous pouvons observer sur cette figure que le nombre de brins de cette carte est beaucoup moins important que celui du niveau précédent. Cette carte est plus difficile à visualiser étant donné que les faces ne sont plus forcément planaires, contrairement aux niveaux précédents.

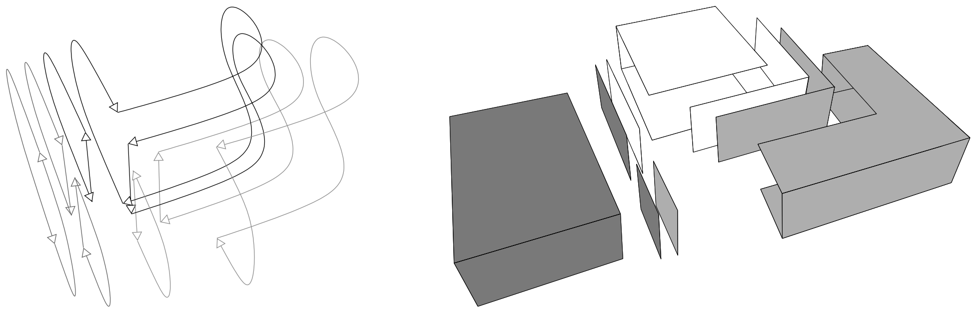
Nous rencontrons pour ce niveau exactement les mêmes problèmes que pour la définition de la carte de niveau 2. Nous fusionnons les faces incidentes à des arêtes de degré un ou deux, afin que la définition ne dépende pas de l'ordre dans lequel ces fusions sont effectuées. Le même problème de déconnexion de face peut se produire, mais il sera présenté et résolu dans le chapitre 5.3, en même temps que le problème de déconnexion du niveau 2. En effet ces deux problèmes sont topologiquement équivalents. La seule différence se situe au niveau de la géométrie des faces fusionnées, ce qui ne rentre pas du tout en compte pour ce problème qui est uniquement topologique.

5.2.6 Le niveau 5 : la carte topologique

Il ne reste plus qu'à fusionner les arêtes incidentes à un sommet de degré deux afin d'obtenir la carte de niveau 5 qui est le dernier niveau de simplification. En effet, nous avons effectué toutes les fusions possibles conservant les informations topologiques. Il est donc impossible de simplifier quoi que ce soit dans ce dernier niveau de carte, sans entraîner la perte d'informations. C'est pour cette raison que nous appelons ce niveau 5 la *carte topologique*.

Définition 28 (carte de niveau 5) *La* carte de niveau 5 *est la carte obtenue à partir de la carte de niveau* 4*, en fusionnant chaque couple d'arêtes adjacentes incidentes à un sommet de degré deux.*

Les arêtes fusionnées lors de cette étape sont forcément non alignées, étant donné que celles qui sont alignées ont déjà été fusionnées lors de la construction de la carte



a. La carte de niveau 5. b. Les frontières intervoxel qu'elle représente.

FIG. 5.11 – La carte de niveau 5, et les faces frontières représentées.

des frontières. Cette fusion, comme pour la carte des frontières, ne peut pas entraîner de déconnexion. De plus, nous ne fusionnons pas les arêtes incidentes à un sommet de degré un, pour les raisons déjà présentées lors de la définition de la carte de niveau 3.

La carte de niveau 5 de notre image d'exemple est présentée figure 5.11. Nous avons représenté la carte de niveau 5 figure 5.11.a, ainsi que les faces frontières de cette image figure 5.11.b. En effet, la carte topologique représente de manière minimale ces faces frontières ainsi que toutes les relations d'incidence et d'adjacence, comme nous pouvons le vérifier sur cette figure. Ces faces frontières sont représentées par leurs bords.

La carte de niveau 5 de notre exemple est composée uniquement de 18 brins, si l'on ne considère pas les 6 brins représentant la région infinie. Ce nombre est beaucoup moins important que, par exemple, les 88 brins de la carte des frontières représentant la même image. Rappelons que les cinq niveaux de simplification représentent exactement les mêmes informations topologiques, propriété démontrée chapitre 6.

Mais cette carte de niveau 5 n'est pas vraiment la carte topologique. En effet, nous n'avons pas résolu les problèmes de déconnexion, et cette carte ne représente donc pas forcément toutes les informations topologiques de l'image. La carte topologique est donc la carte de niveau 5 légèrement modifiée afin de résoudre ces problèmes de déconnexion et conserver effectivement toutes ces informations.

5.3 Problèmes de déconnexion et solutions : arbres d'inclusions et éléments fictifs

Nous avons vu lors des définitions des différents niveaux de simplification que deux problèmes de déconnexion peuvent se produire. Afin que la carte topologique conserve correctement toutes les informations topologiques, nous devons résoudre ces problèmes. Nous présentons les solutions à ces deux problèmes, en montrant dans un premier temps que le problème de déconnexion de volume peut se résoudre sans difficulté par l'ajout d'un arbre d'inclusion de régions, de manière similaire à la solution présentée en dimension 2, mais que le problème de déconnexion de face est plus difficile et nécessite la conservation

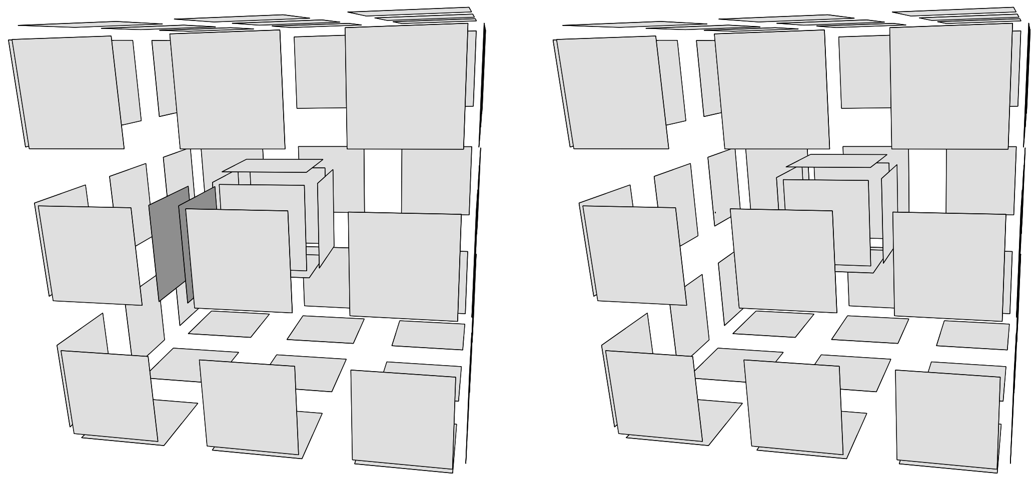


FIG. 5.12 – Un exemple de déconnexion lors de la fusion de volumes.

d'arêtes fictives. La conservation de ces éléments particuliers doit être prise en compte pour la définition de la carte topologique. Nous étudions donc les propriétés de ces arêtes particulières afin de comprendre comment elles interagissent avec la carte topologique, et comment les gérer afin de garantir la minimalité de cette carte.

5.3.1 La déconnexion de volume

Ce type de déconnexion peut survenir lors de la fusion de volumes, lorsqu'un volume est totalement inclus dans un autre. Nous pouvons voir figure 5.12 un exemple pour lequel une région est totalement incluse dans une autre. La figure 5.12.a montre une carte de niveau 1 en cours de construction, avant d'avoir effectué la dernière fusion de volumes le long de la face grise. Notons que cette carte est connexe. Après cette fusion de volume, nous obtenons la carte présentée figure 5.12.b, dans laquelle la surface extérieure du gros cube est déconnectée de la surface intérieure représentant un trou dans ce cube. L'information permettant de relier ces deux surfaces et de les positionner l'une par rapport à l'autre est perdue. Elle doit être conservée sous peine de ne pas caractériser entièrement les différents objets.

Ce type de déconnexion est similaire à celui déjà rencontré en dimension 2, où nous avions alors des déconnexions lors de fusion de faces. Étant donné que nous travaillons en dimension supérieure, nous rencontrons maintenant ces déconnexions pour la fusion de volumes. La solution est la même que pour la dimension 2, et consiste à ajouter un *arbre d'inclusion* des régions. Cet arbre possède un nœud par région de l'image. Sa racine est toujours la région infinie, et un nœud N_1 est fils d'un nœud N_2 si et seulement si la région R_1 correspondant à N_1 est directement incluse dans la région R_2 correspondant à N_2.

À l'aide de cet arbre, nous conservons la relation d'inclusion qui permet de positionner les différentes composantes connexes de la carte les unes par rapport aux autres. Remarquons qu'une autre solution consiste à ne pas effectuer la dernière fusion de volumes, et donc

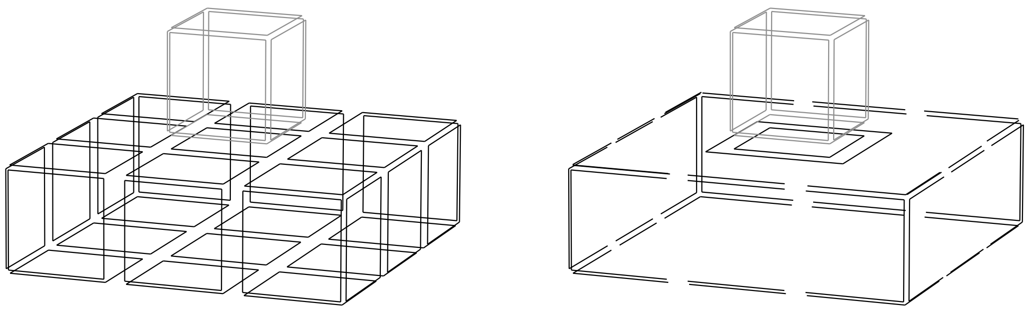


Fig. 5.13 – Un exemple de déconnexion lors de la fusion de faces coplanaires.

à conserver la carte présentée figure 5.12.a comme représentation finale. Cette deuxième solution est équivalente à la première au niveau topologique, mais pas au niveau de sa mise en œuvre. En effet, cette solution entraîne la conservation de faces ayant des propriétés particulières, étant donné qu'elles sont incidentes à une seule et même région. Ces faces auraient normalement dû être fusionnées lors de la construction de la carte de niveau 1, et le fait de les conserver va nous obliger à considérer deux cas différents pour chaque opération, suivant que le brin traité appartient à une de ces faces particulières ou non. De plus, ces faces servent uniquement à représenter la relation d'inclusion, et ne portent aucune autre information topologique.

Ces raisons nous font préférer la première solution, qui permet de s'affranchir de ces faces particulières et de simplifier les traitements. De plus, c'est cette solution qui a été retenue en dimension 2, et que nous étendons donc ici. Enfin, l'information principale utilisée par des algorithmes de traitement porte sur les régions de l'image. L'arbre d'inclusion pourra donc être utilisé afin de contenir d'autres informations sur les régions, comme par exemple leurs couleurs, ou leurs nombres de voxels.

5.3.2 La déconnexion de face et les arêtes fictives

La déconnexion de face peut se produire lors des fusions de faces, donc pour la construction des niveaux 2 et 4. En effet, la différence entre ces niveaux se situe uniquement sur l'aspect géométrique. Pour le niveau 2 seules les faces coplanaires sont fusionnées, alors que pour le niveau 4 ce sont les faces non-coplanaires. Mais ce problème de déconnexion est uniquement topologique et ne dépend pas de la géométrie.

Nous pouvons voir figure 5.13 un exemple pour lequel il se produit une déconnexion de face lors de la construction de la carte de niveau 2. Cette figure présente tout d'abord la carte de niveau 1 d'un objet composé d'un pavé sur lequel est posé au centre un cube, puis la carte de niveau 2 de ce même objet. Pour cette deuxième carte, nous avons déconnecté le bord extérieur de la face supérieure du pavé de son bord intérieur. Nous sommes alors incapables de positionner topologiquement ces deux composantes connexes.

Il est possible d'appliquer la même solution que pour le problème de déconnexion de volume, en ajoutant un arbre d'inclusion des faces qui pour chaque face donnerait l'ensemble

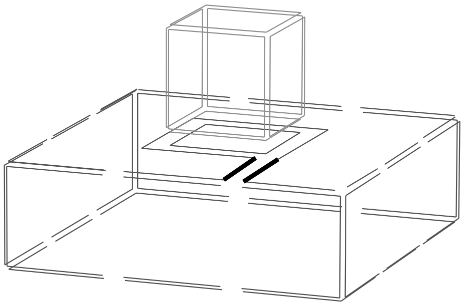


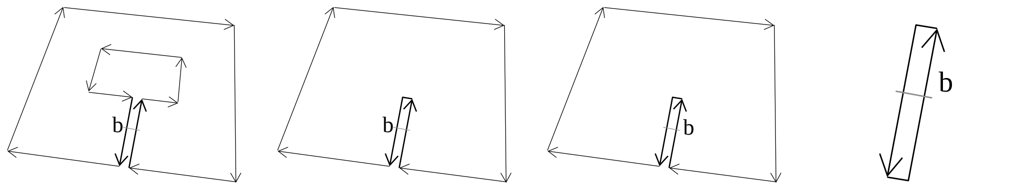
Fig. 5.14 – L'objet présenté figure 5.13 avec une arête fictive.

des faces incluses. Mais cette première solution n'est pas satisfaisante. En effet, contrairement au problème de déconnexion de volume, l'information perdue n'est pas uniquement une information d'inclusion. En effet, par exemple lorsqu'un volume est représenté par une surface fermée, cette surface n'a pas de bord et ne serait alors pas du tout représentée dans la carte de niveau 5. Nous aurions alors perdu le genre de l'objet, malgré la présence des arbres d'inclusion de faces.

Une solution adéquate consiste à conserver une arête permettant de relier le bord extérieur d'une face et son bord intérieur. Nous pouvons voir figure 5.14 la carte de niveau 2 de l'objet déjà présenté figure 5.13, mais pour lequel nous avons conservé une arête reliant le bord extérieur de la face supérieure du pavé avec son bord intérieur. Cette arête permet de conserver la notion de face introduite lors de la présentation des cartes combinatoires, c'est-à-dire au sens orbite $< \beta_1 >$. En effet, si nous partons de n'importe quel brin de la face supérieure du pavé et effectuons un parcours suivant cette orbite, nous parcourons bien l'ensemble des brins de cette face, ce qui n'était pas le cas lorsque ces deux bords étaient déconnectés.

Ces arêtes particulières peuvent être caractérisées sans difficulté, étant donné qu'elles sont de degré un et incidentes deux fois à la même face. Les arêtes de degré un sont normalement supprimées lors de la construction du niveau 2 ou du niveau 4 (suivant la géométrie de la face), mais ces arêtes particulières ne doivent pas l'être afin de conserver chaque face connexe. De plus, ces arêtes n'ont pas de plongement, étant donné qu'elles ne représentent pas le bord d'une surface frontière de l'image. Elles ont une existence uniquement topologique. C'est pour ces raisons que nous les appelons des *arêtes fictives*. Par opposition, nous appelons les autres arêtes des *arêtes réelles*. Étant donné que ces arêtes fictives n'ont qu'une existence topologique, elles ne sont pas figées et peuvent être déplacées à l'intérieur de la face, à condition toutefois de la conserver connexe. Nous verrons section 5.3.3 que cette propriété est très importante, et que son utilisation garantit que la carte topologique est bien minimale.

Afin de conserver ces arêtes fictives, nous modifions les définitions 25 et 27 des niveaux 2 et 4. En effet, c'est uniquement pour la construction des cartes de ces deux niveaux que le problème de déconnexion de face peut se produire. C'est donc uniquement pour ces niveaux qu'il s'agit éventuellement de conserver des arêtes fictives.



a. $\beta_0(b) \neq \beta_2(b)$ $\beta_1(b) \neq \beta_2(b)$. b. $\beta_0(b) = \beta_2(b)$ $\beta_1(b) \neq \beta_2(b)$. c. $\beta_1(b) = \beta_2(b)$ $\beta_0(b) \neq \beta_2(b)$. d. $\beta_0(b) = \beta_2(b)$ $\beta_1(b) = \beta_2(b)$.

FIG. 5.15 – Les quatre configurations possibles de faces autour d'une arête de degré un.

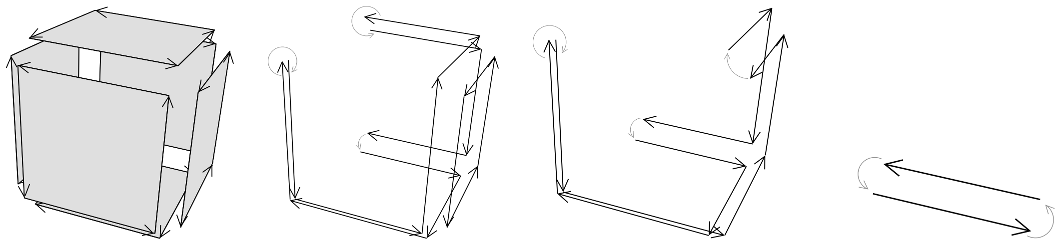
Définition 29 (carte de niveau 2) *La* carte de niveau 2 *est la carte obtenue à partir de la carte de niveau* 1*, en fusionnant chaque couple de faces adjacentes, coplanaires et incidentes à une arête de degré un ou deux, n'entraînant pas de déconnexion de face.*

Définition 30 (carte de niveau 4) *La* carte de niveau 4 *est la carte obtenue à partir de la carte de niveau* 3*, en fusionnant chaque couple de faces adjacentes et incidentes à une arête de degré un ou deux, n'entraînant pas de déconnexion de face, ni la suppression totale de la face.*

Nous pouvons voir sur les définitions 29 et 30 que nous avons simplement ajouté une condition supplémentaire afin de ne pas effectuer une fusion de faces lorsqu'elle entraîne une déconnexion. Remarquons qu'avec ces définitions modifiées, nous obtenons bien pour notre exemple de la figure 5.13, la carte de niveau 2 de la figure 5.14. Nous pouvons noter sur cette carte la présence d'une arête fictive qui permet de conserver la connexité de la face supérieure du pavé. Avec ces nouvelles définitions, chaque face de ces cartes est homéomorphe à un disque topologique.

Pour tester si une fusion de faces entraîne une déconnexion ou non, il suffit d'effectuer un parcours de l'orbite $< \beta_1 >$ d'origine un brin b de l'arête le long de laquelle doit être effectuée la fusion. Si durant ce parcours, nous atteignons le brin $\beta_2(b)$, alors l'arête est de degré un car les deux brins b et $\beta_2(b)$ appartiennent à la même face. Dans ce cas la fusion le long de cette arête va entraîner une déconnexion si les deux brins b et $\beta_2(b)$ ne sont pas cousus entre eux par β_0 et par β_1. Nous pouvons voir figure 5.15 les quatre configurations possibles de faces autour d'une arête de degré un, où nous pouvons vérifier que le seul cas pouvant entraîner déconnexion est celui présenté figure 5.15.a.

Le dernier cas de cette figure est un autre cas particulier, qui bien que n'entraînant pas de déconnexion nécessite une condition supplémentaire. Nous ne devons pas effectuer la fusion de deux faces si cela entraîne la suppression totale d'une face. Cette condition est ajoutée dans la définition 30 de la carte de niveau 4. Ce cas ne peut pas se produire pour le niveau 2, étant donné que chaque surface représente le bord d'un volume, et que ce bord ne peut jamais être constitué uniquement d'une face plane fermée. Il se produit lorsque les deux brins b et $\beta_2(b)$ formant l'arête le long de laquelle doit être effectuée la fusion de faces, sont cousus entre eux par β_1. Ce cas survient lorsque l'objet représenté est une sphère totalement incluse dans une autre région.



a. Le niveau 3. b. Deux étapes intermédiaires. c. Les niveaux 4 et 5.

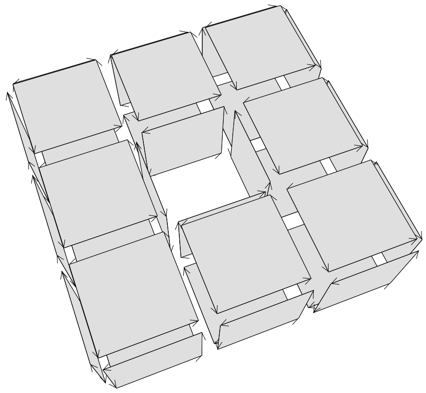
FIG. 5.16 – Le cas de la sphère pour les niveaux 3, 4 et 5.

Nous présentons figure 5.16.*a* la carte de niveau 3 d'une sphère topologique (ici un cube) entièrement incluse dans un autre volume. Nous pouvons remarquer que chaque arête de cette carte est de degré deux. Lors de la construction de la carte de niveau 4 à partir de cette carte, nous pouvons effectuer les fusions de faces le long de toutes ces arêtes sans entraîner de déconnexion de face.

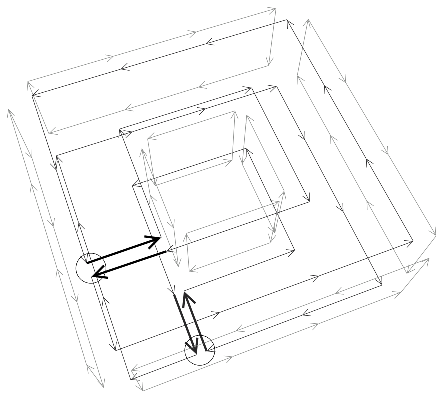
Nous pouvons voir figure 5.16.*b* deux étapes intermédiaires de cette construction, où nous avons seulement effectué certaines fusions de faces. Sur cette figure, nous avons représenté explicitement β_1 par des arcs de cercle gris lorsque les brins concernés ne sont pas dessinés de manière consécutive. Nous voyons sur ces étapes intermédiaires que les fusions de face entraînent progressivement la suppression de faces. Nous pouvons également remarquer que les différentes fusions ne peuvent pas être effectuées dans n'importe quel ordre. En effet, une fusion peut entraîner une déconnexion si elle est effectuée avant une autre, alors que ce ne serait pas le cas si ces fusions étaient effectuées dans l'ordre inverse. Mais il existe un ordre de fusion qui n'entraîne aucune déconnexion, et nous pouvons donc effectuer la fusion de faces le long de chaque arête de la carte. Cela peut se vérifier sur la figure 5.16, où intuitivement nous commençons par fusionner les faces le long des arêtes aux extrémités de la carte.

Dans ce cas, nous supprimons tous les brins représentant cette face ainsi que, du même coup, le volume complet. C'est pour cette raison que nous avons ajouté dans la définition 30, la contrainte selon laquelle nous n'effectuons pas une fusion de faces le long d'une arête si cette fusion entraîne la suppression totale de la face. Avec cette contrainte supplémentaire, nous obtenons alors la carte de niveau 4 présentée figure 5.16.*c*. Cette carte est également la carte de niveau 5, étant donné que les deux sommets de cette carte sont de degré un, et ne vont donc pas être supprimés par une fusion d'arêtes. Nous obtenons bien une représentation minimale de la sphère. En effet, cette carte possède une face, une arête et deux sommets, et la formule d'Euler nous permet de vérifier que le genre de cet objet est égal à zéro.

Cette représentation minimale n'est pas unique, car avec le même nombre de brins, nous pouvons représenter une sphère par deux faces composées d'un brin chacune et β_2 cousus entre eux. Cette autre représentation possède donc deux faces, une arête et un sommet, et donc est également de genre zéro. Ces deux représentations minimales sont duales l'une de l'autre. Mais nous préférons la première représentation, tout d'abord car



a. La carte de niveau 1.



b. La carte de niveau 2.

FIG. 5.17 – Les cartes de niveau 1 et 2 d'un tore totalement inclus dans une autre région.

c'est celle que nous obtenons à partir de notre définition, mais également car la deuxième représentation possède deux faces différentes. Dans ce cas, nous ne représentons plus les faces frontières du volume correspondant. En effet, lorsqu'un objet est totalement inclus dans un autre, il y a une seule face frontière entre ces deux objets. Cette information n'apparaît pas immédiatement avec la deuxième représentation minimale, alors que la première représentation possède bien une seule face topologique correspondant à cette face frontière.

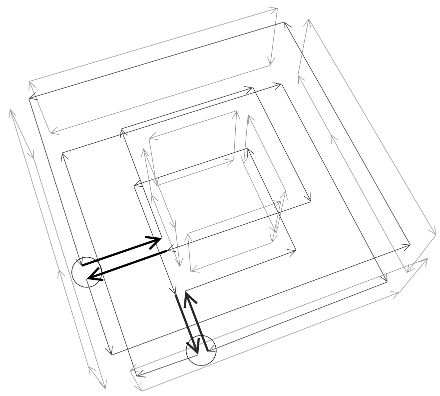
Avec ces définitions modifiées des niveaux 2 et 4, nous représentons maintenant correctement toutes les configurations possibles, en conservant chaque face homéomorphe à un disque topologique au moyen d'arêtes fictives lorsque c'est nécessaire. Mais la carte topologique doit être la représentation minimale de l'image. Nous étudions maintenant comment gérer ces arêtes fictives afin de garantir cette minimalité.

5.3.3 Gestion des arêtes fictives et représentation minimale

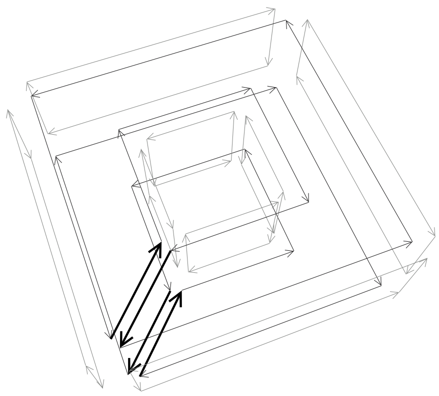
Le fait de conserver des arêtes fictives pour les cartes de niveau 2 et 4 permet de résoudre le problème de déconnexion de face. Mais ces arêtes introduisent de nouveaux problèmes lors de la construction des cartes de niveau 3 et 5, c'est-à-dire lors de fusions d'arêtes. Leur résolution garantit l'obtention de la représentation minimale.

Un premier problème se pose lors de la construction des cartes de niveau 3 et 5, lorsque nous sommes en présence d'arêtes fictives. En effet, ces arêtes fictives peuvent, suivant leur position dans la carte, empêcher la fusion de deux arêtes qui l'auraient été autrement. Ce n'est pas du tout satisfaisant car le rôle des arêtes fictives est uniquement de conserver les faces connexes. Elles ne doivent pas intervenir dans la construction des autres niveaux.

La figure 5.17 permet de visualiser ce problème. La figure 5.17.a montre la carte de niveau 1 d'un tore totalement inclus dans une autre région. Cette carte représente donc une seule face frontière qui est fermée. La figure 5.17.b montre la carte de niveau 2 de ce même tore. Nous pouvons noter la présence de deux arêtes fictives (représentées en noir



a. Problème pour le niveau 3.



b. La carte de niveau 3 correcte.

FIG. 5.18 – Le problème rencontré lors du calcul de la carte de niveau 3 du tore si l'on ne considère pas différemment les arêtes fictives, et la carte de niveau 3 correcte.

épais sur le dessin) qui permettent de conserver les faces supérieure et inférieure du tore connexes. Nous avons sur cette figure représenté en foncé les brins appartenant à ces deux faces, étant donné que ce sont elles qui nous intéressent, et en plus clair les autres brins.

Nous pouvons voir que les arêtes fictives sont positionnées, par rapport aux autres brins de la face, de manière totalement arbitraire étant donné que leurs positions dépend de l'ordre des fusions de faces et que cet ordre est lui même totalement arbitraire. Cela ne pose aucun problème pour cette carte de niveau 2, mais en pose lors du calcul du niveau suivant. En effet, lors de la construction de la carte de niveau 3, nous fusionnons chaque couple d'arêtes adjacentes, alignées, et incidentes à un sommet de degré deux. Or les deux sommets de la carte de niveau 2 (figure 5.17.b) entourés de cercles noirs ne sont pas de degré deux à cause de la présence des arêtes fictives. Si nous n'effectuons pas de distinction entre les arêtes fictives et les autres arêtes lors de la construction du niveau 3, nous obtenons la carte présentée figure 5.18.a.

Sur cette carte, nous voyons que ces deux sommets n'ont pas été supprimés par une fusion d'arêtes, alors qu'ils l'auraient été sans la présence des arêtes fictives, ou si elles avaient été positionnées de manière différente. La figure 5.18.b présente la carte de niveau 3 que nous souhaitions obtenir. En effet, les arêtes fictives servent uniquement à conserver chaque face connexe, mais ne doivent pas empêcher la fusion de certaines arêtes. Nous devons régler ce problème afin de ne pas avoir la carte de niveau 3 qui dépend de l'ordre dans lequel les fusions de faces coplanaires ont été effectuées. En effet, le cas échéant nous n'aurions pas une représentation unique de cette carte, et donc de la carte topologique. De plus, la représentation obtenue ne serait alors pas minimale, étant donné que certaines arêtes ne seraient pas fusionnées. Pour cela, nous considérons de manière différente les arêtes fictives des autres arêtes lors des calculs des niveaux 3 et 5, c'est-à-dire pour les fusions d'arêtes. Nous modifions donc les définitions 26 et 28 en conséquence.

Définition 31 (carte de niveau 3) *La* carte de niveau 3 *est la carte obtenue à partir de la carte de niveau* 2*, en fusionnant chaque couple d'arêtes* (a_1, a_2) *adjacentes, alignées et incidentes à un sommet* s *tel qu'il n'existe pas d'arête réelle différente de* a_1 *et de* a_2 *incidente à* s*.*

Définition 32 (carte de niveau 5) *La* carte de niveau 5 *est la carte obtenue à partir de la carte de niveau* 4*, en fusionnant chaque couple d'arêtes* (a_1, a_2) *adjacentes, n'étant pas des boucles et incidentes à un sommet* s *tel qu'il n'existe pas d'arête réelle différente de* a_1 *et de* a_2 *incidente à* s*.*

Par rapport aux anciennes définitions de ces deux niveaux, nous avons remplacé la condition que le sommet soit de degré deux par la nouvelle condition qu'il n'existe pas d'arête réelle différente de a_1 et de a_2 incidente à ce sommet. Remarquons que ces deux conditions sont équivalentes lorsqu'il n'y a pas d'arête fictive. Mais cette nouvelle condition permet de ne pas dépendre de la position des arêtes fictives qui ne vont plus entrer en compte lors des fusions d'arêtes.

Pour la définition du niveau 5, une contrainte supplémentaire impose que a_1 et a_2 ne soient pas des boucles. En effet, il est impossible de fusionner une boucle avec une autre arête. La fusion d'arête peut être considérée comme un étirement de la première arête afin qu'elle « couvre » la seconde, ce qui n'est pas possible à réaliser lorsqu'une des deux arêtes est une boucle. Cette condition est inutile pour le niveau 3 étant donné que nous ne pouvons pas avoir de boucle, puisque nous fusionnons uniquement des arêtes alignées.

Avec ces définitions modifiées, nous pouvons donc fusionner deux arêtes réelles le long du sommet s, malgré la présence d'arêtes fictives incidentes à s. Mais dans ce cas, afin de réaliser cette fusion, nous devons effectuer préalablement un traitement particulier. En effet, une fusion d'arête peut être réalisée uniquement le long d'un sommet de degré deux. Pour cela, avant d'effectuer la fusion de deux arêtes a_1 et a_2, nous déplaçons simplement toutes les arêtes incidentes à s, à l'exception de a_1 et a_2, sur le deuxième sommet de a_1.

En effet, ces arêtes sont forcément des arêtes fictives, et nous avons vu qu'elles ont uniquement un sens topologique, et peuvent donc être déplacées, à condition de conserver la carte connexe. De plus, nous savons que a_1 et a_2 ne sont pas des boucles et donc il existe bien un autre sommet différent de s incident à a_1. Remarquons enfin que nous pourrions choisir de décaler les arêtes fictives sur le deuxième sommet de a_2 sans que cela ne change quoi que ce soit à la suite des traitements.

Nous présentons l'algorithme 17 effectuant le décalage de toutes les éventuelles arêtes fictives incidentes à un sommet donné. Cet algorithme est composé de deux boucles très similaires. La première traite les éventuelles arêtes fictives se trouvant entre b_1 et b_2, en tournant dans le sens trigonométrique. Pour chacune de ces arêtes, elle appelle simplement l'algorithme 18 qui se charge de décaler cette arête fictive entre les brins b_1 et $\beta_1(b_1)$. La deuxième boucle effectue un traitement similaire pour chaque arête fictive se trouvant entre b_1 et b_2, en tournant dans le sens trigonométrique inverse. Ces arêtes fictives sont quant à elles décalées entre les brins $\beta_2(b_1)$ et $\beta_{20}(b_1)$.

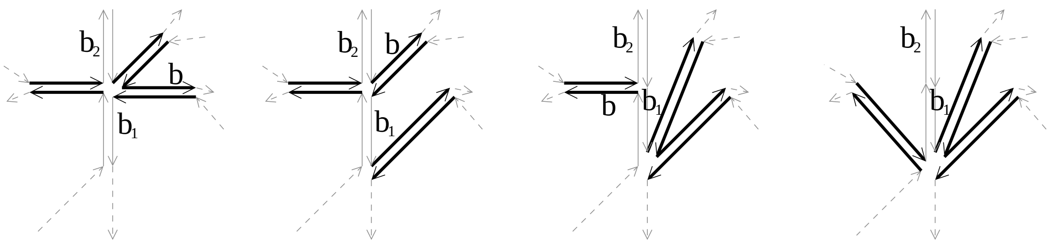
Nous devons effectuer ces deux traitements de manière différente car suivant la position originale des arêtes fictives, elles ne doivent pas être décalées au même endroit. En effet,

Algorithme 17 Décalage de toutes les arêtes fictives incidentes à un sommet donné

Entrée : Deux brins b_1 et b_2 incident au même sommet

Résultat : Les arêtes fictives (différentes des arêtes incidentes à b_1 et b_2), incidente au même sommet sont décalées sur le sommet incident à $\beta_1(b_1)$.

$b \leftarrow \beta_{02}(b_1)$;
tant que $b \neq b_2$ **faire**
  $b_{next} \leftarrow \beta_{02}(b)$;
  Décaler_arête($b, \beta_1(b_1)$) ;
  $b \leftarrow b_{next}$;
$b \leftarrow \beta_{21}(b_1)$;
tant que $b \neq b_2$ **faire**
  $b_{next} \leftarrow \beta_{21}(b)$;
  Décaler_arête($b, \beta_2(b_1)$);
  $b \leftarrow b_{next}$;



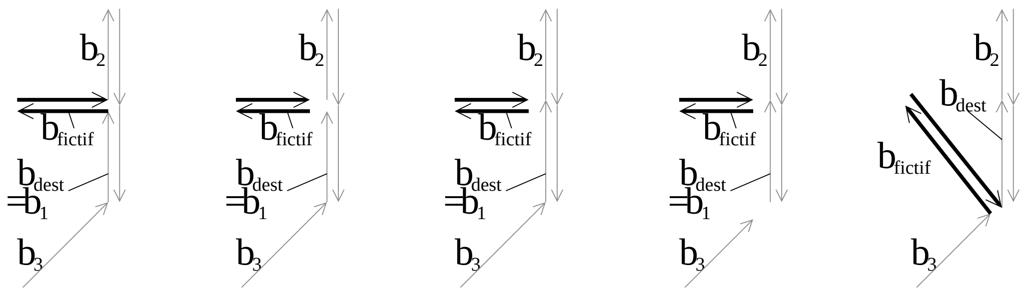
a. Initialisation. b. Après un décalage. c. Après deux décalages : fin de la première boucle. d. Fin de l'algorithme.

FIG. 5.19 – Déroulement de l'algorithme de décalage des arêtes fictives.

lorsque f_1 la face incidente à b_1 n'est pas incidente à b_2, les arêtes fictives conservant la face f_1 connexe doivent être décalées à l'intérieur de la face f_1. Par contre, les arêtes fictives conservant f_2 la face incidente à b_2 connexe doivent être décalées à l'intérieur de f_2.

Nous présentons figure 5.19 le déroulement de cet algorithme. La figure 5.19.a montre une configuration où deux arêtes réelles (incidentes à b_1 et b_2) sont incidentes à un sommet tel qu'il n'existe pas d'autre arête réelle incidente à ce sommet. Les flèches en pointillés gris symbolisent la présence de brins n'étant pas concernés par cet algorithme de décalage, mais rappellent leur existence.

La première boucle commence à traiter la première arête fictive après b_1 en tournant dans le sens trigonométrique. La figure 5.19.b montre l'état de la carte après le premier décalage de cette arête, et la figure 5.19.c son état après le deuxième décalage. Après ce décalage, nous avons $b = b_2$, ce qui entraîne la sortie de la première boucle. La deuxième boucle effectue pour cet exemple un seul décalage et nous obtenons au final la carte présentée figure 5.19.d. Nous avons bien décalé toutes les arêtes fictives incidentes au sommet incident à b_1 et à b_2 : ce sommet est désormais de degré deux.



a. Initialisation. | b. Décousure de l'arête incidente à b_{fictif}. | c. β_1-couture de b_1 et b_2. | d. β_1-décousure de b_3. | e. Couture de l'arête incidente à b_{fictif}.

FIG. 5.20 – Déroulement de l'algorithme décalant une seule arête fictive.

L'algorithme « Décaler_arête », présenté algorithme 18, effectue simplement le décalage de l'arête fictive incidente à b_{fictif} entre les brins $\beta_0(b_{dest})$ et b_{dest}.

Algorithme 18 Décalage d'une arête fictive donnée

Entrée : Deux brins b_{fictif} et b_{dest}

Résultat : L'arête fictive incidente à b_{fictif} est décalé entre les brins $\beta_0(b_{dest})$ et b_{dest}.

$b_1 \leftarrow \beta_0(b_{fictif})$; $b_2 \leftarrow \beta_{21}(b_{fictif})$; $b_3 \leftarrow \beta_0(b_{dest})$;

1 β_0-découdre(b_{fictif}) ; β_1-découdre($\beta_2(b_{fictif})$); β_1-découdre($\beta_3(b_{fictif})$) ; β_0-découdre($\beta_{23}(b_{fictif})$);

2 β_1-coudre(b_1, b_2) ; β_0-coudre($\beta_3(b_1), \beta_3(b_2)$);

3 β_1-découdre(b_3) ; β_0-découdre($\beta_3(b_3)$);

4 β_1-coudre(b_3, b_{fictif}) ; β_1-coudre($\beta_2(b_{fictif}), b_{dest}$); β_0-coudre($\beta_3(b_3), \beta_3(b_{fictif})$) ; β_0-coudre($\beta_{23}(b_{fictif}), \beta_3(b_{dest})$);

Ce décalage s'effectue en 4 étapes que nous illustrons sur l'exemple présenté figure 5.20. Après la phase d'initialisation (figure 5.20.a), la première étape de l'algorithme effectue la décousure de l'arête fictive incidente à b_{fictif} (figure 5.20.b). Une arête fictive est forcément composée de 4 brins. En effet, il existe deux brins cousus entre eux par β_2, par demi-face, et une face topologique est composée de deux demi-faces identiques β_3-cousues. Nous ne représentons pas la deuxième demi-face sur nos figures pour des raisons de lisibilité.

La deuxième étape de l'algorithme β_1-coud b_1 et b_2, les deux brins qui ont été décousus lors de la première phase (figure 5.20.c). Afin de vérifier les contraintes des cartes combinatoires, nous effectuons la couture similaire pour les brins correspondant dans la deuxième demi-face. La troisième étape β_1-découd b_3 pour insérer l'arête fictive (figure 5.20.d). De même que précédemment, nous effectuons la décousure similaire pour $\beta_3(b_3)$. Enfin, la dernière étape (figure 5.20.e) insère l'arête fictive entre les brins b_3 et b_{dest} (ainsi qu'entre $\beta_3(b_{dest})$ et b_3 pour l'autre demi-face).

La complexité de cet algorithme est en temps constant, étant donné que nous effectuons uniquement des opérations unitaires de couture et de décousure. Remarquons que cet algorithme fonctionne quelle que soit la configuration de la carte de départ, la seule contrainte étant que l'arête incidente à b_{fictif} doit être une arête fictive. La complexité de l'algorithme 17 effectuant le décalage de l'ensemble des arêtes fictives incidentes à un sommet est en *O(nombre d'arêtes fictives décalées)*. En effet, les deux boucles « tant que » parcourent l'ensemble des arêtes fictives une à une, et effectuent uniquement des opérations en temps constant. Remarquons enfin que cet algorithme fonctionne même s'il n'y a aucune arête fictive à décaler, et dans ce cas n'entraîne pas de modification de la carte.

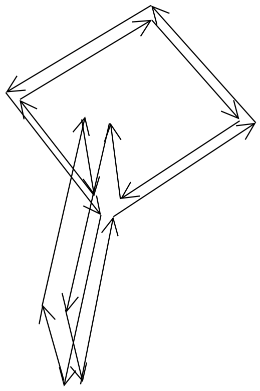
Lors des calculs des cartes de niveaux 3 et 5, nous testons si deux arêtes a_1 et a_2 doivent être fusionnées, à l'aide des contraintes présentées définitions 31 et 32. Si c'est le cas, nous appelons l'algorithme 17, qui va décaler les éventuelles arêtes fictives incidentes à ce sommet. Enfin, nous effectuons la fusion des deux arêtes a_1 et a_2 restantes de manière classique, étant donné que le sommet est désormais de degré 2.

Lorsque nous appliquons ce principe à la carte de niveau 2 pour notre exemple du tore (cf. figure 5.17.b), nous obtenons bien la carte de niveau 3 correcte présentée figure 5.18.b, où les arêtes fictives n'ont pas empêché de fusion d'arêtes. La position des arêtes fictives dépend de l'ordre dans lequel nous avons effectué les fusions d'arêtes et dans lequel nous avons choisi les brins de ces arêtes. Nous obtenons donc une représentation qui est unique à la position des arêtes fictives près.

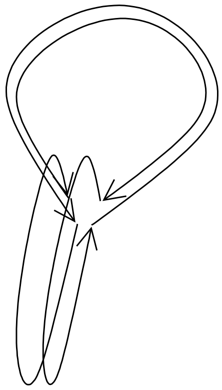
Afin de calculer la carte de niveau 4, nous fusionnons maintenant chaque couple de faces adjacentes incidentes à une arête de degré un ou deux n'entraînant pas de déconnexion de face, ni la suppression totale de la face (d'après la définition 30). Nous observons sur la carte de niveau 3 du tore (cf. figure 5.18.b) que toutes ses arêtes sont de degré deux. Seules les conditions supplémentaires sur la déconnexion et la suppression totale vont nous contraindre à ne pas effectuer certaines fusions.

La carte de niveau 4 de cet exemple est présentée figure 5.21.a. Cette carte est composée d'une seule face. Il est impossible d'effectuer une fusion de faces le long de n'importe quelle arête de cette carte sans entraîner une déconnexion. Cette carte ne peut donc plus être simplifiée par des fusions de faces. De ce fait, chaque arête de cette carte est une arête fictive. En effet, toute ces arêtes sont de degré un et n'ont pas été fusionnées uniquement car cela entraînait une déconnexion. Mais c'est bien ce que nous voulions obtenir, étant donné que la face frontière que représente cette carte est fermée. Elle n'a donc pas de bord, et par conséquence pas d'arête réelle.

Nous pouvons maintenant construire la carte de niveau 5 à partir de cette carte, en appliquant la définition 32. Remarquons que, lors de cette construction, nous fusionnons des arêtes fictives. En effet, les nouvelles définitions des niveaux 3 et 5 imposent qu'il n'existe pas d'arête réelle incidente au même sommet différente des deux arêtes à fusionner, mais il n'y a pas de contrainte sur ces deux arêtes. Elles peuvent donc être indifféremment deux arêtes réelles ou deux arêtes fictives. Mais il n'est pas possible de fusionner une arête réelle avec une arête fictive. En effet, s'il n'existe pas d'autre arête réelle incidente au même sommet, c'est que l'arête réelle est une boucle et dans ce cas nous n'effectuons pas la fusion.



a. La carte de niveau 4.



b. La carte de niveau 5.

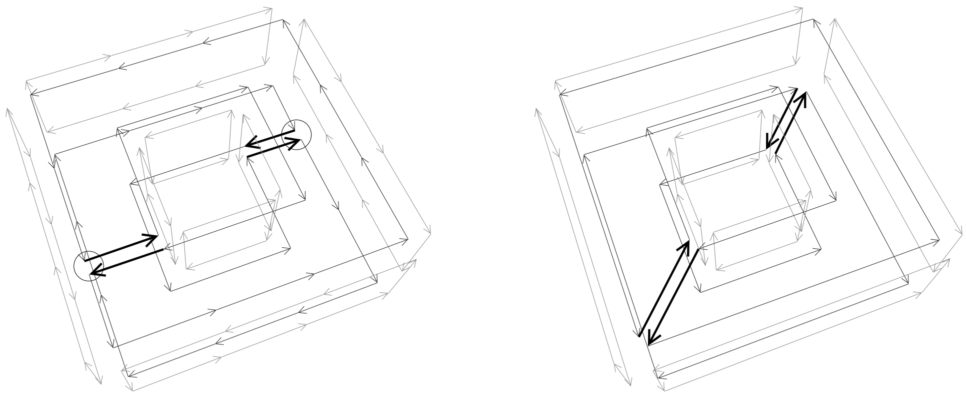
FIG. 5.21 – Suite de l'exemple du tore pour les niveaux 4 et 5.

Pour notre exemple du tore, nous obtenons finalement la carte présentée figure 5.21.*b*. En effet, tous les sommets de la carte de niveau 4 sont de degré deux, à l'exception du sommet central où se croisent les deux boucles. Il ne reste donc plus que ce sommet dans la carte de niveau 5, et nous obtenons bien une représentation minimale classique du tore, composée d'une face, de deux arêtes et d'un sommet.

Nous pouvons vérifier sur cet exemple que les arêtes fictives portent plus d'information que celle d'inclusion, et donc que la solution consistant à ajouter un arbre d'inclusion à chaque face n'est pas suffisante. En effet, ces arêtes fictives sont utiles afin de palier le problème de déconnexion de face, mais permettent également la représentation d'objets ayant une surface fermée comme surface frontière, comme nous venons de le voir. Avec ces nouvelles définitions des niveaux de simplification, nous avons obtenu, dans l'exemple que nous venons de traiter, la représentation minimale de manière directe. En effet, la carte de niveau 5 a été obtenue directement à partir de la carte de niveau 4 en fusionnant les arêtes incidentes à des sommets de degré deux. Cette représentation minimale est unique étant donné que le nombre de faces est fixée : il est égal au nombre de faces frontières de l'image.

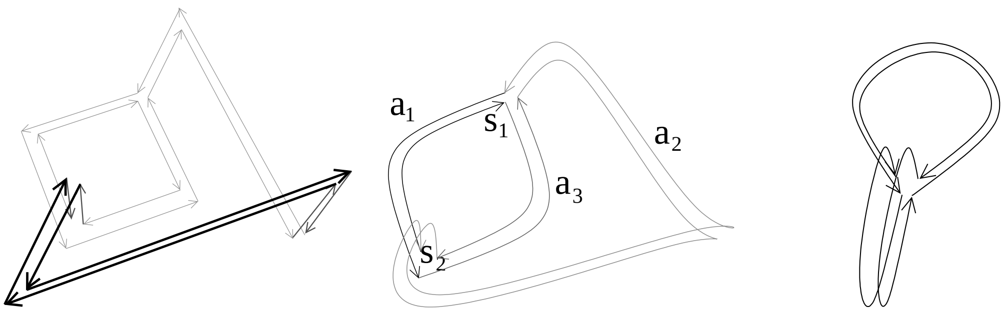
Mais ce n'est pas forcément toujours le cas. Pour illustrer cela, nous reprenons le cas du tore, mais avec d'autres positions des arêtes fictives dans la carte de niveau 2. En effet, ces positions dépendent de l'ordre dans lequel sont réalisées les fusions de faces, étant donné que l'arête fictive s'obtient en n'effectuant pas la dernière fusion. Nous pouvons voir figure 5.22.*a* la carte de niveau 2 du tore de l'exemple précédent, mais avec d'autres positions des arêtes fictives, et figure 5.22.*a* la carte de niveau 3 correspondante. Pour ces deux niveaux, les nouvelles définitions donnent bien les résultats corrects. De plus, ces deux niveaux sont identiques au deux déjà présentés, à la position des arêtes fictives près.

Lorsque nous calculons la carte de niveau 4, nous obtenons alors la carte présentée figure 5.23.*a*. Cette carte est composée d'une seule face, de dix arêtes et de neuf sommets. La caractéristique d'Euler nous permet de vérifier qu'elle représente bien un tore. Pour vérifier que cette carte possède une seule face, il suffit de partir d'un brin quelconque et de



a. La carte de niveau 2. b. La carte de niveau 3.

FIG. 5.22 – Les cartes de niveau 2 et 3 du tore de l'exemple précédent avec d'autres positions des arêtes fictives.

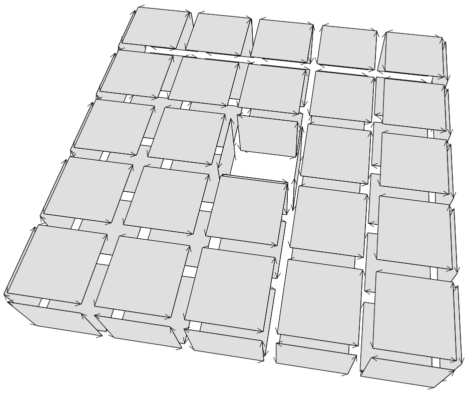


a. La carte de niveau 4. b. Le niveau 5 en cours de construction. c. La carte de niveau 5.

FIG. 5.23 – Les cartes de niveau 4 et 5, et une étape intermédiaire de construction.

parcourir l'orbite $< \beta_1 >$. Nous parcourons alors l'ensemble des brins de cette carte. Cette carte est bien la carte de niveau 4, car nous ne pouvons pas effectuer d'autre fusion de faces le long de n'importe quelle arête sans entraîner une déconnexion. Remarquons que cette carte n'est pas identique à celle que nous avons obtenu précédemment (figure 5.21.a). En effet, ce niveau de carte est dépendant de la géométrie, car nous avons, pour le moment, fusionné uniquement les arêtes alignées.

Nous calculons la carte de niveau 5 à partir de cette carte. La figure 5.23.b montre la carte obtenue en fusionnant uniquement les arêtes incidentes à des sommets de degré deux. Cette carte n'est pas une représentation minimale d'un tore, et n'est pas celle que nous voulons obtenir. Mais avec notre définition modifiée de la carte de niveau 5, nous fusionnons chaque couple d'arêtes n'étant pas des boucles tel qu'il n'existe pas d'autre arête réelle incidente au même sommet. Nous voyons sur la figure 5.23.b que certains couples d'arêtes répondent à ce critère et nous allons les fusionner. Il y a exactement trois couples


a. La carte de niveau 1.

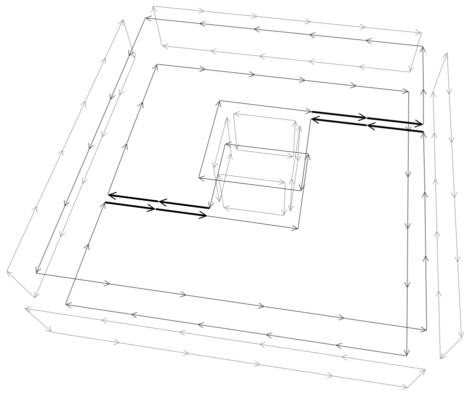

b. La carte de niveau 2.

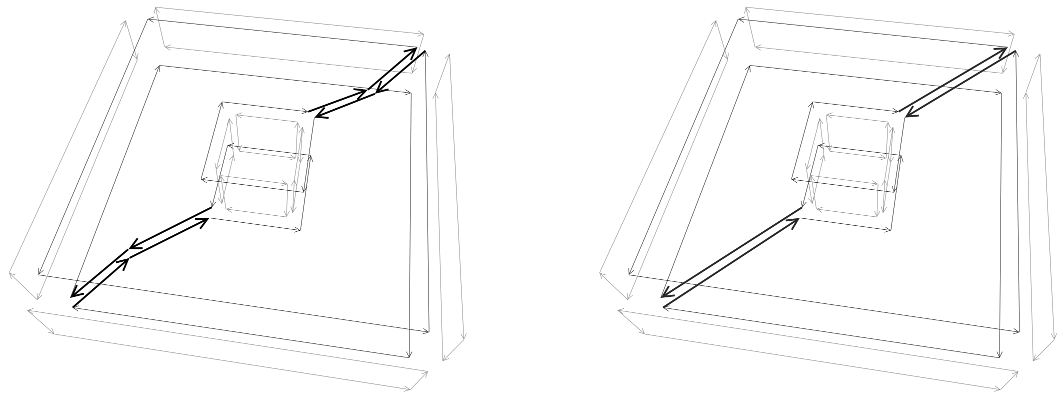
FIG. 5.24 – Les cartes de niveau 1 et 2 d'un autre tore totalement inclus dans une région.

d'arêtes possibles : (a_1, a_2), (a_1, a_3) et (a_2, a_3). Comme nous ne contrôlons pas l'ordre dans lequel sont effectuées les fusions, c'est une de ces trois fusions qui va être réalisée. Mais quel que soit le couple d'arêtes choisi, nous obtenons le même résultat qui est la carte de niveau 5 présenté figure 5.23.*c*. Remarquons que cette carte est la même que celle obtenue avec l'exemple précédent (cf. figure 5.21.*b*). Nous avons bien une représentation unique et minimale.

Afin d'obtenir cette carte, nous utilisons le principe déjà présenté, consistant à déplacer les éventuelles arêtes fictives incidentes au même sommet avant d'effectuer la fusion. Par exemple, si nous fusionnons les deux arêtes a_1 et a_2 le long du sommet s_1, nous commençons par décaler l'arête a_3 incidente également à ce sommet, sur s_2 qui est le deuxième sommet incident à a_1. Ce décalage rend l'arête a_3 deux fois incidente au même sommet : c'est une boucle. Ensuite, nous effectuons la fusion des deux arêtes a_1 et a_2, étant donné que le sommet s_1 est de degré deux. Cette fusion entraîne la création de la deuxième boucle, et nous obtenons alors la carte de niveau 5 finale.

Ces deux exemples mettent en évidence que les définitions modifiées des niveaux de simplification et notre principe de décalage d'arêtes fictives permettent bien d'obtenir la carte topologique, quelle que soit la position des arêtes fictives. Mais il reste encore un cas particulier à traiter afin d'avoir totalement étudié la gestion des arêtes fictives. Ce cas se produit lors de la fusion de deux arêtes fictives lors du calcul de la carte de niveau 3.

Il est illustré figure 5.24, à nouveau pour un tore, mais contrairement aux deux exemples précédents, celui-ci est de taille $5 \times 5 \times 1$ voxels. Nous pouvons remarquer sur la carte de niveau 2 représentant ce tore (figure 5.24.*b*), la présence d'arêtes fictives permettant de conserver les faces inférieure et supérieure de ce tore connexes. Mais contrairement aux exemples précédents, il y a maintenant deux arêtes fictives pour chacune de ces faces, étant donné que les bords extérieur et intérieur de ces faces sont éloignés de deux lignels. Ce n'est pas gênant pour cette carte de niveau 2, étant donné que chaque arête de cette carte représente un lignel. Mais cela pose problème lors du calcul de la carte de niveau 3.



a. La carte de niveau 3 posant problème. b. La carte de niveau 3 correcte.

FIG. 5.25 – Carte de niveau 3 d'un tore totalement inclus dans une autre région : le problème et la carte correcte.

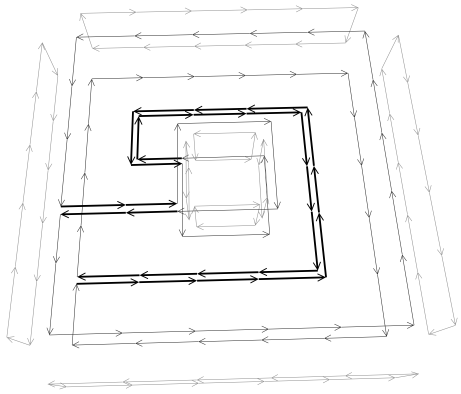
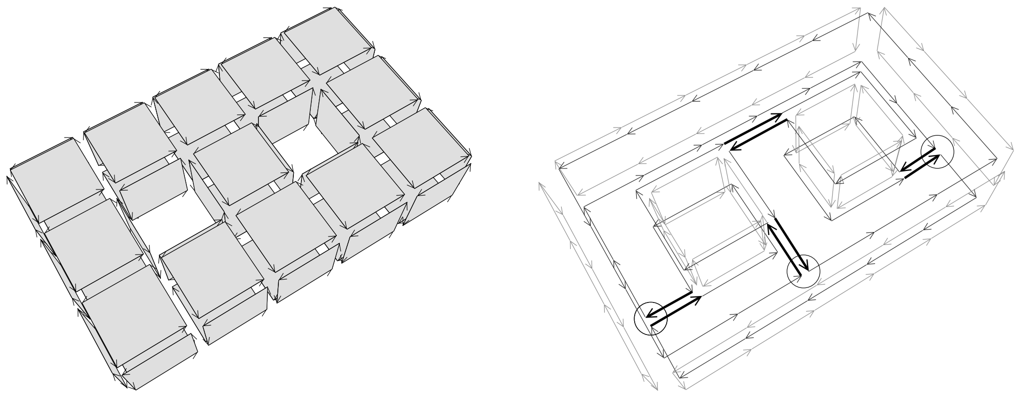


FIG. 5.26 – La carte de niveau 2 du même tore, avec de nombreuses arêtes fictives.

En effet, d'après la définition 31, nous fusionnons chaque couple d'arêtes adjacentes et alignées tel qu'il n'existe pas d'autre arête réelle incidente au même sommet. Si nous appliquons cette définition telle quelle, ainsi que notre principe de décalage d'arêtes fictives, nous obtenons alors la carte présentée figure 5.25.a, sur laquelle nous voyons qu'il y a toujours deux arêtes fictives pour les faces supérieure et inférieure du tore. En effet seule les arêtes alignées ont été fusionnées, or les arêtes fictives non pas de plongement, et n'ont donc pas été fusionnées.

Il est clair que ce n'est pas la carte que nous souhaitons obtenir. En effet, les arêtes fictives permettent de conserver chaque face connexe. Le fait d'en conserver plusieurs n'apporte aucune information supplémentaire. Il n'y a donc aucun intérêt à conserver ici deux arêtes au lieu d'une. De plus, pour cet exemple nous avons seulement deux arêtes par face, mais suivant l'ordre des fusions de faces, nous pourrions en avoir beaucoup plus, comme pour l'exemple présenté figure 5.26, où nous avons douze arêtes fictives pour la face



a. Le niveau 1. b. Le niveau 2 et ses arêtes fictives.

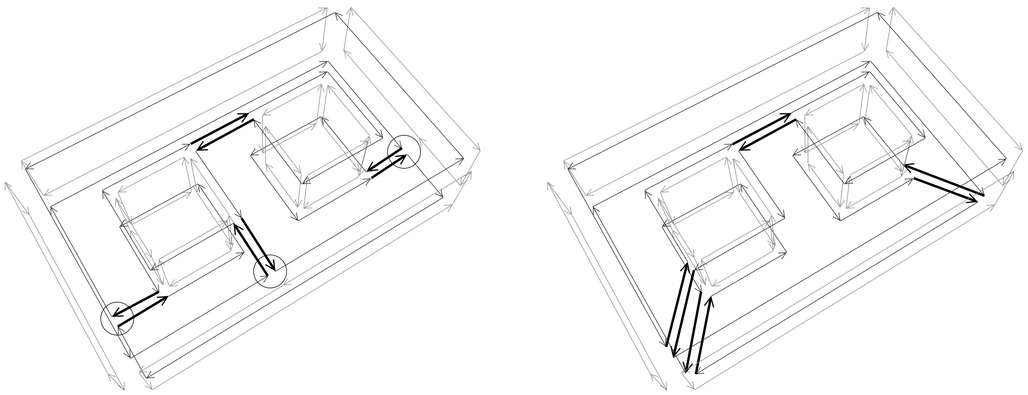
Fig. 5.27 – Les cartes de niveaux 1 et 2 d'un tore à deux trous.

inférieure du tore, plus deux pour la face supérieure.

Afin de régler ce problème et obtenir la carte de niveau 3 présenté figure 5.25.*b*, nous considérons simplement que deux arêtes fictives adjacentes sont alignées. Cela peut être envisagé car ces arêtes n'ont pas de plongement. Cette simple modification permet de résoudre ce dernier problème. Nous obtenons alors la carte de niveau 3 ayant une seule arête fictive par face trouée, y compris dans le dernier cas. À partir de cette carte, nous continuons le calcul des différents niveaux de simplification, que nous ne détaillons pas ici étant donné que nous obtenons exactement les mêmes cartes de niveaux 4 et 5 que pour le tore précédent (cf. figure 5.23).

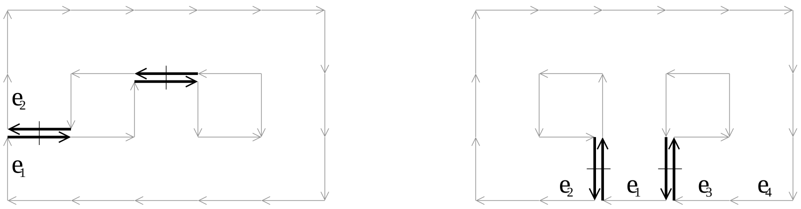
Nous étudions maintenant un dernier exemple un peu plus complexe, non pas pour introduire un nouveau problème, mais simplement pour vérifier le résultat obtenu pour un objet plus complexe : c'est le cas du tore à deux trous. Nous pouvons voir figure 5.27.*a* la carte de niveau 1 d'un tel tore, totalement inclus dans une autre région comme pour nos précédents exemples. Figure 5.27.*b* est présentée la carte de niveau 2 correspondante. Comme les faces supérieure et inférieure du tore possèdent chacune deux trous, nous avons donc pour chacune de ces faces deux arêtes fictives permettant de les conserver connexes. La figure 5.28.*a* présente la carte de niveau 3 obtenue en fusionnant uniquement les arêtes incidentes à des sommets de degré deux, et la figure 5.28.*b* la carte de niveau 3 correcte que nous obtenons avec les définitions modifiées.

Afin de bien comprendre le passage entre le niveau 2 et le niveau 3, et par la même occasion le principe d'extraction modifiée, nous l'étudions pour deux configurations différentes d'arêtes fictives. Pour simplifier les schémas et en faciliter la compréhension, nous nous limitons à l'étude de la face supérieure de ce tore. Nous présentons figure 5.29 cette face supérieure de la carte de niveau 2 présentée 5.27.*b*. Sur cette figure, nous avons dessiné comme précédemment les arêtes fictives en noir épais, et représenté la relation β_2 au moyen des petits segments de droite. Nous avons représenté deux possibilités pour cette face supérieure, suivant la position des arêtes fictives, mais il en existe d'autres.



a. Problème pour le niveau 3. b. Le niveau 3 correct.

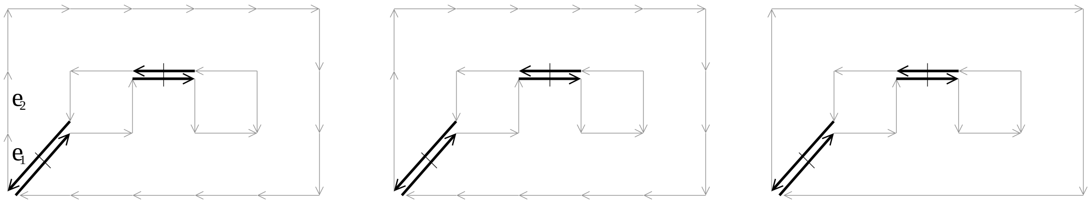
FIG. 5.28 – Le niveau 3 du tore à deux trous : le problème rencontré si l'on ne traite pas les arêtes fictives de manière particulière, et la carte de niveau 3 correcte.



a. b.

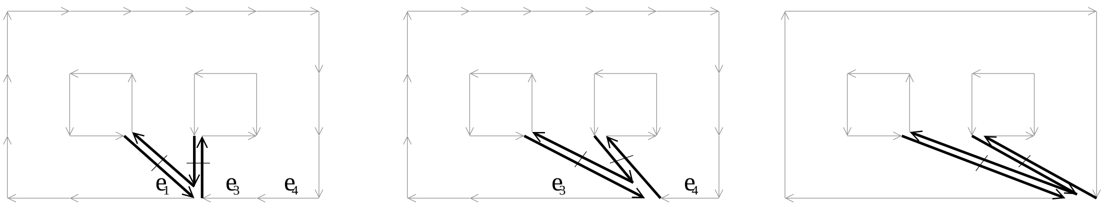
FIG. 5.29 – La face supérieure du tore à deux trous avec différentes positions des arêtes fictives.

Nous pouvons voir figure 5.30.a le résultat du déplacement de l'arête fictive, pour la face supérieure du tore présenté figure 5.29.a. Après avoir décalé cette arête, le sommet incident à e_1 et e_2 est vraiment de degré 2, même en tenant compte des arêtes fictives.



a. Après avoir décalé l'arête fictive. b. Après la fusion de e_1 et e_2. c. La carte de niveau 3 finale.

FIG. 5.30 – La face supérieure du tore à deux trous (figure 5.29.a) : quelques étapes avant d'arriver à la carte de niveau 3.



a. Après la fusion de e_1 et e_2. b. Après la fusion de e_3 et e_1. c. La carte de niveau 3 finale.

FIG. 5.31 – La face supérieure du tore à deux trous (figure 5.29.b) : quelques étapes avant d'arriver à la carte de niveau 3.

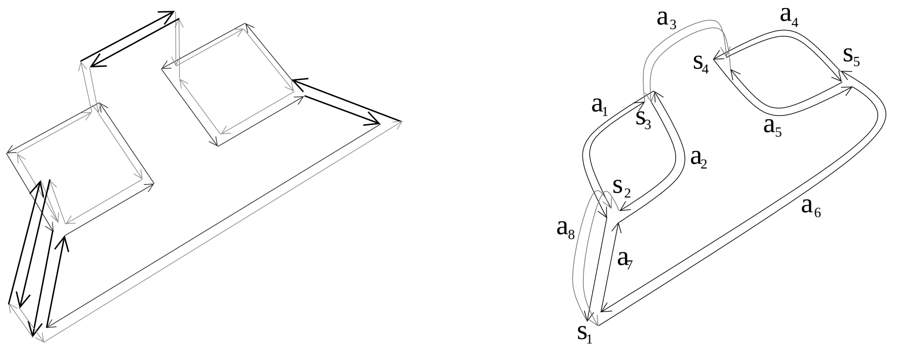
Nous pouvons donc ensuite effectuer la fusion d'arêtes de manière normale, et obtenir la carte présentée figure 5.30.b. Au final, nous obtenons, pour la face supérieure du tore, la carte de niveau 3 présentée figure 5.30.c. Nous pouvons vérifier, sur cette carte, que les fusions d'arêtes réelles ont toutes été réalisées de manière identique aux fusions effectuées si nous n'avions pas d'arête fictive.

Nous montrons maintenant quelques étapes intermédiaires de la construction de la carte de niveau 3 pour la deuxième configuration de la face supérieure du tore à deux trous présentée figure 5.29.b. Cette face diffère de la précédente par la position des arêtes fictives. La figure 5.31.a montre la carte obtenue après la fusion des arêtes e_1 et e_2. Après cette fusion, deux arêtes fictives sont incidentes au même sommet. L'opération de décalage effectuée avant la fusion des deux arêtes e_3 et e_1 va donc décaler deux arêtes fictives, et non plus une seule comme pour les exemples précédents. La figure 5.31.b montre la carte obtenue après la fusion de e_3 et e_1, et la figure 5.31.c la carte de niveau 3 obtenue au final. Cette carte est identique à la carte de niveau 3 obtenue en partant de l'autre configuration, à l'exception de la position des arêtes fictives.

Pour le tore entier, nous obtenons la carte de niveau 3 déjà présentée figure 5.28.b. À partir de ce niveau, nous fusionnons chaque couple de faces adjacentes et incidentes à une arête de degré un ou deux, n'entraînant pas de déconnexion de face, ni la suppression totale de la face (définition 30 du niveau 4 modifié). La carte de niveau 4 obtenue est présentée figure 5.32.a. Cette carte est composée d'une seule face, de dix-sept arêtes et de quatorze sommets, et la formule d'Euler donne bien un genre égal à deux.

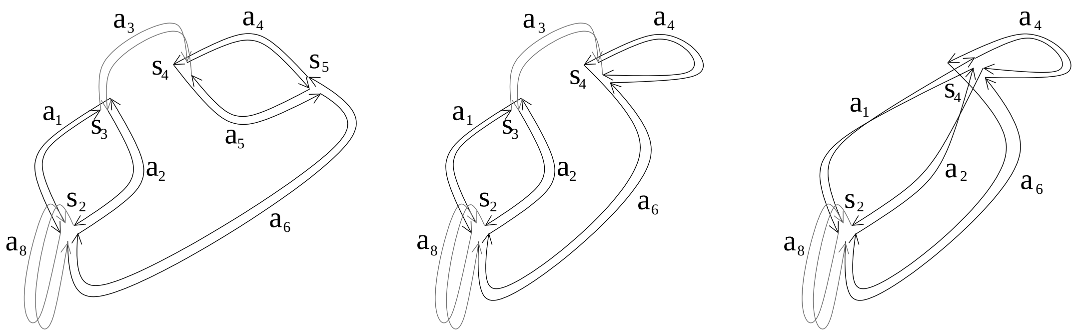
La figure 5.32.b montre la carte de niveau 5 en cours de construction. Nous avons pour le moment fusionné uniquement les arêtes adjacentes et incidentes à des sommets de degré deux. Cette carte est maintenant composée d'une face, de huit arêtes et de cinq sommets : nous caractérisons toujours un double dore. Mais cette carte n'est pas une représentation minimale. Nous continuons donc de fusionner les arêtes respectant les contraintes de la définition 32 du niveau 5 modifié.

Nous illustrons de manière progressive la construction de la carte de niveau 5 à partir de cette carte. Remarquons que, comme pour le cas du tore étudié figure 5.23, nous montrons ici un exemple de fusions d'arêtes. En effet, l'ordre dans lequel sont effectuées ces fusions



a. Le niveau 4. b. Le niveau 5 en cours de construction.

FIG. 5.32 – Le niveau 4 du tore à deux trous, et le niveau 5 en cours de construction, obtenu en fusionnant seulement les arêtes incidentes à des sommets de degré deux.



a. Après fusion de a_8 et a_7 le long de s_1. b. Après fusion de a_6 et a_5 le long de s_5. c. Après fusion de a_1 et a_3 le long de s_3.

FIG. 5.33 – Quelques étapes intermédiaires de la construction du niveau 5 du tore à deux trous.

est totalement arbitraire et aurait donc pu être différent. Mais nous pouvons vérifier que quel que soit cet ordre, nous obtenons toujours la même carte finale.

La figure 5.33.a présente la carte obtenue à partir de la carte précédente (cf. figure 5.32.b) en fusionnant les arêtes a_8 et a_7 le long du sommet s_1. Les arêtes fictives différentes de a_8 et a_7 sont dans un premier temps décalées sur le deuxième sommet incident à a_8, c'est-à-dire s_2. En pratique, une seule arête répond à ce critère et est donc décalée : c'est l'arête a_6.

Nous fusionnons ensuite les arêtes a_6 et a_5 autour du sommet s_5. Nous obtenons alors la carte présentée figure 5.33.b. Avant cette fusion, l'arête a_6 est décalée sur s_4 qui est le deuxième sommet incident à a_6. La figure 5.33.c montre la carte obtenue après la fusion des arêtes a_1 et a_3 le long de s_3. Nous pouvons remarquer que chaque fusion entraîne la suppression d'un sommet et d'une arête topologique, car nous fusionnons forcément deux

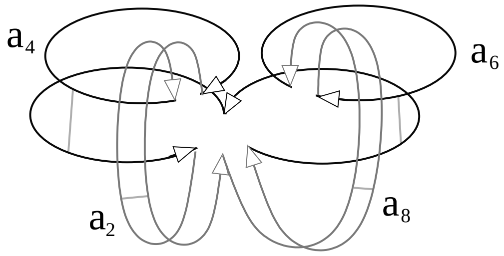


FIG. 5.34 – La carte de niveau 5 du tore à deux trous, obtenue par exemple après la fusion de a_6 et a_1 le long de s_2.

arêtes n'étant pas des boucles. Cette avant-dernière carte possède deux sommets, cinq arêtes et une face.

Enfin, nous fusionnons les arêtes a_6 et a_1 le long du sommet s_2. Nous obtenons alors la carte topologique présentée figure 5.34. Nous avons réorganisé les arêtes pour rendre cette figure plus compréhensible. Cette carte est la seule représentation minimale du tore à deux trous ayant une seule face. Elle est composée d'un sommet, quatre arêtes et une face. Chaque arête est une boucle, et nous ne pouvons donc plus effectuer aucune fusion. Nous obtenons bien, une nouvelle fois, la représentation minimale.

5.3.4 Preuve de la minimalité

Les exemples étudiés permettent de vérifier que les nouvelles définitions des niveaux de simplification amènent bien chaque fois à la représentation minimale. Cette représentation minimale est unique car le nombre de faces est fixée : il est égal au nombre de faces frontières de l'image. Nous avons étudié uniquement des objets totalement inclus dans une autre région, car ce sont ceux pour lesquels la gestion des arêtes fictives est primordiale afin d'obtenir cette représentation. De plus, nous avons vu lors de la présentation de notre premier exemple qui ne contenait pas d'arête fictive, que la représentation minimale est obtenue directement et sans problème. Nous donnons ici la preuve informelle afin de montrer que la carte topologique est la représentation minimale. La preuve formelle est possible mais serait très longue et technique ; nous préférons en exposer seulement les idées générales.

Nous classons les régions en deux catégories distinctes : celles représentées par des surfaces fermées et les autres. Remarquons tout d'abord que les seuls objets topologiques représentés sont des sphères et des tores à n trous. En effet, chaque surface est orientable et fermé. Avec la relation d'inclusion, nous pouvons caractériser plus finement une région, en parlant par exemple de sphère contenant un tore, mais toujours uniquement à partir de sphères et de n-tores. Nous ne traitons pas ici le cas de la sphère, pour lequel nous avons déjà montré section 5.3.2 que la représentation obtenue est bien minimale.

Pour les objets représentés par une surface fermée, la carte de niveau 3 correspondante est composée uniquement d'arêtes de degré un ou deux. En effet, cet objet est forcément

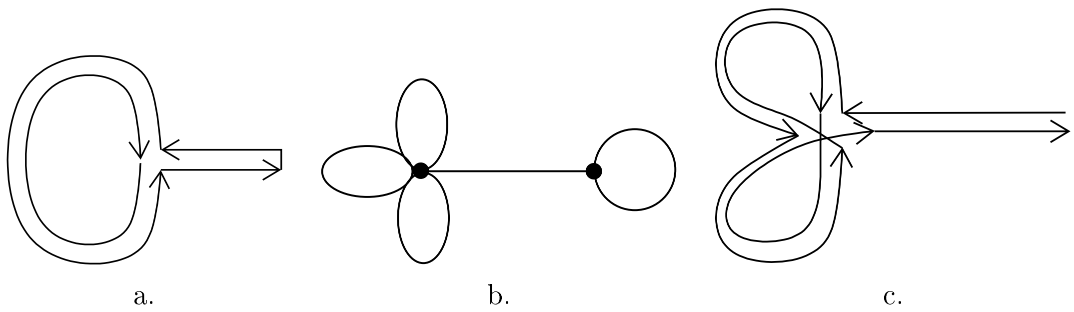


FIG. 5.35 – Trois configurations impossible à obtenir pour la carte de niveau 5.

totalement inclus dans une autre région, sinon sa surface ne serait pas fermée. Au cours du calcul de la carte de niveau 4, nous fusionnons toutes les faces le long de toutes les arêtes, à l'exception des fusions entraînant des déconnexions. Une déconnexion de face peut survenir uniquement pour une fusion de deux faces le long d'une arête de degré un (cf. section 5.3.2). Comme la carte de niveau 4 représentant cet objet est connexe et que toutes ses arêtes sont fictives, nous en déduisons que cette carte est composée d'une seule face. Nous allons ensuite, lors du calcul de la carte de niveau 5 fusionner chaque couple d'arêtes non boucles, étant donné que la carte est entièrement composée d'arêtes fictives. Nous arrivons forcément à une carte ayant un seul sommet et uniquement des arêtes en boucles.

Afin de prouver cela, il suffit de montrer qu'il est impossible d'arriver à une configuration semblable à celles présentées figure 5.35. En effet, ce type de configuration possède deux sommets différents, et une seule arête non boucle. Dans ce cas, il n'y a plus de couple d'arêtes non boucle à fusionner, et la carte de niveau 5 finale ne possède pas un seul sommet. Mais ces configurations ne peuvent pas survenir. En effet, lors de la construction de la carte de niveau 4, toutes les faces pouvant être fusionnées sans entraîner de déconnexion l'ont été. Or, dans ces cas, ainsi que dans les cas similaires, certaines fusions de faces n'entraînent pas de déconnexion. C'est évident pour le cas présenté figure 5.35.a, un peu plus difficile à voir pour celui de la figure 5.35.b qui ne représente pas la carte mais la schématise par ses sommets et arêtes.

Intuitivement, la carte de niveau 4 est composée uniquement d'arêtes de degré un, et ne possède aucun sommet de degré un. En effet, un tel sommet est adjacent à une seule arête, et la fusion de faces le long de cette arête n'entraîne pas de déconnexion. De plus, si la carte de niveau 4 n'a pas de sommet de degré un, alors la carte de niveau 5 n'en n'a pas non plus. En effet, seules les arêtes incidentes à des sommets de degrés deux sont fusionnées, et nous ne pouvons pas obtenir une carte composée seulement d'une boucle. Les seules configurations possibles ressemblant à celle de la figure 5.35.b vérifiant ces deux contraintes sont similaires à celle présentée figure 5.35.c. La configuration des brins appartenant aux deux boucles est la seule possible afin de ne pas avoir de sommet de degré un. Cette configuration est extensible à un nombre quelconque de boucles. Dans ce cas, nous pouvons vérifier que la fusion de faces le long d'une de ces deux boucles n'entraîne pas de déconnexion, ce qui montre que ce type de cas ne peut pas se produire.

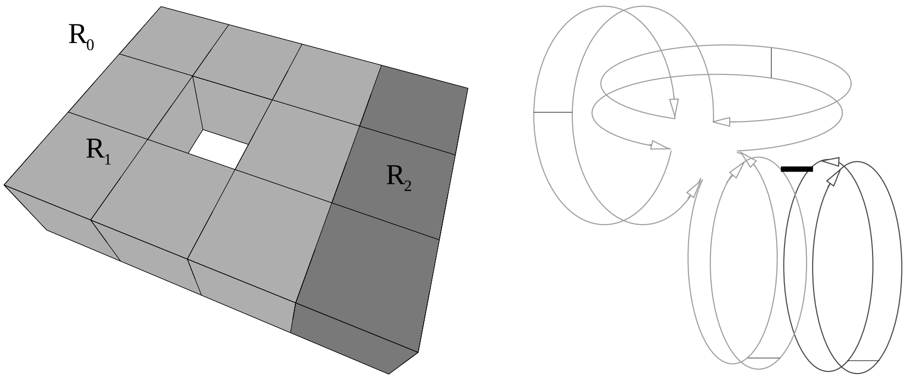


FIG. 5.36 – Un tore adjacent à un autre volume.

Comme ces configurations n'existent jamais, lorsque nous avons plus de deux sommets, il existe toujours au moins deux arêtes n'étant pas des boucles et pouvant donc être fusionnées. Chaque fusion d'arêtes diminue les nombres de sommets et d'arêtes de un, au final la carte obtenue possède toujours un seul sommet et plusieurs arêtes, suivant le genre du tore. Un tore à n trous, est donc représenté par une face, un sommet et $2n$ arêtes, ce qui est bien la représentation minimale.

Lorsqu'un objet est adjacent à d'autres régions, il est représenté par un certain nombre de faces frontières. Ces faces frontières ne peuvent pas être fermées car une telle face représente entièrement un volume, et nous sommes alors dans le cas d'un objet totalement inclus dans une autre région. Ces faces possèdent toutes un ou plusieurs bords, représentés par des arêtes réelles. Ces bords sont minimaux car chaque couple d'arêtes incidentes à des sommets de degré deux ont été fusionnées. Lorsqu'une face frontière est représentée par k bords, il y a $k-1$ arêtes fictives permettant de conserver cette face connexe. Ces arêtes fictives sont bien en nombre minimal. Mais certaines faces peuvent représenter plus d'information que seulement leurs bords.

Ce cas est illustré par la figure 5.36. Cet exemple représente un tore (région R_1) adjacent à un cube topologique (région R_2), tous deux inclus dans la région infinie (R_0). Sur la carte topologique correspondante, nous ne représentons pas les brins de la région infinie. Le tore est représenté par deux faces, une pour la frontière entre R_1 et R_2, et une autre pour celle entre R_1 et R_0. La région R_1 est représentée par deux faces, trois arêtes et un sommet, ce qui correspond bien au genre d'un tore. La première face est simplement une boucle, car tous les sommets appartenant à son bord sont de degré deux. La deuxième face est composée de trois arêtes et d'un sommet. En effet, cette face « contient » l'information que l'objet modélisé est un tore. Elle est composée de deux arêtes fictives, plus un brin appartenant à l'arête réelle. Le brin appartenant à l'arête réelle ne peut être ni déplacé, ni supprimé. Par contre, les autres arêtes étant fictives, elles peuvent être déplacées. Intuitivement, elles vont être positionnées sur le sommet incident à l'arête réelle. Elles vont être ensuite fusionnées lors du calcul de la carte de niveau 5, tant qu'il existe deux arêtes non boucles. Nous obtenons finalement uniquement des boucles incidentes à un seul sommet, en montrant que les configurations présentées figure 5.35 ne peuvent pas non plus se produire ici.

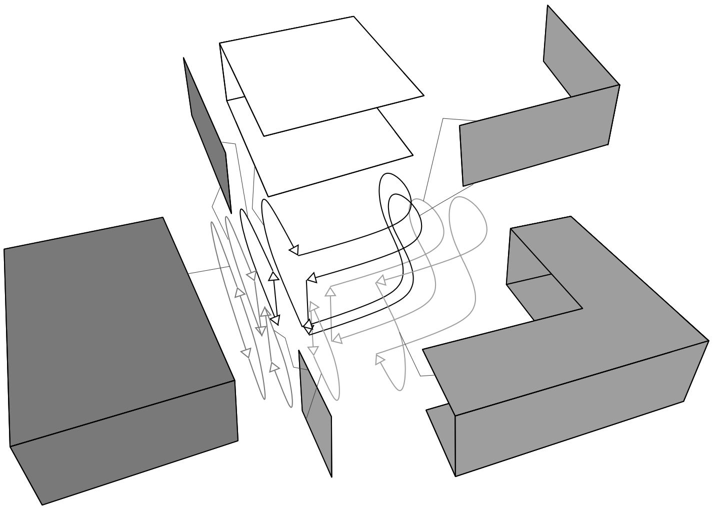


Fig. 5.37 – La carte de niveau 5 de notre premier exemple, avec son plongement.

5.4 Le plongement de la carte topologique

Nous étudions maintenant la manière de plonger la carte topologique. Nous nous limitons à cette carte de niveau 5, tout d'abord car c'est celle qui nous intéresse le plus, mais également car c'est la plus délicate à plonger. En effet, pour les niveaux 1, 2 et 3, les solutions déjà présentées en dimension 2 peuvent s'étendre facilement. Par exemple pour la carte de niveau 3, nous pouvons associer à chaque sommet topologique les coordonnées d'un point géométrique. Étant donné que chaque arête de ce niveau est un segment de droite, ce plongement simple suffit pour reconstruire le plongement de chaque cellule topologique.

5.4.1 Plongement face

Une première solution afin de plonger la carte topologique consiste à associer une surface à chaque face de la carte. Cette solution est relativement simple à mettre en œuvre, étant donné que nous plongeons un seul type de cellule. Nous pouvons voir figure 5.37 ce plongement pour la carte topologique du premier exemple que nous avons étudié. Nous pouvons vérifier sur cet exemple que deux demi-faces topologiques (β_3-cousues entre elles) sont associées à la même surface, comme pour la face distinguée figure 5.38. Mais afin de reconstruire correctement les plongements de chaque cellule, les liaisons entre les brins de la carte et le plongement doivent être réalisées « correctement ». Un brin b doit désigner le bord de la surface de plongement représentant l'arête incidente à b, et le sommet incident à b.

Sur notre exemple figure 5.38, le brin a désigne l'arête a_1 du plongement, du côté du sommet entre les arêtes a_1 et a_2, et le brin d (qui est β_3-cousu au brin a) désigne l'arête a_3 du côté du sommet entre les arêtes a_3 et a_2. Pour récupérer le plongement

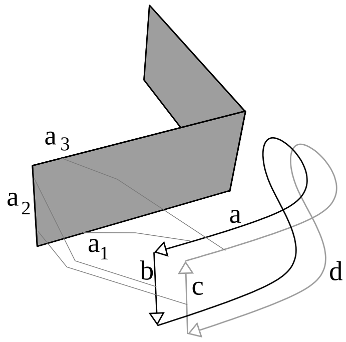


Fig. 5.38 – Zoom sur une face topologique et son plongement.

d'un sommet topologique, il suffit de récupérer les coordonnées du sommet du bon côté de l'arête désignée par ce brin. Pour récupérer le plongement d'une arête topologique incidente à un brin b, il faut parcourir le bord de la surface de plongement à partir de l'arête désignée par b, jusqu'à l'arête de plongement désignée par $\beta_3(b)$. Nous obtenons une suite de coordonnées qui est une courbe 1d représentant le plongement de cette arête. Pour récupérer le plongement d'une face topologique, il suffit de récupérer toute la surface de plongement, et enfin pour récupérer le plongement d'un volume topologique, il faut récupérer l'ensemble des plongements des faces topologiques qui le compose, et les mettre en contact suivant les β_2-coutures de la carte.

Une possibilité intéressante pour représenter ce plongement est l'utilisation des 2-G-cartes. Nous devons en effet représenter des surfaces qui sont donc des objets de dimension 2. De plus, il est intéressant ici d'utiliser les G-cartes au lieu des cartes afin de profiter de leur absence d'orientation. Si nous utilisons les cartes, chaque surface de plongement est représentée par une carte, donc avec une certaine orientation. Certaines opérations vont entraîner la fusion de deux surfaces de plongement. Si les deux orientations de ces surfaces ne sont pas identiques, nous devons inverser une des deux cartes avant de pouvoir les fusionner. Ce problème est résolu par l'utilisation des G-cartes, étant donné qu'elles ne sont pas orientées. Ces G-cartes sont l'équivalent des cartes topologiques en dimension 2, c'est-à-dire qu'elles ne contiennent pas de sommet de degré deux, et que nous associons à chacune de leurs arêtes un plongement arête ouverte et à chacun de leurs sommets les coordonnées d'un point 3d.

Ces 2-G-cartes de plongement représentent la géométrie des faces topologiques de la carte, comme nous pouvons le voir sur l'exemple de la figure 5.39. Chaque brin d'une même face désigne un brin du bord de la même 2-G-carte de plongement, à l'exception des brins appartenant à des arêtes fictives. En effet, une arête fictive est une arête « à l'intérieur » d'une même face, et sera donc associée à une arête n'appartenant pas au bord de la 2-G-carte de plongement. De manière intuitive, ce plongement peut être vu comme le complémentaire de la carte topologique. Tous les brins supprimés lors de la construction des niveaux 4 et 5 sont ajoutés dans des 2-G-cartes de plongement. En effet, ce plongement est un peu l'équivalent de la carte de niveau 3 étant donné qu'il représente les arêtes alignées de longueur maximale et uniquement les faces coplanaires. De plus,

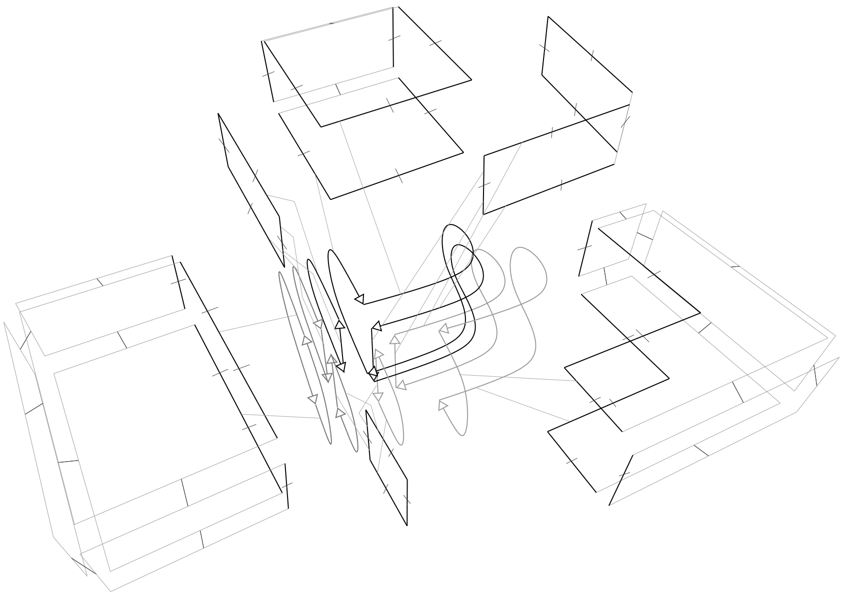


FIG. 5.39 – Le plongement face réalisé avec des 2-G-cartes.

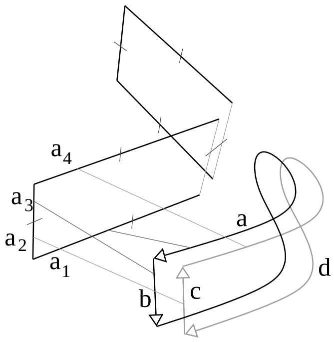


FIG. 5.40 – Zoom sur une face topologique et son plongement réalisé avec des 2-G-cartes.

comme chaque fusion effectuée lors de la construction des niveaux 4 et 5 entraîne une perte d'information géométrique, cette information est conservée dans les 2-G-cartes. Nous pouvons voir figure 5.40 comment est réalisé ce plongement pour la face présenté figure 5.38. Les deux demi-faces topologiques pointent vers des brins du bord de la même 2-G-carte.

L'inconvénient de ce plongement est le même que celui rencontré en dimension 2 : il représente des informations géométriques de manière redondante. En effet, un sommet sera représenté dans chaque G-carte de plongement, donc autant de fois qu'il y a de faces incidentes à ce sommet dans la carte. Pour l'exemple figure 5.39, le sommet central est incident à chacune des 6 faces topologiques. Il est donc représenté 6 fois, une fois dans chaque G-carte de plongement. Ces informations redondantes entraînent une perte en espace mémoire, mais également en temps d'exécution. En effet, lorsque nous voulons modifier les

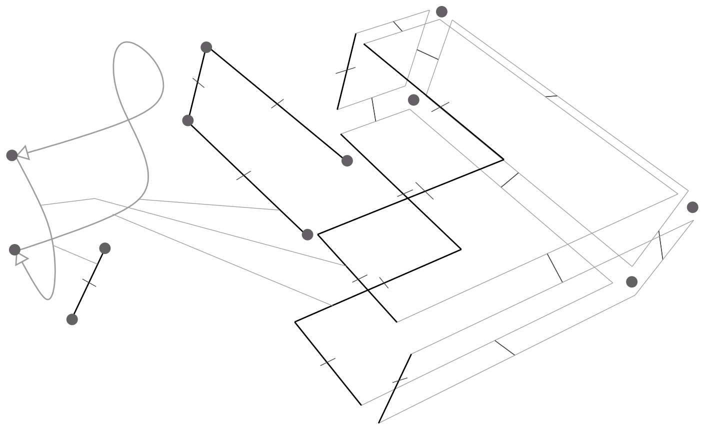


FIG. 5.41 – Le plongement face ouverte, arête ouverte et sommet pour une face topologique de notre exemple.

coordonnées d'un sommet topologique, nous devons le faire pour chacune des G-cartes de plongements le contenant.

5.4.2 Plongement face ouverte, arête ouverte et sommet

Un deuxième plongement, permettant d'éviter la représentation redondante de cellule géométrique, consiste à plonger chaque face topologique par une surface ouverte (c'est-à-dire sans son bord), chaque arête topologique par une courbe 1d ouverte (sans ses sommets extrémités) et chaque sommet topologique par un point géométrique.

Comme pour le plongement précédent, une manière d'implanter ce plongement consiste à associer à chaque face une 2-G-carte représentant la surface géométrique correspondante, mais cette fois sans en plonger le bord. Ce plongement sera en effet reconstruit à l'aide des plongements arêtes et sommets. Nous associons également à chaque arête topologique une 1-G-carte de plongement, représentant la courbe 1d ouverte et enfin à chaque sommet topologique les coordonnées du point géométrique correspondant.

Nous ne donnons pas d'exemple précis de ce plongement, car les figures correspondantes sont illisibles, mais présentons simplement figure 5.41 le plongement d'une face de la carte topologique utilisée pour le plongement précédent. Sur cette figure, nous voyons que les deux sommets topologiques de la face sont associés à deux sommets géométriques (symbolisés par les points). Les deux arêtes pointent vers des 1-G-cartes représentant le plongement des arêtes ouvertes. Enfin, chaque brin pointe vers un brin du bord de la 2-G-carte de plongement, qui est surface « ouverte », c'est-à-dire que son bord ne possède pas de plongement.

L'avantage de ce plongement est que, contrairement au précédent, il ne code aucune information géométrique de manière redondante. Mais nous voyons sur cet exemple qu'il est plus complexe à mettre en œuvre. De plus, afin de récupérer le plongement d'une cellule topologique, nous devons le reconstruire en combinant les plongements sommets, arêtes, et faces, ce qui entraîne un surcoût de complexité. Enfin, chaque brin doit maintenant conserver des informations sur trois types de plongements différents, ce qui entraîne une augmentation de l'espace mémoire nécessaire au codage d'un brin.

Il est clair que suivant les utilisations de la carte topologique et les besoins spécifiques d'une application particulière, nous utiliserons plutôt un plongement que l'autre. Mais une étude plus complète devra être effectuée afin de comparer l'espace mémoire et la complexité en temps d'exécution dans le cadre précis d'une application, et d'une implantation particulière. En effet, ces deux complexités dépendent fortement de la manière dont sont implantés les cartes topologiques et les plongements associés. C'est pour cette raison qu'il est difficile de faire ici cette étude de manière générale.

Dans la suite de ce travail, nous utilisons le premier plongement présenté afin de simplifier les explications. Mais ce choix n'est en aucun cas une contrainte, car il est facile de passer d'un plongement à l'autre, et car des fonctions génériques manipulent ce plongement sans dépendre de la manière dont il est implanté. De ce fait, pour changer le plongement utilisé, il suffira de modifier ces fonctions de base et non les fonctions de plus haut niveau manipulant la carte topologique.

5.5 Un premier algorithme d'extraction

Les définitions des différents niveaux de carte présentées section 4.2 donnent immédiatement, comme pour la dimension 2, un premier algorithme d'extraction qui en est leur transcription. Mais avant de présenter cet algorithme, nous revenons brièvement sur les opérations de fusion en 3d, qui sont à la base de ces définitions et de cet algorithme.

5.5.1 Les fusions dans les 3-cartes

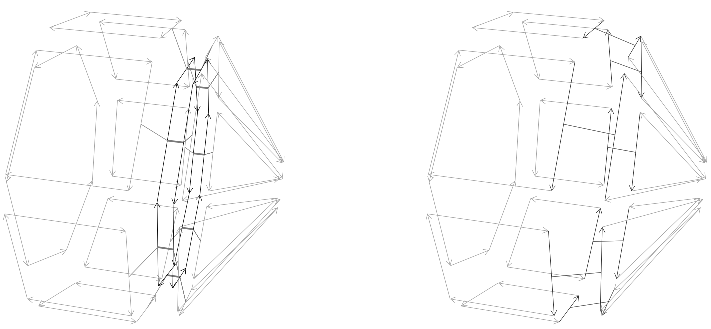
Nous avons déjà vu, section 4.4.1, le principe général des fusions en dimension n. Étant donné que nous travaillons ici en dimension 3, nous avons trois types différents de fusions qui sont la fusion de volumes (dimension 3), la fusion de faces (dimension 2) et la fusion d'arêtes (dimension 1). La fusion de deux i-cellules C_1 et C_2 le long d'une $(i-1)$-cellule C n'est possible uniquement lorsque C est de degré 1 ou 2.

La figure 5.42 présente un exemple de fusion de volumes. La figure 5.42.a montre une carte représentant deux objets β_3-cousus le long de la face dessinée en foncé, et la figure 5.42.b le résultat de la fusion de ces deux volumes.

L'algorithme 19 effectuant cette opération est simple à définir. Il est local car il traite un ensemble de brins (ici les brins appartenant à la demi-face[2] incidente à b) sans ordre particulier, en effectuant un traitement dépendant uniquement de la configuration locale autour de ce brin. De manière générale, cette technique simplifie le nombre de cas différents à traiter et donc l'écriture des algorithmes.

Pour chaque brin d de cette demi-face, nous commençons par conserver les brins qui sont β_2-cousus à d et à $\beta_3(d)$ (figure 5.43.a), puis nous β_2-décousons ces deux brins (fi-

[2]Rappelons que les brins de la demi-face incidente à un brin b sont ceux de l'orbite $< \beta_1 > (b)$.



a. Avant la fusion. b. Après la fusion.

FIG. 5.42 – Fusion de volumes en 3d.

Algorithme 19 Fusion de volumes en 3d

Entrée : Un brin b β_3 cousu

Résultat : Les deux volumes incidents à b et à $\beta_3(b)$ sont fusionnées.

pour chaque *brin d appartenant à la demi-face incidente à b* **faire**

  $b_1 \leftarrow \beta_2(d)$; $b_2 \leftarrow \beta_{32}(d)$;

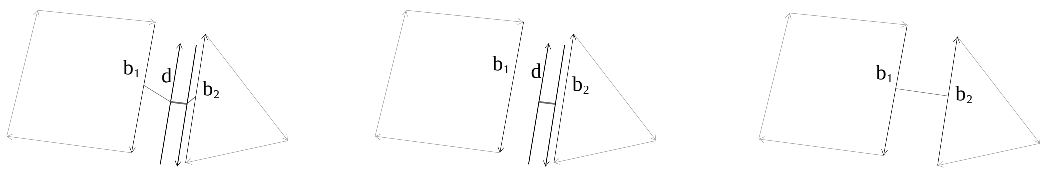
  β_2-découdre(d) ; β_2-découdre($\beta_3(d)$);

  β_2-coudre(b_1, b_2);

  détruire les brins $\beta_3(d)$ et d;

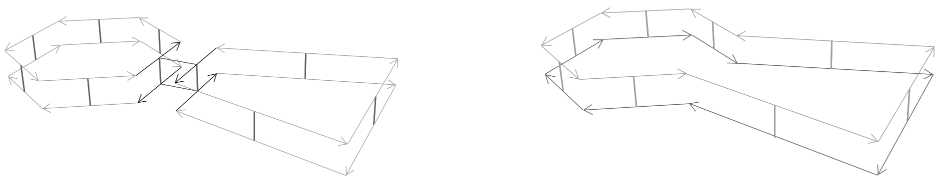
---

gure 5.43.b), et enfin nous β_2-cousons b_1 et b_2, les deux brins initialement cousus à d et $\beta_3(d)$, et détruisons les brins $\beta_3(d)$ et b (figure 5.43.c). Ce traitement étant effectué pour chaque brin de la demi-face, nous obtenons bien au final la carte présentée figure 5.42.b, résultat de la fusion des deux volumes. La complexité de cet algorithme est linéaire en le nombre de brins de la demi-face incidente au brin donné en entrée de l'algorithme, et ce quelle que soit la taille des deux régions. Nous voyons ici un exemple de traitement où les cartes combinatoires permettent un gain important en complexité, par rapport au traitement similaire par exemple sur la matrice de voxels. De plus, cet algorithme fonctionne quelle que soit la configuration de la carte, à condition que le brin de départ de l'algorithme soit β_3-cousu. Il fonctionne de manière identique lorsque la face le long de laquelle nous



a. Avant de traiter d. b. β_2 décousures de d et $\beta_3(d)$. c. β_2 couture de b_1 et b_2.

FIG. 5.43 – Fusion de volumes en 3d, un traitement local.



a. Avant la fusion. b. Après la fusion.

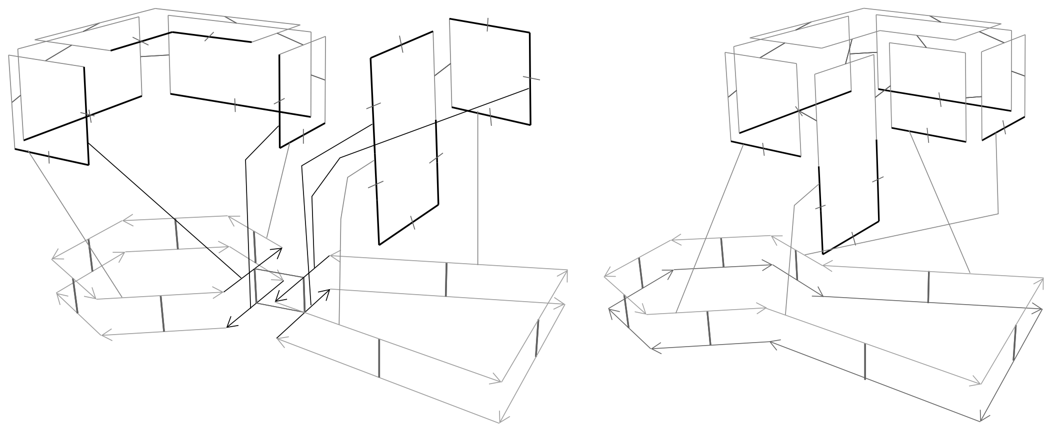
FIG. 5.44 – Fusion de faces en 3d.

effectuons la fusion est de degré un (cas d'une face à l'intérieur d'un même volume), ce qui permet de ne pas différencier ces deux cas.

Cet algorithme présente seulement la partie topologique de la fusion de volumes. Mais aucune modification de plongement n'est nécessaire. En effet, nous ne modifions aucun plongement de face, nous détruisons seulement celui associé à la face topologique supprimée. Si nous avons un plongement sommet, arête ouverte et face ouverte, les opérations de couture et décousure se chargent des mises à jour nécessaires. Par exemple, si lors d'une décousure, nous déconnectons un sommet topologique en deux, le plongement sommet correspondant sera dupliqué afin que chaque sommet topologique ait exactement un plongement sommet. De manière réciproque, si une opération de couture regroupe deux orbites sommets distinctes, l'un des deux plongements sommets sera supprimé. Ces opérations sont faites de manière transparente, ce qui permet de changer le plongement sans avoir à changer quoi que ce soit dans les opérations de haut niveau.

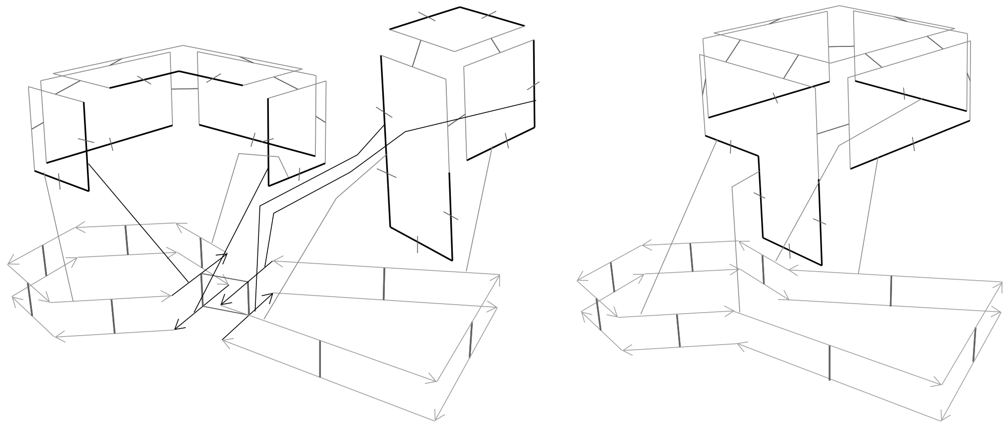
La fusion de faces en 3d peut être vue comme deux fusions de faces 2d. En effet, une face topologique en dimension 3 est composée de deux demi-faces identiques β_3-cousues entre elles. La figure 5.44.a montre deux faces adjacentes, et incidentes à une arête de degré deux, et la figure 5.44.b le résultat de leur fusion de long de cette arête. L'algorithme effectuant cette fusion peut se ramener à deux appels consécutifs de l'algorithme effectuant la fusion de faces en 2d, une fois pour le brin b et une deuxième fois pour le brin $\beta_3(b)$. Nous ne le détaillons donc pas ici. Cet algorithme fonctionne quelle que soit la configuration des deux faces, à condition que le brin donné à l'algorithme soit β_2-cousu.

La partie topologique effectuant cette fusion de faces ne pose donc pas de problème. Nous présentons maintenant brièvement les modifications du plongement. Nous considérons ici que nous avons un plongement uniquement face, implanté avec des 2-G-cartes. La figure 5.45.a présente les deux mêmes faces topologiques de l'exemple précédent, mais maintenant avec leur plongement. Afin de ne pas rendre les figures illisibles, nous avons seulement représenté quelques liaisons entre la carte et le plongement. Ces deux faces étant différentes, elles possèdent donc deux G-cartes de plongement distinctes. Après la fusion de ces deux faces, nous devons modifier le plongement afin qu'il tienne compte de cette fusion. Nous pouvons voir le résultat de cette modification figure 5.45.b. Afin d'effectuer cette modification, nous devons simplement α_2-coudre deux à deux tous les brins de plongement représentant l'arête supprimée lors de la fusion. Cette opération s'effectue sans difficulté particulière. De plus, nous utilisons les fonctions de la 2-G-carte, qui suivant son plongement font les mises à jour nécessaires afin qu'il reste valide. De ce fait, nous n'avons pas



a. Avant la fusion. b. Après la fusion.

FIG. 5.45 – Modification du plongement lors de fusion de faces 3d.

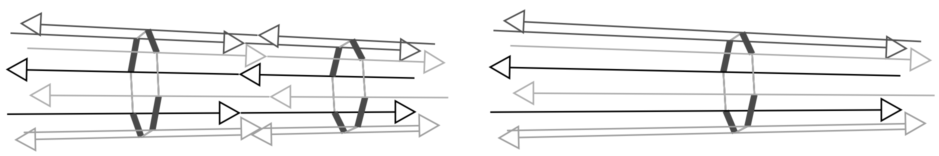


a. Avant la fusion. b. Après la fusion.

FIG. 5.46 – Un autre exemple de modification du plongement lors de fusion de faces 3d.

besoin de nous préoccuper de la manière dont sont plongées les 2-G-cartes. Nous pouvons également remarquer qu'il n'y a pas d'autre modification à faire : chaque brin de la carte conserve le même lien vers un brin de plongement, si l'on excepte les 4 brins supprimés lors de la fusion.

Mais cette opération de modification de plongement n'est pas tout le temps une α_2-couture, comme nous pouvons le voir sur l'exemple présenté figure 5.46. En effet, dans ce cas, nous α_2-cousons des faces coplanaires. Afin de conserver les G-cartes minimales, nous devons alors fusionner ces faces après les avoir cousues. Ce traitement est effectué de manière transparente par la G-carte sans traitement supplémentaire. De plus, après ces opérations de fusions, certaines arêtes peuvent être fusionnées lorsqu'elles sont incidentes à un sommet de degré deux. Mais ces fusions sont également effectuées de manière trans-



a. Avant la fusion. b. Après la fusion.

FIG. 5.47 – Fusion d'arêtes en 3d.

parente par la G-carte. C'est pour ces raisons que nous ne détaillons pas dans la plupart de nos algorithmes la gestion du plongement. Lorsque nous devons modifier le plongement de manière précise, nous employons alors des méthodes génériques, comme par exemple l'affectation des coordonnées d'un sommet topologique, l'affectation d'un plongement à une face, ou encore la fusion de deux plongements faces sans détailler ces opérations.

Nous étudions figure 5.47 un exemple de fusion d'arêtes le long d'un sommet de degré deux. Sur cette figure, nous avons représenté uniquement les deux arêtes à fusionner. Nous avons dessiné de même couleur deux brins cousu par β_2, (relation représentée par les trais gris clair fin), et représenté β_3 par les traits épais et foncés. Les deux arêtes de la figure 5.47.a sont composées de 8 brins chacune. De manière générale, une arête possède au minimum 4 brins (car nous avons toujours des cartes fermées) et au maximum 8 brins car nous représentons des voxels et avons donc au maximum 4 faces incidentes à une arête (et deux brins par face). La figure 5.47.a montre le résultat de la fusion d'arêtes.

L'algorithme 20 effectue cette fusion d'arêtes. Pour qu'il puisse fonctionner correctement, le sommet entre les deux arêtes à fusionner doit être de degré deux. Cet algorithme n'est pas local, contrairement à celui effectuant la fusion de volumes, car la définition des cartes combinatoires n'est pas homogène pour la définition des sommets[3]. De ce fait, nous devons parcourir les brins du sommet incident au brin de départ dans l'ordre, en alternant une involution β_2 et une involution β_3, et en effectuant un traitement légèrement différent dans ces deux cas.

Afin de comprendre le fonctionnement de cet algorithme, nous étudions son déroulement sur l'exemple présenté figure 5.48. Nous avons pris un exemple assez simple où les deux arêtes à fusionner sont composées chacune de 6 brins. La figure 5.48.a montre la configuration de départ, avant de rentrer dans la boucle « Tant que ». Nous avons initialisé certains brins, et β_2-décousu le brin b qui est le brin de départ. Dans la boucle, nous traitons le brin courant, d, et conservons le brin de la dernière arête traitée afin de le coudre correctement à la nouvelle arête, c'est le brin old. Nous commençons par effectuer le traitement local consistant à découdre d pour β_0 et β_1 puis par β_1-coudre b_1 et b_2 qui étaient les deux brins β_0 et β_1 cousus à d. Ensuite nous différencions deux cas, suivant que old est cousu par β_2 ou par β_3 à d. Ces deux cas se produisent alternativement, étant donné que nous utilisons β_2 et β_3 à tour de rôle. Pour cette première itération, nous sommes dans le premier cas où nous β_2-décousons old puis le β_2-cousons avec b_1 qui est localement le brin résultat de la fusion. Puis nous affectons le prochain brin à traiter, qui sera $\beta_{31}(d)$ (figure 5.48.b). Nous pouvons alors détruire le brin d, après l'avoir β_3-décousu, et traiter

[3]L'algorithme similaire pour les G-cartes serait lui local et donc beaucoup plus simple.

Algorithme 20 Fusion d'arêtes en 3d

Entrée : Un brin b incident à un sommet de degré deux

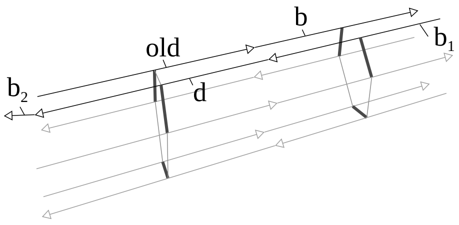
Résultat : Les deux arêtes incidentes à b et à $\beta_0(b)$ sont fusionnées.

```
d ← β21(b);
β2-découdre(b);
old ← β0(b);
pair ← faux;
tant que d ≠ b faire
    b1 ← β0(d) ;  b2 ← β1(d);
    β0-découdre(d) ;  β1-découdre(d);
    β1-coudre(b1, b2);
    si pair est faux alors
        β2-découdre(d) ;  β2-coudre(old, b1);
        new ← β31(d) ;  β3-découdre(d);
    sinon
        β3-découdre(d) ;  β3-coudre(old, b1);
        new ← β21(d) ;  β2-découdre(d);
    pair ← ¬pair ;  old ← b1;
    détruire le brin d;
    d ← new;
b1 ← β0(d) ;  b2 ← β1(d);
β0-découdre(d) ;  β1-découdre(d);
β1-coudre(b1, b2);
β3-découdre(d) ;  β3-coudre(old, b1);
détruire le brin d;
```

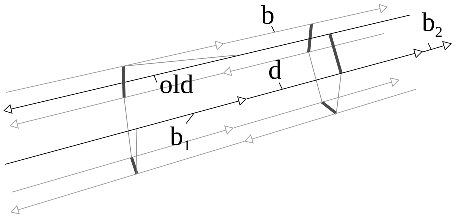
le prochain brin. Ce prochain brin aura un traitement similaire, sauf que nous sommes maintenant dans le deuxième cas étant donné que *old* est β_3-cousu à d. Le traitement de ce cas est similaire au précédent, en échangeant β_2 et β_3.

Nous traitons ainsi chaque brin de l'arête (cf. figures 5.48.b à e), et retombons finalement sur le brin de départ (cf. figure 5.48.f). Nous devons alors traiter ce dernier brin en dehors de la boucle, car son traitement est légèrement différent des autres brins. Nous obtenons alors la carte présentée figure 5.48.g qui est bien le résultat de la fusion des deux arêtes initiales.

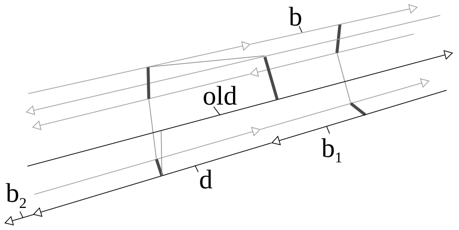
Cet algorithme fonctionne dans n'importe quelle configuration, à condition que le sommet supprimé soit de degré deux, et que la carte soit fermée. Sa complexité est linéaire en le nombre de brins de l'arête. La modification du plongement est faite de manière transparente lors des opérations de coutures et décousures. Si les deux arêtes fusionnées sont alignées, les deux arêtes de plongement correspondantes seront également fusionnées. Sinon le traitement est dépendant du plongement utilisé. Nous pouvons par exemple, si les 2-G-cartes de plongement possèdent un plongement arête, fusionner quand même les deux arêtes et ajouter le plongement du sommet supprimé dans celui-ci. Si elles ont uniquement un plongement sommet, nous n'effectuons alors pas la fusion.



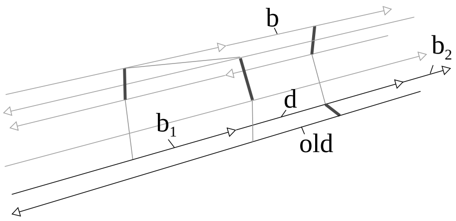
a. Configuration initiale.



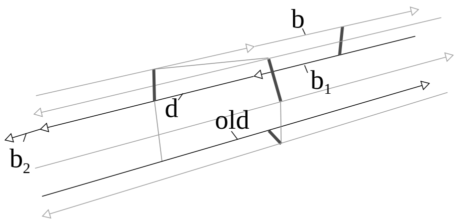
b.



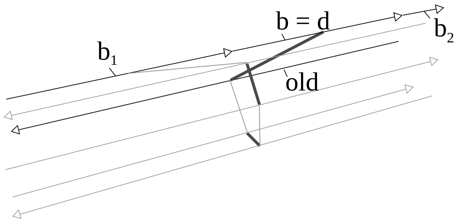
c.



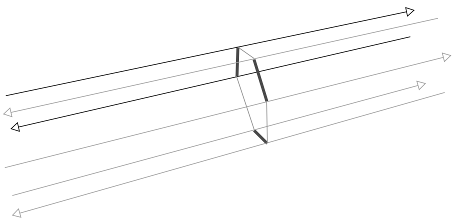
d.



e.



f.



g. Le résultat final de la fusion.

FIG. 5.48 – Déroulement de l'algorithme de fusion d'arêtes en 3d.

5.5.2 L'algorithme d'extraction naïf

L'algorithme 21 calculant la carte de niveau n d'une image 3d segmentée en régions est simplement la transcription des définitions de nos cinq niveaux de simplification. De plus, cet algorithme est l'extension directe de l'algorithme naïf en dimension 2. Il prend en entrée une image 3d segmentée en régions, et le niveau de simplification souhaité (un entier entre 1 et 5), et retourne la carte de ce niveau représentant l'image.

La première ligne de cet algorithme commence par construire la carte de niveau 0 correspondant à l'image I, c'est-à-dire la carte complète représentant chaque voxel de cette image plus une face représentant la région infinie et fermant cette carte. Ensuite, chaque ligne de l'algorithme permet de passer au niveau de simplification suivant, par application directe de la définition correspondante.

Algorithme 21 Extraction « naïve » de la carte de niveau n en 3d

Entrée : Une image I de $n_1 \times n_2 \times n_3$ voxels
n le niveau de la carte que nous voulons extraire
Sortie : La carte de niveau n de l'image I.

1 Construire la carte de niveau 0 correspondant à I;
2 Fusionner tout couple de volumes adjacents appartenant à la même région;
si $n \geq 2$ **alors**
3   Fusionner tout couple de faces adjacentes, coplanaires, incidentes à une arête de degré un ou deux et n'entraînant pas de déconnexion de face;
    si $n \geq 3$ **alors**
4   Fusionner tout couple d'arêtes adjacentes et alignées a_1 et a_2 incidentes à un sommet s tel qu'il n'existe pas d'arête réelle différente de a_1 et de a_2 incidente à s;
        si $n \geq 4$ **alors**
5   Fusionner tout couple de faces adjacentes, incidentes à une arête de degré un ou deux et n'entraînant pas de déconnexion de face ni la suppression totale de la face;
            si $n = 5$ **alors**
6   Fusionner tout couple d'arêtes adjacentes a_1 et a_2 n'étant pas des boucles et incidentes à un sommet s tel qu'il n'existe pas d'arête réelle différente de a_1 et de a_2 incidente à s;

7 Calculer l'arbre d'inclusion des régions;

La ligne 2 construit la carte de niveau 1 correspondant à I, en fusionnant chaque couple de volumes adjacents appartenant à la même région. C'est uniquement pour la construction de ce niveau que nous effectuons le lien entre l'image à représenter et la carte. Tous les autres niveaux vont ensuite utiliser uniquement le degré des cellules et des tests de déconnexion. Comme pour la dimension 2, cette étape peut être réalisée en complexité linéaire en le nombre de brins de la carte. En effet, nous parcourons chaque face de la carte, testons si les deux volumes incidents sont de même région, et lorsque c'est le cas effectuons la fusion. Nous parcourons chaque face exactement une seule fois, car lorsqu'une fusion n'est pas effectuée à un moment de l'algorithme, nous savons qu'elle ne pourra jamais être réalisée et n'avons donc pas besoin de tester à nouveau cette face. De plus, cette technique de construction permet de traiter les cas où nous devons fusionner un volume avec lui même, et les cas où nous devons fusionner plusieurs fois le même couple de volumes le long de différentes faces.

La ligne 3 construit la carte de niveau 2, en appliquant la définition 29 qui tient compte des problèmes de déconnexion de face et conserve éventuellement des arêtes fictives afin de les résoudre. Pour réaliser cette étape, nous parcourons chaque arête de la carte et regardons si elle vérifie les conditions de la définition de ce niveau. Pour cela, nous testons si cette fusion va entraîner la déconnexion de la face, en parcourant l'orbite β_1. Cette opération est en *O(nombre de brins de la face)*. De plus, contrairement au niveau précédent, une fusion peut ne pas être possible à un certain moment car elle entraîne une déconnexion, mais ne plus entraîner de déconnexion et donc être possible plus tard. Cela oblige, après une fusion de faces, à recommencer à tester toutes les arêtes de la carte. La complexité est alors quadratique en le nombre de brins de la carte. Lors de la construction de ce niveau, nous fusionnons uniquement des faces coplanaires, et donc pouvons effectuer exactement les mêmes fusions dans les 2-G-cartes de plongements. De plus, lorsqu'une fusion entraîne la déconnexion de la carte, nous ne l'effectuons pas afin de conserver une arête fictive. Dans ce cas, nous α_2-cousons les deux arêtes correspondantes dans la carte de plongement, ce qui crée une arête fictive dans celle-ci. Pour ce niveau, chaque face de la carte topologique possède une face de plongement identique.

Nous construisons ensuite la carte de niveau 3 à la ligne 4 de l'algorithme. Pour cela, nous fusionnons chaque couple d'arêtes adjacentes et alignées a_1 et a_2, incidentes à un sommet s tel qu'il n'existe pas d'arête réelle différente de a_1 et a_2 incidente à s. Nous parcourons chaque sommet de la carte, et cherchons s'il existe deux arêtes alignées incidentes au sommet courant s. Cette opération s'effectue en *O((nombre d'arêtes incidentes à s)^2)*. De plus, au cours de cette boucle autour du sommet, nous pouvons en même temps compter le nombre d'arêtes réelles incidentes à s[4]. À la fin de cette boucle, si nous avons trouvé deux arêtes réelles et que ce nombre est 2, nous pouvons effectuer la fusion. Mais si les deux arêtes sont fictives[5] et que ce nombre est 0, nous pouvons également effectuer la fusion. Dans ces deux cas, nous appelons tout d'abord l'algorithme décalant toutes les arêtes fictives incidentes à s, puis effectuons la fusion des deux arêtes de manière classique. Cet algorithme de décalage a une complexité en *O(nombre d'arêtes incidentes à s)*. La boucle principale qui parcours l'ensemble des sommets de la carte est linéaire en le nombre de brins de la carte. En effet, contrairement au niveau précédent, lorsque une fusion de deux arêtes n'est pas possible, elle ne le sera jamais, et nous n'avons donc pas besoin de tester à nouveau ce sommet. En majorant largement, nous obtenons une complexité quadratique en le nombre de brins de la carte. Nous n'avons pas besoin d'effectuer un calcul de complexité plus fin, étant donné que l'étape précédente est déjà en complexité quadratique. Les modifications du plongement sont, comme pour le niveau précédent, exactement les mêmes que celles de la carte. En effet, nous fusionnons les arêtes alignées et ces fusions peuvent donc être réalisées également dans les cartes de plongement.

La construction de la carte de niveau 4 (effectuée ligne 4 de l'algorithme) est très proche de celle du niveau 2. La seule différence entre ces deux niveaux porte sur le fait qu'au niveau 2 ces faces sont uniquement coplanaires alors que pour le niveau 4 non. Nous utilisons donc la même technique, avec les mêmes problèmes, et employons les mêmes solutions. Les deux complexités sont donc identiques. La seule différence se situe au niveau du plongement, où contrairement à la carte de niveau 3, les G-cartes de plongements ne sont pas fusionnées mais seulement α_2-cousues, afin de conserver chaque face de plongement plane.

Enfin, nous construisons la carte topologique à la ligne 6 de l'algorithme. Ici aussi, la construction de ce niveau est identique à celle du niveau 3, exception faite de la géométrie. Mais la géométrie n'entre en jeu ni pour la technique de construction, ni pour la complexité.

[4]Afin de tester en *O(1)* si une arête est réelle ou fictive, il suffit de marquer chaque arête fictive lors du calcul du niveau précédent ainsi que pour le calcul du niveau 4. En effet, c'est uniquement pour ces deux niveaux que nous pouvons créer des arêtes fictives.

[5]Rappelons que nous considérons que deux arêtes fictives sont toujours alignées.

Nous effectuons un test supplémentaire afin de déterminer si une arête est une boucle ou non, mais ce test peut s'effectuer simplement en *O(nombre d'arêtes incidentes à s)*. Nous obtenons alors exactement la même complexité que celle du calcul du niveau 3. Nous avons maintenant construit la carte de niveau n représentant l'image I. Nous devons construire l'arbre d'inclusion des régions qui permet de résoudre le problème de déconnexion de volumes.

5.5.3 Calcul de l'arbre d'inclusion

Cet algorithme 22 est quasiment le même que celui présenté en dimension 2 (section 4.4.3).

Algorithme 22 Calcul de l'arbre d'inclusion en 3d

Entrée : La carte C d'une image I segmentée en régions
La liste L des régions de l'image I
Sortie : L'arbre d'inclusion $\mathcal{A}$ correspondant.

Démarquer chaque région de L;
Ajouter un nœud associé à R_0 dans $\mathcal{A}$;
1 **pour chaque** *région* R_i *de* L *non marquée* **faire**
    $b_{init} \leftarrow représentant(R_i)$;
    parent $\leftarrow$ le nœud associé à $région(\beta_3(b_{init}))$;
2   **pour chaque** *brin* b *de la composante connexe incidente à* b_{init} **faire**
        si $région(b)$ *non marquée* **alors**
            Ajouter un nœud associé à $région(b)$ dans $\mathcal{A}$;
            Ajouter ce nœud comme fils de *parent* dans $\mathcal{A}$;
            Marquer $région(b)$;
retourner $\mathcal{A}$

La seule différence entre ces deux algorithmes se situe lors du calcul de *parent*, la région contenant la composante connexe en cours de traitement. En effet, nous récupérons ici la région du brin β_3 cousu au brin représentant de la région en cours de traitement, alors que c'était le brin β_2 cousu en dimension 2. Le principe de l'algorithme est donc le même : parcourir les régions de l'image, dans l'ordre haut-bas, derrière-devant et droite-gauche, et chaque fois que nous trouvons une région non traitée parcourir la composante connexe incidente à cette région. Toutes les régions de cette composante connexe sont incluses dans la région du brin β_3 cousu au brin représentant de la région en cours de traitement, de par la contrainte que nous fixons sur ce brin représentant.

En effet, pour que cet algorithme fonctionne, nous devons vérifier trois contraintes, de manière similaire à la dimension 2 :

- chaque brin doit connaître sa région d'appartenance ;
- chaque région r connaît un des brins de la carte associée à cette région. Nous appelons un tel brin le *représentant* de sa région. Ce brin est soit le brin ayant comme plongement sommet le pointel le plus en haut à gauche et derrière de r, soit, si un tel brin n'existe pas, le brin ayant ce même pointel dans son plongement face ;

– nous connaissons la liste de toutes les régions de l'image, sans la région infinie. Cette liste doit être ordonnée de sorte qu'une région R_i est inférieure à une autre région R_j si et seulement si R_i se rencontre avant R_j lors du parcours de l'image de haut en bas, de derrière à devant et de gauche à droite.

Ces trois propriétés sont assurées par nos algorithmes d'extraction. La première lors de la construction de la carte de niveau 0, la deuxième également lors de cette construction, puis en nous assurant de la conserver lors de chaque fusion, et enfin la dernière par un parcours préalable de l'image, en classant les régions rencontrées selon cet ordre. Ce sont ces propriétés qui nous assurent que la région contenant la composante connexe en cours de traitement est bien la région *parent*. Comme pour la dimension 2, cet algorithme est linéaire en le nombre de brins de l'image. De plus, il fonctionne de manière identique, quel que soit le niveau de la carte lui étant fournie, à condition qu'elle respecte les trois contraintes citées.

Nous avons présenté de manière détaillée chaque étape de l'algorithme naïf d'extraction d'une carte à partir d'une image. La complexité globale de cet algorithme est finalement quadratique en le nombre de brins de la carte. Cela peut être gênant pour traiter de grosses images, qui vont être représentées par beaucoup de brins. Mais il est possible d'améliorer cette complexité en cherchant un ordre sur les fusions de faces, afin d'éviter de tester à nouveau chaque brin après chaque fusion. Nous verrons que ce point fait partie d'une de nos perspectives de recherche. Mais cette complexité importante n'est finalement pas très gênante, étant donné que cet algorithme est uniquement l'algorithme naïf, et que nous présentons maintenant l'algorithme optimal qui a une complexité linéaire.

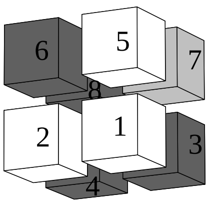
5.6 Les précodes et l'algorithme optimal d'extraction

L'algorithme optimal d'extraction est l'extension à la dimension 3 de celui présenté en dimension 2. Il effectue un balayage de l'image de haut en bas, de derrière à devant et de gauche à droite avec une fenêtre de $2 \times 2 \times 2$ voxels, et exécute le traitement correspondant à la configuration de cette fenêtre. Nous étudions les configurations à traiter afin d'extraire chaque niveau de simplification. Pour chaque niveau, nous devons traiter des nouvelles configurations en plus de celles du niveau précédent. Cette manière de définir les précodes permet de factoriser les configurations se traitant de manière similaire. Nous revenons tout d'abord sur la notion de précode et de précode partiel en dimension 3.

5.6.1 Les précodes en 3d

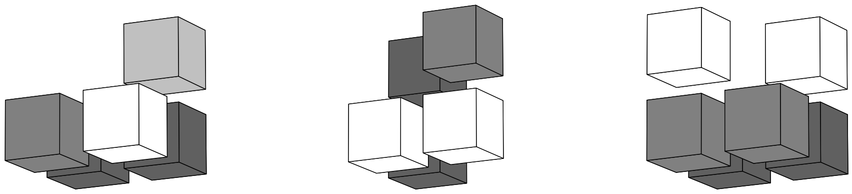
Nous avons déjà présenté les notions de précode et précode partiel section 4.5.1, et ce en dimension n. En dimension 3, nous balayons l'image de haut en bas, de derrière à devant et de gauche à droite. Au cours de ce balayage, nous appelons *voxel courant* le voxel en cours de traitement (celui en bas, devant et à droite de la fenêtre courante).

Un *précode* en dimension 3 est une partition de l'ensemble des 8 voxels de l'hypercube de longueur 2. Nous pouvons voir figure 5.49 quatre exemples de précodes 3d. Comme pour la dimension 2, nous représentons un précode en affectant une couleur différente à chaque élément de la partition. Deux voxels de même couleur appartiennent à la même



a. b. c. d.

FIG. 5.49 – Quatre exemples de précodes en 3d.



a. b. c.

FIG. 5.50 – Deux exemples de précodes partiels (a et b) et un contre-exemple (c).

région et réciproquement. Si nous numérotons les voxels de la manière présentée figure 5.49.a, (le voxel courant a le numéro 1), ces quatre précodes sont équivalents respectivement aux partitions suivantes : $\{\{1,2,5\},\{3,4,6,8\},\{7\}\}$, $\{\{1,2\},\{3,7\},\{5\},\{4,6,8\}\}$, $\{\{1,2,6,7,8\},\{3\},\{4\},\{5\}\}$ et $\{\{1,6,7,8\},\{2,5\},\{3,4\}\}$.

Un précode 3d est *non variété* si et seulement s'il existe dans ce précode une partition n'étant pas 6-connexe. Les trois premiers précodes de la figure 5.49 sont des précodes variété, alors que le quatrième est un précode non variété. Un précode *partiel* est une partition d'un sous-ensemble des cellules d'un précode, chaque ensemble de la partition devant être un ensemble 6-connexe. Nous pouvons voir figure 5.50 deux exemples de précodes partiels en dimension 3 (a et b) ainsi qu'un contre-exemple de précode partiel (c). La partition associée au précode présenté figure 4.17.c n'est pas un précode partiel car un ensemble de voxels appartenant à la même région n'est pas 6-connexe.

Nous avons vu lors de la présentation des précodes, section 4.5.1, que les nombres de Stirling et de Bell[6] permettent de calculer le nombre de précodes en dimension n. Cette formule nous a permis de calculer qu'il existait 15 précodes en dimension 2, ce qui est facilement vérifiable de manière exhaustive. Cette même formule nous donne que le nombre de précodes en dimension 3 est égal à B_8 qui vaut 4140. Il existe donc beaucoup trop de précodes pour en donner la liste exhaustive. Un autre nombre intéressant, est le nombre de précodes variétés. En dimension 2, nous l'avons calculé en regardant pour chacun des 15 précodes lesquels étaient des variétés. Mais cette solution n'est pas applicable ici. Nous avons dans un premier temps cherché une formule de dénombrement permettant de calculer

[6]Les nombres de Bell sont donnés par la formule de récurrence $B_l = \sum_{i=0}^{l-1} \frac{B_i.l!}{i!(l-i)!}$ avec $B_0 = B_1 = 1$.

ce nombre en dimension n, malheureusement sans succès pour le moment. Ce point fait partie de nos perspectives de recherches.

Afin de palier l'absence de formule, nous avons calculé ce nombre par programme. Nous avons pour cela généré les précodes de manière automatique, et testé ensuite chaque précode. Nous utilisons pour cela le graphe représentant l'hypercube de dimension n. Chaque sommet représente une n-cellule et deux sommets sont reliés par une arête si les deux n-cellules correspondantes sont $2n$-voisins. Chaque sommet est étiqueté par la région à laquelle il appartient. Il suffit ensuite de tester si chaque ensemble de sommets ayant la même étiquette est bien un ensemble connexe. Ce programme nous a permis de trouver qu'il existe 958 précodes variétés en dimension 3. Ce nombre est petit en regard des 4140 précodes existant au total. Mais il est encore trop important pour envisager l'écriture de chaque méthode de traitement correspondant. Nous allons voir que nous n'avons pas besoin de traiter l'ensemble de ces 958 cas, car un grand nombre d'entre eux se traitent de manière similaire. C'est ici que la définition de la carte topologique de manière hiérarchique apporte pleinement son intérêt.

5.6.2 L'algorithme optimal et général d'extraction

L'algorithme 23 général et optimal d'extraction est simple, comme en dimension 2, puisqu'il se résume par un balayage de l'image en exécutant le code associé au précode courant. Comme pour la dimension 2, cet algorithme fonctionne pour tout type d'image : segmentées ou non, étiquetage fort ou faible.

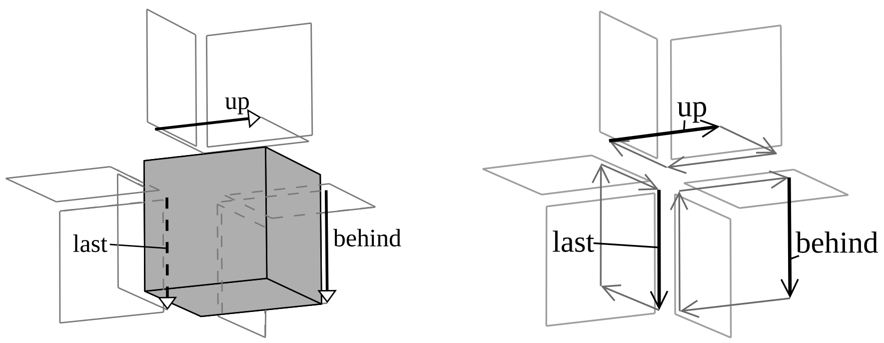
Algorithme 23 Extraction optimale de la carte de niveau n en 3d

Entrée : Une image I segmentée en régions de $n_1 \times n_2 \times n_3$ voxels
n le niveau de la carte que l'on veut extraire
Sortie : La carte de niveau n de l'image I.

```
1 last ← Construire le bord supérieur de la carte;
2 pour k = 1 à n3 + 1 faire
      pour j = 1 à n2 + 1 faire
          pour i = 1 à n1 + 1 faire
              last ← Exécuter le code associé au précode (i, j, k) du niveau n;
3 Calculer l'arbre d'inclusion des régions;
```

L'image I en entrée de cet algorithme doit être segmentée en régions. Elle est composée des voxels de coordonnées $1 \dots n_1$ en x, de $1 \dots n_2$ en y et de $1 \dots n_3$ en z. Elle est entourée d'une région infinie contenant l'ensemble des voxels n'appartenant pas à l'image. Le précode (i, j, k) est le précode ayant comme voxel courant (celui en bas, devant et à droite de la fenêtre $2 \times 2 \times 2$) le voxel (i, j, k).

Avant de balayer l'image, nous commençons par créer le bord supérieur de la carte. En effet, comme pour la dimension 2, la carte correspondant aux voxels plus anciens pour l'ordre de parcours doit être déjà construite. Cet invariant garantit, à tout instant, qu'il existe une face à gauche, derrière et au-dessus du voxel courant. Ces trois faces permettent de raccrocher le voxel courant à la carte déjà construite. Afin de désigner ces faces, nous



a. Avec le voxel courant. b. Et sans.

FIG. 5.51 – Les brins *last*, *up* et *behind* par rapport au voxel courant.

conservons un brin particulier sur chacune, que nous appelons respectivement *last*, *up* et *behind* (ou l, u et b en abrégé). Nous pouvons voir figure 5.51 la position de ces brins particuliers, tout d'abord par rapport au voxel courant, puis sans ce voxel afin de mieux voir leurs positions. Sur ces figures, nous n'avons pas représenté l'ensemble des orientations des brins afin de les rendre plus lisibles, mais elles peuvent se déduire aisément à partir des seules orientations représentées. La position de ces brins à l'intérieur des faces a été choisie de manière arbitraire. Nous devons simplement fixer cette position qui est primordiale lors des algorithmes associés aux précodes.

Avant de parcourir l'image, le voxel courant (qui est le prochain voxel à traiter) est celui ayant comme coordonnées $(1, 1, 1)$. Afin que notre invariant soit conservé, nous devons avant de traiter ce voxel, avoir construit la carte des voxels plus anciens, c'est-à-dire le voxel à gauche $(0, 1, 1)$, les voxels derrière $(i, 0, 1)$, $\forall i$ et les voxels dessus $(i, j, 0)$, $\forall i, j$. C'est cette carte qui est appelée bord supérieur de l'image. De plus, comme pour la dimension 2, nous créons ce bord de manière particulière afin de nous affranchir des problèmes liés aux précodes de bord de l'image. Nous verrons section 5.6.8 comment ce bord est construit afin de régler ces problèmes, et comment il intervient lors de l'extraction d'une carte.

Lors du balayage de l'image, nous conservons uniquement le brin *last*, car les brins *up* et *behind* peuvent se retrouver aisément à partir de ce brin à l'aide des algorithmes 24 et 25.

Algorithme 24 Calcul du brin *up* en 3d.

Entrée : *last* le brin à gauche du voxel courant
Sortie : *up* le brin au-dessus du voxel courant.

$up \leftarrow \beta_{02}(last)$;
tant que up *est* β_3*-cousu* **faire**
  $up \leftarrow \beta_{32}(up)$)
$up \leftarrow \beta_1(up)$;
retourner up;

Ces deux algorithmes fonctionnent suivant le même principe. L'algorithme 24 commence par initialiser le brin *up* par $\beta_{02}(last)$. Ce brin appartient à l'arête incidente aux

Algorithme 25 Calcul du brin *behind* en 3d.

Entrée : *last* le brin à gauche du voxel courant
Sortie : *behind* le brin derrière le voxel courant.

$behind \leftarrow \beta_2(last)$;
tant que behind *est* β_3*-cousu* **faire**
  $behind \leftarrow \beta_{32}(up)$)
$behind \leftarrow \beta_{11}(behind)$;
retourner behind;

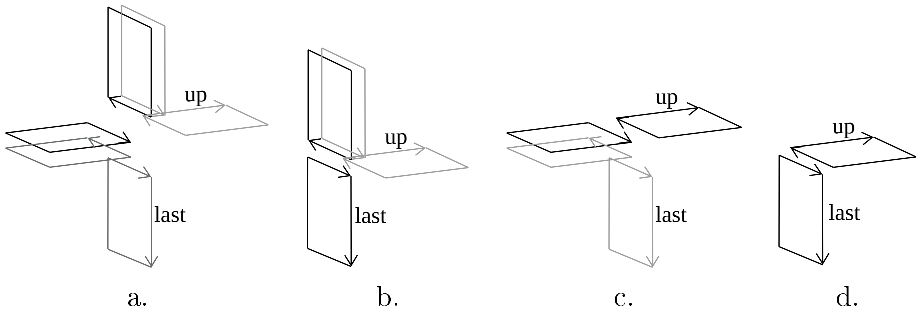


Fig. 5.52 – Les quatre seules configurations possible des faces entre celle contenant *last* et celle contenant *up*.

deux faces contenant *last* et *up*. Nous tournons ensuite autour de cette arête, en utilisant β_{32} tant que le brin courant est β_3-cousu. En effet, toutes les faces entre la face contenant *last* et celle contenant *up* sont β_3-cousus. Lorsque le brin courant n'est pas β_3-cousu, c'est qu'il appartient à la face contenant *up*, et il suffit alors d'appliquer β_1 pour trouver le brin *up*. L'algorithme 25 effectue le même traitement mais pour un brin initial différent, ainsi que pour le « déplacement » final.

La complexité de ces deux algorithmes est linéaire en le nombre de faces incidentes à l'arête entre la face contenant *last* et celle contenant *up* pour le premier, et la face contenant *last* et celle contenant *behind* pour le deuxième. Ce nombre de faces est toujours égal à 0, 1 ou 2, étant donné que chaque face représente un surfel. Nous pouvons voir figure 5.52 les quatre seules configurations possibles des faces entre celle contenant *last* et celle contenant *up*. Sur cette figure, nous avons dessiné deux brins de même couleur lorsqu'ils appartiennent à la même région. Nous n'avons pas représenté tous les brins afin de ne pas surcharger les schémas. Nous voyons donc que, dans le pire des cas présenté figure 5.52.*a*, l'algorithme calculant le brin *up* va effectuer uniquement 2 itérations. Dans le meilleur des cas présenté figure 5.52.*d*, ce même algorithme ne va pas entrer dans cette boucle, car dès l'initialisation le brin *up* est β_3-libre. Il en est exactement de même pour le calcul du brin *behind* et les configurations possibles de faces entre celle contenant *last* et celle contenant *behind*.

Après avoir créé le bord supérieur de l'image dans l'algorithme 23, le brin *last* est initialisé afin d'être sur la face de gauche par rapport au voxel $(1, 1, 1)$. Ensuite, l'algorithme effectue le balayage de l'image en exécutant le traitement correspondant au précode

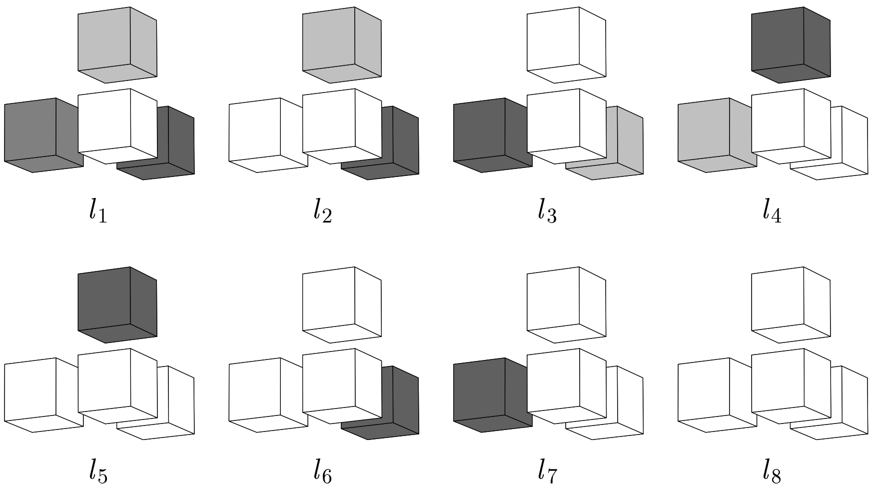


FIG. 5.53 – Les huit précodes à traiter afin de calculer la carte lignel en 3d.

courant. Chacun de ces précodes modifie localement la carte, en utilisant les trois brins distingués *last*, *up* et *behind*, et en retournant le prochain brin *last* pour le traitement du précode suivant. Nous verrons section 5.6.8 que de par la manière particulière dont le bord est construit, aucun traitement particulier n'est nécessaire pour les précodes se trouvant au bord de l'image, ce qui simplifie beaucoup l'algorithme.

Enfin, après le balayage de l'image, nous avons construit la carte de niveau n représentant l'image I. Il ne reste plus qu'à calculer l'arbre d'inclusion des régions afin de positionner les éventuelles différentes composantes connexes de la carte les unes par rapport aux autres. Pour cela, nous utilisons l'algorithme 22 déjà présenté section 5.5.3. Comme pour la dimension 2, le parcours préalable de l'image afin d'obtenir la liste des régions triées selon leur ordre d'apparition n'est plus nécessaire pour ce second algorithme d'extraction. En effet, cette liste peut maintenant être créée durant le même parcours que celui construisant la carte. Cet algorithme optimal d'extraction est générique car il permet d'extraire n'importe quel niveau de simplification. Pour cela, il faut définir pour chacun de ces niveaux les précodes à distinguer et la manière de les traiter.

5.6.3 Les précodes pour la carte lignel

Afin de calculer la carte lignel, nous effectuons la fusion de chaque couple de volumes adjacents et appartenant à la même région. De par notre ordre de parcours, il est uniquement possible de fusionner le voxel courant avec ses voisins gauche, derrière et dessus. Les différents cas à traiter sont simplement l'ensemble des combinaisons de fusion ou non du voxel courant avec ces trois voisins. Il y a donc 8 précodes différents à traiter, présentés figure 5.53 (que nous appelons l_i pour précode lignel numéro i). Le précode l_1 représente le cas où le voxel courant n'est fusionné avec aucun de ses voisins (C_3^0), les précodes l_2, l_3 et l_4 les cas où il est fusionné avec un seul de ses voisins (C_3^1), les précodes l_5, l_6 et l_7 les

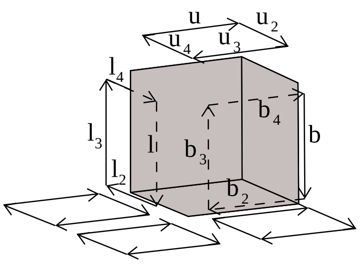
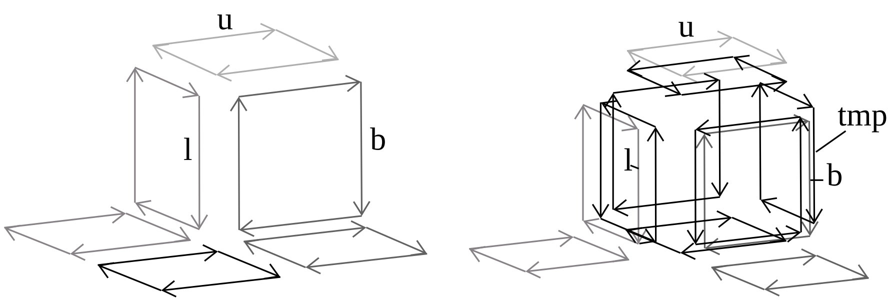


FIG. 5.54 – Les brins obligatoirement présents avant de traiter un précode de niveau 1.



a. Carte avant le traitement.

b. Le résultat.

FIG. 5.55 – Le précode l_1 avant et après son traitement.

cas où il est fusionné avec deux de ses voisins (C_3^2) et enfin le précode l_8 le cas où il est fusionné avec ses trois voisins (C_3^3).

La configuration de la carte avant le traitement de chacun de ces précodes est la même, et est présentée figure 5.54. En effet, nous savons qu'il existe les trois faces autour du voxel courant de par notre invariant. Ce sont les faces contenant les brins *last*, *up* et *behind*. Nous ne connaissons aucune autre face, présente obligatoirement avant le traitement d'un précode, car ces huit précodes sont des précodes partiels et nous ne connaissons pas la région des voxels libres. Nous avons également représenté trois autres faces entre les voxels (i, j, k) et $(i, j, k+1)$, mais ces faces n'entrent pas en compte pour les différents traitements, elles permettent seulement de mieux visualiser la figure. Afin de simplifier la désignation des brins, nous notons $l_2 = \beta_1(l)$, $l_3 = \beta_1(l_2)$ et $l_4 = \beta_1(l_3)$, les trois autres brins de la face contenant le brin *last*. Nous notons de manière similaire les autres brins appartenant aux faces contenant les brins *up* et *behind*.

Comme pour la dimension 2, le traitement d'un précode consiste à donner la suite d'opérations transformant la carte courante avant de traiter ce précode en la carte à obtenir après ce traitement. Ces opérations simulent les fusions correspondant à ce précode. De par le nombre important de précodes différents, nous ne donnons pas ici, contrairement à la dimension 2, chaque algorithme de traitement. La figure 5.55.a montre la carte courante avant de traiter le précode l_1, et la figure 5.55.b celle à obtenir. L'algorithme 26 permettant de passer de la première carte à la seconde est simple, étant donné qu'il consiste simplement

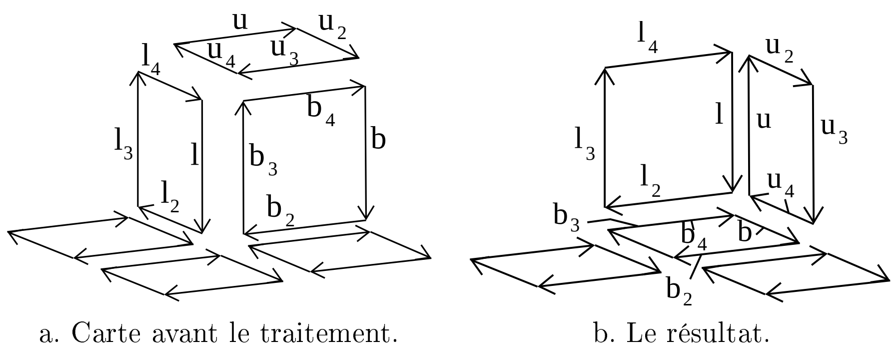


FIG. 5.56 – Le précode l_8 avant et après son traitement.

à créer un cube, et à β_3-coudre sa face de gauche à la face contenant *last*, sa face de dessus à la face contenant *up* et sa face de derrière à la face contenant *behind*.

Algorithme 26 Code associé au précode l_1 en 3d

Entrée : *last*, *up* et *behind*
(x, y, z) les coordonnées du nouveau sommet
Sortie : le « prochain » *last*.

tmp ← créer un cube topologique ;
β_3-coudre la face contenant *last* et celle contenant $\beta_{2112}(tmp)$;
β_3-coudre la face contenant *up* et celle contenant $\beta_{021}(tmp)$;
β_3-coudre la face contenant *behind* et celle contenant $\beta_2(tmp)$;
Le plongement du sommet incident à $\beta_{11}(tmp) \leftarrow (x, y, z)$;
retourner *tmp*;

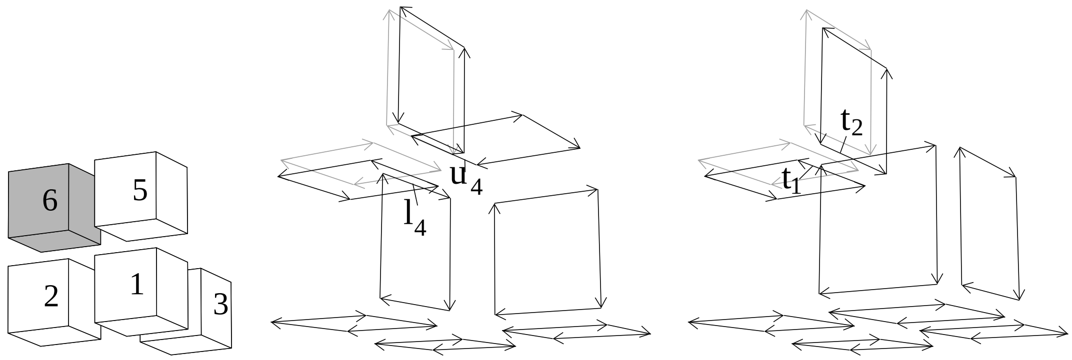
Pour le plongement, nous associons simplement les coordonnées du nouveau sommet au plongement sommet du brin $\beta_{11}(tmp)$. En effet, c'est le seul sommet n'ayant pas de plongement, tout les autres sommets appartiennent à une orbite possédant déjà un plongement dans la carte avant le traitement. Suivant le plongement utilisé cette mise à jour du plongement se fera de manière différente. Mais la majorité des mises à jour de plongement sont faites lors des opérations de couture. Par exemple pour un plongement face, lors d'une β_3-couture de deux faces entre elles, celle n'ayant pas de plongement pointera désormais sur le plongement de la face qui en possède un.

La figure 5.56.a montre la configuration de la carte avant de traiter le précode l_8 et la figure 5.56.b celle à obtenir. L'algorithme 27 effectue la transformation de la première carte en la seconde. Cet algorithme présente une manière d'effectuer cette transformation, mais il est possible d'effectuer le même traitement de manière différente. Nous avons ici privilégié la conservation des brins de la carte initiale, afin d'éviter des allocations et désallocations mémoire successives coûteuses en temps d'exécution. De plus, nous avons essayé d'être minimal en nombre d'opérations. Par exemple, nous conservons les brins l_3, b_2 et u_2 à leurs places initiales, ce qui évite une β_2-décousure et une β_2-couture pour chacun de ces brins.

Algorithme 27 Code associé au précode l_8 en 3d

Entrée : *last*, *up* et *behind*
(x, y, z) les coordonnées du nouveau sommet
Sortie : le « prochain » *last*.

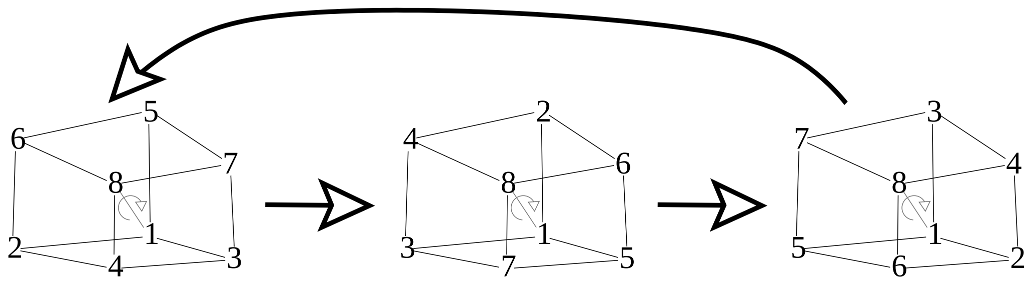
1 $t_1 \leftarrow \beta_2(l_4)$; $t_2 \leftarrow \beta_2(u_4)$; β_2-découdre(l_4) ;
si $t_1 \neq u_4$ **alors**
$\quad$ β_2-découdre(u_4) ; β_2-coudre(t_1, t_2) ;
2 $t_1 \leftarrow \beta_2(l)$; $t_2 \leftarrow \beta_2(b_3)$; β_2-découdre(l) ;
si $t_1 \neq b_3$ **alors**
$\quad$ β_2-découdre(b_3) ; β_2-coudre(t_1, t_2) ;
3 $t_1 \leftarrow \beta_2(u_3)$; $t_2 \leftarrow \beta_2(b_4)$; β_2-découdre(u_3) ;
si $t_1 \neq b_4$ **alors**
$\quad$ β_2-découdre(b_4) ; β_2-coudre(t_1, t_2) ;
4 $t_1 \leftarrow \beta_2(l_2)$; β_2-découdre(l_2) ; β_2-coudre(t_1, b_3);
5 $t_1 \leftarrow \beta_2(b)$; β_2-découdre(b) ; β_2-coudre(t_1, u_3);
6 $t_1 \leftarrow \beta_2(u)$; β_2-découdre(u) ; β_2-coudre(t_1, l_4);
7 β_2-coudre(l, u) ; β_2-coudre(l_2, b_4) ; β_2-coudre(u_4, b) ;
Le plongement du sommet incident à $u \leftarrow (x, y, z)$;
retourner u_3;



a. Le cas à traiter. b. Carte avant le traitement. c. Le résultat.

Fig. 5.57 – Le cas traité par la ligne 1 de l'algorithme 27 lorsque le voxel 6 n'est pas de la même région que ses trois voisins dans le précode l_8.

La première modification effectuée par cet algorithme (ligne 1) permet de traiter la configuration présentée figure 5.57. Ce cas se produit lorsque, comme nous l'avons représenté figure 5.57.*a*, le voxel numéro 6 n'est pas de la même région que les voxels 1, 2 et 5. La carte courante avant de traiter ce précode est présentée figure 5.57.*b*, où nous pouvons noter la présence d'une face entre les voxels 2 et 6, et une autre entre les voxels 5 et 6. Dans ce cas, $\beta_2(l_4) \neq u_4$ et nous devons donc β_2-découdre les deux brins l_4 et u_4 puis β_2-coudre les brins qui étaient β_2-cousus à eux (que nous avons conservé dans t_1 et t_2). Par contre, si $\beta_2(l_4) = u_4$, alors c'est que le voxel 6 est de la même région que les voxels 1, 2 et 5, et dans ce cas nous devons simplement β_2-découdre l_4. Ce traitement est similaire pour traiter les



a. Un précode p_1. b. p_2 =rotation(p_1). c. p_3 =rotation(p_2).

FIG. 5.58 – La permutation *rotation* qui transforme un précode.

TAB. 5.1 – La permutation *rotation*.

x	1	2	3	4	5	6	7	8
rotation(x)	1	5	2	6	3	7	4	8

deux cas où les voxels numéro 4 et 7 ne sont pas de la même région que leurs trois voisins (lignes 2 et 3 de l'algorithme).

Après avoir traité ces trois cas possibles, les lignes 4, 5 et 6 effectuent le remplacement respectivement du brin l_2 par b_3, du brin b par u_3 et du brin u par l_4. Enfin, la ligne 7 effectue les β_2-coutures qui n'ont pas encore été effectuées. Notons que tous les brins traités ici ont déjà été β_2-décousus. Les plongements divers vont être mis à jour de manière transparente lors des coutures et décousures ; le seul que nous devons affecter explicitement est celui du sommet incident au brin u. En effet, c'est le seul sommet topologique qui n'existait pas dans la carte courante avant de traiter ce précode.

Nous pouvons donner pour les précodes l_2 à l_7 chaque algorithme de traitement correspondant. Ces algorithmes vont ressembler aux deux que nous venons de présenter. Mais il n'est pas nécessaire de définir un algorithme par précode. Nous regroupons ces traitements pour les précodes *isomorphes par rotation*, ce qui réduit le nombre d'algorithmes à écrire. Afin de savoir si deux précodes sont isomorphes par rotation, nous définissons pour cela la permutation appelée *rotation* sur l'ensemble $1, \ldots, 8$. Cette fonction associe à un numéro de voxel son image par rotation, de la manière présentée figure 5.58. La figure 5.58.a représente un précode en numérotant chacun de ses huit voxels, ainsi qu'en symbolisant les relations de 6-voisinage afin de faciliter la visualisation. La figure 5.58.b montre l'image de ce précode par application de *rotation*, et la figure 5.58.c l'image de ce deuxième précode. Nous définissons explicitement cette permutation en donnant l'image de chaque élément au moyen du tableau 5.1. Nous pouvons remarquer que lorsque nous appliquons trois fois cette fonction sur n'importe quel précode, nous obtenons toujours le précode initial. Remarquons également que les deux voxels numéro 1 et 8 sont invariants par rotation. En effet, cette rotation peut être vue comme la rotation de 120 degrés autour de l'axe allant du voxel 1 au voxel 8, et il est donc normal que ces deux voxels soient invariants. Deux précodes sont *isomorphes par rotation* si à partir d'un précode nous pouvons obtenir le deuxième en appliquant *rotation* autant de fois que nécessaire.

L'ensemble des précodes isomorphes par rotation peuvent être traités avec un seul algorithme appelé avec des brins initiaux différents. Cela nous permet de diminuer le nombre de traitements à écrire. Par exemple pour les précodes lignels, les précodes l_2, l_3 et l_4 sont isomorphes par rotation. Pour s'en convaincre, il suffit d'écrire la partition correspondant au précode l_2 et d'appliquer ensuite *rotation*. La partition correspondant au précode l_2 est $\{\{1,2\},\{3\},\{5\}\}$. Le résultat de *rotation* sur ce précode est la partition $\{\{1,5\},\{2\},\{3\}\}$, obtenue en remplaçant chaque élément par son image. Cette partition est celle correspondant au précode l_3. Le résultat d'une nouvelle application de *rotation* est la partition $\{\{1,3\},\{5\},\{2\}\}$ correspondant au précode l_4. Nous pouvons donc définir un seul algorithme de traitement pour ces trois précodes, qui est présenté algorithme 28.

Algorithme 28 UneFusionVolumes3d

Entrée : d_1, f_1 et f_2 trois brins
(x, y, z) les coordonnées du nouveau sommet
Sortie : le « prochain » *last*.

$d_2 \leftarrow \beta_1(d_1)$; $d_3 \leftarrow \beta_1(d_2)$; $d_4 \leftarrow \beta_1(d_3)$;
β_1-découdre(d_1) ; β_1-découdre(d_2) ; β_1-découdre(d_3) ; β_1-découdre(d_4) ;

créer une face carrée autour de d_1 ; créer une face carrée autour de d_2 ;
créer une face carrée autour de d_3 ; créer une face carrée autour de d_4 ;
tmp $\leftarrow$ créer une face carrée topologique ;

β_2-coudre($\beta_1(d_1), \beta_0(d_2)$) ; β_2-coudre($\beta_{11}(d_1), \beta_0(tmp)$) ;
β_2-coudre($\beta_0(d_1), \beta_1(d_4)$) ; β_2-coudre($\beta_1(d_3), \beta_0(d_4)$) ;
β_2-coudre($\beta_1 1(d_3), \beta_1(tmp)$) ; β_2-coudre($\beta_0(d_3), \beta_1(d_2)$) ;
β_2-coudre($tmp, \beta_{11}(d_2)$) ; β_2-coudre($\beta_{11}(tmp), \beta_{11}(d_4)$) ;

β_3-coudre la face contenant f_1 et celle contenant d_1 ;
β_3-coudre la face contenant f_2 et celle contenant d_2 ;

Le plongement du sommet incident à $\beta_{11}(tmp) \leftarrow (x, y, z)$;
retourner *tmp*;

Cet algorithme prend en entrée trois brins d_1, f_1 et f_2 et modifie la carte courante en fusionnant le voxel courant avec celui incident à d_1 et en le β_3-cousant aux deux faces f_1 et f_2. Nous commençons par nommer les trois autres brins de la face incidente à d_1, puis β_1-décousons ces quatre brins. Nous créons ensuite quatre faces carrées autour de chacun de ces brins. Nous créons également une nouvelle face carrée supplémentaire. Le traitement suivant consiste simplement à β_2-coudre ensemble chacune de ces faces. À part les brins d_1 à d_4 qui étaient initialement β_2-cousus, et dont nous conservons les coutures, les autres brins viennent tous d'être créés et sont donc β_2-libres. Enfin, le dernier traitement de cet algorithme consiste à β_3-coudre les deux faces incidentes aux deux brins f_1 et f_3 avec leurs faces respectives en vis-à-vis. Nous devons affecter le plongement du sommet incident au brin $\beta_{11}(tmp)$ car il appartient au nouveau sommet topologique. Les autres plongements sont mis à jour de manière automatique, principalement lors des deux β_3-coutures. Le résultat de cet algorithme pour les trois traitements des précodes l_2, l_3 et l_4 est donné respectivement sur les figures 5.59, 5.60 et 5.61.

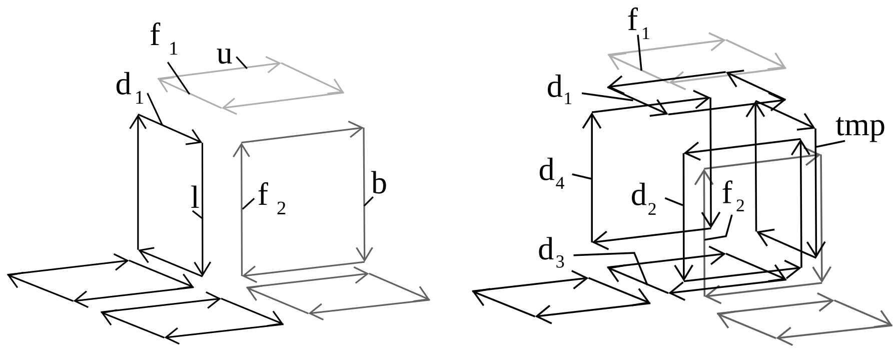
Algorithme 29 Code associé au précode l_2 en 3d

Entrée : *last*, *up* et *behind*

$c = (x, y, z)$ les coordonnées du nouveau sommet

Sortie : le « prochain » *last*.

$d_1 \leftarrow \beta_0(last)$; $f_1 \leftarrow \beta_0(up)$; $f_2 \leftarrow \beta_{11}(behind)$;
$tmp \leftarrow$ UneFusionVolumes3d(d_1, f_1, f_2, c);
retourner *tmp*;



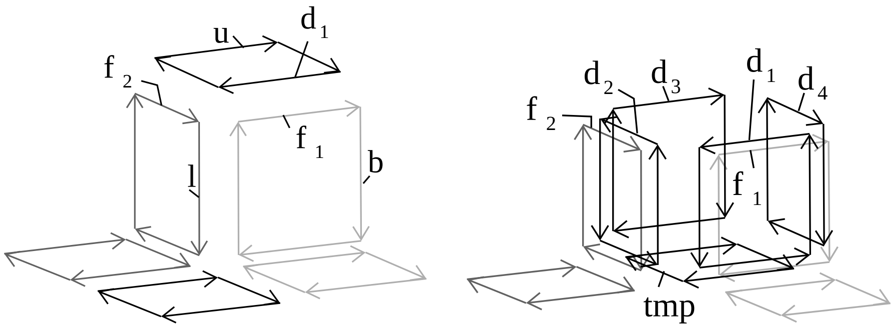
a. Carte avant le traitement. b. Le résultat.

FIG. 5.59 – La carte avant et après le traitement du précode l_2.

Maintenant que nous avons défini cet algorithme, nous devons l'appeler avec les brins initiaux correspondant au précode effectif. L'algorithme 29 montre comment cet algorithme est appelé pour le traitement du précode l_2. 29. Cet algorithme consiste simplement à appeler *UneFusionVolumes3d* et à retourner le prochain brin *last* pour le traitement du précode suivant. La figure 5.59.*a* montre la position des brins initiaux passés à l'algorithme *UneFusionVolumes3d*. La carte obtenue après cet algorithme est présentée figure 5.59.*b*.

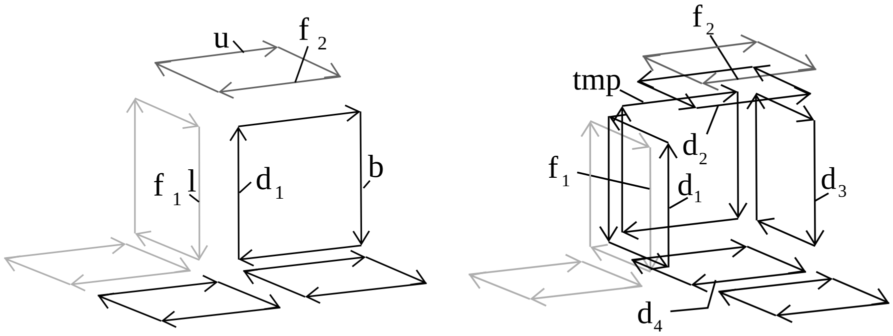
Le traitement du précode l_3 est identique, à l'exception de la position des brins initiaux et du brin retourné. Nous pouvons voir figure 5.60.*a* la position de ces brins avant le traitement de ce précode, et figure 5.60.*b* le résultat de ce traitement, avec la position du brin retourné par l'algorithme *UneFusionVolumes3d*. Nous avons $d_1 = \beta_{11}(up)$, $f_1 = \beta_0(behind)$ et $f_2 = \beta_0(last)$. Le brin retourné par cet algorithme est maintenant $\beta_{1120}(tmp)$. Le traitement du précode l_4 est également identique à celui du précode l_2, comme nous pouvons le vérifier figure 5.61. Pour ce traitement, nous avons $d_1 = \beta_{11}(behind)$, $f_1 = last$ et $f_2 = \beta_{11}(up)$. Le brin retourné est $\beta_{1211}(tmp)$.

Remarquons que pour chacun de ces algorithmes, nous avons un brin sur chaque face incidente à *last*, *up* et *behind*. En effet, ces brins désignent les faces supprimées par une fusion de volumes ou conservées et β_3-cousues avec une face du volume représentant le voxel courant. De plus, afin de fixer la position de ces brins, il suffit de la choisir pour le premier précode que nous traitons, ici l_2, et d'écrire l'algorithme *UneFusionVolumes3d* pour ce précode. L'écriture des traitements des autres précodes se fait alors simplement en retrouvant l'image de chaque brin utilisé par le premier traitement par *rotation*.



a. Carte avant le traitement. b. Le résultat.

FIG. 5.60 – La carte avant et après le traitement du précode l_3.



a. Carte avant le traitement. b. Le résultat.

FIG. 5.61 – La carte avant et après le traitement du précode l_4.

Les trois précodes restant, l_5, l_6 et l_7, sont isomorphes par rotation, et nous pouvons donc également définir un seul traitement pour les trois. Cet algorithme, que nous appelons *DeuxFusionsVolumes3d* car le voxel courant est fusionné avec deux de ses voisins, prend en paramètre deux brins d_1 et e_1 appartenant aux deux voxels à fusionner avec le voxel courant, et un brin f_1 appartenant au voxel qui ne va pas être fusionné. Remarquons que pour cet algorithme, il faut un traitement similaire à celui effectué en début de l'algorithme 27 traitant le précode l_8 afin de coudre les éventuelles faces se trouvant entre d_1 et e_1. Nous ne donnons pas ici cet algorithme, qui peut s'écrire simplement en regardant la carte courante avant le traitement et celle à obtenir.

Nous allons maintenant voir comment cet algorithme est appelé pour traiter les trois précodes l_5, l_6 et l_7. Cela permet de bien comprendre les traitements de précodes isomorphes par rotation. De plus, nous montrons pour chacun de ces précodes la carte courante avant et après le traitement, avec la position de chaque brin, ce qui permet ensuite pour les personnes désireuses d'implanter l'algorithme optimal d'extraction d'écrire facilement la fonction correspondant à *DeuxFusionsVolumes3d*.

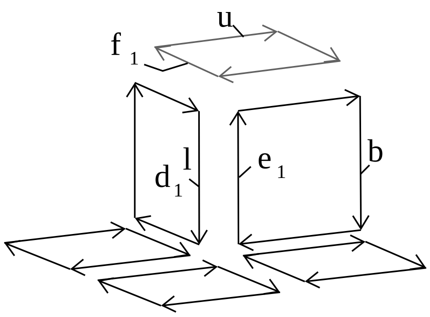
Algorithme 30 Code associé au précode l_5 en 3d

Entrée : *last*, *up* et *behind*

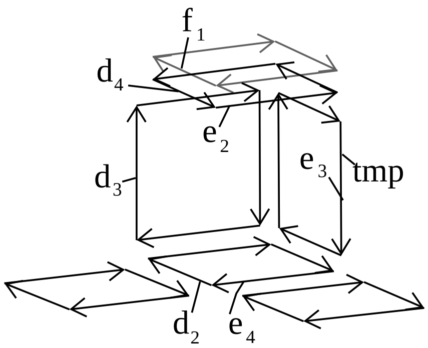
$c = (x, y, z)$ les coordonnées du nouveau sommet

Sortie : le « prochain » *last*.

$d_1 \leftarrow last$; $e_1 \leftarrow \beta_{11}(behind)$; $f_1 \leftarrow \beta_0(up)$;
$tmp \leftarrow$ DeuxFusionsVolumes3d(d_1, e_1, f_1, c);
retourner *tmp*;



a. Carte avant le traitement.

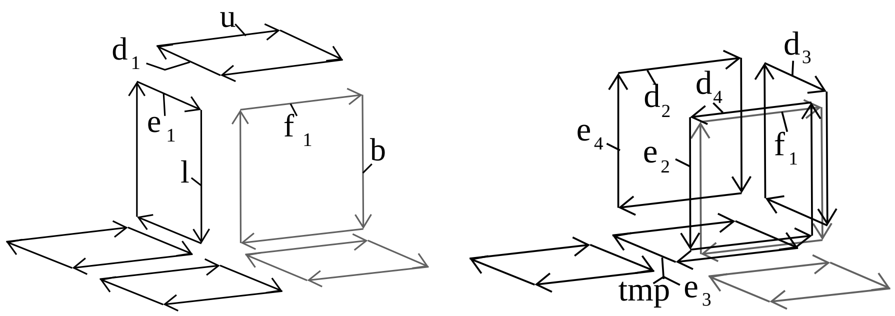


b. Le résultat.

FIG. 5.62 – La carte avant et après le traitement du précode l_5.

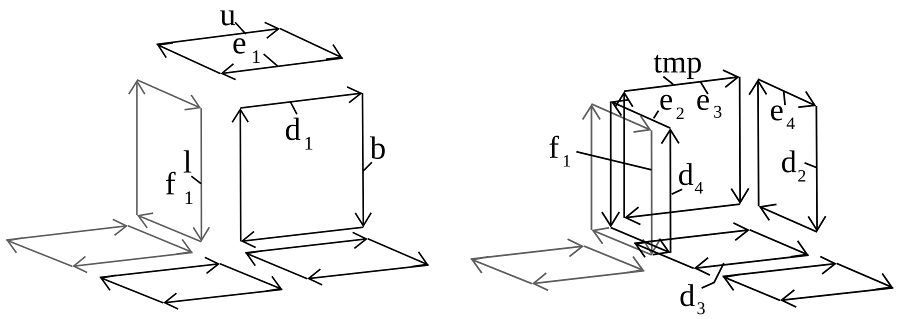
Nous présentons tout d'abord l'algorithme 30 traitant le précode l_5, et figure 5.62 la carte courante avant de traiter ce précode, et celle à obtenir après son traitement. Sur cette figure, nous pouvons observer la position des brins passés en paramètre à l'algorithme *DeuxFusionsVolumes3d*. d_1 et e_1 appartiennent aux deux faces qui doivent être fusionnées avec le voxel courant. Afin de faciliter la désignation d'un brin particulier, nous appelons d_2, d_3 et d_4 les brins appartenant à la même face que d_1, dans l'ordre donné par β_1, et de manière similaire les brins de la face incidente à e_1. Nous pouvons voir figure 5.62.*b* que nous avons conservé certains de ces brins afin d'éviter des destructions et des allocations de mémoire. De plus, nous conservons ces brins à leur place initiale, ils n'ont donc pas besoin d'être β_2-décousus puis β_2-cousus à un autre endroit. La face incidente à f est β_3-cousue à celle incidente à d_4, ce qui entraîne des mises à jour des plongements. Enfin, le seul sommet topologique qui n'existait pas dans la carte avant le traitement de ce précode est celui incident à $\beta_0(e_4)$ et nous lui affectons donc les coordonnées du nouveau sommet passé en paramètre à l'algorithme. Enfin le brin retourné est *tmp* qui est le brin retourné par l'algorithme *DeuxFusionsVolumes3d*.

Le traitement du précode l_6 est identique à celui du précode l_5, à l'exception de la position des brins initiaux et du brin retourné. Comme nous pouvons voir figure 5.63, nous avons $d_1 = \beta_0(up)$, $e_1 = \beta_0(last)$, $f_1 = \beta_0(behind)$ et le brin retrourné qui est $\beta_{1120}(tmp)$. De même pour le traitement du précode l_7, où comme nous pouvons vérifier figure 5.64 nous avons $d_1 = \beta_0(behind)$, $e_1 = \beta_{11}(up)$, $f_1 = last$ et le brin retourné qui est $\beta_{1211}(tmp)$.



a. Carte avant le traitement. b. Le résultat.

FIG. 5.63 – La carte avant et après le traitement du précode l_6.



a. Carte avant le traitement. b. Le résultat.

FIG. 5.64 – La carte avant et après le traitement du précode l_7.

Nous avons présenté le traitement de chaque précode lignel. L'algorithme optimal d'extraction va simplement tester quel est parmi ces huit précodes, le précode courant, et exécuter le traitement correspondant qui modifie localement la carte. Cet algorithme est linéaire, car chaque traitement est en *O(1)*, et chaque voxel de l'image est parcouru exactement une fois. De plus, nous avons vu pour ces divers traitements que nous créons exactement le nombre de brins nécessaires, sans effectuer de destruction ou d'allocation inutile.

Nous pouvons montrer que ces huit précodes lignels couvrent bien l'ensemble des précodes existant en dimension 3. Mais contrairement à la dimension 2, nous ne pouvons pas donner ici pour chaque précode la liste des précodes qu'il couvre, étant donné leur nombre beaucoup trop important. Nous allons dénombrer les configurations couvertes par chaque précode, et vérifier que c'est bien le nombre de configurations possibles. Une configuration (notion présentée section 4.5.1) s'obtient à partir d'un précode en affectant à chaque élément de la partition un identifiant de région, que nous appelons ici abusivement couleur. Il est plus facile de compter les configurations que les précodes, étant donné que deux configurations sont différentes si elles n'ont pas exactement les mêmes couleurs pour les mêmes voxels, ce qui n'est pas le cas des précodes qui tiennent compte de l'appartenance ou non à la même région des voxels voisins. Nous considérons que nous avons huit régions différentes. En effet, un précode 3d est composé de huit voxels, et peut donc dans le pire des cas avoir huit couleurs différentes. Nous pouvons sans problème effectuer les mêmes calculs en considérant que nous avons k régions différentes, mais cela les complique pour arriver exactement au même résultat.

Le nombre de configurations possibles est simplement 8^8. En effet, nous pouvons avoir n'importe quelle configuration dans l'image. La contrainte sur la segmentation en régions, imposant que chaque région soit 6-connexe, n'entraîne en effet aucune configuration interdite localement. Étant donné qu'une configuration est de taille $2 \times 2 \times 2$, nous avons donc huit voxels différents, et chacun peut avoir n'importe laquelle des huit couleurs. Il existe donc $16\,777\,216$ configurations différentes.

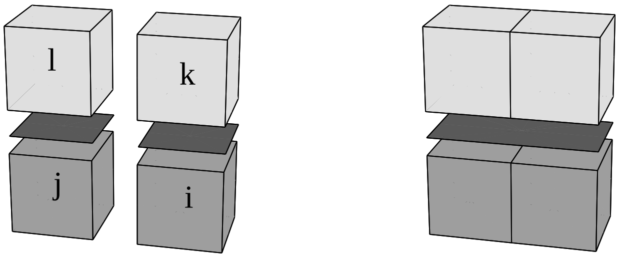
Nous comptons maintenant le nombre de configurations couvertes par chaque précode lignel. Le précode l_1 en couvre $8 \times 7 \times 7 \times 7 \times 8^4 = 11\,239\,424$. En effet, le voxel numéro 1 peut avoir n'importe quelle couleur parmi les 8, ses trois voisins ne sont pas de la même couleur donc peuvent avoir une des 7 couleurs restantes, et les 4 voxels libres peuvent avoir n'importe quelle couleur. Le précode l_2 couvre $8 \times 1 \times 7 \times 7 \times 8^4 = 1\,605\,632$ configurations, car le voxel 1 peut avoir n'importe laquelle des 8 couleurs, son voisin de gauche est forcément de la même couleur que lui, ses deux autres voisins ne sont pas de cette couleur, donc peuvent avoir une des 7 couleurs restantes et enfin les 4 voxels libres peuvent être de n'importe quelle couleur. Ce nombre de configurations couvertes est identique pour les précodes l_3 et l_4 car ces trois précodes sont isomorphes par rotation. Nous pouvons d'ailleurs facilement le vérifier en utilisant la même démarche que pour le précode l_2. Le précode l_5 couvre $8 \times 1 \times 1 \times 7 \times 8^4 = 229\,376$ configurations, car deux voisins du voxel 1 sont de la même couleur que lui. Les précodes l_6 et l_7 couvrent chacun ce même nombre de configurations. Enfin, le précode l_8 couvre $8 \times 1 \times 1 \times 1 \times 8^4 = 32\,768$ configurations.

Nous obtenons finalement que ces huit précodes lignels couvrent $11\,239\,424 + (3 \times 1\,605\,632) + (3 \times 229\,376) + 32\,768 = 16\,777\,216$, qui est bien le nombre de configurations possibles. Cela prouve que les 8 précodes lignels couvrent bien l'ensemble des précodes possibles.

5.6.4 Les précodes pour la carte de niveau 2

Pour ce niveau de simplification, nous fusionnons chaque couple de faces adjacentes, coplanaires, et incidentes à une arête de degré un ou deux (cf. définition 25). Ce type de fusion est possible uniquement si deux voxels adjacents i et j sont de même région et qu'il existe un autre voxel k adjacent à i étant d'une région différente, tel que le voxel adjacent à la fois à k et à j soit de la même région que k. Cette formulation compliquée est illustrée figure 5.65. Nous devons effectuer ce type de fusion uniquement lorsqu'un des quatre voxels i, j, k ou l est le voxel courant. En effet, le cas échéant nous avons déjà effectué ce traitement lors d'un précode précédent.

De manière un peu plus intuitive, nous pouvons effectuer une fusion de faces coplanaires s'il existe deux voxels adjacents de même région, et deux autres voxels adjacents d'une autre région « parallèle » aux deux premiers. Nous avons représenté figure 5.65.a ces quatre voxels, et schématisé les deux faces coplanaires qui vont être fusionnées. Si l'un des quatre voxels représenté sur cette figure appartient à une autre région, la fusion de faces est alors



a. Avant la fusion. b. Et après.

FIG. 5.65 – Une représentation partielle d'un précode où l'on peut fusionner deux faces coplanaires.

impossible. En effet, nous aurions alors une face supplémentaire entre ce voxel et un de ses voisins. De ce fait, l'arête au centre des quatre voxels ne serait plus de degré deux mais de degré trois.

Afin de trouver tous les précodes pour lesquels nous pouvons effectuer une fusion de faces coplanaires, il suffit d'étudier les différentes cas où une telle configuration peut se produire, ainsi que toutes les combinaisons possibles entre ces cas. Nous pouvons ainsi trouver de manière exhaustive chacun de ces précodes. Nous obtenons les 18 précodes présentés figures 5.66 et 5.67 (que nous appelons fc_i pour précode faces coplanaires numéro i).

Sur ces deux figures, nous avons représenté les précodes que nous devons traiter afin d'extraire la carte de niveau 2, en les classant afin de faciliter la lecture. Sur chaque ligne, nous avons représenté un premier précode, puis son image par l'application de la fonction *rotation* puis l'image de ce deuxième précode par cette même fonction. Chaque ligne représente donc trois précodes isomorphes par rotation. De ce fait, nous obtenons immédiatement un deuxième classement en colonne. La première colonne de la figure 5.66 montre les sous-cas du précode l_2, la deuxième les sous-cas du précode l_3 et la troisième ceux du précode l_4. De manière similaire, la première colonne de la figure 5.67 montre les précodes sous-cas du précode l_5, la deuxième ceux du précode l_6 et la troisième ceux du précode l_7.

Certains de ces précodes partiels sont composés de 5 voxels, et d'autres de 6. Par exemple, le précode fc_7 fixe une contrainte sur le voxel numéro 6 alors que le précode fc_1 non. Ce voxel peut donc, *a priori*, appartenir à n'importe quelle région. Mais en fait il ne peut pas appartenir à la même région que le voxel numéro 5 (son voisin droite). En effet, dans ce cas nous obtenons le précode fc_7. Ce voxel peut donc être de la même région que son voisin du bas, le voxel numéro 2, mais pas de la même région que son voisin de gauche. Il en est de même pour chaque précode étant composé seulement de 5 voxels. Remarquons que dans ces deux cas nous effectuons exactement le même traitement, pour le précode fc_1 une fusion de volumes et une fusion des deux faces coplanaires situées entre les voxels 1 et 3 pour la première, et entre les voxels 2 et 4 pour la deuxième.

En utilisant le même principe que pour les précodes lignels, nous pouvons définir un seul traitement pour chaque ensemble de précodes isomorphes par rotation. Nous devons

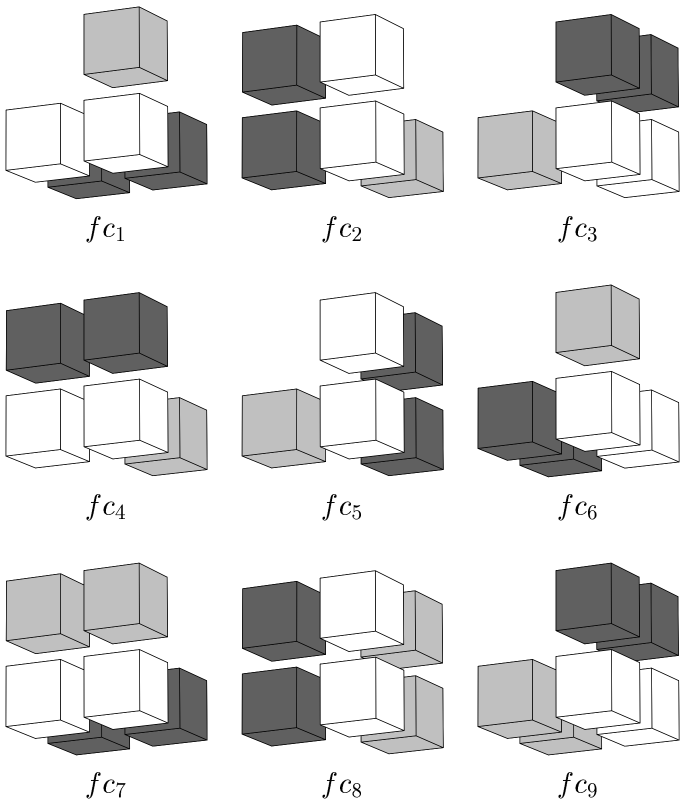


FIG. 5.66 – Les 9 premiers des 18 précodes à traiter afin de calculer la carte de niveau 2 en 3d, sous-cas des précodes l_2, l_3 et l_4.

donc définir six traitements différents, qui sont appelés sur des brins initiaux différents pour des précodes isomorphes par rotation. Ces traitements sont similaires et peuvent se trouver simplement en étudiant la carte de départ et celle que nous voulons obtenir. Nous donnons donc uniquement le traitement générique des trois premiers précodes, fc_1, fc_2 et fc_3, présenté algorithme 31. Nous pouvons vérifier pour ces trois précodes, que cet algorithme effectue bien la modification de la carte correspondant au précode traité, en étudiant son déroulement sur les figures 5.68, 5.69 et 5.70.

Cet algorithme prend trois brins en paramètres : b_1 qui appartient à la face du voxel qui va être fusionné avec le voxel courant, f_1 qui appartient à la face du voxel qui ne va pas être fusionné, mais pour lequel nous effectuons la fusion des deux faces incidentes à ce brin, et e_1 pour lequel nous n'effectuons aucune fusion. Il est assez proche de l'algorithme traitant les précodes l_2, l_3 et l_4. La principale différence concerne les traitements numérotés 1, 2 et 3, qui effectuent la fusion des deux faces incidentes au brin f_1. Cette fusion est effectuée de manière locale, autour du brin f_1, puis de manière un peu différente pour la face en vis à vis étant donné qu'il y a une seule demi-face présente. Au vu des modifications locales effectuées, nous avons ensuite seulement besoin de β_3-coudre les trois brins qui étaient β_3-libres, les autres brins conservant leur coutures. La suite de cet algorithme est très proche de l'algorithme pour les précodes lignels. La seule différence se situe au niveau de

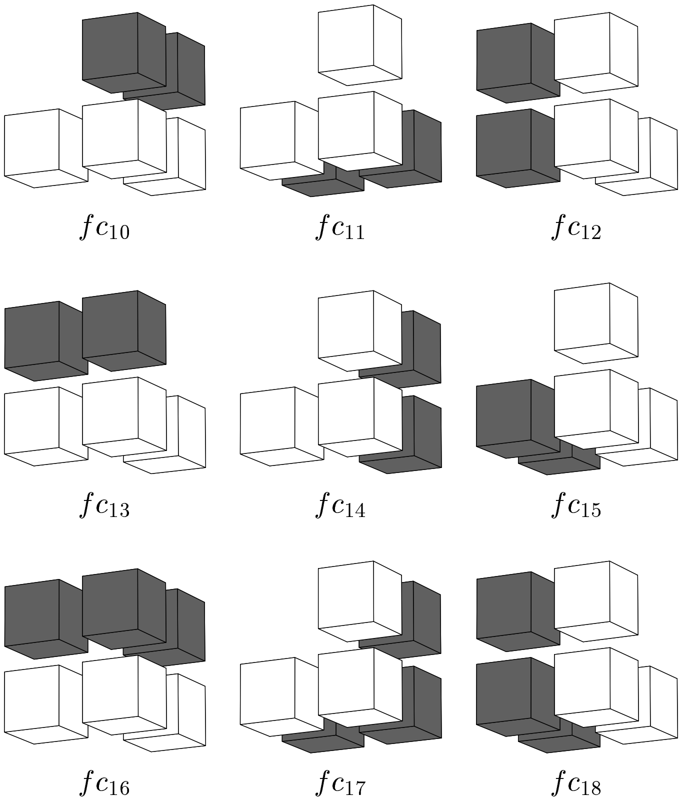


Fig. 5.67 – Les 9 derniers des 18 précodes à traiter afin de calculer la carte de niveau 2 en 3d, sous-cas des précodes l_5, l_6 et l_7.

la création de la première face carrée, effectuée autour des brins d_1 et d_2. Cela revient à dire que nous créons deux nouveaux brins, et β_1-cousons les quatre brins pour former une face carrée. Nous récupérons ici le brin d_2, qui n'a plus d'utilité, ce qui évite une allocation mémoire suivie d'une destruction. La plupart des modifications de plongement sont faites lors de opérations de couture et décousures. Nous affectons seulement les coordonnées du nouveau sommet au sommet topologique créé.

Nous pouvons voir sur les figures 5.68, 5.69 et 5.70, la carte courante avant de traiter les précodes, fc_1, fc_2 et fc_3, ainsi que la position des différents brins passés en paramètres à l'algorithme, puis le résultat de ce traitement. Nous ne détaillons pas ici les trois algorithmes de ces précodes, qui consistent uniquement, comme pour les précodes lignels, à initialiser les brins de départ, à appeler l'algorithme général, puis à retourner le prochain brin *last* pour le précode suivant. Les autres algorithmes généraux de traitement sont très proches de celui que nous venons de présenter. Suivant les précodes traités, il faut effectuer une ou deux fusions de volumes, et une ou deux fusions de faces coplanaires. Nous ne détaillons pas plus ces traitements parce que cela s'avèrerait long et fastidieux, alors qu'ils ne sont pas trop difficiles à retrouver en dessinant la carte courante et celle à obtenir.

Algorithme 31 UneFusionVolumesUneFusionFacesCoplanaires3d

Entrée : d_1, f_1 et e_1 trois brins
(x, y, z) les coordonnées du nouveau sommet
Sortie : le « prochain » *last*.

$d_2 \leftarrow \beta_1(d_1)$; $d_3 \leftarrow \beta_1(d_2)$; $d_4 \leftarrow \beta_1(d_3)$;
β_1-découdre(d_1) ; β_1-découdre(d_2) ; β_1-découdre(d_3) ; β_1-découdre(d_4) ;

1 $t_1 \leftarrow \beta_{20}(f_1)$; $t_2 \leftarrow \beta_1(f_1)$;
β_1-découdre(t_1) ; β_0-découdre(t_2) ; β_1-coudre(t_1, t_2) ;

2 $t_1 \leftarrow \beta_0(f_1)$; $t_2 \leftarrow \beta_{21}(f_1)$;
β_1-découdre(t_1) ; β_0-découdre(t_2) ; β_1-coudre(t_1, t_2) ;

3 $a_1 \leftarrow \beta_2(d_2)$; $a_2 \leftarrow \beta_2(f_1)$; $a_3 \leftarrow \beta_1(a_1)$;
β_1-découdre(a_1) ; β_2-découdre(a_2) ;
β_1-coudre(a_1, a_2) ; β_1-coudre(a_2, f_1) ; β_1-coudre(f_1, a_3) ;

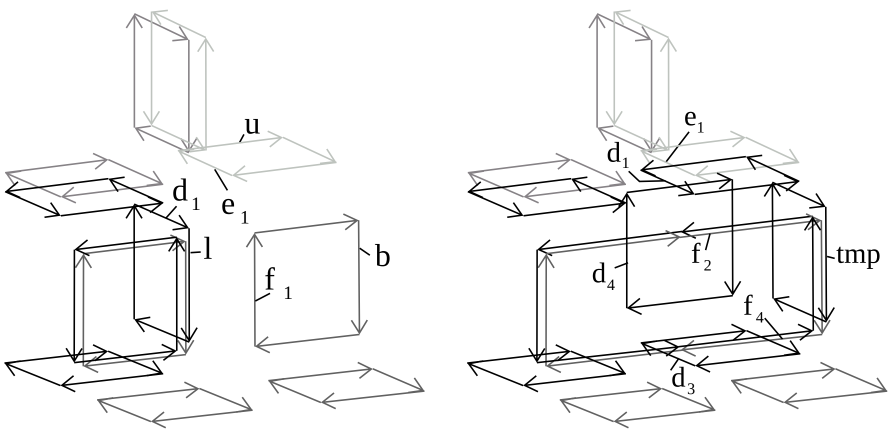
β_3-coudre(a_1, t_1) ; β_3-coudre($a_2, \beta_0(t_1)$) ; β_3-coudre($f_1, \beta_{00}(t_1)$) ;

créer une face carrée autour de d_1 et de d_2 ; créer une face carrée autour de d_3 ;
créer une face carrée autour de d_4 ; *tmp* $\leftarrow$ créer une face carrée topologique ;

β_2-coudre($\beta_1(d_1), f_1$) ; β_2-coudre($\beta_{11}(d_1), \beta_0(tmp)$) ;
β_2-coudre($\beta_0(d_1), \beta_1(d_4)$) ; β_2-coudre($\beta_1(d_3), \beta_0(d_4)$) ;
β_2-coudre($\beta_{11}(d_3), \beta_1(tmp)$) ; β_2-coudre($\beta_0(d_3), a_1$) ;
β_2-coudre(tmp, a_2) ; β_2-coudre($\beta_{11}(tmp), \beta_{11}(d_4)$) ;

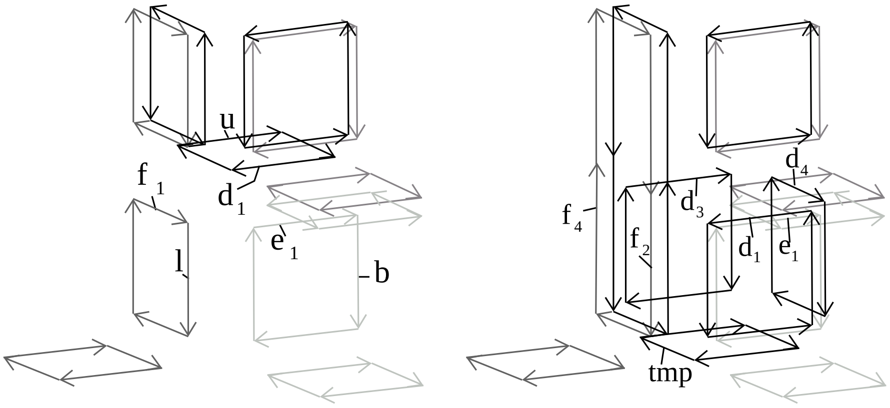
β_3-coudre la face contenant e_1 et celle contenant d_1 ;

Le plongement du sommet incident à $\beta_{11}(tmp) \leftarrow (x, y, z)$;
retourner *tmp*;



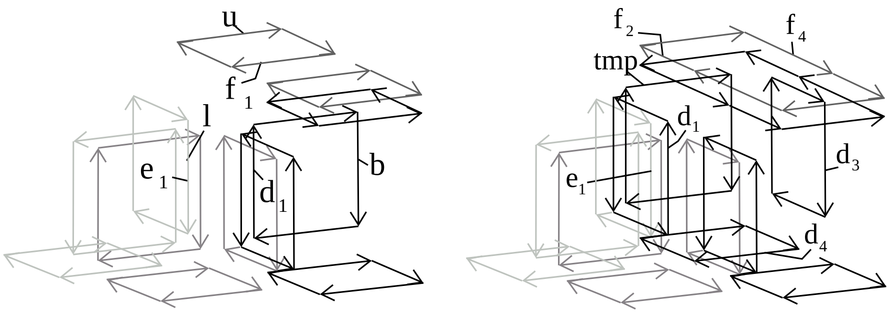
a. Carte avant le traitement. b. Le résultat.

FIG. 5.68 – La carte avant et après le traitement du précode fc_1.



a. Carte avant le traitement. b. Le résultat.

FIG. 5.69 – La carte avant et après le traitement du précode fc_2.



a. Carte avant le traitement. b. Le résultat.

FIG. 5.70 – La carte avant et après le traitement du précode fc_3.

Nous avons vu lors de la définition de la carte de niveau 2, (définition 29) les problèmes de déconnexion de faces qui pouvaient se produire, et la solution consistant à conserver des arêtes fictives. Lorsque nous avons défini l'algorithme d'extraction naïf, il nous a suffit de ne pas faire les fusions de faces qui entraînaient des déconnexions. Maintenant que nous balayons l'image et construisons directement la carte de niveau quelconque avec les précodes, ce n'est plus exactement pareil. Nous pouvons remarquer qu'une déconnexion de face peut se produire uniquement pour les précodes fc_{16}, fc_{17} et fc_{18}. En effet, c'est seulement dans ces cas que nous pouvons déconnecter le bord extérieur de la face de son bord intérieur, étant donné que nous effectuons deux fusions de faces coplanaires.

Ces trois précodes étant isomorphes par rotation, ils sont donc traités par le même algorithme qui effectue deux fusions de volumes et deux fusions de faces coplanaires. Cet algorithme va commencer par tester si la fusion des deux faces entraîne une déconnexion. Pour cela, il parcourt l'orbite $< \beta_1 >$ d'un brin d'une des deux faces. Si durant ce parcours un brin de la deuxième face est rencontré, c'est que la fusion entraîne une déconnexion, à condition que le deux brins ne soient pas cousus entre eux par β_0 et β_1. Dans ce cas, une seule des deux fusions est effectuée afin de conserver une arête fictive. Pour cela, il suffit d'exécuter le traitement de l'algorithme effectuant deux fusions de volumes et une seule fusion de faces, traitant les précodes fc_{10}, fc_{11} et fc_{12} ou celui traitant les précodes fc_{13}, fc_{14} et fc_{15}. En effet, le choix de la fusion de faces à effectuer n'est pas important, étant donné que la position de l'arête fictive conservée peut être quelconque.

Avec ce simple test et éventuellement un traitement différent, nous conservons les arêtes fictives permettant de régler les problèmes de déconnexion de faces. Cela se fait quand même au détriment de la complexité, puisque le traitement des précodes fc_{16}, fc_{17} et fc_{18} n'est plus effectué en *O(1)* étant donné que nous parcourons l'ensemble des brins d'une des deux faces. Nous conservons pour ce niveau de simplification éventuellement plusieurs arêtes fictives, étant donné que chaque arête représente un lignel de l'image.

Afin de nous assurer que ces 18 précodes couvrent bien toutes les possibilités de fusion de faces coplanaires, nous utilisons la même démarche que pour le niveau précédent. Nous dénombrons le nombre de cas existants où il faut effectuer des fusions de faces coplanaires, et comparons ensuite ce nombre aux cas couverts pas les 18 précodes.

Afin d'effectuer une fusion de faces coplanaires, le voxel courant et un de ses voisins doivent être de même région, et deux voxels parallèles doivent appartenir à une autre région. Remarquons que lorsque nous choisissons un des trois voisins du voxel courant, il y a seulement deux possibilités pour choisir les voxels parallèles, étant donné qu'il « reste » seulement deux directions possibles. Le nombre de cas possibles pour une de ces deux directions, pour un des trois voisins du voxel courant est de $8 \times 1 \times 7 \times 1 \times 8^4 = 229\,376$ cas. En effet, il y a 8 couleurs pour le voxel courant, le voisin choisi est de la même couleur, le voxel dans la direction choisi est d'une autre couleur (il y a donc 7 possibilités), et son voisin est de la même couleur. Enfin, les 4 voxels restant peuvent être de n'importe quelle couleur. Nous comptons maintenant le nombre de cas si les deux pixels choisis sont dans la deuxième direction possible. Ils sont au nombre de $8 \times 1 \times 7 \times 1 \times 8 \times 7 \times 8^2 = 200\,704$ cas. En effet, c'est identique pour les deux couples de voxels choisis, mais par contre le deuxième couple dans la première direction choisie ne doit pas être de la même région car ce cas a déjà été compté dans la première formule. Enfin les 2 voxels restants peuvent être de n'importe quelle couleur.

Pour un voisin du voxel courant, nous avons donc $229\,376 + 200\,704 = 430\,080$ cas possible, qu'il faut multiplier par trois étant donné que le voxel courant à trois voisins. Nous obtenons donc au total $1\,290\,240$ cas possibles pour lesquels nous devons effectuer une ou plusieurs fusions de faces coplanaires.

Nous dénombrons maintenant le nombre de cas couverts par chaque précode fc_1 à fc_{18}. Nous savons que le nombre de cas couverts par deux précodes isomorphes par rotation est identique, ce qui nous laisse seulement 6 cas à étudier.

Pour le précode fc_1, nous avons $8 \times 7 \times 7 \times 7 \times 8^2 = 175\,616$ cas. Le voxel courant est de la même couleur que son voisin gauche, pas de la même que les deux voxels derrière, ni que celle de son voisin du dessus. Ce dernier voxel n'est pas de la même couleur que son voisin gauche, mais peut-être de la même couleur que le voxel courant. Les deux voxels restants peuvent être de n'importe quelle couleur. Pour le précode fc_4 nous obtenons le même résultat par application d'un raisonnement similaire. Le précode fc_7 couvre $8 \times 7 \times 7$

$\times\ 8^2 = 25\,088$ cas, étant donné que deux couples de voxels sont de même région. Pour les précodes f_{10} et f_{13}, nous obtenons le même nombre de cas couverts : $8 \times 7 \times 7\ \times\ 8^2 = 25\,088$ cas. Enfin, le précode f_{16} couvre $8 \times 7\ \times\ 8^2 = 3\,584$ cas différents.

Nous obtenons finalement 430 080 cas différents qu'il faut multiplier par trois car nous avons compté uniquement les cas couverts par un précode de chaque classe d'équivalence. Nous obtenons bien finalement 1 290 240, le même nombre que le nombre de cas possibles pour la fusion de faces coplanaires. Comme pour les précodes lignels, cela prouve que les 18 précodes couvrent bien tous les cas pour lesquels nous devons effectuer une ou plusieurs fusions de faces coplanaires.

5.6.5 Les précodes pour la carte des frontières

Pour extraire ce niveau de simplification, nous devons maintenant fusionner chaque couple d'arêtes adjacentes, alignées et incidentes à un sommet de degré deux. Cette définition n'est plus valable lorsque nous sommes en présence d'arêtes fictives. Mais nous commençons par mettre de côté la gestion de ces arêtes, nous y reviendrons après avoir défini les précodes concernés par cette fusion.

Afin d'effectuer ce type de fusion, il est nécessaire que chaque couple de voxels dans une même direction soit composé de deux voxels de même région. En effet, afin que le sommet au centre d'un précode soit de degré deux, il faut qu'il y ait seulement deux arêtes qui lui soient incidentes. Pour trouver tous les précodes à traiter, il suffit d'étudier tous les cas où une telle configuration peut se produire. Ces cas étant encore peu nombreux, nous pouvons les définir de manière exhaustive et obtenir les 27 précodes présentés figures 5.71, 5.72 et 5.73.

Nous avons découpé ces précodes en trois parties, afin de pouvoir représenter sur chaque ligne les trois précodes isomorphes par rotation. Pour la figure 5.71, nous avons par exemple $rotation(f_1) = f_2$, $rotation(f_2) = f_3$ et $rotation(f_3) = f_1$. De plus, cette figure montre sur sa première colonne, les sous-cas du précode l_2, sur sa deuxième colonne ceux du précode l_3 et sur sa dernière colonne ceux du précode l_4. La figure 5.72 montre respectivement sur ses trois colonnes les sous-cas des précodes l_5, l_6 et l_7. Enfin la figure 5.73 montre les trois sous-cas du précode l_8. Remarquons que les deux précodes présentés figure 5.74 (et leurs rotations) ne sont pas des précodes à traiter pour l'obtention de la carte des frontières. En effet, après les fusions de faces coplanaires, il n'y a plus d'arête au centre du précode, et donc pas de fusion d'arêtes alignées à effectuer.

Comme pour les précodes du niveau précédent, nous donnons seulement l'algorithme 32 correspondant au traitement des précodes f_1, f_2 et f_3. Nous pouvons vérifier sur les trois figures 5.75, 5.78 et 5.79 le résultat de ce traitement pour ces trois précodes, ainsi que la position des brins initiaux passés à cet algorithme.

Nous avons découpé cet algorithme en 5 parties que nous expliquons maintenant pour le traitement du précode f_1. Nous notons d_2, d_3 et d_4 les trois brins de la face incidente à d_1 suivant l'ordre donné par β_1, et de manière similaire les brins des faces incidentes à e_1 et à f_1. La carte 5.75.a montre la configuration courante avant de traiter ce précode, ainsi que la position des trois brins passés en paramètre à l'algorithme 32. Afin d'expliquer ces traitements, nous déroulons chaque partie de cet algorithme. Pour simplifier la visualisation de la carte, nous représentons uniquement la partie concernée par le traitement en cours. Le

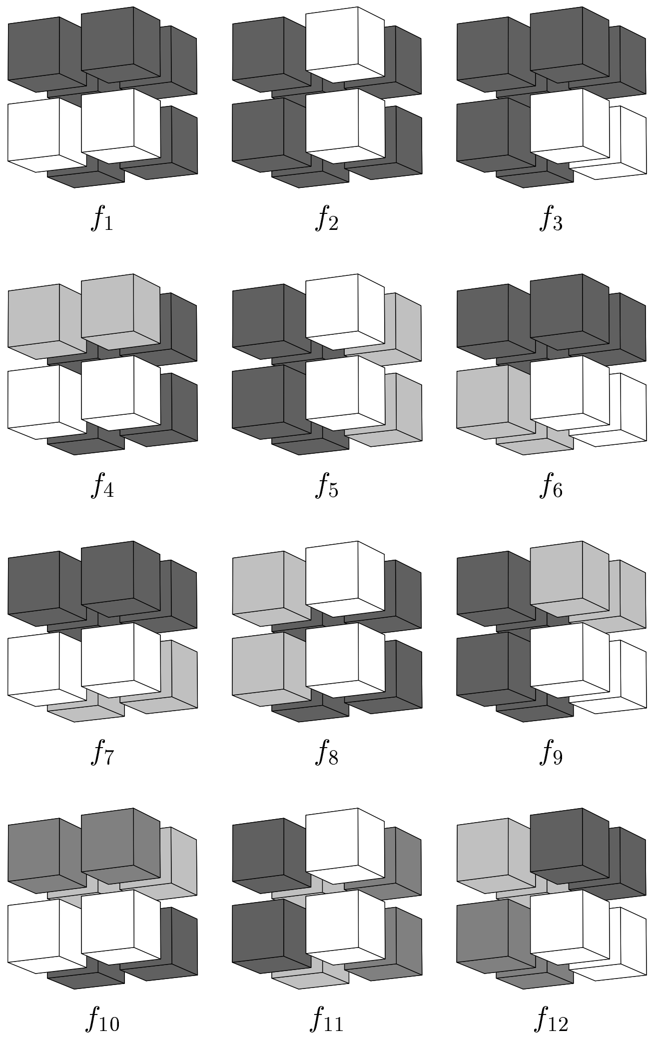


FIG. 5.71 – Les 12 premiers des 27 précodes à traiter afin de calculer la carte des frontières en 3d, sous-cas des précodes l_2, l_3 et l_4.

premier traitement modifie les deux faces incidentes à f_1 et $\beta_2(f_1)$ présentées figure 5.76.a. La première ligne effectue la fusion de ces deux faces du côté où les arêtes ne doivent pas être fusionnées. Nous obtenons la carte de la figure 5.76.b. La deuxième ligne effectue la fusion de l'autre côté. Afin d'effectuer en même temps la fusion d'arêtes, nous β_1-cousons t_1 à f_3. Nous enlevons donc le brin f_2 de la deuxième arête et obtenons la carte présentée figure 5.76.c. Les brins $\beta_2(f_1)$, f_1 et f_2 sont maintenant inutiles. Ils sont décousus afin d'être utilisés plus tard, au lieu d'être détruits et de créer ensuite de nouveaux brins.

La deuxième partie de l'algorithme 32 effectue un traitement similaire pour les deux faces incidentes à e_1 et $\beta_2(e_1)$. La troisième partie traite la face incidente à $\beta_2(d_1)$. Elle insère un nouveau brin n_1 sur cette face et β_3-coud ce brin ainsi que $\beta_2(d_1)$ respectivement à f_2 et à f_3. Les autres brins de cette face conservent leurs β_3-coutures. La quatrième partie

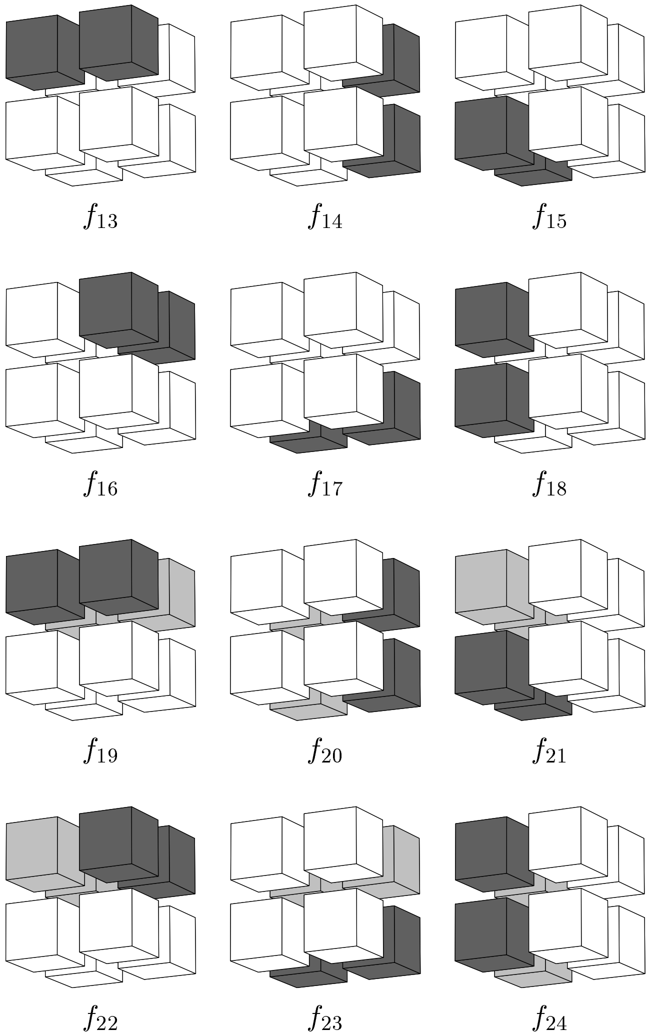


FIG. 5.72 – Les 12 précodes suivants des 27 précodes à traiter afin de calculer la carte des frontières en 3d, sous-cas des précodes l_5, l_6 et l_7.

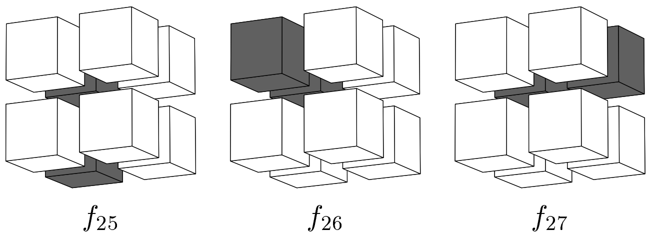


FIG. 5.73 – Les 3 derniers précodes des 27 précodes à traiter afin de calculer la carte des frontières en 3d, sous-cas du précode l_8.

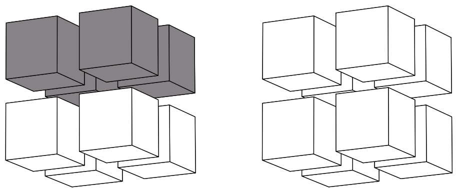


FIG. 5.74 – Deux précodes à ne pas traiter pour la construction de la carte des frontières.

Algorithme 32 Traitement_f_1_f_2_f_3_3d

Entrée : d_1, f_1 et e_1 trois brins
(x, y, z) les coordonnées du nouveau sommet
Sortie : le « prochain » *last*.

1 $t_1 \leftarrow \beta_{20}(f_1)$; β_1-découdre(t_1) ; β_0-découdre(f_2) ; β_1-coudre(t_1, f_2) ;
$t_1 \leftarrow \beta_{21}(f_1)$; β_0-découdre(t_1) ; β_1-découdre(f_3) ; β_1-coudre(f_3, t_1) ;
$m_1 \leftarrow \beta_2(f_1)$; β_2-découdre(f_1) ; β_0-découdre(f_1) ; β_3-découdre(m_1) ;

2 $t_1 \leftarrow \beta_{21}(e_1)$; β_0-découdre(t_1) ; β_1-découdre(e_4) ; β_1-coudre(e_4, t_1) ;
$t_1 \leftarrow \beta_{20}(e_1)$; β_1-découdre(t_1) ; β_0-découdre(e_3) ; β_0-coudre(e_3, t_1) ;
$n_1 \leftarrow \beta_2(e_1)$; β_2-découdre(n_1) ; β_1-découdre(e_1) ;
β_2-découdre(e_2) ; β_3-découdre(n_1) ;

3 $t_1 \leftarrow \beta_{21}(d_1)$; β_0-découdre(t_1) ; β_0-coudre(t_1, n_1) ; β_1-coudre($\beta_2(d_1), n_1$) ;
β_3-coudre(f_2, n_1) ; β_3-coudre($f_3, \beta_2(d_1)$) ;

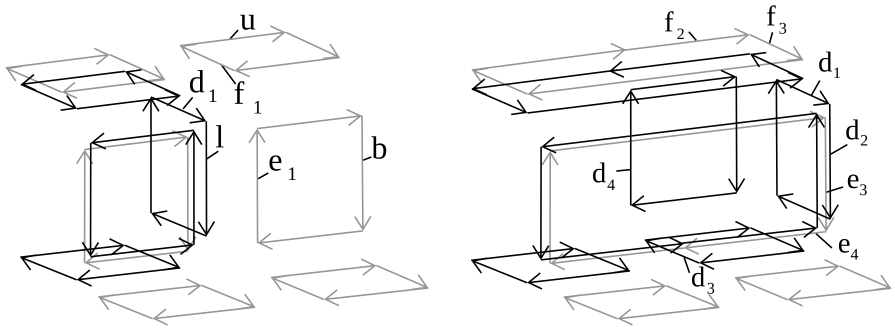
4 $t_1 \leftarrow \beta_{20}(d_2)$; β_1-découdre(t_1) ; β_1-coudre(t_1, m_1) ; β_1-coudre($m_1, \beta_2(d_2)$) ;
β_3-coudre(e_4, m_1) ; β_3-coudre($e_3, \beta_2(d_2)$) ;

5 β_1-découdre(d_2) ; β_1-découdre(d_3) ; β_1-découdre(d_4) ;
créer une face carrée autour de d_1 et de d_2 ;
créer une face carrée autour de d_3, f_1 et f_4 ;
créer une face carrée autour de d_4, e_1 et e_2 ;
β_2-coudre($m_1, \beta_0(d_3)$) ; β_2-coudre($f_4, \beta_1(d_2)$) ; β_2-coudre($f_1, \beta_0(d_4)$) ;
β_2-coudre($e_2, \beta_0(d_1)$) ; β_2-coudre(n_1, e_1) ;
Le plongement du sommet incident à $f_4 \leftarrow (x, y, z)$;
retourner d_2;

de l'algorithme effectue un traitement similaire pour la face incidente à $\beta_2(d_2)$. Enfin, la dernière partie de l'algorithme crée les trois faces restantes, en utilisant pour cela les brins libérés par les opérations précédentes. Il ne reste plus qu'à β_2-coudre correctement ces brins, et à affecter les coordonnées du nouveau sommet au plongement du sommet topologique créé lors de cet algorithme.

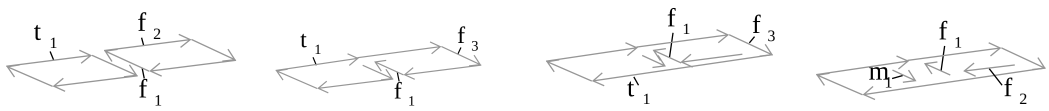
Nous pouvons vérifier que cet algorithme, appliqué sur la carte de la figure 5.75.a, produit bien celle de la figure 5.75.b. De même pour le précode f_2 où nous montrons figure 5.78.a la carte avant le traitement et figure 5.78.b le résultat obtenu, et pour le précode f_3 de manière identique sur la figure 5.79.

Nous ne donnons pas les algorithmes correspondant à ces trois précodes, qui consistent simplement à appeler l'algorithme général sur les brins de départ particuliers, présentés



a. Carte avant le traitement. b. Le résultat.

FIG. 5.75 – La carte avant et après le traitement du précode f_1.

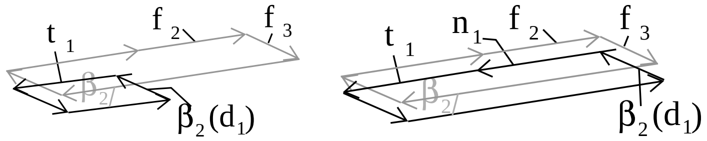


a. b. c. d.

FIG. 5.76 – Le premier traitement de l'algorithme 32 appliqué sur le précode f_1.

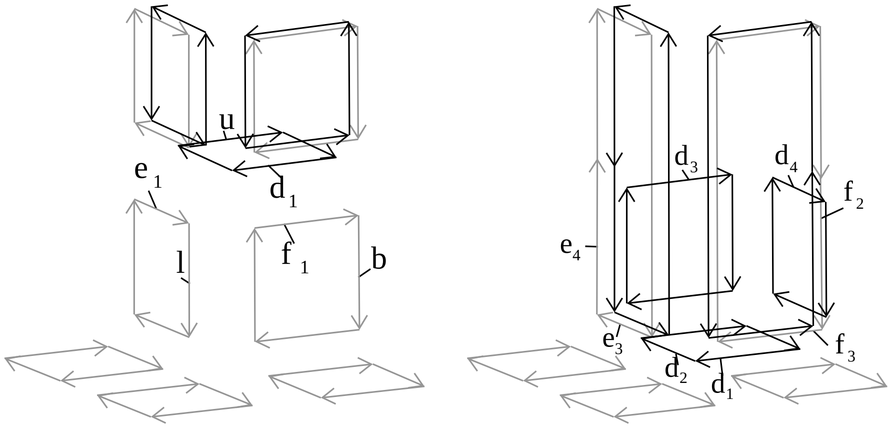
sur chaque figure, et à retourner ensuite le prochain brin *last* pour le traitement du précode suivant. Nous ne détaillons pas les traitements des autres précodes, assez proches de ceux déjà présentés. De plus, certains traitements peuvent être regroupés afin de ne pas développer l'ensemble des 9 algorithmes.

En effet, de nombreux précodes effectuent les mêmes fusions de volumes et de faces coplanaires, comme par exemple les précodes f_1, f_4, f_7 et f_{10}. Pour ces quatre précodes, la seule différence porte sur le nombre de brins des deux arêtes alignées à fusionner, où sur la position de ces brins. Nous pouvons définir un seul traitement pour ces précodes, qui effectue la fusion de volumes et des faces coplanaires de manière optimisée, puis la fusion des deux arêtes alignées au moyen de l'algorithme 20. Cette solution permet d'écrire un seul traitement pour les précodes $f_1 \dots f_9$, un autre pour les précodes f_{13}, f_{14}, f_{15}, f_{19}, f_{20} et f_{21}, un autre pour f_{16}, f_{17}, f_{18}, f_{22}, f_{23} et f_{24}, et enfin un dernier pour les trois



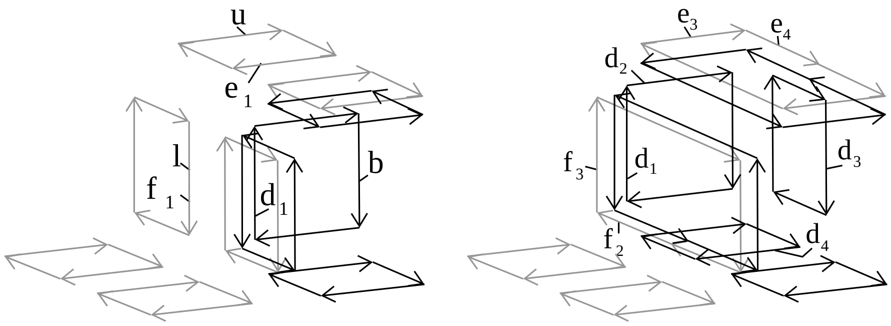
a. b.

FIG. 5.77 – Le troisième traitement de l'algorithme 32 appliqué sur le précode f_1.



a. Carte avant le traitement. b. Le résultat.

FIG. 5.78 – La carte avant et après le traitement du précode f_2.



a. Carte avant le traitement. b. Le résultat.

FIG. 5.79 – La carte avant et après le traitement du précode f_3.

précodes restants. Seulement 4 algorithmes différents sont à écrire au lieu de 9 si nous écrivons chaque méthode de manière optimale. Bien sûr, cette solution entraîne une légère perte en complexité, étant donné que nous ne connaissons pas exactement la configuration locale, et effectuons donc des tests, ou des boucles, ce qui n'est pas le cas sinon. Mais ces deux solutions permettent de choisir entre temps de développement plus important et complexité.

Afin d'extraire directement la carte des frontières d'une image au moyen de l'algorithme optimal, il suffit de tester quel est le précode courant parmi les 53 possibles (8 pour le niveau 1, 18 pour le niveau 2 et 27 pour le niveau 3) et simplement exécuter le traitement correspondant à ce précode. Mais nous n'avons pas encore étudié le problème des arêtes fictives. Nous avons vu section 5.3.3 quels problèmes ces arêtes pouvaient engendrer lors de la fusion d'arêtes et comment nous devions les résoudre. Nous devons maintenant voir comment cette gestion intervient dans l'algorithme optimal basé sur les précodes.

Nous avons vu pour le niveau précédent, que la conservation de ces arêtes fictives ne posaient aucun problème. Pour les 27 précodes de fusion d'arêtes, nous utilisons l'algorithme 17 décalant toutes les arêtes fictives incidentes à un sommet. En effet, chacun de ces précodes fusionne les deux arêtes incidentes au sommet central du précode. L'appel de cet algorithme avant chacun de ces précodes décale les éventuelles arêtes fictives sur un autre sommet, et le traitement du précode pourra ensuite se dérouler de manière normale. Mais un autre problème n'est pas résolu par cette solution : lorsque deux bords d'une face sont reliés par un chemin composé de plusieurs arêtes fictives. Nous voulons dans ce cas obtenir une seule arête fictive dans la carte des frontières. Ce problème ne peut pas être résolu uniquement pour les précodes frontières, étant donné que les arêtes fictives sont conservées par les précodes faces coplanaires. De ce fait, nous devons modifier le traitement des trois précodes fc_{16}, fc_{17} et fc_{18}.

Nous utilisons le pré-traitement présenté algorithme 33 pour le précode fc_{16}, afin de savoir quelle opération nous devons effectuer.

Algorithme 33 Pré-traitement du précode fc_{16} en 3d

Entrée : *last*, *up* et *behind*
$c = (x, y, z)$ les coordonnées du nouveau sommet
Sortie : le « prochain » *last*.

1 **si** $\beta_{020}(last)$ *est un brin d'une arête fictive* **alors**
  Décaler_arête($\beta_{0202}(last), \beta_{02}(behind)$);
  Exécuter le traitement du précode fc_{16};
sinon
2 **si** $\beta_{020}(last) \neq \beta_{02}(behind)$ ***et*** $\beta_{02}(last) \in <\beta_1> (\beta_{02}(behind))$ **alors**
    // Les deux fusions de faces entraînerait déconnexion
    Marquer l'arête incidente à $\beta_{02}(behind)$ comme étant une arête fictive ;
    Exécuter le traitement du précode fc_{13};
  sinon
3   *// Le cas normal, il n'y a pas d'arête fictive ni de déconnexion*
    Exécuter le traitement du précode fc_{16};

Nous distinguons trois cas différents. Le premier lorsque le brin $\beta_{020}(last)$ appartient à une arête fictive. Ce cas est présenté figure 5.80.a. C'est seulement dans ce cas (et les deux cas isomorphes par rotation) que nous créons des « chemins » d'arêtes fictives. Pour éviter ces chemins, nous décalons l'arête fictive entre le brin $\beta_{02}(behind)$ et $\beta_{020}(behind)$. Ce décalage est effectué à l'aide de l'algorithme 18 déjà présenté lors de la définition de la carte topologique. Nous obtenons alors la carte présentée figure 5.80.b et nous pouvons maintenant exécuter le traitement normal du précode fc_{16} qui effectue les deux fusions de faces coplanaires et donne la carte présentée figure 5.80.c. Si le précode suivant est à nouveau fc_{16}, nous allons rencontrer ce même cas particulier, et recommencer à « pousser » l'arête fictive. Nous obtenons au final une seule arête fictive de longueur quelconque, qui relie deux bords distincts de la face.

Le deuxième cas de ce pré-traitement teste si la fusion des deux faces coplanaires entraîne une déconnexion. Si c'est le cas, nous appliquons simplement le traitement du précode fc_{13}, qui effectue les mêmes fusions que celles du précode fc_{16}, à l'exception

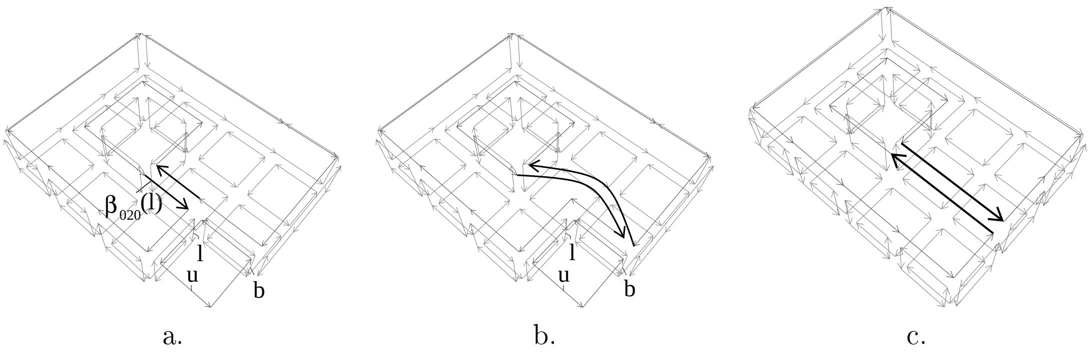


FIG. 5.80 – Un exemple du traitement particulier des précodes fc_{16}, fc_{17} et fc_{18} pour gérer les arêtes fictives.

d'une fusion de faces coplanaires. Cela nous permet de conserver une arête fictive, que nous marquons afin de tester en *O(1)* si un brin est incident à une de ces arêtes. Enfin, le dernier cas est le cas « normal » où nous exécutons le traitement du précode fc_{16}. Ce principe est appliqué de manière similaire pour les deux autres précodes fc_{17} et fc_{18}. Il faut alors tester d'autres brins que ceux du pré-traitement du précode fc_{16} afin de savoir dans quel cas l'on se trouve, et changer le précode appelé dans le deuxième cas.

Nous montrons maintenant que nous avons bien traité tous les cas de fusion d'arêtes alignées. Pour cela, nous utilisons le même principe que pour les deux niveaux précédents, en comptant le nombre de cas où une telle fusion est possible, et en comparant ce nombre au nombre de cas couverts par les 27 précodes. Une manière simple de compter le nombre de cas où une fusion d'arêtes alignées est possible, est de faire le lien avec la dimension 2. En effet, une telle fusion est possible si chaque couple de voxels d'un précode dans une même direction est composé de deux voxels de même région. Cela revient à prendre un précode en 2d, à le placer dans une des trois directions possibles d'un précode 3d, et à le dupliquer pour les 4 voxels restants. Nous pouvons vérifier que cette technique de construction nous permet bien d'obtenir chacun des 27 précodes. Mais il y a trois précodes 2d avec lesquels cette technique n'est pas valide. Pour le précode 2d composé de 4 pixels de même couleur, nous obtenons le précode 3d composé de 8 voxels de même couleur. Dans ce cas, il n'y a pas d'arêtes à fusionner, étant donné qu'il n'y a pas d'arêtes au milieu de ce précode. Les deux autres précodes 2d pour lesquels cette technique n'est pas valide sont les précodes ayant 2 pixels voisins de même couleur, et les 2 autres pixels d'une autre couleur. En effet, ces deux précodes étendus en 3d donnent un précode similaire à celui présenté figure 5.74. Ce précode n'a pas d'arête en son centre, et il n'y a donc pas de fusion d'arêtes à effectuer.

Finalement, le nombre de précodes 3d pour lesquels une fusion d'arêtes alignées est nécessaire est donc de $3 \times (8^4 - 8 - 2 \times 8 \times 7)$. 8^4 est le nombre de cas en 2d, moins 8 pour enlever les cas où un précode a une seule couleur, et moins $2 \times 8 \times 7$ pour enlever les deux précodes en 2d ayant deux pixels voisins de même couleur et les deux autres pixels d'une autre couleur. Il faut multiplier par trois le nombre obtenu, car il y a trois directions possibles pour placer chaque précode 2d. Nous obtenons donc 11 928 cas différents.

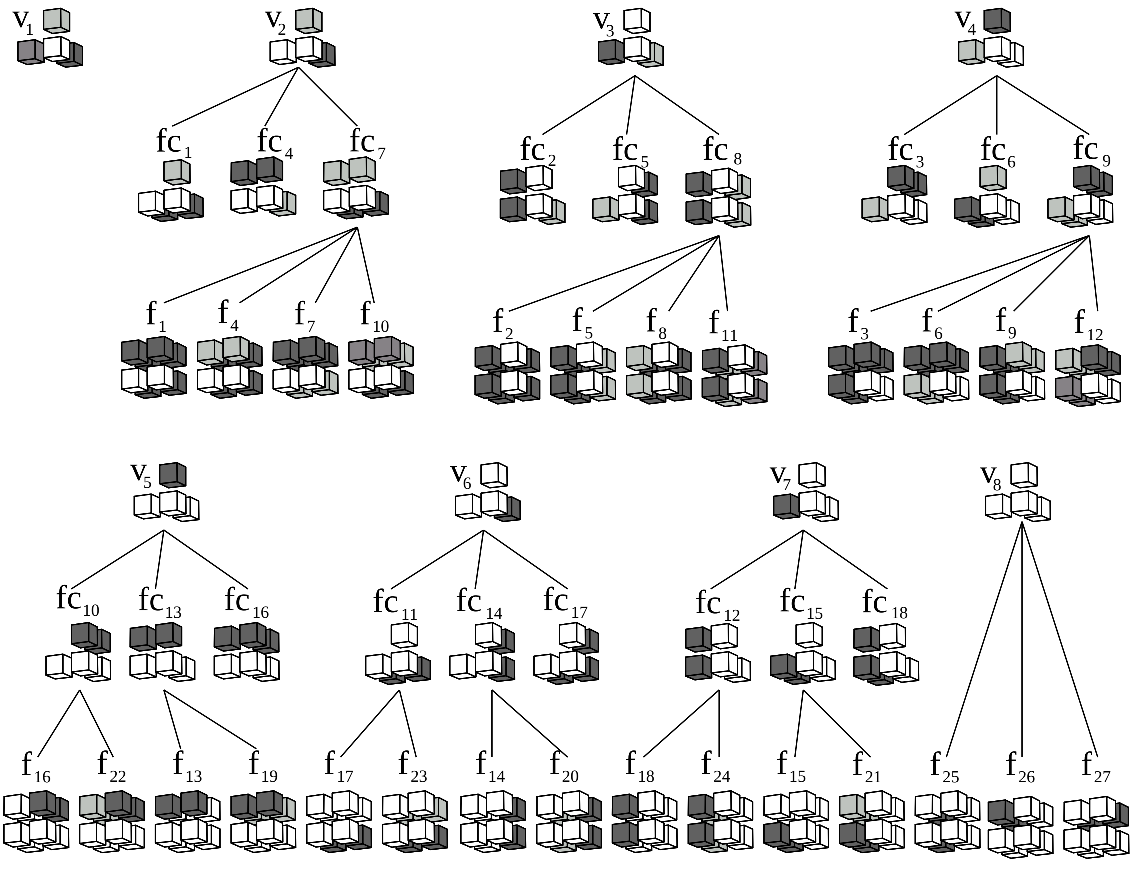
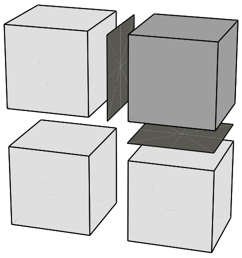


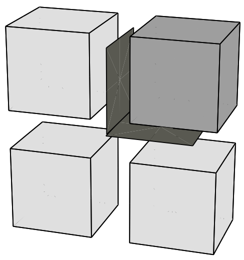
FIG. 5.81 – Les arbres de couverture pour les trois premiers niveaux de simplification.

Nous dénombrons maintenant les cas couverts par chaque précode de niveau 3. Pour les 12 précodes ayant seulement deux régions différentes, il est de $8 \times 7 = 56$. Pour les 12 précodes ayant trois régions différentes, il est de $8 \times 7 \times 6 = 336$. Et pour les 3 précodes ayant quatre régions différentes, il est de $8 \times 7 \times 6 \times 6 + 8 \times 7 \times 7 = 2408$. Le premier terme de cette somme compte les cas de variétés, et le deuxième les cas de non variétés. Nous obtenons finalement que le nombre de cas couverts par les 27 précodes est de $12 \times 56 + 12 \times 336 + 3 \times 2408 = 11\,928$: nous couvrons bien tous les cas possibles de fusion d'arêtes alignées.

Nous présentons figure 5.81 les arbres de couverture pour les précodes des trois premiers niveaux de simplification. Ces arbres sont au nombre de 8, chacun ayant pour racine un précode lignel. Un précode p_1 est fils d'un autre précode p_2 lorsque p_1 est couvert par p_2. Le précode l_1 ne possède pas de fils, étant donné qu'il n'y a aucun précode couvert par lui. En effet, pour ce précode, le voxel courant est différent de ses trois voisins, et il ne peut donc pas y avoir ni fusion de faces coplanaires, ni fusion d'arêtes alignées. Le précode l_8 n'a pas de fils pour la carte de niveau 2, mais en a trois pour la carte des frontières.



a. Avant la fusion.



b. Et après.

Fig. 5.82 – Une représentation partielle d'un précode où l'on peut fusionner deux faces non coplanaires.

5.6.6 Les précodes pour la carte de niveau 4

Nous devons maintenant fusionner les faces non coplanaires adjacentes et incidentes à une arête de degré deux. Ce type de fusion se produit lorsque trois voxels d'un même plan sont de la même région, et que le quatrième est d'une autre région, comme nous pouvons le voir figure 5.82. Nous devons effectuer en même temps les fusions de faces coplanaires, et d'arêtes alignées, et traiter toutes les combinaisons possibles.

Nous pouvons à nouveau trouver tous les précodes à traiter de manière exhaustive, en faisant une étude de cas à partir des 8 précodes lignels. Par exemple si nous cherchons les conditions pour le précode l_1 pour lesquelles nous devons fusionner deux faces non coplanaires, nous obtenons trois possibilités. La première lorsque les voxels 2, 5 et 6 sont de la même région et le voxel 1 d'une autre, les deux autres obtenues par rotation. En effet, étant donné que nous sommes partis du précode l_1, nous savons que le voxel courant n'est pas de la même région que ses trois voisins, et il n'existe pas d'autre cas possible. Nous devons alors étudier toutes les combinaisons possibles avec ces trois cas. Soit un seul de ces trois cas est vrai, ce qui donne alors 3 précodes, soit deux sont vrais, cela donne également 3 précodes, et enfin un précode où les trois sont vrais en même temps. Remarquons qu'il ne peut pas y avoir de fusion de faces coplanaires, étant donné que pour cela il faut que le voxel courant soit de la même région qu'au moins un de ses voisins. Pour le précode l_1, nous obtenons donc les 7 précodes de niveau 4 présentés figure 5.83 (que nous appelons précode nc_i pour précode non coplanaire numéro i). Les trois premiers de ces précodes sont les cas où une seule fusion est à faire. Remarquons que ces trois cas sont isomorphes par rotation. Les précodes nc_4, nc_5 et nc_6 sont les trois précodes où nous devons fusionner deux couples de faces non coplanaires, et enfin le dernier précode est le cas où nous devons fusionner les trois couples de faces non coplanaires.

Nous pouvons utiliser le même principe afin de trouver les sous-cas du précode l_2. C'est un peu plus complexe car nous devons maintenant tenir compte des faces coplanaires et de toutes les combinaisons possibles entre ces deux types de fusions. Afin de ne pas oublier un cas, nous effectuons ce raisonnement à l'aide d'un arbre de décision présenté figure 5.84. Pour le précode l_2, nous savons que le voxel 1 est de la même région que le voxel 2, et qu'il est d'une région différente des voxels 3 et 5. Nous étudions donc toutes les possibilités pour

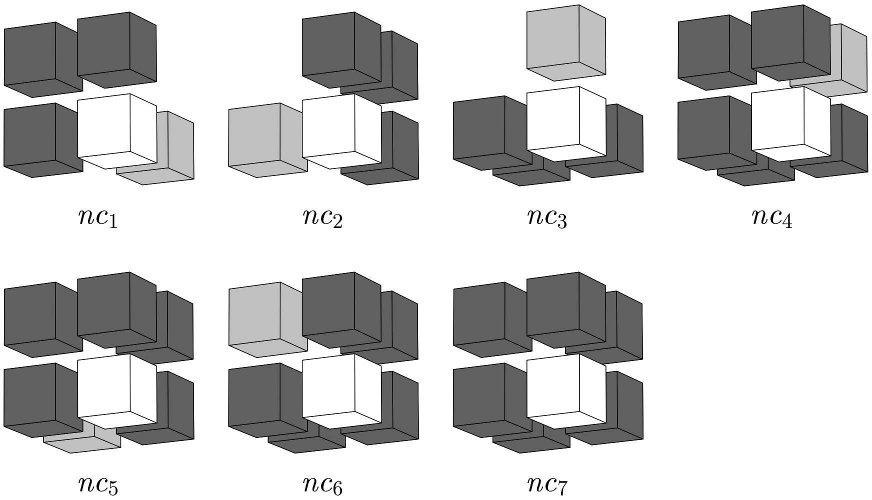


FIG. 5.83 – Les sept sous-cas du précode l_1 pour calculer la carte de niveau 4 en 3d.

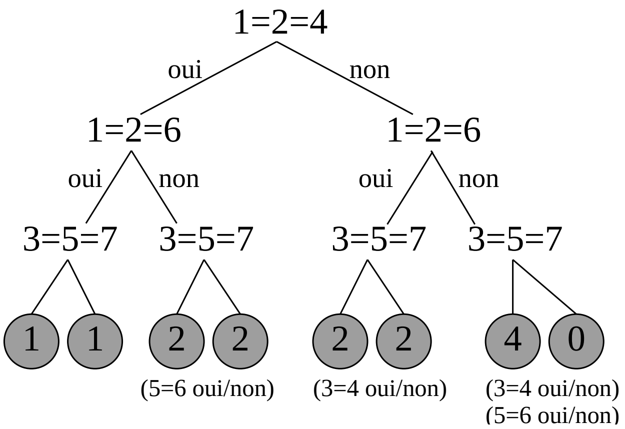


FIG. 5.84 – L'arbre de décision pour trouver les précodes de niveau 4 sous-cas du précode l_2.

les voxels restants, en tenant compte de ces contraintes. Nous écrivons $1 = 2 = 4$ pour dire que les voxels 1, 2 et 4 sont de la même région et le voxel 3 d'une région différente. Il existe les trois possibilités isomorphes par rotation. Remarquons que si aucune n'est vraie, il n'y a alors pas de fusion de faces non coplanaires à faire, nous sommes dans la branche à droite de l'arbre. Il n'y a alors pas de précode de niveau 4 correspondant. Pour les cas intermédiaires, il peut exister des fusions de faces coplanaires. Enfin, pour le cas où seul les trois voxels 3, 5 et 7 sont de même région, il existe 4 précodes différents, qui sont toutes les combinaisons possibles si les voxels 3 et 4 sont de la même région ou pas, et 5 et 6 de même région ou pas. Nous obtenons au final 14 cas différents, sous-cas du précode l_2. Nous obtenons le même nombre de cas pour les précodes l_3 et l_4, en effectuant un raisonnement identique. Nous présentons quatre de ces précodes figure 5.85. Nous pouvons

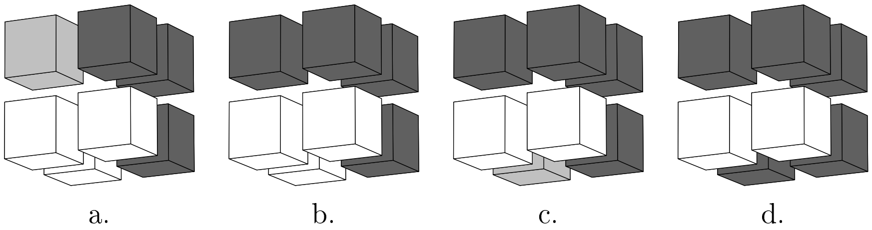


FIG. 5.85 – Quatre précodes de niveau 4 en 3d, sous-cas du précode l_2.

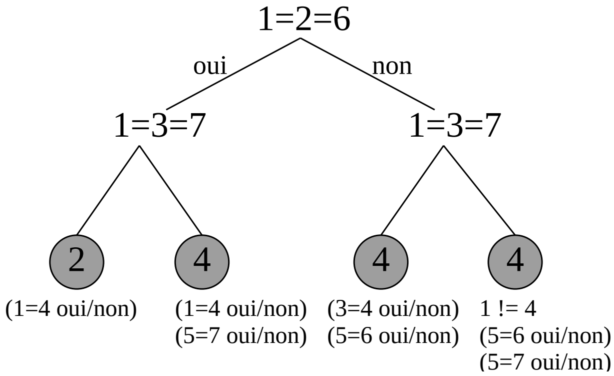


FIG. 5.86 – L'arbre de décision pour trouver les précodes de niveau 4 sous-cas du précode l_5.

obtenir l'ensemble des précodes sous-cas du précode l_2 en parcourant l'arbre de décision et en affectant les voxels correspondant suivant les cas. Par exemple le précode présenté figure 5.85.a correspond au cas où $1 = 2 = 4$, non $1 = 2 = 6$, $3 = 5 = 7$ et non $5 = 6$. Le précode de la figure 5.85.a correspond au même cas, sauf que ici $5 = 6$. La différence de traitement entre ces deux précodes est que pour le deuxième il existe une fusion de faces coplanaires qui n'est pas effectuée pour le premier.

Pour le précode l_5, nous effectuons le même raisonnement à l'aide de l'arbre de décision présenté figure 5.86. La feuille la plus à droite de cet arbre regroupe 4 précodes différents pour lesquels le voxel 1 n'est pas de la même région que le voxel 4, puis les 4 combinaisons possibles suivant si les voxels $5 = 6$ ou non et les voxels $5 = 7$ ou non. Nous obtenons donc 14 précodes supplémentaires sous-cas du précode l_4. Nous obtenons les mêmes nombres pour les sous-cas des précodes l_6 et l_7 en appliquant la même démarche.

Il ne reste plus qu'à trouver les sous-cas du précode l_8. Pour ce précode, nous avons $1 = 2 = 3 = 5$. Les trois cas de fusion de faces non coplanaires sont lorsque le voxel 4 ou le voxel 6 ou le voxel 7 n'appartiennent pas à la même région que les voxels 1, 2, 3 et 5. Il existe donc trois précodes où une seule de ces conditions est vérifiée, trois précodes où deux sont vérifiées, et un où les trois sont vérifiées. Nous obtenons donc les 7 précodes présentés figure 5.87.

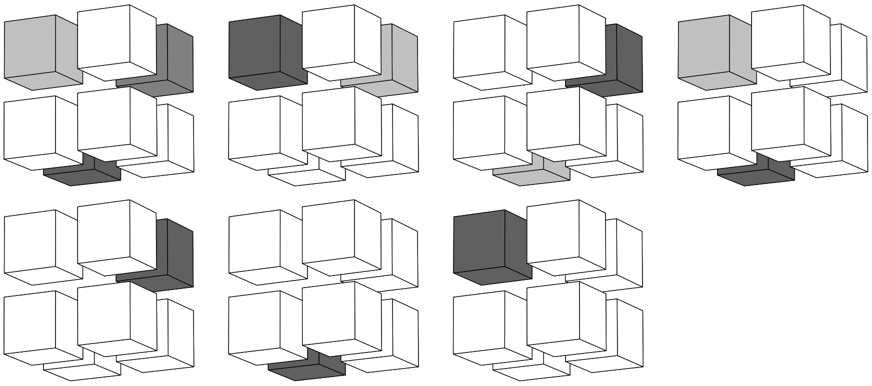


FIG. 5.87 – Les sept sous-cas du précode l_8 pour calculer la carte de niveau 4 en 3d.

Au total nous avons donc 98 précodes afin d'extraire la carte de niveau 4, présentés annexe A. Tous ces précodes comprennent au moins une fusion de faces non coplanaires, et éventuellement des fusions de faces coplanaires. Mais il n'y a pas de cas de fusion d'arêtes alignées. En effet, si nous regardons les 27 précodes frontières pour lesquels nous effectuons une telle fusion, nous pouvons noter que pour 12 d'entre eux il faut fusionner des faces non coplanaires. Ce sont les 12 précodes présentés figure 5.88. Mais chacun de ces précodes est déjà couvert par un des 98 précodes de niveau 4. Par exemple le précode f_1 est couvert par le précode figure 5.85.d. Le traitement correspondant à ce précode effectue la fusion de faces non coplanaires. Après cette fusion, il n'y a plus d'arête au centre du précode, et donc pas de fusion d'arêtes alignées possible. En pratique, cela revient à dire que lorsque nous calculons la carte de niveau 4, nous testons en priorité les précodes de ce niveau avant les précodes de niveau 3. Si le précode courant est un précode de niveau 4, nous exécutons son traitement. Le cas échéant, nous testons si le précode courant est un de niveau 3.

Nous devons comme pour le niveau 2, ne pas effectuer de fusion de faces lorsque cela entraîne une déconnexion. Nous utilisons la même solution que celle du niveau 2, en exécutant un pré-traitement pour les précodes susceptibles d'entraîner une déconnexion. Nous présentons figure 5.89 deux précodes de niveau 4 pouvant entraîner une déconnexion de face. Le premier précode (figure 5.89.a) est un sous-cas du précode l_8. Ce précode doit fusionner la face entre les voxels 2 et 6 et celle entre les voxels 5 et 6. Nous testons si ces faces sont différentes ou non. Lorsque c'est une seule face, la fusion va entraîner une déconnexion. Dans ce cas, nous exécutons simplement le traitement du précode l_8, étant donné qu'il n'y a pas d'autre fusion à effectuer. Le précode présenté figure 5.89.c peut également entraîner une déconnexion. Ce précode effectue une fusion de faces non coplanaires ainsi qu'une fusion de faces coplanaires. Cette deuxième fusion ne peut pas entraîner de déconnexion. Si la première fusion en entraîne une, nous exécutons le traitement du précode fc_{13} qui est le précode effectuant uniquement la fusion des deux faces coplanaires. Dans ces deux cas, nous n'effectuons pas la fusion qui entraînerait une déconnexion, et créons donc une arête fictive.

Nous ne détaillons pas l'écriture des algorithmes correspondant à ces précodes de niveau 4. La technique utilisée est toujours la même : dessiner la carte courante avant le

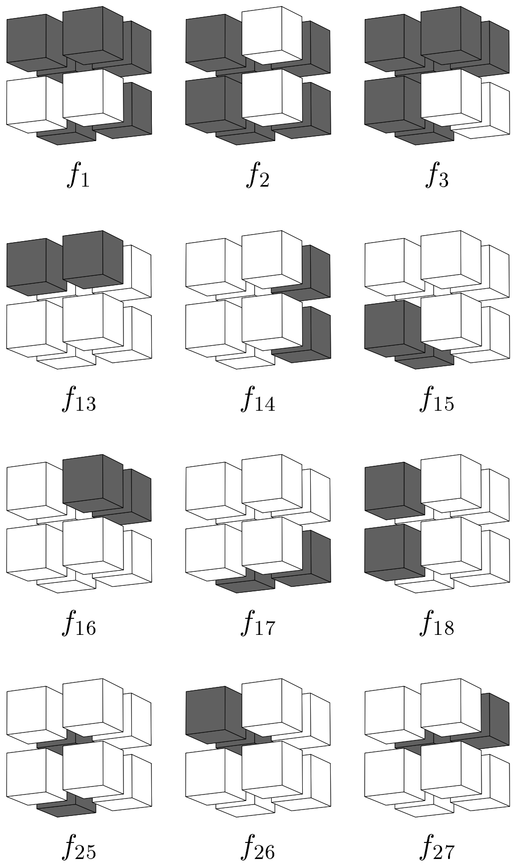


FIG. 5.88 – Les 12 précodes frontières pour lesquels nous effectuons une fusion de faces non coplanaires afin de calculer la carte de niveau 4.

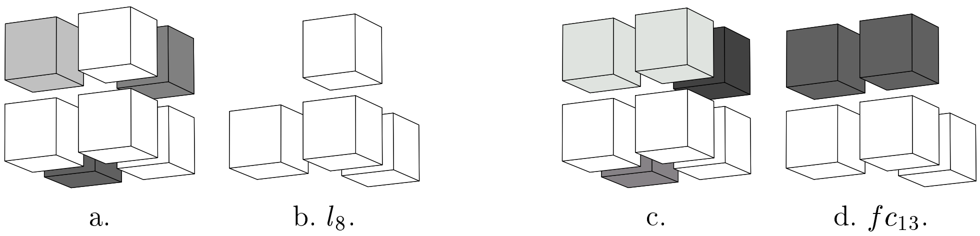


FIG. 5.89 – Deux précodes de niveau 4 pour lesquels une déconnexion de faces peut se produire, et leurs précodes de « remplacement ».

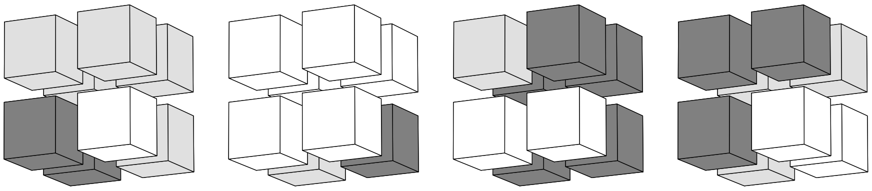


FIG. 5.90 – Quatre exemples de précodes de niveau 5 en 3d.

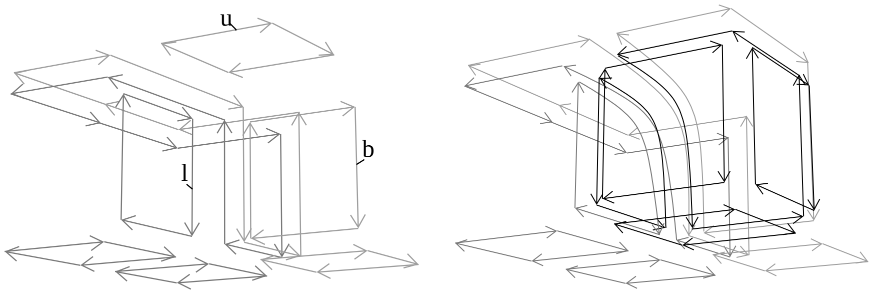
traitement d'un précode, la carte à obtenir, et écrire la suite des opérations transformant la première carte en la seconde. Notons que nous pouvons avoir désormais des faces non coplanaires, et que nous devons donc conserver un plongement face afin de garder la géométrie des régions représentées. Cela n'était pas obligatoire jusqu'à ce niveau, étant donné que nous avions uniquement des arêtes topologiques représentant des segments de droite. Il existe pour ce niveau 98 précodes différents, mais « seulement » 34 classes de précodes, obtenues en regroupant les précodes isomorphes par rotation. En effet, nous avons un précode sous-cas de l_1 (cf. figure 5.83) et un précode sous-cas de l_8 (cf. figure 5.87) qui sont stables par rotation. Nous obtenons donc $(98-2)/3+2=34$ classes d'équivalences. Ce nombre de méthodes reste raisonnable et permet d'envisager d'implanter le calcul de ce niveau de manière optimale, d'autant plus que des compromis peuvent être faits entre efficacité et nombre de méthodes à développer comme pour le niveau 3.

5.6.7 Les précodes pour la carte topologique

Pour ce dernier niveau nous devons fusionner les arêtes non alignées incidentes à des sommets de degré deux, si nous mettons momentanément de côté la gestion des arêtes fictives. Nous présentons figure 5.90 quatre exemples de précodes pour lesquels nous devons effectuer une telle fusion. Comme nous définissons ici les précodes de notre dernier niveau de simplification, nous devons également effectuer toutes les autres fusions possibles par rapport au voxel courant (volumes, faces coplanaires, arêtes alignées et faces non coplanaires). Par exemple pour le précode 5.90.a une fusion de faces non coplanaires et une fusion d'arêtes non alignées. Pour le précode 5.90.c il faut effectuer une fusion de volumes, une de faces coplanaires et une de faces non coplanaires.

Nous pouvons voir figure 5.91.a la carte correspondant au précode de la figure 5.90.a, avant son traitement, et figure 5.91.b la carte à obtenir. Nous observons sur cette carte que deux arêtes non alignées ont été fusionnées. Nous pouvons également vérifier que cette fusion a bien été réalisée autour d'un sommet de degré 2, ce qui est assez difficile à voir sur le précode correspondant.

Afin de traiter les arêtes fictives, nous utilisons le même principe que pour les précodes de niveau 3. Avant une fusion d'arêtes, nous décalons toutes les arêtes fictives incidentes au sommet supprimé lors de cette fusion. De plus, nous devons également, pour chaque précode de niveau 4 susceptible d'entraîner une déconnexion de faces, effectuer un pré-traitement similaire à celui présenté pour les précodes de niveau 3. Ce pré-traitement permet, suivant



a. Carte avant le traitement. b. Le résultat.

FIG. 5.91 – Un exemple de précode où l'on fusionne deux arêtes non alignées en 3d.

TAB. 5.2 – Récapitulatif du nombre de précodes pour chaque niveau de simplification.

Niveau	1	2	3	4	5
Précodes pour ce niveau	8	18	27	98	228
Nombre total de précodes	8	26	53	151	379
Classes d'équivalences pour ce niveau	4	6	9	34	76
Nombre total de classes d'équivalences	4	10	19	53	129

la configuration locale, de ne pas effectuer une fusion si elle entraîne une déconnexion, ce qui crée une arête fictive, où d'étirer une arête fictive afin d'éviter de créer des chemins d'arêtes fictives. Nous ne revenons pas ici sur ce pré-traitement qui va être très proche de celui déjà présenté section 5.6.5.

Nous avons calculé par programme le nombre de précodes supplémentaires à traiter pour ce dernier niveau : 228. Ces précodes sont présentés annexe B. Il existe « seulement » 76 classes d'équivalences de précodes isomorphes par rotation, ce qui diminue de manière conséquente le nombre de méthodes différentes à écrire. De plus, si nous effectuons uniquement les fusions de volumes et de faces de manière optimisée, nous devons seulement écrire 50 méthodes différentes. En effet, par cette technique, nous regroupons tous les précodes qui effectuent les mêmes fusions, mais où la fusion d'arêtes diffère uniquement par le nombre de brins de ces arêtes. Ces fusions d'arêtes étant, pour cette solution, effectuées à l'aide de l'algorithme générique, le traitement sera alors le même pour ces différents cas. Cette solution permet d'envisager des implantations intermédiaires, plus ou moins optimisées, mais plus ou moins longues en temps de développement.

Le tableau 5.2 récapitule le nombre de précodes supplémentaires et total pour chaque niveau de simplification. Nous obtenons donc au total 379 précodes à traiter pour extraire la carte topologique en une seule passe de l'image. Ce nombre est important, mais pas en regard des 4140 précodes existant en dimension 3, ni même en regard des 958 précodes variétés. En effet, nous avons regroupé 579 de ces précodes variétés, ce qui est de beaucoup supérieur à nos prévisions initiales, estimées après avoir défini la carte topologique

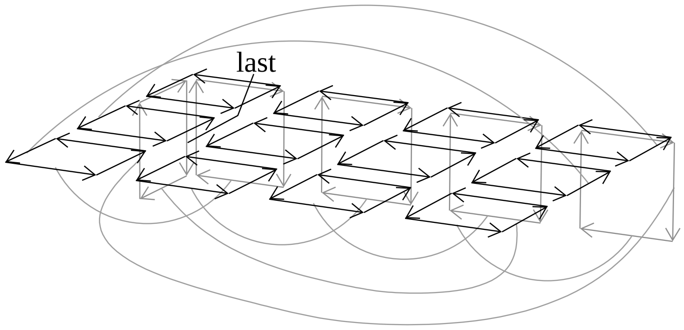


FIG. 5.92 – Le bord créé avant de commencer le traitement de l'image. Ici sur une image de $3 \times 2 \times n_3$.

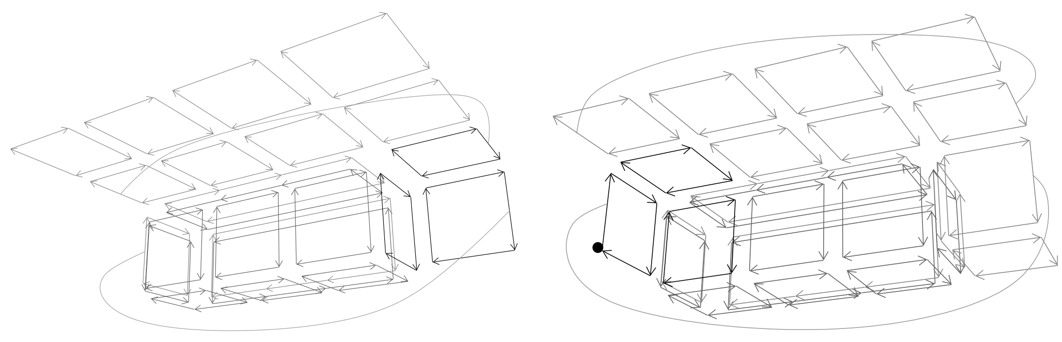
en dimension 2. De plus, nous avons seulement 129 classes d'équivalences de précodes isomorphes par rotation.

La complexité globale de l'algorithme optimal d'extraction est linéaire en le nombre de brins de l'image, multiplié par le nombre moyen de brins par face. Ce nombre est difficilement calculable étant donné qu'il dépend fortement du type d'image traitée. Mais nous verrons section 5.8 sur quelques images de tests, qu'il est relativement faible, et en moyenne égale à 7.

5.6.8 Le bord de l'image

Comme en dimension 2, c'est la définition du bord initial de l'image qui permet de s'affranchir des problèmes qui lui sont inhérents, et évite des cas particuliers et des traitements dupliqués. L'image en entrée de l'algorithme optimal est composée des voxels de $\{1 \dots n_1\} \times \{1 \dots n_2\} \times \{1 \dots n_3\}$, les autres voxels appartiennent tous à la région infinie. Nous plaquons cette image sur un tore, afin que le traitement d'un précode sur le bord droit de l'image crée la face de gauche du premier précode de la ligne suivante, et que le traitement d'un précode de la dernière ligne d'une plaque de voxels crée la face de derrière du voxel de la plaque suivante. Pour cela, nous identifions les voxels de coordonnées (n_1+1, y, z) aux voxels $(0, y+1, z)$ (le dernier voxel d'une ligne et le premier voxel de la ligne suivante en y), ainsi que les voxels de coordonnées $(x, y+1, z)$ aux voxels $(x, 0, z+1)$ (le dernier voxel d'une colonne, et le premier voxel de la même colonne de la plaque suivante).

Nous présentons figure 5.92 ce bord initial pour une image de taille $3 \times 2 \times n_3$. Ce bord initial est constitué des faces supérieures de l'image, représentant le bord inférieur de la région infinie. Ces faces vont de $1 \dots n_1 + 1$ en x et $1 \dots n_2 + 1$ en y. En effet, lors de notre balayage de l'image, nous « débordons » d'un voxel par rapport à l'image dans les 3 directions de l'espace, afin de tenir compte des frontières avec la région infinie. Ce bord contient également les faces derrière la première ligne de voxels de l'image, plus la face de gauche du premier voxel. De ce fait, le premier voxel possède bien les trois faces à sa gauche, derrière et au-dessus de lui. Les autres voxels de cette première ligne possèdent



a. Avant de traiter ce précode. b. Après traitement du précode.

FIG. 5.93 – Configuration d'une carte en cours de construction, avant et après le traitement du dernier précode de la première ligne.

une face au-dessus et derrière, la face de gauche étant à chaque fois créée par le traitement du voxel précédent.

Nous pouvons voir sur la figure 5.92 comment les faces aux extrémités du bord sont cousues afin d'identifier les voxels de la manière présentée ci-dessus. Sur cette figure, nous avons représenté les β_2-coutures de ces brins particuliers avec les courbes grises. Nous pouvons remarquer que cette carte initiale est 2-fermée. Nous β_2-cousons la dernière face d'une ligne avec la première face de la ligne suivante. De ce fait, le traitement du dernier précode d'une ligne crée la face de gauche du premier voxel de la ligne suivante, comme nous pouvons le vérifier figure 5.93. Cette figure montre la configuration d'une carte en cours de construction, avant de traiter le dernier précode de la première ligne. L'image traitée est ici de taille $3 \times 2 \times n_3$, et nous sommes en train d'extraire la carte de niveau 3. En effet, nous pouvons vérifier figure 5.93.a que nous avons déjà effectué des fusions de faces coplanaires et d'arêtes alignées. Le dernier précode de la première ligne est toujours le même, quelle que soit l'image, étant donné qu'un seul voxel appartient à l'image, tous les autres appartiennent à la région infinie. Nous pouvons vérifier figure 5.94 qu'il est impossible d'avoir n'importe quel précode sur les bords de l'image. Pour les 8 angles, nous avons toujours un seul précode possible. Pour les « arêtes » de l'image, il existe deux précodes possibles étant donné que deux voxels appartiennent à l'image, ils peuvent donc être de même région ou non. Enfin, nous n'avons pas représenté sur cette figure les précodes au centre des faces, mais il en existe 15 possibles par face. En effet, seuls 4 voxels de ces précodes appartiennent à l'image, et nous retrouvons donc les 15 précodes existant en 2d.

Pour l'exemple de la figure 5.93, nous traitons le dernier précode de la première ligne et nous fusionnons donc le voxel courant avec ses voisins de derrière et de dessus (précode l_7). Le calcul du nouveau sommet, passé en paramètre à chaque précode, est calculé à l'aide d'un modulo afin de simuler le tore. Lorsque nous traitons le voxel (i, j, k), ces coordonnées sont simplement $(i \text{ modulo } (n_1+1), j + (i \text{ div } (n_1+1), k)$. Ce nouveau sommet aura donc, pour ce dernier précode où le voxel courant est $(4, 1, 1)$, pour coordonnées $(0, 2, 1)$. Ce sommet est représenté par le point noir sur la figure 5.93.b. De par la construction initiale du bord et ce calcul des coordonnées, la face de droite du voxel courant se retrouve à gauche

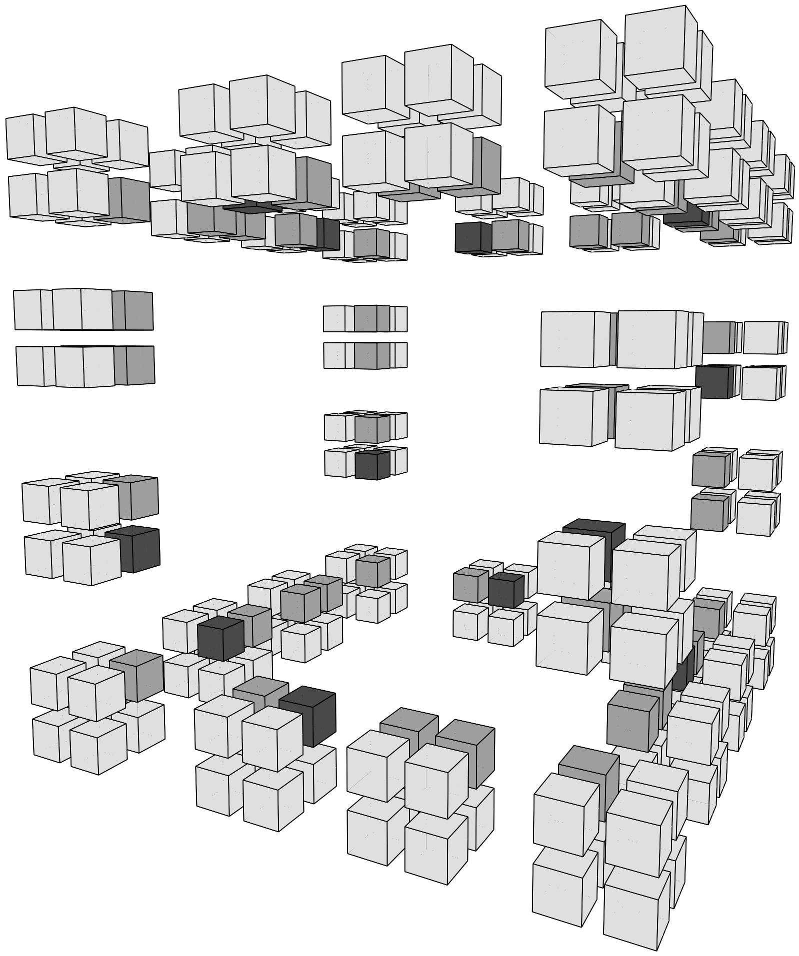


FIG. 5.94 – Les précodes pouvant se trouver sur les bords de l'image 3d.

du premier voxel de la ligne suivante. De plus, le brin *last* est positionné correctement sur cette face, et nous pouvons donc traiter le prochain précode sans cas particulier. Il en est de même pour les précodes de la dernière ligne de l'image, qui vont créer de manière similaire les faces derrière le premier voxel de la même colonne de la plaque suivante. Pour cela, nous devons effectuer le calcul des coordonnées du nouveau sommet modulo la deuxième coordonnées : $(i \text{ modulo } (n_1+1), (j+(i \text{ div } (n_1+1))) \text{ modulo } (n_2+1), k+(j+(i \text{ div } (n_1+1))) \text{ div } (n_2+1))$. Intuitivement, ce deuxième modulo revient à effectuer le deuxième repliement de la surface pour en faire un tore. Le traitement du dernier voxel d'une plaque va créer la face de derrière du premier voxel de la même colonne et de la plaque suivante, ainsi que la face de gauche du premier voxel de la plaque suivante.

Chaque précode fonctionne donc de manière identique qu'il soit au bord de l'image ou pas. De plus, la construction du bord initial ne dépend pas de la hauteur de l'image, étant donné que ce sont les précodes de bord qui créent les faces de bords des plaques suivantes. Enfin, de manière similaire à la dimension 2, après avoir fini le balayage de l'image, nous retrouvons le bord initial en-dessous et déconnecté de celle-ci. Il est alors inutile et peut être détruit.

5.7 Un algorithme d'extraction intermédiaire

Nous proposons ici un dernier algorithme d'extraction qui est une solution intermédiaire intéressante entre les algorithmes d'extractions naïf et optimal. En effet, l'algorithme naïf est en complexité quadratique en le nombre de brins de l'image. Il est donc difficilement utilisable sur de grosses images. L'algorithme optimal est bien meilleur en complexité, mais demande beaucoup de travail de développement. Même en regroupant les précodes isomorphes par rotations, nous devons quand même développer 129 méthodes différentes afin de calculer la carte topologique. C'est envisageable mais reste un travail important. C'est pour cette raison que nous proposons l'algorithme 34 qui est une troisième approche permettant d'implanter rapidement un algorithme d'extraction ayant une complexitée linéaire, mais n'étant pas optimal.

Cet algorithme reprend le principe de l'algorithme optimal. Il commence par construire le bord de l'image de manière identique à cet algorithme, puis effectue un balayage de l'image selon le même ordre. Nous conservons le même invariant que pour l'algorithme optimal : le voxel courant possède toujours une face à sa gauche, dessus et derrière, désignées par les brins *last*, *up* et *behind*.

Au cours de ce balayage, cet algorithme crée un cube topologique, (ligne 1) correspondant au voxel courant, et le β_3-coud correctement à ces trois faces. Ce traitement correspond au précode l_1 pour lequel aucune fusion n'est effectuée. Afin de faciliter la lecture de l'algorithme, nous nommons ensuite les différentes cellules incidentes à ces trois faces, et susceptibles d'être supprimées par une fusion. Les traitements suivants sont comparables aux cinq traitements effectués pour l'algorithme d'extraction naïf. La différence est que ces traitements sont ici effectués uniquement pour les cellules incidentes au cube qui vient d'être créé. Par exemple, pour le niveau 1, nous testons si les trois volumes v_1, v_2 et v_3 (incidents aux brins *last*, *up* et *behind*) sont de la même région que le voxel courant, et lorsque c'est le cas nous effectuons la fusion de ces volumes avec le voxel courant. Les traitements sont ensuite similaires pour les autres niveaux. Remarquons que pour la fusion de faces, nous avons seulement trois arêtes le long desquelles nous pouvons effectuer une fusion, et pour la fusion d'arêtes un seul sommet.

Nous utilisons les mêmes principes que ceux déjà présentés lors de l'algorithme optimal afin de gérer correctement les arêtes fictives. Pour une fusion de deux faces, nous testons si cela entraîne une déconnexion, et lorsque c'est le cas nous testons s'il existe une arête fictive pouvant être étirée afin de ne pas créer des chemins d'arêtes fictives. Lors d'une fusion d'arêtes, nous appelons préalablement l'algorithme décalant l'ensemble des arêtes fictives incidentes à un sommet.

Cet algorithme est intermédiaire entre les deux algorithmes précédents. Il utilise le principe de balayage de l'image de l'algorithme optimal afin d'éviter de créer initialement toute la carte de niveau 0, donc énormément de brins, pour ensuite en supprimer beaucoup lors des différentes fusions. Mais il utilise le principe des fusions génériques de l'algorithme

Algorithme 34 Extraction « intermédiaire » de la carte de niveau n en 3d

Entrée : Une image I segmentée en régions de $n_1 \times n_2 \times n_3$ voxels
n le niveau de la carte que l'on veut extraire

Sortie : La carte de niveau n de l'image I.

- *last* $\leftarrow$ Construire le bord supérieur de la carte;
- **pour** $k = 1$ *à* $n_3 + 1$ **faire**
  - **pour** $j = 1$ *à* $n_2 + 1$ **faire**
    - **pour** $i = 1$ *à* $n_1 + 1$ **faire**
      - *tmp* $\leftarrow$ créer un cube topologique ;
      - β_3-coudre la face contenant *last* et celle contenant $\beta_{2112}(tmp)$;
      - β_3-coudre la face contenant *up* et celle contenant $\beta_{021}(tmp)$;
      - β_3-coudre la face contenant *behind* et celle contenant $\beta_2(tmp)$;
      - $x \leftarrow i$ modulo $(n_1 + 1)$;
      - $y \leftarrow (j + (i \text{ div } (n_1 + 1)))$ modulo $(n_2 + 1)$;
      - $z \leftarrow k + (j + (i \text{ div } (n_1 + 1)))$ div $(n_2 + 1)$;
      - Le plongement du sommet incident à $\beta_{11}(tmp) \leftarrow (x, y, z)$;
      - // Notons v_1, v_2 et v_3 les volumes incidents à *last*, *up* et *behind*;
      - // Notons f_1, f_2 et f_3 les faces incidentes à *last*, *up* et *behind*;
      - // Notons a_1, a_2 et a_3 les arêtes incidentes à $\beta_0(last)$, $\beta_0(up)$ et $\beta_{11}(behind)$;
      - // Notons s le sommet incident à *last*;
      - **si** *v_1 (resp. v_2, v_3) est de la même région que le voxel courant* **alors**
        - Fusionner v_1 (resp. v_2, v_3) avec le voxel courant le long de f_1 (resp. f_2, f_3);
      - **si** $n \geq 2$ **alors**
        - **si** *a_1 (resp. a_2, a_3) est de degré un ou deux, que les deux faces incidentes à cette arêtes sont coplanaires et que la fusion de ces deux faces n'entraîne pas déconnexion* **alors**
          - Fusionner les deux faces incidentes à a_1 (resp. a_2, a_3);
        - **si** $n \geq 3$ **alors**
          - **si** *s est de degré deux et que les deux arêtes incidentes à ce sommet sont alignées* **alors**
            - Fusionner les deux arêtes incidentes à s;
          - **si** $n \geq 4$ **alors**
            - **si** *a_1 (resp. a_2, a_3) est de degré un ou deux et que la fusion de ces deux faces n'entraîne pas déconnexion* **alors**
              - Fusionner les deux faces incidentes à a_1 (resp. a_2, a_3);
            - **si** $n \geq 5$ **alors**
              - **si** *s est de degré deux* **alors**
                - Fusionner les deux arêtes incidentes à s;
      - *last* $\leftarrow$ *tmp*;
- Calculer l'arbre d'inclusion des régions;

naïf, afin d'éviter de définir les traitements des précodes. La perte en complexité par rapport à l'algorithme optimal est relativement faible. En effet, nous créons ici des brins pour le cube initial, pour les détruire éventuellement ensuite lors des fusions, ce qui entraîne une perte en temps d'exécution. De plus, les fusions sont effectuées de manière générique, et nous allons donc effectuer des boucles et des tests, ce qui n'est pas le cas pour l'algorithme optimal. Mais l'ordre de complexité général reste le même que celui de l'algorithme optimal, et l'espace mémoire nécessaire est de l'ordre de l'espace mémoire du niveau calculé. Ce n'est pas le cas pour l'algorithme naïf, qui requiert beaucoup d'espace mémoire pour la construction du niveau 0.

Enfin, cette solution peut également être améliorée en calculant certains niveaux de simplification au moyen des précodes, et en effectuant les autres fusions de manière classique. Nous pouvons par exemple envisager de traiter les 8 précodes lignels, ce qui évite la construction des brins pour les faces entre deux voxels de même région, et permet d'avoir à définir seulement les 4 traitements correspondant au niveau 1.

5.8 Expérimentations et analyse

Nous avons implanté l'algorithme optimal d'extraction à base de précodes pour les trois premiers niveaux de simplification. En effet, le plongement utilisé pour ces trois premiers niveaux est un plongement sommet, où nous associons à chaque sommet topologique les coordonnées d'un point 3d. Ce plongement n'est plus valable pour les niveaux 4 et 5 étant donné que les faces ne sont plus forcément planes. Pour ces niveaux, nous avons utilisé le plongement face, tout d'abord implanté avec des 2-cartes. Mais les premiers résultats obtenus ont été catastrophiques en temps d'exécution.

À titre de comparaison, pour une image de taille $64 \times 64 \times 64$, le calcul de la carte de niveau 4 ayant environ 55 000 brins prend plus de 40 minutes avec le plongement 2-cartes, et 1.6 secondes avec le plongement 2-G-cartes. Cette image a environ 210 000 brins dans les 2-cartes de plongements, et il y a plus de 400 millions retournements de ces brins. Cela explique le temps d'exécution important. En effet, nous retournons une carte de plongement lorsqu'elle n'a pas la même orientation qu'une autre carte, et qu'elles doivent être fusionnées. Mais la carte résultante de cette fusion pourra être à nouveau retournée lors d'autres fusions.

Nous avons donc abandonné l'idée de représenter les surfaces avec des 2-cartes au profit des 2-G-cartes. En effet, même si l'espace mémoire occupé par ces dernières est deux fois plus important que celui occupé par les 2-cartes, le gain en temps d'exécution justifie complètement cette perte d'espace mémoire. Nous avons implanté le plongement face (présenté section 5.4) car il est plus simple que le plongement face ouverte, arête ouverte et sommet. Mais ce plongement n'est pas optimal en espace mémoire, étant donné qu'il représente plusieurs fois le plongement des sommets et des arêtes de la carte topologique. De ce fait, mais également car nous utilisons les 2-G-cartes pour représenter les surfaces, l'espace mémoire occupé par la carte topologique est généralement un peu plus important que celui occupé par la carte de niveau 3. Il faudrait effectuer les mêmes comparaisons en implantant le plongement face ouverte, arête ouverte et sommet afin de comparer les espaces mémoires « optimaux » de ces niveaux de simplification. De plus, il est possible

TAB. 5.3 – Cœur, $64 \times 64 \times 64$, 204 régions.

	Niveau 1	Niveau 2	Niveau 3	Niveau 4	Niveau 5
Nombre de brins	429 552	88 140	38 382	23 370	15 934
Nombre de sommets géométriques	107 756	37 562	12 183	12 928	12 928
Nombre de brins par face	4	12	5,2	9,8	6,7
Nombre d'arêtes fictives	0	55	4	209	85
Espace mémoire en méga-octets	12,3	2,67	1,12	1,64	1,37
Temps de calcul en secondes	1,73	1,57	1,54	1,6	1,52

d'utiliser les 2-G-cartes lors de la construction de la carte topologique, afin de ne pas avoir de nombreux retournements à effectuer, puis de les transformer en 2-cartes afin de diminuer l'espace mémoire utilisé. Si les modifications effectuées par la suite ne demandent pas trop de retournements, la perte en temps d'exécution sera minime et le gain en espace mémoire important.

Ces 2-G-cartes représentent le plongement de chaque face frontière de la carte topologique. Elles sont implantées de manière similaire aux cartes topologiques 2d, en fusionnant les arêtes incidentes à des sommets de degré deux. Elles sont plongées arêtes ouvertes et sommets, le plongement des arêtes étant effectué avec des 1-G-cartes. Nous obtenons un modèle hiérarchique, qui a pour avantage de pouvoir facilement s'étendre en dimension supérieure. De plus, ce plongement réutilise le travail effectué sur la carte topologique 2d ainsi que certaines fonctions, comme par exemple les fusions de faces et d'arêtes en 2d. Par manque de temps, nous n'avons pas implanté l'algorithme optimal d'extraction pour les niveaux 4 et 5. Nous avons utilisé l'algorithme intermédiaire, en effectuant les traitements des 8 précodes lignels de manière optimisée, puis les autres fusions de manière classique. Cela montre l'avantage de cet algorithme, et permet de voir que la perte en temps d'exécution n'est pas trop importante.

Nous présentons les résultats obtenus pour 5 images différentes. Le manque de temps et d'images 3d segmentées ont en effet limité le nombre de tests effectués. Ces images sont en majorité des images médicales, étant donné que c'est le domaine d'application utilisant le plus les images 3d. Le tableau 5.3 présente les résultats de l'extraction des 5 niveaux de simplification pour une image IRM de cœur de taille $64 \times 64 \times 64$ et composée de 204 régions. Les tableaux 5.4, 5.5 et 5.6 présentent les résultats pour trois IRM de cerveaux. Le tableau 5.7 présente les résultats pour une image provenant de l'institut français du pétrole représentant un bloc géologique afin d'étudier les failles sismiques. Cette image est composée de seulement deux régions, car le but de la segmentation est de différencier les voxels appartenant au bord des failles des autres voxels.

La première ligne de ces tableaux donne le nombre de brins nécessaires à la représentation de chaque niveau de simplification. Nous voyons que ce nombre diminue de manière impressionnante, mais ce n'est pas étonnant en regard de tous les éléments fusionnés. Ces nombres montrent également que le temps des traitements concernant uniquement la partie topologique vont être grandement améliorés étant donné le nombre d'éléments beaucoup moins important à traiter. Nous présentons sur la deuxième ligne de ces tableaux le nombre de sommets géométriques représentés, c'est-à-dire les sommets du plongement de la carte.

TAB. 5.4 – Cerveau1, 256 × 256 × 69, 52909 régions.

	Niveau 1	Niveau 2	Niveau 3	Niveau 4	Niveau 5
Nombre de brins	10 688 776	6 892 772	6 001 862	3 246 718	2 493 866
Nombre de sommets géométriques	2 753 856	2 286 360	1 840 905	1 882 640	1 882 640
Nombre de brins par face	4	5,5	4,8	8,9	6,9
Nombre d'arêtes fictives	0	1 557	477	103 729	58 824
Espace mémoire en méga-octets	308	203	175	244	217
Temps de calcul en secondes	577	563	562	1097	890

TAB. 5.5 – Cerveau2, 256 × 256 × 72, 14627 régions.

	Niveau 1	Niveau 2	Niveau 3	Niveau 4	Niveau 5
Nombre de brins	6 222 152	2 899 120	2 437 356	1 003 178	566 782
Nombre de sommets géométriques	1 584 064	984 246	753 364	658 659	658 659
Nombre de brins par face	4	5,8	4,9	11,3	6,4
Nombre d'arêtes fictives	0	3 230	468	30 291	13 286
Espace mémoire en méga-octets	178	85	71	87	71
Temps de calcul en secondes	104	94	95	197	138

TAB. 5.6 – Cerveau3, 256 × 256 × 83, 17995 régions.

	Niveau 1	Niveau 2	Niveau 3	Niveau 4	Niveau 5
Nombre de brins	7 758 816	4 218 088	3 549 748	1 604 840	1 041 050
Nombre de sommets géométriques	1 961 598	1 427 914	1 093 744	1 006 657	1 006 657
Nombre de brins par face	4	5,9	5	13,2	8,6
Nombre d'arêtes fictives	0	1 072	404	74 146	38 068
Espace mémoire en méga-octets	222	124	103	133	113
Temps de calcul en secondes	125	121	121	623	510

TAB. 5.7 – Failles, 512 × 45 × 121, 2 régions.

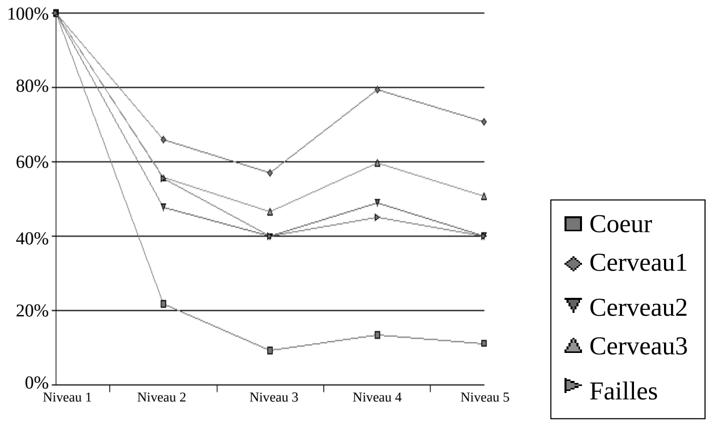
	Niveau 1	Niveau 2	Niveau 3	Niveau 4	Niveau 5
Nombre de brins	8 738 672	4 710 036	3 434 058	1 073 686	706 442
Nombre de sommets géométriques	2 218 060	1 734 460	1 096 471	836 695	836 695
Nombre de brins par face	4	9,6	6,9	12,4	8,2
Nombre d'arêtes fictives	0	28 387	28 387	165 131	93 595
Espace mémoire en méga-octets	250	139	100	112	99
Temps de calcul en secondes	65	47	49	532	421

Cela permet d'étudier pour les trois premiers niveaux de carte le nombre de sommets topologiques, car pour ces niveaux chaque sommet topologique correspond à un sommet géométrique. Mais cela permet également de vérifier que les niveaux 4 et 5 représentent certains de ces sommets de manière redondante. Par exemple, pour l'image du cœur, nous voyons que le niveau 3 représente 12 183 sommets géométriques alors que les niveaux 4 et 5 en représentent chacun 12 928. Ce nombre devrait normalement rester constant pour ces trois niveaux étant donné qu'ils représentent la même géométrie. Cela provient du plongement face utilisé, où les sommets au bord de ces faces sont représentés de manière redondante. Mais nous voyons que ce nombre de sommets représenté « en trop » n'est pas très important et la perte en espace mémoire est finalement assez faible.

Les deux lignes suivantes donnent le nombre moyen de brins par face topologique, et le nombre d'arêtes fictives. Ces nombres sont intéressants afin d'estimer la complexité de l'algorithme optimal qui parcourt une face afin de savoir si sa fusion va entraîner une déconnexion ou non. Pour le niveau 1, chaque face est forcément composée de 4 brins étant donné qu'aucune fusion de faces n'a encore été effectuée. Ce nombre augmente entre les niveaux 1 et 2 mais aussi entre les niveaux 3 et 4, étant donné que le passage entre ces niveaux effectue des fusions de faces ce qui augmente le nombre moyen de brins par face. Par contre, le passage entre les niveaux 2 et 3 et entre les niveaux 4 et 5 diminue ce nombre étant donné que certaines arêtes sont fusionnées. Pour les cinq images, nous obtenons environ 7 brins par face, ce qui est relativement faible. De ce fait, le test afin de savoir si une fusion va entraîner une déconnexion est effectué en parcourant en moyenne 7 brins, ce qui limite le surcoût dû à cette opération.

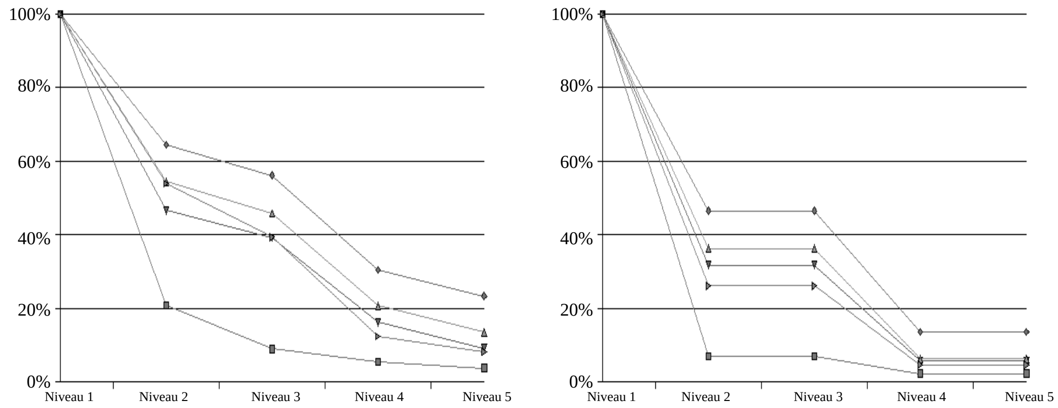
L'avant-dernière ligne donne l'espace mémoire nécessaire à chaque niveau, en tenant compte du plongement ainsi que de l'arbre d'inclusion. Nous constatons une perte en espace mémoire entre les niveaux 3 et 4, dû au changement de plongement. Nous utilisons en effet un plongement 2-G-cartes, qui est relativement coûteux. De plus, ce plongement duplique le plongement des sommets et des arêtes du bord des surfaces. Malgré ça, la carte de niveau 5 ne requiert pas beaucoup d'espace mémoire supplémentaire par rapport à la carte de niveau 3. Enfin le temps d'exécution, donné en dernière ligne, augmente de manière non négligeable entre les niveaux 3 et 4, car nous changeons alors d'algorithme d'extraction. Le niveau 3 est calculé directement de manière optimale, à l'aide des 53 précodes, alors que les niveaux 4 et 5 sont calculés à l'aide de l'algorithme intermédiaire, en utilisant seulement les 8 précodes lignels. De nombreux brins sont donc créés et détruits, ce qui augmente le temps d'exécution, mais nous effectuons également le test de déconnexion de face avant chaque fusion, contrairement à l'algorithme optimal du niveau 3 où il est effectué uniquement pour les trois précodes fc_{16}, fc_{17} et fc_{18}.

Nous récapitulons et regroupons ces résultats sur des schémas qui permettent de mieux visualiser l'évolution de ces caractéristiques en fonction du niveau de simplification. La figure 5.95 montre l'évolution de l'espace mémoire en pourcentage du niveau 1. Nous pouvons vérifier que l'espace mémoire augmente entre les niveaux 3 et 4, à cause du changement de plongement. Mais l'espace mémoire occupé par le niveau 5 est en moyenne du même ordre que celui du niveau 3, malgré le plongement 2-G-carte et les sommets représentés de manière redondante. La figure 5.96.a montre l'évolution du nombre de brins en pourcentage du niveau 1 et la figure 5.96.b l'évolution du nombre de faces, toujours en pourcentage du niveau 1. Nous remarquons sur cette figure que le nombre de faces reste constant entre les



a. Espace mémoire. b. Légende.

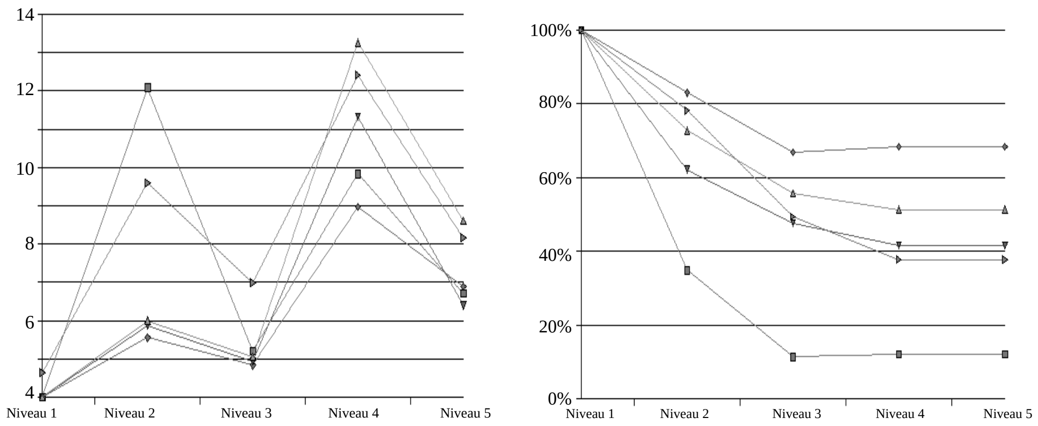
Fig. 5.95 – Évolution de l'espace mémoire en pourcentage du niveau 1.



a. Nombre de brins. b. Nombre de faces.

Fig. 5.96 – Évolution du nombre de brins et du nombre de faces en pourcentage du niveau 1.

niveaux 2 et 3 et entre les niveaux 4 et 5, étant donné que seules des arêtes sont fusionnées entre ces niveaux. La figure 5.97.a présente l'évolution du nombre moyen de brins par face. Ce nombre est le plus élevé pour le niveau 4, environ de 11 brins par face, mais diminue ensuite pour le niveau 5, pour être en moyenne d'environ 7 brins par face. La figure 5.97.b présente l'évolution du nombre de sommets géométriques en pourcentage du niveau 1. Nous vérifions que ce nombre augmente légèrement entre les niveaux 3 et 4 étant donné que nous changeons de plongement, et représentons les sommets se trouvant au bord des faces de manière redondante. Mais cette augmentation est assez faible, en regard du nombre total de sommets représentés.



a. Nombre moyen de brins par face. b. Nombre de sommets géométriques.

FIG. 5.97 – Évolution du nombre moyen de brins par face et du nombre de sommets géométriques.

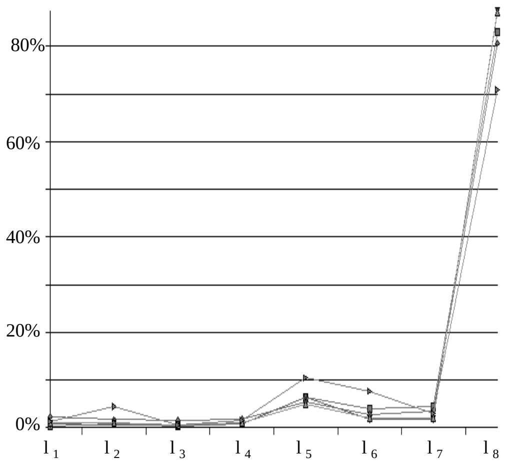
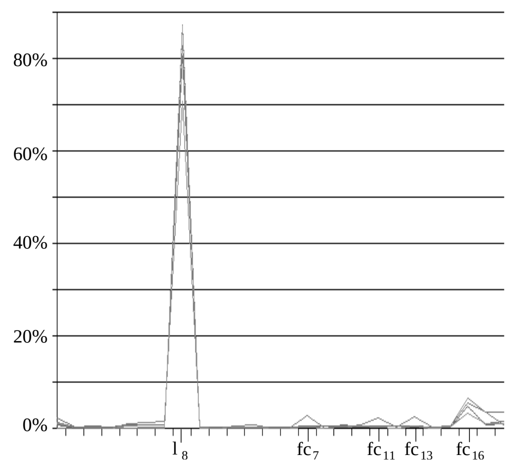
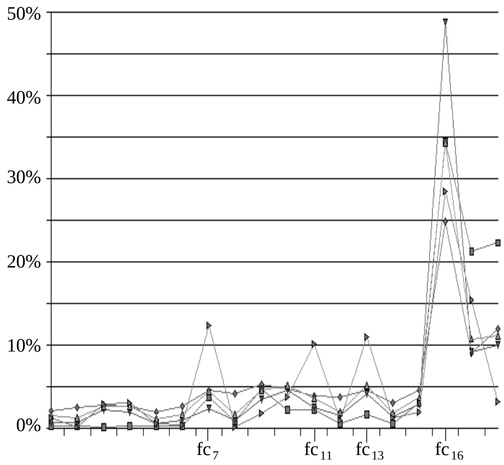


FIG. 5.98 – Nombre de précodes lignels utilisés lors de l'extraction de la carte de niveau 1, en pourcentage du nombre total de précodes.

Nous avons également étudié le nombre de précodes utilisés lors de l'extraction des trois premiers niveaux de simplification. Cela permet de voir quels sont les précodes les plus fréquents. Ce sont ces précodes qu'il va falloir optimiser au maximum afin d'améliorer le temps d'extraction. Nous avons effectué cette étude uniquement pour les trois premiers niveaux étant donné que les deux derniers sont calculés à l'aide de l'algorithme intermédiaire. La figure 5.98 montre le nombre de précodes utilisés pour l'extraction de la carte lignel, en pourcentage du nombre total de précodes dans l'image. Nous voyons que, pour les 5 images utilisées pour nos tests, le précode l_8 apparaît dans environ 80% des précodes



a. En pourcentage du nombre total de précodes.



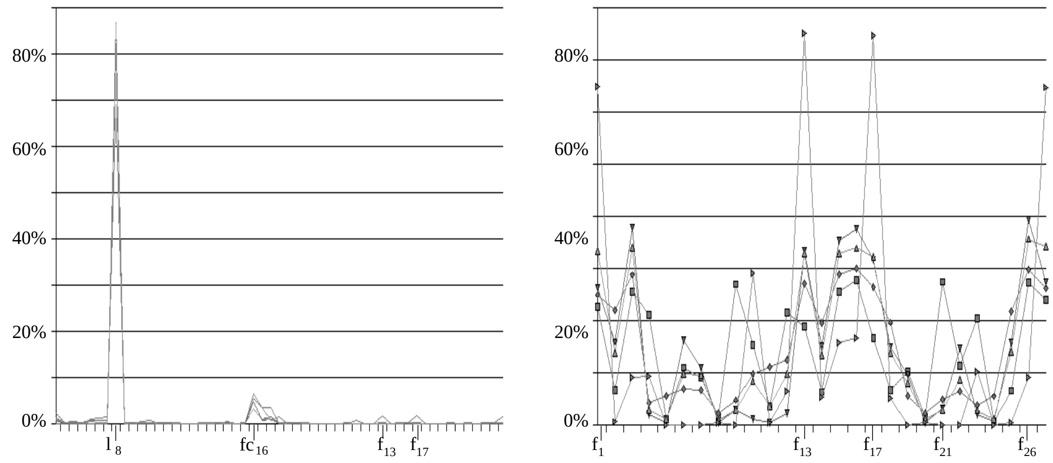
b. En pourcentage du nombre de précodes faces coplanaires.

FIG. 5.99 – Nombre de précodes faces coplanaires utilisés lors de l'extraction de la carte de niveau 2.

totaux de l'image. En effet, les images étant segmentées en régions, la majorité des précodes se trouvent au milieu d'une région, et sont donc traités par le précode l_8. Cela explique la perte de temps d'exécution pour les niveaux 4 et 5, étant donné que le traitement du précode l_8 effectue des tests afin de savoir si des faces ou des arêtes doivent être fusionnées, ce qui n'est pas le cas pour l'algorithme optimal.

La figure 5.99.a montre le nombre de précodes utilisés pour l'extraction de la carte lignel, en pourcentage du nombre total de précodes dans l'image. Comme pour le niveau 1, le précode l_8 apparaît dans environ 80% des cas. Les autres pourcentages sont de ce fait tous assez faibles. Les seuls qui ressortent sur les cinq images sont les précodes fc_{16}, fc_{17} et fc_{18}. Ce sont les trois précodes pour lesquels il faut tester la déconnexion de face. Ces précodes apparaissent plus que les autres car ce sont ceux ayant le plus de voxels de même région. Il est donc normal de les retrouver en plus grand nombre que les précodes ayant beaucoup de voxels différents. Afin de mieux voir les précodes utilisés pour extraire la carte de niveau 2, la figure 5.99.b montre le pourcentage d'apparition de ces précodes par rapport au nombre total de précodes faces coplanaires. Cette figure confirme l'apparition plus fréquente des précodes fc_{16}, fc_{17} et fc_{18}. Les précodes fc_7, fc_{11} et fc_{13} ont des pourcentages supérieurs aux autres, mais uniquement pour l'image *failles*. Cela n'est donc pas vraiment représentatif. Il faudrait effectuer des tests plus nombreux afin d'obtenir une moyenne sur ces pourcentages d'apparition qui soit représentative.

La figure 5.100.a montre le nombre de précodes utilisés pour l'extraction de la carte de niveau 3. Le précode l_8 apparaît toujours dans 80% des cas. Les seuls précodes un peu plus utilisés sont les précodes fc_{16}, fc_{17} et fc_{18} pour la fusion de faces coplanaires, et les précodes f_{13}, f_{17} pour la fusion d'arêtes alignées. Mais ces deux derniers précodes ressortent principalement pour l'image *failles* et pas vraiment pour les autres images. Pour le vérifier, la figure 5.100.b présente le pourcentage d'apparition des précodes frontières par



a. En pourcentage du nombre total de précodes.

b. En pourcentage du nombre de précodes frontières.

FIG. 5.100 – Nombre de précodes frontières utilisés lors de l'extraction de la carte de niveau 3.

rapport au nombre total de ces précodes. Nous voyons sur cette figure que les courbes sont très irrégulières, et même si certains précodes apparaissent plus souvent que d'autres ce n'est pas forcément pour toutes les images.

Ces différences d'apparitions sont assez faibles. Finalement, ce qui ressort de cette étude est que, pour extraire la carte de niveau 3, environ 85% des précodes utilisés sont des précodes lignels, avec 80% uniquement pour le précode l_8. Les précodes faces coplanaires représentent environ 10% des précodes totaux, et les précodes frontières 5%. Cela montre que les précodes les plus importants sont finalement les précodes lignels, et principalement le précode l_8. Cela relativise un peu l'impact de l'algorithme optimal et de l'utilisation des précodes pour calculer la carte de niveau 5. Malgré cela, les résultats d'extraction des cartes de niveaux 4 et 5 montrent que la perte en temps d'exécution est non négligeable. En effet, l'utilisation de l'algorithme intermédiaire entraîne des tests afin d'effectuer toutes les fusions nécessaires, mais également des tests de déconnexion. Ces tests sont effectués pour tous les précodes, y compris le précode l_8. C'est pour ce précode que la perte de temps est alors la plus conséquente. L'algorithme optimal se justifie non pas afin de traiter de manière optimale tout les précodes, étant donné le faible pourcentage d'apparition de la majorité d'entre eux, mais car il évite des traitements non optimaux pour le précode l_8 et en ce sens diminue la complexité de l'extraction. En effet, seule l'implantation complète des 5 niveaux de simplification permet, lors du calcul de la carte topologique, de traiter le précode l_8 de manière optimale sans aucun test supplémentaire. Dans ce cas, le gain en temps d'exécution est conséquent étant donné le fort pourcentage d'apparition de ce précode.

5.9 Conclusion

Dans ce chapitre, nous avons défini la carte topologique en dimension 3. Ce modèle combinatoire permet de représenter les images 3d segmentées en régions de manière minimale. Il représente toute la topologie des régions de l'images, (adjacence, incidence, subdivisions...) mais également leur géométrie. Il est stable par rotation, translation et homothétie, c'est-à-dire que deux images isomorphes pour ces transformations auront la même carte topologique, malgré des géométries différentes.

Afin de définir ce modèle, nous avons étendu la notion de niveau de simplification, introduite en dimension 2. Nous avons alors obtenu 5 niveaux de simplification, définis dans un premier temps de manière assez simple. Nous avons ensuite étudié les problèmes que posaient ces premières définitions. Nous avons rencontré un problème de déconnexion de volume, résolu par l'adjonction d'un arbre d'inclusion des régions, de manière similaire à la dimension 2. Le problème de déconnexion de face a été plus délicat à résoudre. Pour cela, nous avons conservé des arêtes fictives pour que chaque face de la carte soit homéomorphe à un disque topologique.

Ces arêtes fictives permettent de régler le problème de déconnexion de face. Mais elles jouent un rôle différent des arêtes réelles. Nous avons donc étudié précisément leur rôle, les problèmes que leur conservation posaient, et la manière de les régler. Ce point est très important, car c'est la gestion adéquate de ces arêtes qui nous permet d'obtenir la représentation minimale. Nous avons montré, sur plusieurs exemples, comment ces arêtes fictives intervenaient, principalement pour les objets représentés uniquement par une face frontière fermée. En effet, c'est pour ces objets que la gestion des arêtes fictives est la plus importante, étant donné qu'ils sont représentés uniquement par ce type d'arêtes.

Nous avons ensuite étudié la manière de construire ce modèle à partir d'images 3d segmentées. Un premier algorithme naïf peut être défini par application directe des définitions des différents niveaux de simplification. Cet algorithme a l'avantage d'être simple à mettre en œuvre, mais a une complexité quadratique en le nombre de brins de l'image. De plus, il requiert un espace mémoire beaucoup plus important que celui nécessaire à la carte topologique.

Nous avons donc étudié un deuxième algorithme d'extraction, basée sur la notion de précodes. Cet algorithme est optimal, car il crée directement le bon nombre de brins, en utilisant un nombre minimal d'opérations. Sa complexité est linéaire en le nombre de brins de l'image, malgré la constante due au parcours de faces afin de savoir si une fusion va entraîner une déconnexion. Nous avons étudié précisément les précodes à traiter pour chaque niveau de simplification. Cette approche nous a permis de factoriser les précodes similaires, et d'obtenir au final seulement 379 précodes à traiter, au lieu des 4140 existants. De plus, nous avons présenté la notion de précodes isomorphes par rotation, qui permet de factoriser davantage de cas, sans perte en efficacité. Avec cette deuxième factorisation, nous devons définir « seulement » 129 traitements différents. Même si ce nombre est encore important, il est relativement faible en comparaison au nombre total de cas. Enfin, nous avons présenté un troisième algorithme d'extraction, qui est une solution intermédiaire entre les deux premiers algorithmes. Il autorise une implantation assez rapide et simple de l'algorithme d'extraction, tout en restant d'une complexité linéaire. De plus, cet algorithme peut être plus ou moins optimisé en incorporant certains traitements de précodes.

Nous envisageons maintenant la définition de la carte topologique en dimension n. Cela devrait pouvoir se réaliser sans trop de problème, en étendant les solutions proposées en dimension 2 et 3. Nous savons définir les $2n - 1$ niveaux de simplification, en mettant dans un premier temps de côté les problèmes de déconnexion. Nous réglons le problème de déconnexion en nd en ajoutant un arbre d'inclusion des régions. Pour les déconnexions de dimensions $n - 1$ à 2, il est facile de conserver des éléments fictifs permettant d'avoir uniquement des i-cellules connexes. Enfin, il faut s'intéresser à la gestion particulière de ces éléments fictifs et c'est là que se trouvent les difficultés éventuelles. Intuitivement, cela devrait ressembler à la gestion des arêtes fictives en 3d. Il s'agit de « pousser » les éléments fictifs afin qu'ils n'empêchent pas la fusion d'autres éléments. Il faut s'intéresser à la manière dont ces éléments sont « poussés », et aux conditions pour lesquelles c'est possible. De plus, après avoir donné ces définitions, nous obtenons immédiatement l'algorithme d'extraction naïf. L'algorithme optimal semble difficilement envisageable de par le nombre important de cas à traiter. Mais l'algorithme intermédiaire pourra être une bonne solution afin de limiter l'espace mémoire nécessaire et conserver une complexité linéaire.