Tétraèdres en Diamants : Structure de Données Compacte pour Maillages Tétraèdriques

Stage de M2 MPRI (Master Parisien de Recherche en Informatique)

Gabriel Beauplet [*] Luca Castelli Aleardi [†] Olivier Devilliers [‡]

13 août 2019

1 Résumé

De nombreuses structures de données compactes ont été développées pour les triangulations 2D mais très peu pour les tétraèdrisations 3D. Nous introduisons dans ce rapport, une nouvelle structure de donneés compacte pour les tétraèdrisations. Nous nous concentrons seulement sur les informations de connectivité plutôt que sur les informations géométriques (les coordonnées des sommets). L'idée principale est de regrouper les tétraèdres en diamants de telle manière à économiser les références d'adjacence entre deux tétraèdres du même diamant. Nous définissons un diamant comme un ensemble de tétraèdres formant un cycle autour d'une arête. Notre structure de données permet de représenter la connectivité d'un maillage en utilisant en moyenne 2.4 références par tétraèdre tout en permettant l'exécution de requêtes simples. Nous présentons des résultats pratiques de notre structure de donneés ainsi que des résultats plus théoriques.

2 Introduction

Un maillage représente un domaine géometrique en le discretisant en formes simples. Les maillages permettent de représenter des objets géométriques en 1, 2 ou 3D à des fins scientifiques ou industrielles par exemple. En 2D, les maillages representent des surfaces et sont constitués de polygones (triangles, carrés...) reliés deux à deux par une arête. En 3D, les maillages représentent des volumes à l'aide de polyèdres (tétraèdres, pyramides...) partageant une face commune. Les maillages sont très utilisés pour la visualisation de volume, calculs de solutions pour des équations aux dérivées partielles... Cependant, les maillages sont des structures complexes qui peuvent devenir très volumineuses et dont on essaye de réduire la taille mémoire.
La compression de données est omniprésente en informatique, avec des formats compressés génériques comme *gzip* mais aussi dédiés comme *mp3* pour les fichiers audios. Ce besoin de compresser les données est grandissant car de plus en plus de fichiers sont stockés à distance sur des serveurs et la moindre économie de stockage a d'importantes répercussions. Néanmoins, sous certains formats compressés, les données originales deviennent inutilisables (ex : *rar*). Cela pose problème quand l'on souhaite accéder aux données sans passer par l'étape de décompression.
Une structure de données est une manière d'organiser les données pour faciliter leur traitement. Les listes, arbres et graphes sont des exemples de structures de données. Leur but n'est pas de limiter l'usage mémoire mais seulement de faciliter l'utilisation des données. Ainsi, pour certains formats volumineux, des structures de données compactes ont été inventées. Ce sont des structures de données compressées, c'est à dire des structures de données dont l'utilisation de la mémoire est limitée.
Si l'on revient aux maillages, les algorithmes de compression limitent au maximum l'usage

[*]LIX, Ecole Polytechnique, Palaiseau, France
[†]LIX, Ecole Polytechnique, Palaiseau, France
[‡]INRIA, CNRS, Loria, Université de Lorraine, France

mémoire du maillage et il devient inutilisable sous forme compressé tandis que la structure de données pré-traite le maillage en réduisant l'usage mémoire mais celui-ci reste utilisable. Les maillages en deux dimensions sont majoritairement utilisés car ils sont plus léger, et permettent de représenter implicitement des volumes (la frontière). Par conséquent, de nombreuses structures de données compactes ont été créees afin de faciliter leur utilisation. Les maillages 3D étant beaucoup moins utilisés pour l'instant, peu de structure de données compactes leur sont dévolues. Néanmoins, leur utilisation croissante incite à procéder de même. La suite de ce rapport sera principalement consacrée aux structures de données compactes pour maillages 3D tétraèdriques.

De manière générale, un structure de données pour maillage stocke trois types d'informations :
- La géometrie, c'est à dire les positions des sommets
- La connectivité, les relations d'adjacence entre les tétraèdres
- Des attributs (par sommet, par arête, par face, par tétraèdre)

La structure de données doit supporter des requêtes simples :
- Quels sont les sommets de la ième face ?
- Quel est le degré du ième sommet ?
- Quelles sont les tétraèdres adjacents au ième tétraèdre ?
- ...

Par ailleurs, suivant l'utilisation ciblée, la structure de données devra être en mesure de satisfaire des opérations de modification :
- Ajouter/Enlever un sommet
- Ajouter/Enlever/Séparer un tetra

On évalue une telle structure de données en analysant le temps nécessaire à sa construction, à l'exécution d'une requête, à une opération de modification et surtout en observant la quantité de stockage nécessaire pour l'utiliser.

Contributions. Dans ce rapport, nous présentons une structure de données permettant de représenter la connectivité d'un maillage tétraèdrique en utilisant en moyenne 2.4 références par tétraèdre. Notre structure s'appelle **Tétraèdres en Diamants**. Elle permet l'accès au ième sommet, au ième tétrahèdre en temps constant et à l'hypersphère d'un sommet en temps proportionnel au degré du sommet. Son implémentation est simple et l'utilisation d'un tableau d'entiers afin de représenter les références permet une interopérabilité entre les languages de programmation.
Nous allons d'abord définir les principaux termes utilisés et rapeller les algorithmes de compression et structures de données déjà développés en deux et trois dimensions. Puis nous présenterons le fonctionnement de notre structure ainsi que les requêtes possibles.

2.1 Définitions

Simplexe. Un simplex σ^p de dimension p est l'enveloppe convexe de $p+1$ points $\{v_0, v_1, ...v_p\}$, où $v_i \in R^n$ et les vecteurs $v_1 - v_0, v_2 - v_0...$ sont linéairement indépendants. Les simplexes de dimensions 0, 1, 2 et 3 sont respectivement les sommets, arêtes, triangles et tétraèdres.

Complexe simplicial. Un complexe simplicial est un ensemble K de simplexes d'un espace affine tel que toutes les faces de chaque simplexe de K appartiennent aussi à K et si deux simplexes σ et τ de K sont adjacents alors $\sigma \cap \tau \neq \emptyset$.

Variété. Une variété topologique (manifold en anglais) M de dimension n est un espace topologique connexe séparé localement homéomorphe à un ouvert de $\mathbb{R}^n$. C'est à dire que chaque point de M admet un voisinage homéomorphe à un ouvert de $\mathbb{R}^n$.

Variété à bord. Une variété à bord est un sous-espace topologique dont les points admettent un voisinage homéomorphe à $\mathbb{R}^n$ (les points intérieurs) ou un voisinage homéomorphe à $\mathbb{R}^{n-1}\text{x}\mathbb{R}^+$ (les points bordants). L'ensemble des points bordants constitue le bord de la variété.

Frontière. Les (k-1)-simplexes d'une k-variété M qui sont incidents à seulement un k-simplexe sont les simplexes frontières. L'ensemble des simplexes frontières est dénoté ∂M.

Maillage. Un maillage est un complexe simplicial représentant un objet géométrique. Il a la même dimension que l'objet qu'il représente. Ainsi, pour tout objet en 1, 2 ou 3D, les maillages respectifs seront en dimensions 1, 2 ou 3. Dans un maillage de dimension d, les simplexes de dimensions (d-1) sont appelées des facettes. Ainsi, les facettes d'un tétraèdre sont ses faces, les facettes d'une face sont ses arêtes et les facettes d'une arête sont ses sommets.

Le degré. Le degré d'un k-simplexe est le nombre de (k+1)-simplexes adjacents. Ainsi le degré d'un sommet est le nombre d'arêtes adjacentes et le degré d'une arête est le nombre de faces adjacentes.

Etoile. L'étoile d'un sommet est l'ensemble des k-simplexes adjacents à un sommet. C'est l'ensemble des triangles (resp. tétraèdres) adjacents à un sommet dans le cas surfacique (resp. volumique). Dans ce dernier cas, on appelle cela l'hypersphère.

2.2 Combinatoire

En mathématique et en optimisation combinatoire, la caracéristique d'Euler χ est un invariant topologique décrivant la forme d'un objet géométrique.

$$\chi = \sum_{i=0} (-1)^i \, |dim(H_i)| = 2 - 2g \tag{1}$$

— H_i est l'ensemble des faces de dimension i
— g est le genre (le nombre de trou de l'objet étudié)

Ainsi, pour un polytope de dimension 4, la formule d'Euler devient :

$$\chi = |V| - |E| + |F| - |T| \tag{2}$$

— V est l'ensemble des sommets
— E est l'ensemble des arêtes
— F est l'ensemble des faces (ie. polygone)
— T est l'ensemble des tétraèdres

Sachant que chaque face qui n'est pas sur les bords du volume est partagée par deux tétraèdres, alors :

$$|F| \simeq 2|T| \tag{3}$$

Ainsi :

$$\chi \simeq |V| - |E| + |T| \tag{4}$$

Il y a donc autant d'arêtes que de tétraèdres et sommets réunis.

3 Etat de l'art

Les maillages sont la pluspart du temps stockés sous forme indexés. Dans un premiers temps, on énumère pour chaque sommet ses coordonnées géométriques. Puis pour chaque face (resp. tétraèdre), les indices de ses 3 (resp. 4) sommets. D'autres attributs peuvent être stockés (normales, couleurs...) mais nous n'en discuterons pas ici. Les formats indexés ne sont pas les formats les plus concis pour sauvegarder des maillages. En effet, la connectivité occupe une place très importante. Tandis que dans les maillages 2D, le degré moyen des sommets est de 6, il est de 22 dans les maillages tétraèdriques. Par conséquent, dans ce type de format, un sommet apparaitra dans 22 tétraèdres différents.

La mesure utilisée pour évaluer la qualité d'une compression est le nombre de bits par sommets (bits per vertex ou bpv) tandis que la mesure utilisée pour évaluer la qualité d'une structure de données compacte est le nombre de références par triangle (resp. tétraèdre) rpt pour les maillages surfaciques (resp. volumiques).
On peut compresser un maillage en le simplifiant (supprimer des sommets...), en encodant la géométrie et/ou la connectivité du maillage. Nous ne nous intéresserons dans cette partie qu'à la compression de la connectivité puisque c'est la plus gourmande en mémoire. Par ailleurs, nous détaillerons d'abord le travail effectué en 2D puis en 3D. Bien que notre travail soit uniquement centré sur les maillages 3D, l'essentiel des travaux effectués jusqu'à présent est en 2D.

3.1 Maillages 2D

3.1.1 Compression

Bande de triangle. Les bandes de triangles ("triangles strips") et les eventails de triangles ("triangle fans") sont des représentations utilisées pour transférer les maillages de la mémoire centrale du PC vers la mémoire du GPU. Une bande de triangle est une séquence de sommet où chaque nouveau sommet défini un triangle avec les deux précedent sommets. En ce qui concerne les bande de triangles, le but est de trouver de très longues bandes. Si les bande de triangles sont suffisament longue alors, cette representation permet de passer de 3N references aux sommets à N+2. L'algorithme de Deering utilise ces bandes de triangles et utilise entre 3.3 et 9.8 bpv [1].

Traversée de triangles. La Cut border Machine [2] est un algorithme de Gumhold qui encode la conectivité en parcourant le graphe en largeur. L'algorithme étend la frontière défini par un triangle initial en travesant itérativement des triangles adjacents. Sept symboles sont utilisés pour préciser si la frontière a été étendu en insérant un nouveau sommets, si la frontière a été séparée ou si deux frontière se sont jointes. Le schéma peut compresser des variétés avec 4 bpv. En revanche, ce résultat est seulement valide pour des maillages réguliers. En effet, quand une jointure est effectuée, un décallage doit etre fait pour désigner les sommets concernés. Par conséquent, il n'y a pas de borne supérieure garantie pour la compression avec cet algorithme.
L'algorithme EdgeBreaker de Rossignac [4] traverse lui aussi le graphe d'un triangle adjacent à un autre et enregistre la connectivité d'un maillage en produisant les sympboles C,L,R,E,S. Cependant, il garantit un cout de 4bpv.

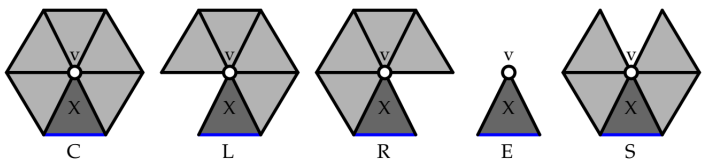

FIGURE 1 – Les cinq configurations dans l'algorithme Edgebreaker. v est le sommet central de la configuration et X est le triangle cible

Codage de la Valence. Une manière de décrire la connectivités de sommets est à travers leurs valence. Le premier travail sur la valence des sommets est le travail de Touma et Gotsman [6]. Le principe est de considérer la frontière d'un triangle initial et de l'étendre en ajoutant itérativement de nouveaux sommets. La connectivité est encodée en utilisant la valence des nouveaux sommets (concentrée autour de 6). Ainsi, La liste de valence des sommets peut être efficacement compressée par un encodeur d'entropie (2.3 bpv). C'est toujours aujourd'hui l'une des méthodes les plus efficace.

3.1.2 Structure de données compacte

Plusieurs structures de données permettent une utilisation très facile des maillages et se focalisent sur l'utilisation des arêtes du graphe. C'est le cas d'Half-Edge, Winged-Edge [11] et Quad-Edge

qui stockent $19n$ références (soit 9.5 rpt). Elles permettent facilement de naviguer dans le maillage et opèrent des requêtes d'adjacence en temps constant. Cependant, elles occupent trop de place pour être considérées comme compactes.
La Corner Table (CT) est à la base de plusieurs structures de données. Elle utilise deux listes V et O de $3|F|$ entiers chacune où F est l'ensemble des faces maillage. La table V stocke les incidences triangle/sommet tel-que les 3 sommets bordant un triangle t sont consécutifs (V[3t],V[3t+1],V[3t+2]) et sont listés dans un ordre consistent avec le maillage. Ainsi, V[c] représente un coin c associé avec une face f et un sommet. La table O stocke la référence entière du coin oposé. Le coin opposé o(c) au coin c est un coin dans un triangle adjacent qui partage la même arête opposée.

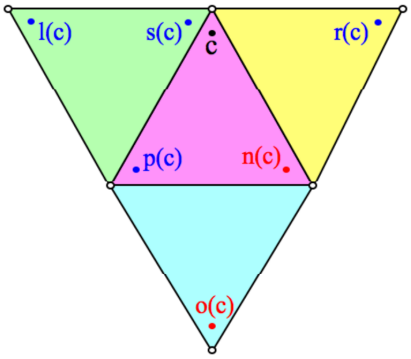

FIGURE 2 – Les opérateurs utilisant les coins pour un maillage triangulaire

VOT. La structure de données VOT (Vertex Opposite Table) est la première structure de données à utiliser cette 'Corner Table'. Elle permet une répresentation simple et efficace des maillages avec 6 références par triangle (3 références pour les sommets dans la table V et 3 références pour les coins dans la table O).
SOT. Développée par Rossignac [9], c'est une amélioration de VOT où la table O est réordonnée et la table V supprimée. Néanmoins, l'accès au coin d'un sommet et à l'étoile d'un sommet sont toujours en temps constant. Cette dernière structure de données utilise 3 rpt en moyenne.

Structure de données	Taille mémoire	Temps de navigation	Accès au sommet	Dynamique
Basées sur les arêtes (Half-edge, Quad-edge, Winged-edge)	18n+n	O(1)	O(1)	oui
Basées sur les triangles	13n	O(1)	O(1)	oui
Corner table	13n	O(1)	O(1)	oui
2D catalog [13]	7.67n	O(1)	O(1)	oui
Star vertices [12]	7n	O(d)	O(1)	non
SOT [9]	6n	O(1)	O(d)	non
SQUAD [14]	$(4+\epsilon)$n	O(1)	O(d)	non

TABLE 1 – Performances des structures de données pour maillages 2D

3.2 Maillages 3D

3.2.1 Compression

Grow&Fold. L'algorithme Grow&Fold [7] combine les idées de l'algorithme Topological Surgery [5] de Taubin et EdgeBreaker [4] de Rossignac. Il construit un arbre couvrant de tétraèdres et un folding string. L'arbre couvrant débute à une face arbitraire et grandit en ajoutant des tétraèdres aux faces externes de l'actuel arbre couvrant. Pour chaque ajout de tétraèdre, 3 bits encodent si d'autres tétraèdres seront attachés aux 3 faces extérieures de ce tétraèdre. Le folding string contient pour chaque triangle externe de l'arbre couvrant un code sur 2 bits permettant de retrouver les relations d'incidences absentes de l'arbre couvrant. L'arbre couvrant contient $|T|$ tétraèdres et il y a $2|T|$ faces externes. Par conséquent, l'usage mémoire est de 7 bpt.

Cut Border Machine. La Cut Border Machine pour les maillages volumiques [2] est directement inspirée de celle pour les maillages surfaciques de Gumhold [3]. L'algorithme étend la frontière défini par une face initiale en traversant des tétraèdres adjacents. Dix symboles sont utilisés pour décrire l'entourage de la frontière lors de l'ajout d'un nouveau sommet pour la construction d'un tétraèdre. Leur algorithme permet de compresser les maillages tétraèdriques en utilisant 2.4 bpt et s'adapte aux non-variétés.

3.2.2 Structure de données compacte

VOT. La Corner Table a été adapté par Rossignac aux maillages tétraèdriques (VOT). Elle demande 8 rpt (4 pour les sommets et 4 pour les coins opposés). Un index dans ces listes identifie un coin particulier à un tétraèdre. Ainsi, les tables O et V ont toutes les deux $4|T|$ entrées. Les coins de chaque tetrahèdre sont consécutifs dans les deux listes (les quatres coins du ième tetra sont stockées aux entrées 4i+j, where j = 0,1,2,3) et sont listés dans un ordre consistent avec l'orientation du tétrahèdre (les sommets des coins j=1,2,3 apparaissent dans le sens inverse des aiguilles d'une montre depuis le sommet du coin 0).
Bande de Triangles. Weiler et al. [8] encode les tétrahèdres en bandes. L'inclusion d'une petite quantité d'informations d'adjacence leur permet d'acceder aux faces voisines en temps constant. Leur algorithme stocke en moyenne 5.1 rpt.
SOT. La dernière structure de données développée est SOT [9] par Rossignac. Elle améliore sa première structure de données VOT en triant la table O et en supprimant la table V. La structure de données utilise 4 références et 9 bits par tétraèdre en moyenne et permet l'accès à l'étoile d'un sommet en temps constant.

Structure de données	Taille mémoire	Temps de navigation	Accès au sommet	Dynamique
VOT	8t	O(1)	O(1)	non
Bande de triangles [8]	5.1t	O(1)		non
SOT [9]	4t	O(1)	O(d)	no
Tétraèdre en diamants	$\simeq$ **2.4t**	**O(1)**	**O(1)**	**no**

Table 2 – Performances des structures de données pour maillages 3D

4 Tétraèdres en Diamants

4.1 Le principe

Dans SQUAD [14], les auteurs traversent le graphe en prodondeur afin d'appareiller les triangles deux à deux avec l'un de leurs sommet partagé. Cela leur permet d'avoir 4 références par quad (i.e pair de triangle) et donc d'économiser une référence par triangle. Quant aux triangles non appareillés, ils sont déguisés comme des quads. D'autre part, tout comme SOT, ils utilisent l'ordre des quads tel-que le ième quad soit associé au ième sommet. Dans un maillage 2D, il y a deux fois plus de triangles que de sommets. Par conséquent, le nombre de quads et le nombres de sommets devrait être assez proche.
Notre algorithme s'inspire fortement de SQUAD. Seulement, appareiller les tétraèdres deux à deux permet seulement d'économiser une référence par cube, c'est à dire de passer de 4 références par tétraèdre à 3.
L'idée ici est de regrouper les tétraèdres partageant une même arête. On appelle un tel regroupement **un diamant**. Il combine plusieurs idées :

- Regroupement des tétraèdres comme SQUAD
- Ancrer un sommet avec un diamant comme SOT
- Ordonner les diamants tel que le ième sommet soit au sein du ième diamant comme SOT
- Passage d'un tétraèdre à l'autre en utilisant les faces (et non les coins)

Un diamant est un ensemble de tétraèdres adjacents deux à deux, partageant une arête commune et formant un cycle (fig. 3). Dès lors que l'ensemble des tétraèdres n'est pas cyclique, la figure géométrique n'est plus un diamant (fig. 4).

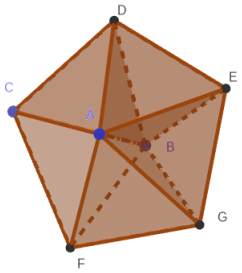

FIGURE 3 – figure
Exemple de diamant contenant 5 tétraèdres et dont l'arête centrale commune est AB

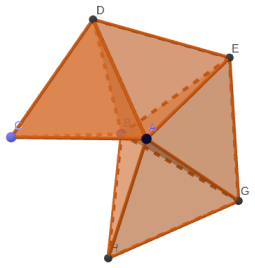

FIGURE 4 – figure
Exemple n'étant pas un diamant car les tétraèdres ne sont pas cycliques (bien que tous les tétraèdres partagent la même arête AB)

Au sein d'un diamant, les tétraèdres sont ordonnés. Ainsi, on peut oublier les références de voisinage entre deux tétraèdres du même diamant. Pour les tétraèdres au sein d'un diamant, seules les références vers des tétraèdres extérieurs sont nécessaires. Un diamant D contenant $|D|$ tétraèdres a $2|D|$ faces extérieures donc $2|D|$ références. Sur la Fig. 3, les faces ABD,ABE,ABG,ABF,ABC n'apparaitront pas dans notre structure car nous savons implicitement que pour passer d'un tétraèdre à l'autre dans ce diamant, nous utilisons une de ces faces.

4.2 Evaluation de notre structure de données

Il est évident que nous devons évaluer notre structure afin de comparer ses performances aux résultats pré-existants. Pour cela, nous avons choisi une dizaine de maillages avec des formes, des tailles, et des tétraèdrisations différentes. La pluspart des maillages ont été réalisés avec la bibliothèque CGAL [10]. Nous ne présentons les résultats de nos algorithmes que sur 6 de ces maillages.

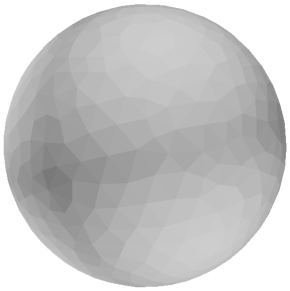

FIGURE 5 – figure
Tétraèdrisation d'une boule

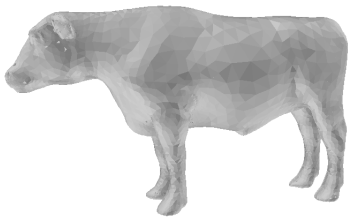

FIGURE 6 – figure
Tétraèdrisation d'une vache

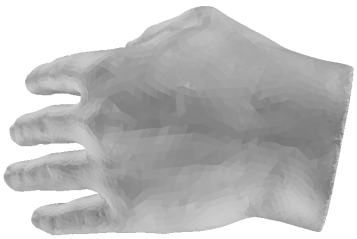

FIGURE 7 – figure
Tétraèdrisation d'une main

FIGURE 8 – figure
Tétraèdrisation d'un crane

Nom de la structure	Nombre de sommets	Nombre d'arêtes	Nombre de tétraèdres	Part de tétraèdres sur les bords	Nombre de tétraèdres par sommet	Nombre d'arêtes par sommet
Boule (B1)	1056	5452	3583	0.45	13.52	10.32
Boule (B2)	15063	101343	83412	0.07	22.15	13.45
Boule (B3)	378532	2647808	2243131	0.05	23.70	13.98
Vache (V1)	30808	182170	134707	0.24	17.48	11.82
Main (M1)	28796	169115	125127	0.24	17.38	11.74
Crane (C1)	37813	217974	156135	0.30	16.51	11.52

TABLE 3 – Caractéristiques des maillages tétraèdriques utilisés pour évaluer notre structure

4.3 Appariement des tétraèdres en diamants

La première étape consiste à regrouper les tétraèdres en diamants. On peut ramener ce problème à un problème d'optimisation dans les graphes. En considérant notre maillage 3D comme un graphe, il s'agit de choisir un ensemble E' d'arêtes tel-que deux arêtes de E' n'appartiennent pas au même tétraèdre. Les arêtes candidates pour appartenir à E' sont toutes les arêtes qui ne sont pas situées sur les bords du maillage. Par exemple, sur Fig. 4, l'arête AB est située sur le bord du maillage et n'est donc pas candidate pour appartenir à E'.
Pour chaque arête selectionnée, le diamant est constitué de tous les tétraèdres possedant cette arête. Tous les tétraèdres n'ayant pas d'arêtes dans E' sont appelés les tétraèdres isolés.
Pour trouver cet ensemble d'arêtes, nous avons essayé plusieurs algorithmes.

4.3.1 Choisir une direction pour chaque sommet

La première méthode consiste à prendre pour chaque sommet, une arête dans une direction pré-définie. Néanmoins, cette méthode a deux inconvénients majeurs : deux arêtes peuvent être choisies et appartenir au même tétraèdre et elle utilise la géometrie du domaine (et donc peut sembler moins générique). Nous avons donc choisi de ne pas utiliser cette méthode.

4.3.2 Choisir l'arête de degré maximum

Le degré d'une arête est le nombre de tétraèdres possédant cette arête. Une heuristique très simple consiste à prendre en priorité les arêtes ayant un degré important. C'est un algorithme glouton dans la mesure où seulement la récompense immédiate nous intéresse. Il est nécessaire de trouver l'arête maximum à chaque itération de l'algorithme. Cela a pour conséquence un temps d'exécution relativement long.

4.3.3 Parcours en largeur des tétraèdres

La troisième approche consiste à parcourir le maillage en largeur. On choisit un tétraèdre au début de l'algorithme puis on regarde pour chacune de ses arêtes si les tétraèdres partageant cette arète forment un diamant et qu'aucun n'appartienne déja à un diamant. Si ces deux conditions sont remplies, on crée un nouveau diamant avec cette arête centrale. Puis on ajoute à la file les tétraèdres adjacents et non visités au tétraèdre choisi. On exécute ainsi cet algorithme tant que la file n'est pas vide.

```
Soit F une file ;
F.ajouter(t) ;
while F n'est pas vide do
    t = F.défiler() ;
    for arête e dans t do
        if e forme un diamant then
            if aucun des tétraèdres ayant e n'appartient à un diamant then
                Creer un diamant avec e comme arête centrale ;
            end
        end
    end
    Marquer t ;
    Ajouter voisins de t non marqués à Q ;
end
```

Algorithm 1: Parcours en profondeur du maillage avec un tétraèdre de départ t

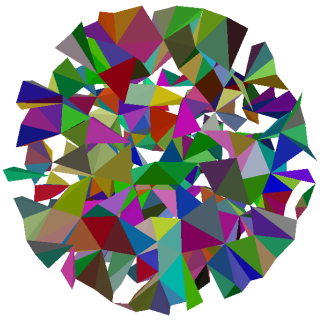

FIGURE 9 – figure
Vue de coupe des tétraèdres isolés après exécution du parcours en largeur pour créer les diamants. Chaque couleur représente un tétraèdre.

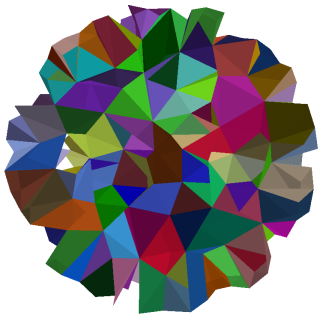

FIGURE 10 – figure
Vue de coupe des diamants après exécution du parcours en largeur pour créer les diamants. Chaque couleur représente un diamant.

Sur Fig. 9, on note une certaine homogénéité des tétraèdres isolés. En analysant plus spécifiquement la concentration des tétraèdres isolés, on remarque sur les Fig. 11 et Fig. 12 qu'ils sont particulièrement situé sur les bords et dans des régions à courbures [1].

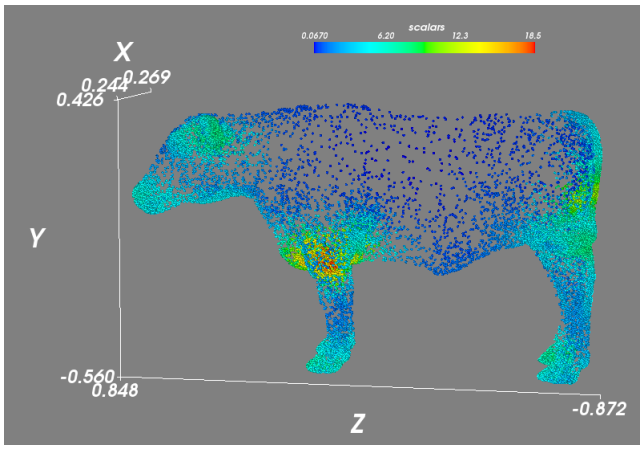

FIGURE 11 – figure
Distribution des tétraèdres isolés. Chaque point représente le barycentre d'un tétraèdre et sa couleur indique la densité (le nombre de tétraèdres isolés à proximité).

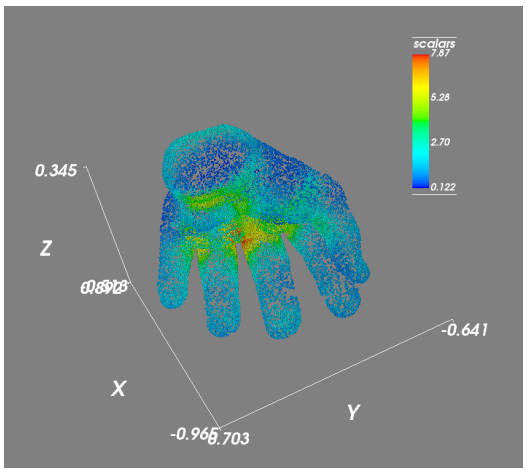

FIGURE 12 – figure
Distribution des tétraèdres isolés. Chaque point représente le barycentre d'un tétraèdre et sa couleur indique la densité (le nombre de tétraèdres isolés à proximité).

1. Afin que la densité ne soit pas biaisée, nous avons soustrait la densité originale du maillage. Le fait qu'une région soit plus densément peuplée en tétraèdres n'influence donc pas le résultat.

Choix du tétraèdre de départ Suivant le premier tétraèdre choisi pour lancer l'algorithme de parcours en largeur (fig. 14), le taux de tétraèdres appareillés dans des diamants varie peu (1% d'écart). Il semble néanmoins plus intéressant de commencer par les régions étroites.

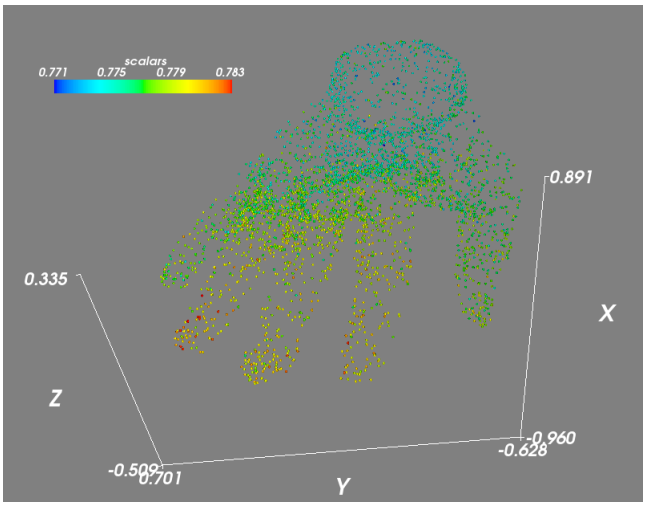

FIGURE 13 – Performance de l'algorithme de parcours en largeur en fonction du tétraèdre de départ.

4.3.4 Optimisation aléatoire

Notre problème s'exprime facilement comme un problème d'optimisation combinatoire en nombres entiers. Malheureusement, la résolution de ces problèmes NP-difficile, ce qui signifie qu'aucun algorithme ne peut trouver une solution optimale en temps polynomial. Néanmoins, on peut utiliser des algorithmes d'optimisation aléatoire afin de trouver une solution approchée. Ce sont des algorithmes très utilisés en pratique qui visitent plus ou moins aléatoirement l'espace des solutions.
Soit f la fonction aléatoire à minimiser, l'idée est de partir d'une solution initiale x et tant que la condition d'arrêt n'est pas remplie, de créer une solution y à partir de x puis de remplacer x par y si f(y)>f(x).
Dans notre cas, une solution est un ensemble d'arêtes. Elle est faisable si pour toute paire d'arêtes de notre solution, aucunes n'appartiennent au même tétraèrdre. On peut donc matérialiser notre solution comme un vecteur de 0 et 1 pour chaque arête du graphe (1 si l'arête appartient à la solution, 0 sinon). Pour calculer la valeur de notre solution, on ajoute pour chaque arête de la solution le nombre de tétraèdre utilisant l'arête et si deux arêtes appartiennent au même tétraèdre alors on inflige une pénalité en soustrayant le nombre de tétraèdres adjacents à ces deux arêtes. L'inconvénient majeur de cet algorithme est sa lenteur. Il permet de trouver des solutions quasi optimales pour des maillages avec quelques milliers d'arêtes en quelques secondes mais ne permet pas de trouver des solutions convenables au dela. Nous avons donc décidé de ne pas retenir cette solution.

4.3.5 Résultats

Pour calculer le nombre de références par tétraèdre (rpt), nous appliquons la formule suivante :

$$\text{rpt} = \frac{2 \cdot \text{Nombre de tétraèdres dans des diamants} + 4 \cdot \text{Nombre de tétraèdres isolés}}{\text{Nombre de tétraèdres total}} \tag{5}$$

	Parcours en largeur			Degré de l'arête		
Nom de la structure	Part des tétraèdres dans des diamants	RPT	Temps (s)	Part des tétraèdres dans des diamants	RPT	Temps (s)
B1	0.76	2.69	0.03	0.81	2.36	0.47
B2	0.80	2.39	1.02	0.85	2.29	497
B3	0.81	2.37	39.79			
V1	0.77	2.44	1.56	0.81	2.36	922
M1	0.77	2.44	1.26	0.83	2.34	794
C1	0.77	2.44	1.48	0.80	2.38	1262

TABLE 4 – Résultats des algorithmes d'appareillage des tétraèdres en diamants

Compte tenu des résultats du tableau 4.3.5, nous avons choisi d'utiliser le parcours en largeur pour appareiller les tétraèdres en diamants. Le parcours en largeur est rapide et permet d'associer en moyenne 76% des tétraèdres en diamants. Il pourrait probablement être amélioré en étant exécuté en parallèle ou en choisissant plus spécifiquement un tétraèdre de départ. Finalement, on remarque aussi que 50% des tétraèdres sont dans des diamants possedant 5 ou 6 tétraèdres (Fig. 14).

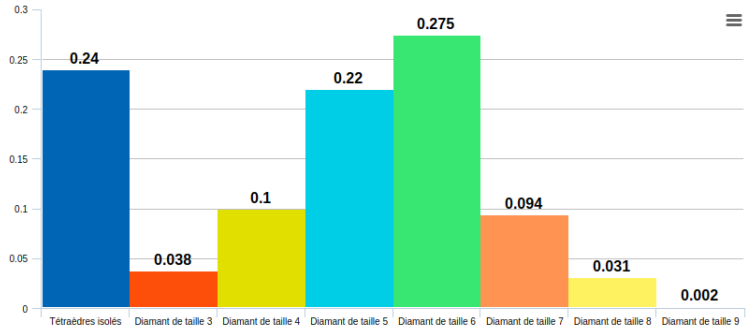

FIGURE 14 – Histograme de la part de tétraèdres parmi les tétraèdres isolés et diamants en utilisant le parcours en largeur

4.4 Choisir l'ancre

Le but de cette section est d'associer chaque sommet à un diamant ou un tétraèdre isolé. Voici les règles pour l'association.

— Un diamant ne peut être associé qu'à un sommet de son arête centrale
— Un tétraèdre isolé peut être associé à n'importe lequel de ses quatres sommets
— On associe en priorité les sommets à des diamants afin de calculer plus rapidement le degré du sommet

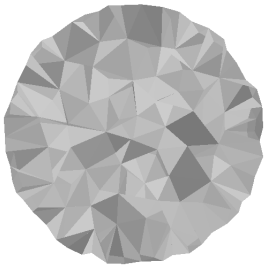

FIGURE 15 – figure
Vue de coupe d'une tétraèdrisation d'une boule

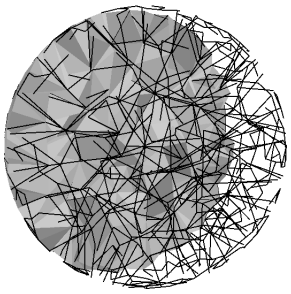

FIGURE 16 – figure
Vue de coupe d'une tétraèdrisation d'une boule avec l'affichage des arêtes centrales de tous les diamants

Les différents cas possibles pour l'association des sommets sont les suivants :

— Le sommet est adjacent à une arête centrale disponible. On associe le sommet à cette arête.
— Le sommet n'est adjacent à aucune arête centrale et il est adjacent à un tétraèdre isolé libre. Alors on associe le sommet à ce tétraèdre libre.
— Le sommet n'est adjacent à aucune arête centrale, n'est pas adjacent à un tétraèdre isolé libre mais est adjacent à un diamant. On "explose" alors le diamant, c'est à dire qu'on fait comme s'il était constitué que de tétraèdres isolés. Sur la Fig. 17, le sommet F n'est pas sur une arête centrale. Si le diamant est déjà associé au sommet A (resp. B) alors on explose le diamant. On peut alors associer le sommet F (Fig. 18) aux tétraèdres ABFG ou ABFC et le sommet A (resp. B) aux tétraèdres ACDB ou ADEB ou ABEG.

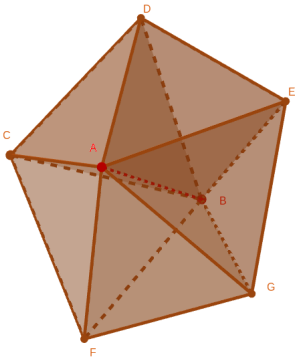

FIGURE 17 – figure
Image d'un diamant contenant 5 tétraèdres dont l'arête centrale est le segment AB et ancré au sommet A

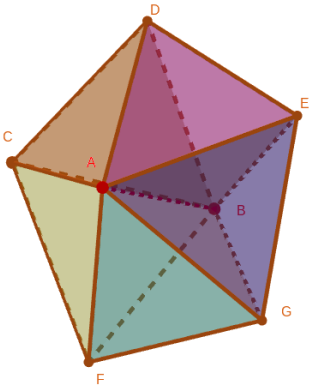

FIGURE 18 – figure
Image de 5 tétraèdres isolés (chaque couleur est un tétraèdre différent)

Nom de la structure	Part de sommets non associés avec des diamants ou tétraèdres isolés	Part de tétraèdres dans des diamants avant choix des ancres	Part de tétraèdres dans des diamants après choix des ancres
B1	0.06	0.76	0.65
B2	8e-4	0.80	0.80
B3	8e-5	0.82	0.82
V1	8e-3	0.77	0.76
M1	7e-3	0.77	0.76
C1	8e-3	0.77	0.76

TABLE 5 – Ancrage des sommets aux diamant/tétraèdres isolés

Afin d'ancrer les sommets aux diamants/tétraèdres isolés, nous appliquons un algorithme glouton affectant en priorité les sommets adjacents à peu d'arêtes centrales et de tétraèdres isolés. On remarque sur le tableau 4.4 qu'en appliquant cet algorithme, une part minime des sommets demeure non ancré. Par ailleurs, on note que la destruction de certains diamants pour associer des sommets à des tétraèdres n'a que peu d'incidence sur les performances tant que le nombre de sommets est important.

4.5 La structure

Désormais, nos diamants sont formés et les sommets sont associés à des diamants (ou des tétraèdres isolés).
Pour rappel, notre structure doit permettre de :
— Accéder au ième tétraèdre en temps constant
— Accéder au ième sommet en temps constant
— Naviguer facilement dans le graphe (entre les tétraèdres)
— Calculer le degré (i.e l'hypersphère) d'un sommet

La structure que nous proposons est un tableau F à une dimension dont la taille est le nombre de faces extérieures des diamants ou tétraèdres isolés. Les faces extérieures sont toutes les faces des tétraèdres isolés et les faces externes des diamants. Le tableau contient des entiers qui sont les indices des faces. Ainsi F[i] indique l'indice dans le tableau de la face adjacente à la ième face. Si la ième face est sur le bord du volume alors F[i]=-1.
Un diamant contenant quatre tétraèdres occupera 8 cellules dans le tableau (car il a 8 faces extérieures) et un tétraèdre isolé en occupera 4. Les diamants et tétraèdres isolés sont ré-ordonnés de tel manière que le ième sommet soit ancré au ième diamant/tétraèdre isolé. Etant donné qu'il y a beaucoup plus de diamants que de sommets, seuls, les $|V|$ premiers diamants/tétraèdres isolés sont ré-ordonnés.
Par ailleurs, pour savoir quand on passe d'un diamant à un autre, nous utilisons des bits de service (1 ou 0) pour chaque face. Une face contient un 1 si c'est la première face d'un diamant/tétraèdre isolé, 0 sinon. Finalement, afin de pouvoir tourner facilement autour d'un sommet, nous utilisons 3 bits de service par face pour représenter la permutation des sommets entre ces deux faces.
Pour résumer, voici notre structure de données :

- Un tableau de tailles |F| où F est l'ensemble des faces exterieures du maillage.
- Un bit de service par face afin de savoir si une face est la première d'un diamant ou d'un tétraèdre isolé
- 3 bits de service par face afin de représenter la permutation des sommets entre deux faces

	1 er diamant								1 er tétraèdre isolé				2 ème tétraèdre isolé				3 ème tétraèdre isolé			
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
F[i]	-1	281	202	758	854	829	239106	307	865	12	289	-1	9	863	239086	-1	861	385	380	239084
1 bit de service	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0
3 bits de service	0,0,0	1,0,2	2,0,1	0,2,1	2,0,1	2,0,1	2,1,0	1,0,2	1,0,2	0,1,2	0,1,2	0,0,0	0,1,2	1,0,2	1,0,2	0,0,0	2,0,1	1,0,2	1,0,2	1,0,2

FIGURE 19 – Notre structure est constituée d'un tableau F contenant les indices des faces adjacentes, d'un bit de service permettant de savoir si une face est la première d'un diamant ou d'un tétraèdre isolé, et de trois autres bits de service afin de connaître la permutation des sommets entre deux faces. Dans cet exemple, la face 0 est sur le bord et la face 1 est adjacente à la face 281.

Ordre des tétraèdres dans un diamant Au sein d'un diamant les tétraèdres sont ordonnés de manière purement arbitraire (Fig. 20). La seule condition est que deux tétraèdres adjacents doivent avoir un ordre consécutif modulo le nombre de tétraèdres dans le diamant (ex : dans un diamant contenant 5 tétraèdres, le troisième tétraèdre doit être adjacent au deuxième tétraèdre et au quatrième).

Ordre des faces dans un diamant L'ordre des faces dans un diamant respecte celui des tétraèdres. Pour le ième tétraèdre dans un diamant, ses faces extérieures seront les faces $2i$ et $2i+1$. Si un diamant est associé à un sommet alors les faces paires sont adjacentes à l'ancre. Sur la Fig. 20, l'ordre des face est donc : ACD,BCD,ADE,BDE,AGE,BGE,AFG,BFG,AFC,BFC.

Ordre des faces dans un tétraèdre isolé Les faces d'un tétraèdre isolé sont ordonnées de manière arbitraire. Seulement, si le tétraèdre isolé est ancré à un sommet, alors la première face doit être la face opposée à l'ancre.

Ordre des sommets dans les diamants Tout comme les tétraèdres, les sommets peuvent être ordonnés au sein d'un diamant. Un diamant D possedant $|D|$ tétraèdres contient $|D|+2$ sommets. On ordonne d'abord les sommets situés entre deux faces (les sommets D,E,G,F,C sur

la Fig. 20), puis les deux sommets communs à toutes les faces (les sommets A et B sur la Fig. 20). Le classement entre ces deux-derniers n'est pas important car on ne compare jamais l'ordre de ces deux sommets.

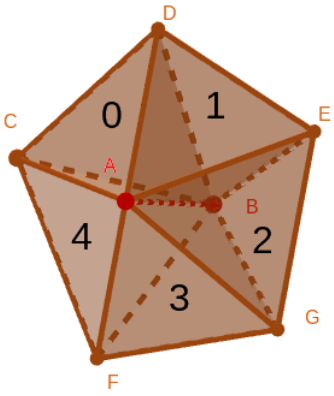

FIGURE 20 – Diamant contenant 5 tétraèdres ordonnés, dont l'arête centrale est AB et ancré au sommet A. L'ordre des tétraèdres est indiqué en noir. L'ordre des sommets est : C,D,E,G,F,A,B.

Ordre des sommets dans les tétraèdres Au sein d'un tétraèdre, les sommets sont ordonnés tel-que le ième sommet est opposé à la ième face. Si un sommet est associé au tétraèdre alors ce sommet est opposé à la première face.

4.6 Calculer les permutations

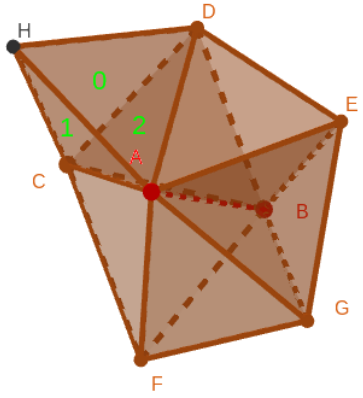

FIGURE 21 – Un diamant partage une face avec un tétraèdre isolé. En vert est l'ordre des face du tétraèdre. L'ordre des sommets du tétraèdre est A,D,C,H et celui du diamant est C,D,E,G,F,A,B.

Pour calculer la permutation vue du tétraèdre isolé de la face CAD, il suffit de comparer l'ordre des sommets du tétraèdre isolé et du diamant. Si l'on garde seulement les 3 sommets qui nous intéresse dans les deux classements, on a : A,D,C et C,D,A. La permutation est donc (2,1,0). Bien entendu, la permutation pour la même face du point de vue du diamant est la même.
Il y a 3 !=6 permutations possibles et donc 3 bits sont nécessaires pour représenter ces permutations.

4.7 Les requêtes

4.7.1 Vérifier qu'un diamant est correct

Cette fonction est essentielle car elle nous permet de savoir si une arête peut être candidate pour être l'arête centrale d'un diamant. Pour cela, on vérifie que deux sommets apparaissent plus de

deux fois, et tous les autres apparaissent exactement deux fois.

```
bool is_cycle(vector<Tetrahedron*>& tetra_list)
{
    map<int,int> count_vertex;
    for (int i=0;i<tetra_list.size();i++)
    {
        for (Vertex* j : tetra_list[i]->get_vertices())
        {
            count_vertex[j->get_id()]+=1;
        }
    }
    for (pair<int,int> i : count_vertex)
    {
        if (i.second<2)
        {
            return false;
        }
    }
    return true;
}
```

4.7.2 Accéder à la ième face

Pour accéder à la ième face, il suffit d'accéder à la cellule F[i] dans laquelle se trouve la face opposée à la ième face.

4.7.3 Acceder au ième diamant ou tétraèdre isolé

Il suffit de parcourir le tableau F et de regarder pour chaque face si le premier bit de service vaut 1. On s'arrête alors dès que l'on a parcouru i faces dont la valeur du premier bit de service est 1. La compléxité est O(n).

```
int ith_tetra_diamond(bool service_bit_array[],int array_size,int i)
{
    int count=0;
    int j=0;
    while(count<i)
    {
        if (service_bit_array[j]==1)
        {
                count++;
        }
    }
    return j;
}
```

4.7.4 Parcourir les tétraèdres d'un diamant

Les faces au sein d'un diamant sont ordonnées. Les faces consécutives dans le tableau (modulo la taille du diamant) sont adjacentes dans le diamant. Par conséquent, pour accéder aux faces du ième tétraèdre d'un diamant D, il suffit d'aller à la $2i$ et $2i+1$ ièmes faces du diamant (modulo $|D|$). La compléxité est O(1).

4.7.5 Accéder au ième tétraèdre

Lors du regroupement des tétraèdres en diamants, l'ordre des diamants n'est plus le même que l'ordre initial (à la lecture du fichier OFF). Néanmoins, on peut re-ordonner les tétraèdres dans le fichier original afin que l'ordre des tétraèdres soit les mêmes. De cette manière on peut acceder au ième tétraèdre en O(n).

4.7.6 Accéder au ième sommet

Bien que nous ne stockions pas les sommets de manière explicite (nous ne stockons que les faces). Nous sommes en mesures de localiser le ième sommet car il est adjacent au ième diamant/tétraèdre isolé. Il suffit donc d'accéder au ième diamant/tétraèdre isolé. Si c'est un diamant, alors le sommet est adjacent à toutes les faces paires du diamant. Si c'est un tétraèdre isolé, alors le sommet est opposé à la première face et est donc adjacent au trois autres faces. La compléxité est O(n).

4.7.7 Degré et hypersphère d'un sommet

Calculer le degré d'un sommet est plus compliqué. On sait que le ième sommet est adjacent au ième diamant/tétraèdre isolé. Si celui-ci est un diamant alors toutes les faces paires de celui-ci sont adjacentes au sommet ciblé. Si c'est un tétraèdre isolé, alors le sommet cible est opposé à la première face et est donc adjacent aux trois autres faces. Pour chaque face adjacente au sommet on accède à la face opposée en utilisant notre tableau F. En utilisant les permutations entre deux faces, nous sommes en mesure de savoir où se situe notre sommet dans la face opposée. Ainsi, nous pouvons accéder aux faces adjacentes au sommet dans ce nouveau diamant/tétraèdre isolé. Pour chaque diamant/tétraèdre isolé, on utilise un bit de service afin de savoir si celui-ci a déjà été visité[2]. On arrête de parcourir les faces quand tous les diamants et tétraèdres isolés adjacents au sommet ont été visités. La complexité est donc O(d).

4.7.8 Parcours en profondeur du graphe

Parcourir en profondeur le graphe est plus facile que de calculer le degré d'un sommet. Il suffit d'utiliser une file et un bit de service pour chaque tétraèdre afin de savoir si celui-ci a été visité. Le passage d'un tétraèdre à l'autre est rendu très facile grace à notre tableau représentant les faces adjacentes.

```
void BFS(int diamond_array[],bool service_bit_array[],int array_size,
int starting_index)
{
    queue<int> wait_list;
    unordered_set<int> visited_polygons;
    int index = ith_diamond(service_bit_array,array_size,starting_index);
    wait_list.push(get_starting_index(service_bit_array,index));
    while(!wait_list.empty())
    {
        index = wait_list.front();
        wait_list.pop();
        int i=index+1;
        int diamond_id = get_starting_index(service_bit_array,index);
        if(visited_polygons.count(diamond_id)==0)
        {
            visited_polygons.insert(diamond_id);
            path.push_back(diamond_id);
            //we iterate over the following faces in the diamond or isolated tetra
            while(service_bit_array[i]!=1 && i<array_size)
            {
                if (diamond_array[i]!=-1)
                {
                    wait_list.push(diamond_array[i]);
                }
                i++;
            }
            //we iterate over the previous faces in the diamond or isolated tetra
            i=index;
            while(service_bit_array[i]!=1 && i<array_size)
            {
                if (diamond_array[i]!=-1)
                {
                    wait_list.push(diamond_array[i]);
```

2. Nous omettons ce bit de service car il y en a en moyenne $\frac{|Diamants|+|Tétraàdres\ isolés|}{|Tétraàdres|} \simeq 0.5$ par tétraèdre

```
                }
                i--;
            }
            if (diamond_array[i]!=-1)
            {
                wait_list.push(diamond_array[i]);
            }
        }
    }
}
```

4.7.9 Retrouver l'indice d'un sommet

Retrouver l'indice d'un sommet est une procédure courante. En effet lorsque on est sur une face, on peut vouloir connaître les trois sommets qui la compose. Pour connaître l'indice d'un sommet, on calcule son hypersphère. Puis pour chaque diamant ou tétraèdre isolé dans l'hypersphère dont l'indice de la première face est inférieur à la limite [3], on calcule alors l'hypersphère du sommet ancré. Il suffit de comparer l'hypersphère de notre sommet cible avec les hypersphère des sommets ancrés et de renvoyer l'indice du sommet ayant la même hypersphère. Cette opération peut s'avérer néanmoins très couteuse. La compléxité pour calculer l'hypersphère d'un sommet est O(d). En utilisant notre algorithme pour appareiller les tétraèdres en diamants, les sommets sont en moyennes adjacents à 8 diamants/tétraèdres isolés. Seulement parmi ces 8 diamants/tétraèdres isolés, en moyenne 65% [4] auront un indice inférieur à la limite. Par conséquent, retrouver l'indice d'un sommet nécessite de calculer en moyenne 6 (0.65*8+1) hypersphères.

4.7.10 Résultats

Nom de la structure	Choisir les arêtes centrales et appareiller sommet/diamant	Accès au ième tétraèdre	Accès au ième diamant	Degré d'un sommet	Parcours en prodondeur
B1	0.06	1e-5	5e-6	1.9e-5	6e-4
B2	21.34	4.1e-4	1.6e-4	1.2e-4	0.01
B3	33.29				
M1	50.6	6.4e-4	2.6e-4	2.2e-4	0.02
C1	96.71	8.1e-4	3.4e-4	2.8e-4	0.02

TABLE 6 – Tableau des temps de chaque requête en secondes sur plusieurs maillages

Les résultats du tableau 4.7.10 sont obtenus en utilisant l'optimisation '-O3' de C++. Les résultats obtenus sont des moyennes calculées après 10 000 essais.

4.8 Borne inférieure théorique

4.9 Sauvegarder notre structure

Les fichiers OFF encodent d'abord les coordonnées géométriques de chaque sommet puis les indices des 3 sommets de chaque face. Ils sont ainsi très faciles à manipuler mais ne sont pas concis (un sommet apparaît en moyenne 22 fois dans une tétraèdrisation).

Encodage de la structure Comme exprimé dans la partie 4.7.9, un sommet dans notre structure est adjacent en moyenne à 8 diamants/tétraèdres isolés. Par conséquent, plutôt que de décrire pour chaque face les 3 indices des 3 sommets, nous pouvons encoder les indices des sommets composant un diamant. Les sommets sont ordonnés dans un diamant (4.5). Par conséquent, en ayant juste l'ordre des sommets, nous sommes en mesure de retrouver les faces (la première face est bordée par le premier, le deuxième et l'avant dernier sommet).
Cependant, notre structure ne stocke pas implicitement les sommets. Nous savons seulement que le ième sommet est adjacent au ième diamant/tétraèdre isolé. Néanmoins, en calculant l'hypersphère d'un sommet, nous informons les faces adjacentes qu'elles possèdent ce sommet.

3. La limite est l'index à partir duquel les diamant et tétraèdres isolés ne sont plus associés à des sommets
4. $\frac{|Diamants|+|Tétraèdres\ isolés|}{|V|}$

En faisant ainsi pour tous les sommets du maillage, nous associons alors à chaque face ses 3 sommets bordants.
Il est alors facile de retrouver l'ordre des sommets dans un diamant à partir des sommets bordants chaque face du diamant. Il suffit de prendre tous les sommets qui apparaissent exactement deux fois dans toutes les faces (i.e les sommets qui n'appartiennent pas à l'arête centrale). Puis de les ordonner afin qu'ils décrivent un cycle.
Par exemple, sur la Fig. 20, à partir du calcul de l'hypersphère de chaque sommet, nous saurions alors que :

- Les faces 0 et 1 contiennent les sommets C et D
- Les faces 2 et 3 contiennent les sommets E et D
- Les faces 4 et 5 contiennent les sommets G et E
- Les faces 6 et 7 contiennent les sommets F et G
- Les faces 8 et 9 contiennent les sommets C et F

Trouver l'ordre des sommets est alors aisé. On commence par trouver le sommet commun à la première et dernière face (i.e C), puis le sommet commun à la première et deuxième face (i.e D) et ainsi de suite pour finalement obtenir : C,D,E,G,F,A,B (les deux derniers sommets sont communs à toute les faces).

Ecriture dans un fichier L'écriture dans un fichier suit le même procédé que pour les fichiers OFF. Nous écrivons d'abord les coordonées géométriques de chaque sommet. Puis nous écrivons pour chaque diamant et tétraèdre isolé l'ordre de ses sommmets.

Ouverture du fichier L'ouverture est similaire à l'encodage du maillage. Pour chaque diamant, l'ordre des sommets nous permet de connaître les faces. Pour chaque tétraèdre isolé, l'ordre n'est pas important étant donné que tous les sommets d'un tétraèdre sont adjacents.

Résultats En théorie comme en pratique, notre structure nous permet de reduire en moyenne la taille du fichier OFF de 44% ($\frac{8}{18}$).

4.10 Améliorations

Références différentielles Dans notre tableau T, chaque face est représenté par un indice sur 32 bits, qui est la taille minimale d'un entier en C++. Néanmoins, nous pourrions représenter l'adjacence par la différence entre les indices des deux faces. Malheureusement, les faces sont seulement ordonneés de manière à ce que le ième sommet soit adjacent au ième diamant. Par conséquent, nous n'avons aucune garanti sur l'éloignement (dans le tableau F) de deux faces opposées. Etant donné que le gain semblait mineur, nous avons choisi de ne pas implémenter cette fonction.

Dynamicité Une structure de données compacte est dîtes dynamique lorsqu'on peut modifier les données localement (i.e de manière instantanée). Dans notre cas, modifier les données peut revenir à ajouter un sommet, ajouter une face, supprimer un tétraèdre...
Malheureusement, nous n'avons encore implémenté encore aucune de ces fonctions.

4.11 Défault

Afin d'appareiller les tétraèdres en diamants, nous réalisons un parcours en largeur de notre maillage. Seulemen, pour parcourir le maillage, nous lisons entièrement le fichier original car nous n'avons aucune garantie géométrique sur la manière dont sont énumérés les tétraèdres. Si les tétraèdres sont énumérés dans le fichier original par proximité alors nous pouvons maintenir un cache de tétraèdres puis les associer au moment voulu. Sans-cela, nous devons lire le fichier entièrement et cela représente un vrai goulot d'étranglement.

5 NP-Complétude

6 Implémentation

Tout est implémenté en C++ natif, sans l'aide d'aucune bibliothèque extérieure. Le code source est compilé avec g++ sous Elementary OS. Tous les algorithmes ont été exécutés sur une machine avec un processeur i5-5300U et 16Go de RAM. L'ensemble du code est open-source et disponible sur github : `https://github.com/beaupletga/3D-Mesh-Compression`.
Des classes représentent chaque forme géométrique (sommet, tétraèdre, diamant) permettant un code modulable et facilement exploitable. Nous utilisons cette représentation sous forme de classe pour la construction de notre structure mais elle est tout à fait absente dans son utilisation.
En ce qui concerne cette dernière, le tableau F est un tableau d'entiers codés sur 32 bits. Bien que nous aurions pu inclure les 4 bits de services dans les 32 bits de chaque entier, nous avons préféré utiliser deux tableaux annexes pour représenter ces bits de service. Le premier bit de service est stocké dans un tableau de booléens et les 3 autres bits de service sont stockés dans un tableau d'entiers. En codant les indices dans le tableau F sur 28 bits et en utilisant les 4 derniers bits comme bits de service, nous pouvons encoder des maillages ayant jusqu'à $2^{28} = 268$ millions de faces.

7 Conclusion

Nous avons présenté dans ce rapport une nouvelle structure de données compacte afin de représenter les maillages tétraèdriques en utilisant en moyenne 2.4 rpt. Cela représente une économie de 60% de références par tétraèdres en comparaison avec l'état de l'art (SOT). Notre structure s'adapte ainsi parfaitement aux maillages très volumineux. La navigation dans le maillage est intuitive car notre structure utilise les faces des tétraèdres. L'accès au ième sommet, ième tétraèdre se fait en temps constant et le calcul de l'hypersphère d'un sommet est proportionnel au degré du sommet. Finalement, nous avons aussi réussi à enregistrer notre structure dans un fichier afin que celle-ci puisse se partager facilement.

Références

[1] Deering, *Geometry Compression.*

[2] Stefan Gumhold, *Improved Cut-Border Machine for Triangle Mesh Compression.*

[3] Stefan Gumhold, Stefan Guthe, Wolfgang Straßer, *Tetrahedral Mesh Compression with the Cut-Border Machine.*

[4] Jarek Rossignac, *Edgebreaker : Connectivity compression for triangle meshes.*

[5] Gabriel Taubin, Jarek Rossignac, *Geometric Compression Through Topological Surgery.*

[6] Costa Touma and Craig Gotsman, *Triangle Mesh Compression.*

[7] Andrzej Szymczak, Jarek Rossignac, *Grow&fold : compression of tetrahedral meshes.*

[8] Manfred Weiler, Paula N. Mallon, Martin Kraus, Thomas Ertl, *Texture-Encoded Tetrahedral Strips.*

[9] Topraj Gurung, Jarek Rossignac, *SOT : Compact representation for tetrahedral meshes.*

[10] , *The CGAL Project. CGAL User and Reference Manual. CGAL Editorial Board, 4.14 edition, 2019*

[11] Bruce G. Baumgart, *Winged Edge Polyhedron Representation*

[12] Marcelo Kallmann and Daniel Thalmann, *Star-vertices : a compact representation for planar meshes with adjacency information*

[13] Catalog based representation of 2d triangulations, *Luca Castelli Aleardi, Olivier Devillers, and Abdelkrim Mebarki*

[14] Topraj Gurung, Daniel Laney, Peter Lindstrom, and Jarek Rossignac, *SQuad : Compact representation for triangle meshes*