الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي

BADJI MOKHTAR- ANNABA UNIVERSI
UNIVERSITE BADJI MOKHTAR - ANNA

جامعة باجي مختار-عنابة

Faculté : Technologie **Année : 2024**

Département : Informatique

THÈSE

Présentée en vue de l'obtention du diplôme de Doctorat 3ème Cycle

Intitulé

Reconstruction des Modèles des Objets Tridimensionnelles à Partir d'un Nuage de Points Quelconque

Option : Réseau et Sécurité Informatique

Par : Tchantchane Zahida

Dr. Abdelaziz Amara Korba	**Université de Badji Mokhtar, Annaba**	**Président**
Pr. Ghoualmi -Zine Nacera	**Université de Badji Mokhtar, Annaba**	**Directrice de thèse**
Pr. Meraihi Yacine	**U_M'Hamed Bougara Boumerdes**	**Examinateur**
Pr. Makhlouf DERDOUR	**Université Larbi Ben Mhidi**	**Examinateur**

Remerciements

L’achèvement de ce travail mené sur plusieurs années procure une grande satisfaction pour le thésard et son entourage. Il est également le moment de se rappeler les différentes difficultés qu’il a fallu surpasser et surtout les personnes qui nous ont encouragé et soutenus pour mener ce travail à terme. Il serait ingrat de renier le fait qu‘une thèse, reste malgré tout, une œuvre collective. La mienne est l‘aboutissement de plusieurs années de travail, de patience et de persévérance. Durant toutes ces années, j‘ai été assisté et conseillé par plusieurs personnes, auxquelles je dois leurs adresser mes plus sincères remerciements.

Tout d’abord je voudrais remercier Mr Abdelaziz Amara Korba et Mr Meraihi Yacine d’avoir accepté de relire cette thèse et d’en être rapporteurs. La version finale de cette thèse est le fruit de leurs lectures très attentives et de leurs remarques pertinentes. Je tiens à remercier Mr Abdelaziz Amara Korba d’avoir accepté d’être président du jury. Je remercie également tous les membres du jury d’avoir accepté d’assister à la présentation de ce travail.

C‘est à Mme Ghoualmi-Zine nassira, Professeur à l’université de ANNABA Directrice de la thèse. à qui revient le mérite d‘avoir dirigé ce travail. Elle m‘a offert d‘excellentes idées et perspectives sur le travail tout au long de son évolution et n‘ont jamais cessé de me conseiller en me permettant de concrétiser mes idées librement. Qu'elle trouve ici l'expression de mes vifs remerciements.

Cette thèse a été menée au sein de l’équipe CFAO du CDTA, dans le cadre de ces protocoles de recherche, à ce titre, Je voudrai remercier les membres de l’équipe qui m’ont soutenue énormément ainsi que tous mes collègues du CDTA. En particulier ; j’adresse mes vifs remerciements à Mr Bey Mohamed chef d’équipe CFAO et Mr Bendifallah mohamed el hassen ingénieur de recherche qui n’ont ménagé aucun effort pour clarifier et orienter mes idées et surtout pour leurs patience tout au long de ce travail.

Mes remerciements vont aussi à ma famille avec cette question récurrente, « quand est-ce que tu la soutiens cette thèse ? », bien qu’angoissante en période fréquente de doutes, m’ont permis de ne jamais dévier de mon objectif final. Merci à ma mère mes sœurs et mes frères. Mes remerciements vont aussi à mon cher époux pour son soutien et son enthousiasme à l’égard de mon travail sans oublier mes trois enfants qui ont grandi en même temps que ma thèse.

Enfin mes remerciements vont également à tous ceux qui ont contribué de près ou de loin à la réalisation de ce travail.

A mon père allah yarhmou Mohamed

A ma mère Nadjia

A mon époux Redouane

A mes enfants, Ishak, Ines et Mohamed

Je dédie ce travail

Tchantchane Zahida.

Résumé

La production de pièces très complexes avec une très grande précision est devenue une obligation. Ces pièces sont conçues selon deux classes d'approches (*i*) approches classiques et (*ii*) approches d'ingénierie inverse. Le temps de traitement très important et la modification successive de la forme sont parmi les inconvénients de la première classe. Quant à la deuxième classe, l'ingénierie inverse, elle suit plusieurs étapes : numérisation de l'objet, recalage, décimation, segmentation du nuage de points et enfin approximation de la surface. Malheureusement, les deux dernières étapes présentent plusieurs difficultés de traitement. Dans le but de rendre la conception des modèles plus souple et moins couteuse, l'approche de *Facettisation* a été proposée. Le travail présenté dans le cadre de cette thèse s'est concentré sur l'approche de *Facettisation*, où les points du nuage sont reliés par des formes géométriques simples (tétraèdres). La triangulation de Delaunay est l'une des techniques de reconstruction les plus utilisées. Elle est générée par l'algorithme « *Insertion incrémentale* » et « *Diviser pour régner* ». L'objectif principal de cette thèse est de développer une nouvelle méthodologie de reconstruction de modèle d'objet 3D à partir d'un nuage de points non-uniforme en un délai de traitement réduit. Cet objectif a été divisé en plusieurs sous objectifs. Le premier but concerne la génération de la triangulation de Delaunay 3D en utilisant, pour la première fois, l'approche de « *Destruction et construction* ». Par la suite, l'approche *FLIP* (*AF*) a été développée pour générer une meilleure triangulation de Delaunay. Après et pour pallier aux inconvénients des approches précédentes, l'approche *FLIP Amélioré* (*AFA*) a été proposée. Ultérieurement, le paradigme « *Diviser pour régner* » a été exploitée. Ainsi, une méthodologie de partitionnement spatiale a été proposée. A la fin, une nouvelle architecture de fusion 3D a été proposée. Les principaux défis sont au nombre de trois : (*i*) proposer une approche rapide et efficace pour localiser le nouveau point à insérer, (*ii*) trouver un meilleur partitionnement qui permet d'améliorer les performances de regroupement des points par une homogénéité des partitions, et enfin (*iii*) proposer une nouvelle architecture de fusion des sous-triangulations. La méthodologie proposée a été mise en œuvre et sa validation a été effectuée à travers plusieurs pièces. Les résultats obtenus ont démontré (*i*) l'efficacité et la performance de la méthodologie développée pour localiser le nouveau point à insérer en termes de temps de calcul, (*ii*) les hautes performances de la méthode K-means global rapide (KGR) en ce qui concerne l'homogénéité des partitions et la charge de travail équilibrée sur les différents processeurs, et enfin (*iii*) la supériorité de la nouvelle architecture de fusion des sous-triangulations proposée en termes de de temps de calcul.

Mots clés : Nuage de points non-structuré, Algorithme « *Diviser pour régner* », Architecture de fusion, Partitionnement, Triangulation de Delaunay 3D.

Abstract

The production of highly complex parts with very high precision has become a requirement. These parts are designed according to two classes of approaches (*i*) classical approaches and (*ii*) reverse engineering approaches. The high processing times and the successive modifications of the shape are among the key drawbacks of the first class. The second class, reverse engineering, follows several steps: digitization of the object, readjustment, decimation, segmentation of the points cloud and finally approximation of the surface. Unfortunately, the last two steps present several processing difficulties. To make the models design more flexible and less costly, the *Faceting approach* is proposed. The work presented in this thesis focused on the *Faceting approach*, where the points of the cloud are connected by simple geometric shapes (i.e., tetrahedra). The Delaunay triangulation is one of the most widely utilized reconstruction techniques. It is generated by the *"Incremental Insertion"* and *"Divide and Conquer"* algorithms. The main objective of the thesis is to develop a new methodology to build a 3D object model from a non-uniform points cloud in a reduced processing time. This purpose has been divided into several sub-objectives. The first goal deals with the generation of the 3D Delaunay triangulation using, for the first time, the *"Destruction and Construction"* approach. Subsequently, the *FLIP Approach (AF*) is developed to generate a better Delaunay triangulation. Afterwards and in order to overcome the drawbacks of the previous approaches, the *Improved FLIP Approach (AFA*) is proposed. Therefore, the *"Divide and Conquer"* paradigm is exploited. Thus, a spatial partitioning methodology is proposed. Finally, a new 3D fusion architecture is proposed. The main challenges are threefold (*i*) propose a fast and efficient approach to localize the new point to be inserted, (*ii*) find a better partitioning to improve the performances of points clustering and partitions homogeneity, and finally (*iii*) propose a new fusion architecture for sub-triangulations. The proposed methodology is implemented and its validation is carried out via several pieces. Results demonstrated (*i*) the efficiency and performance of the developed methodology to localize the new point to be inserted in terms of computation time, (*ii*) the high performances of the *Fast Global K-means* method (*FGK*) with respect to the partitions homogeneity and the balanced workload on the different processors, and finally (*iii*) the superiority of the proposed new fusion architecture of sub-triangulations in terms of computation time.

Keywords: Unstructured points cloud, *"Divide and Conquer"* algorithm, Fusion architecture, Clustering, 3D Delaunay triangulation.

ملخص

لقد أصبح إنتاج القطع المعقدة للغاية بدقة عالية جدًا إلزاميًا. تم تصميم هذه القطع وفقًا لفئتين من الطرق (1) الطرق الكلاسيكية و (2) طرق الهندسة العكسية. تعد مدة المعالجة الطويلة جدًا والتعديل المتتالي للشكل من بين عيوب الطرق الأولى. أما بالنسبة للفئة الثانية، الهندسة العكسية، فهي تتبع عدة خطوات: رقمنة القطعة، التسجيل، التدمير، تجزئة السحابة النقطية، وأخيراً تقريب السطح. لسوء الحظ، تمثل الخطوتان الأخيرتان عدة صعوبات في المعالجة. من أجل جعل تصميم النماذج أكثر مرونة وأقل تكلفة، تم اقتراح نهج Facettisation. ركز العمل المقدم في هذه الأطروحة على منهجFacettisation ، حيث تربط نقاط السحابة بأشكال هندسية بسيطة (رباعي). يعد تثليث Delaunay أحد أكثر تقنيات إعادة البناء استخدامًا. يتم إنشاؤه بواسطة خوارزمية *"الإدراج المتزايد"* و خوارزمية*"فرق تسد"* . الهدف الرئيسي من الأطروحة هو تطوير منهجية جديدة لإعادة بناء نموذج القطعة ثلاثية الأبعاد من سحابة نقطية غير موحدة في وقت معالجة قصير. تم تقسيم هذا الهدف إلى عدة أهداف فرعية. يتعلق الهدف الأول بانشاء تثليث Delaunay ثلاثي الأبعاد باستخدام نهج *"التدمير والبناء"* لأول مرة. بعد ذلك، تم تطوير نهج (*AF*) *FLIP* لإنشاء تثليث Delaunay أفضل. من أجل التغلب على عيوب المناهج السابقة، تم اقتراح نهج *FLIP* المحسن (*AFA*) . بعد ذلك، تم استغلال خوارزمية *"فرق تسد"* . وبالتالي، تم اقتراح منهجية التقسيم المكاني. في النهاية، تم اقتراح بنية دمج جديدة ثلاثية الأبعاد. تتمثل التحديات الرئيسية في (1) اقتراح طريقة سريعة وفعالة لتحديد موقع النقطة الجديدة التي سيتم إدراجها، (2) إيجاد تقسيم أفضل يجعل من الممكن تحسين أداء تجميع النقاط من أجل تجانس الأقسام، وأخيرًا (3) اقتراح بنية جديدة لدمج التثاليث الفرعية. تم تنفيذ المنهجية المقترحة وتم التحقق من صحتها من خلال تجربتها على عدة قطع. أظهرت النتائج التي تم الحصول عليها (1) كفاءة في أداء المنهجية المطورة لتحديد موقع النقطة الجديدة التي سيتم إدراجها من حيث وقت المعالجة، (2) الأداء العالي لطريقة K-means الشاملة السريعة (FGK) فيما يتعلق بتجانس الأقسام وعبء العمل المتوازن على المعالجات المختلفة، وأخيرًا (3) تفوق بنية دمج التثاليث الفرعية الجديدة المقترحة من حيث وقت المعالجة.

الكلمات المفتاحية: سحابة نقط غير منظمة، خوارزمية *"فرق تسد"*، بنية الدمج، التقسيم، تثليث Delaunay ثلاثي الأبعاد.

Valorisation scientifique

Publication internationale

Z. Tchantchane, K. Bouhadja, O. Azouaoui, and N. Ghoualmi-Zine . Enhanced unstructured points cloud subdivision applied for parallel Delaunay triangulation. Cluster Computing, (2022). https://doi.org/10.1007/s10586-022-03699-9

Communications nationale et internationales

1. Z. Tchantchane, M. Bey, O. Azouaoui, and H. Bendifallah, "STL Model of Sculptured Surfaces from 3D Unstructured Cloud of Points," *11èmes Journées de Mécanique de l'EMP (JM'11–EMP)*, pp. 3–7, 2018.
2. Z. Tchantchane, M. Bey, O. Azouaoui, and H. Bendifallah,, " Reconstruction Des Surfaces Frontieres D'un Objet À Partir D'un Nuage De Points 3D," International Conference On Advanced Mechanics And Renewable Energies « ICAMRE ». 28&29 Novembre 2018, Boumerdes – Algerie.
3. Z. Tchantchane, M. Bey, O. Azouaoui, and H. Bendifallah,, " Triangulation D'un Nuage De Points 3D Non Structure Par La Méthode De Flip," International Conference On Advanced Mechanics And Renewable Energies « ICAMRE ». 28&29 Novembre 2018, Boumerdes – Algerie.
4. Z. Tchantchane,O. Azouaoui, N. Ghoualmi-Zine, and M. Bey, "Enhanced Flip in 3D Delaunay Triangulation", International Conference on Cyber Security, Artificial Intelligence and Theoretical Computer Science "ICCSAITCS 2022", 27-28 Novembre 2022, Boumerdes, Algerie
5. Z. Tchantchane,O. Azouaoui, N. Ghoualmi-Zine, and M. Bey, "Enhanced merging order: A novel architecture for merging sub-triangulations", International Conference on Cyber Security, Artificial Intelligence and Theoretical Computer Science "ICCSAITCS 2022", 27-28 Novembre 2022, Boumerdes, Algerie.

Table des Matières

Partie 2 : Triangulation de Delaunay parallèle en utilisant la méthode « Diviser pour régner ».

Chapitre 4 : Partitionnement du nuage de points en sous-nuages

Liste des figures

Chapitre 1 :

Chapitre 2 :

Chapitre 3 :

Chapitre 4 :

Chapitre 5 :

Liste des tables

Chapitre 3 :

Chapitre 4 :

Chapitre 5 :

Chapitre 1
Introduction Générale

Contexte général

Depuis plusieurs années, les progrès des technologies d'acquisition 2D ont favorisé l'essor de la modélisation géométrique d'objets à deux dimensions à partir de données issues de leur numérisation. Malheureusement, cette modélisation est devenue obsolète car son espace de représentation est défini seulement par deux informations, la *longueur* et la *largeur*, ce qui induit à une représentation insuffisante de l'objet.

Avec l'apparition de la 3D, une nouvelle vision dans l'espace a donné la possibilité de visualiser le modèle sous toutes ses perspectives, ce qui permet une meilleure représentation et approximation de l'objet surtout quand il s'agit d'objets très complexes.

Le travail présenté dans le cadre de cette thèse concerne le développement d'approches permettant la reconstruction de modèles d'objets définis par un nuage de points quelconque. Ce travail peut être introduit dans le secteur de l'industrie médicale, l'aéronautique, la conception et la fabrication de moules, etc. où les formes d'objets sont très complexes et doivent répondre à des fonctionnalités très précises.

Problématique et solution

La reconstruction d'objets est une discipline relativement jeune qui concerne un large éventail de domaines liés au génie mécanique, à la vision artificielle et à la robotique, ainsi qu'au génie médical, au génie civil et environnement.

Les applications majeures de la reconstruction d'objets sont représentées par la génération de modèles de pièces, la reconstruction pour l'usinage, l'inspection et le contrôle en génie mécanique, la reconstruction dans la technologie des prothèses en chirurgie orthopédique, la définition des voies d'accès en chirurgie générale, le suivi de l'évolution d'une maladie, l'aide au diagnostic et à la prévention dans le domaine médical, et enfin la modélisation numérique de terrains.

L'utilisation de l'outil informatique a apporté une aide qui est devenue indispensable au fil du temps par la résolution des problèmes de nature complexe. En effet, cet outil offre efficacité, fiabilité et rapidité de traitement qui peuvent être observés dans plusieurs disciplines, notamment en *Enseignement Assisté par Ordinateur* (*EAO*), *Dessin Assisté par Ordinateur* (*DAO*), et *Conception et Fabrication Assistées par Ordinateur* (*CFAO*). Parmi les domaines qui exigent l'utilisation de cet outil, on trouve le domaine de l'industrie, tels que l'industrie automobile, l'industrie aéronautique, l'industrie mécanique, etc. Plusieurs industries ont été contraintes à augmenter la quantité et améliorer la qualité de leurs produits. On cite en particulier l'industrie mécanique qui s'appuie sur la puissance de la CFAO et qui peut apporter la flexibilité et la souplesse dans la conception et la fabrication des pièces de complexités diverses. L'avantage majeur de cette discipline consiste en sa capacité à réaliser les formes gauches telles que les moules des ailes d'avion, des véhicules, etc.

Une forme gauche est une surface gauche qui ne peut pas être définie par une équation mathématique exacte et qui est obtenue par raccordement de plusieurs morceaux de surfaces.

Aujourd'hui, la conception des formes gauches peut être réalisée selon les deux classes d'approches suivantes :

- **Classe des approches classiques** : elle est basée sur la modélisation de l'objet selon un cahier des charges et l'utilisation des fonctions de conception et de manipulation prédéfinies des logiciels de *Conception Assistée par Ordinateur* (*CAO*),
- **Classe des approches d'ingénierie inverse** : cette classe d'approches est basée sur la modélisation de l'objet à partir d'un nuage de points

L'inconvénient de la première catégorie d'approches réside dans la difficulté d'obtenir les formes souhaitées sans recourir à des phases de modification et de validation successives, afin de respecter les contraintes géométriques (tangentes, courbures, torsion, etc.) et fonctionnelles imposées a priori ; ceci induit des temps de traitement très importants.

La deuxième catégorie d'approches basée sur l'ingénierie inverse ou « *retro-conception* » est utilisée dans le cas où les formes des objets sont très complexes et ne peuvent être conçues dans un logiciel de CAO, ou bien si le modèle mathématique n'est pas disponible. Cette approche est devenue de nos jours une pratique très courante dans le monde industriel puisqu'elle permet de réduire les temps de développement d'un produit, comme en témoigne l'essor connu par les procédés de prototypage rapide. Une telle approche repose généralement sur le principe de la duplication d'objets physiques (objets existants) ou la création d'un prototype de nouveaux objets. La démarche globale de la *retro-conception* recouvre généralement les étapes de numérisation de l'objet, de recalage des nuages de points digitalisés, de décimation et de segmentation de nuages de points, et enfin de l'approximation de surface (Figure 1).

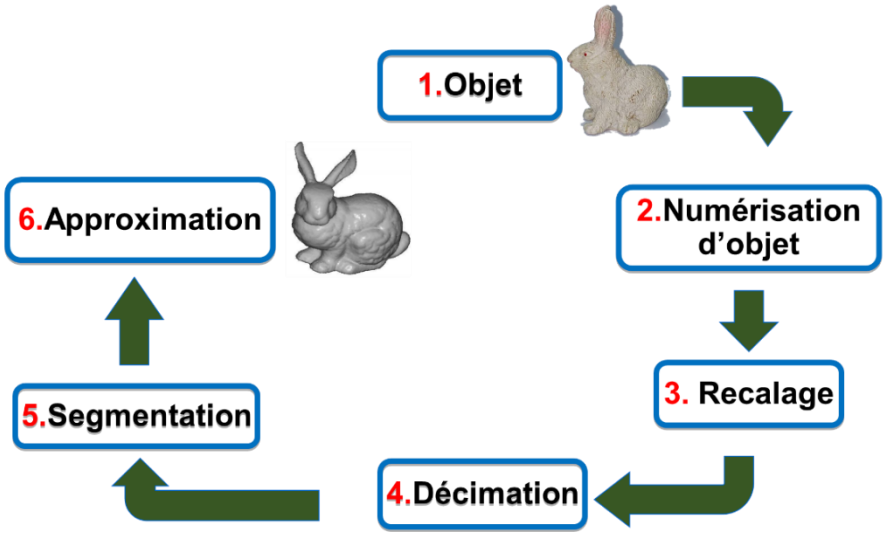


Figure 1 : Démarche globale de la retro-conception

La numérisation de l'objet permet de générer un ou plusieurs ensembles de points en fonction de la complexité géométrique de l'objet et des conditions d'accessibilité aux différentes régions à l'aide des outils de mesure tels que les « *Machines à Mesurer Tridimensionnelle* (*MMT*) » où chaque point mesuré P_i est identifié par les trois coordonnées (x_i, y_i, z_i) [1]. Par la suite, dans l'étape de recalage, les nuages de points nécessitent une mise en correspondance pour exprimer les coordonnées de l'ensemble des points dans un même repère (un seul nuage de points) [2].

Puisque le nuage de points récupéré est très dense et l'utilisation directe de celui-ci sans simplification conduit à l'utilisation d'un espace de stockage très important, à l'augmentation des temps de traitement et à la manipulation non efficiente des données. Dans ce cas, une opération de décimation est nécessaire afin de rendre le nuage de points manipulable et traitable sur ordinateur avec une puissance de calcul et des ressources mémoires raisonnables [3]. Ensuite, l'étape de segmentation consiste à distinguer les différentes parties (régions) présentes dans le nuage de points. Elle vise à décomposer le nuage de points en régions distinctes partageant une ou plusieurs caractéristiques communes [4]. La dernière phase de ce processus consiste à reconstruire le modèle CAO de l'objet à l'aide d'une méthode de reconstruction. L'objectif étant la détermination de la représentation mathématique de l'objet. Les méthodes les plus utilisées sont l'interpolation et l'approximation [5].

La conception de pièces en utilisant ces deux approches avec une bonne précision est une tâche ardue, coûteuse en temps, et les systèmes de CAO exigent une grande interactivité entre le logiciel et l'utilisateur selon les possibilités de modélisation disponibles. En outre, le modèle de l'objet qui en résulte est une approximation du nuage de points qui nécessite très souvent des ajustements. La *retro-conception* présente des difficultés de traitement lors des étapes de segmentation et d'approximation de surface.

Pour rendre la conception des modèles plus souple et moins couteuse, une nouvelle approche a été développée, « *approche de Facettisation* ». Elle consiste à générer un modèle facettisé après l'étape de décimation du nuage de points, et qui est par la suite transmis à l'usinage.

L'approche de facettisation n'est qu'une triangulation (Figure 2). À vrai dire, elle consiste à relier les points du nuage de points par des formes géométriques simple telles que les triangles (en deux dimensions) et les tétraèdres (en trois dimensions).

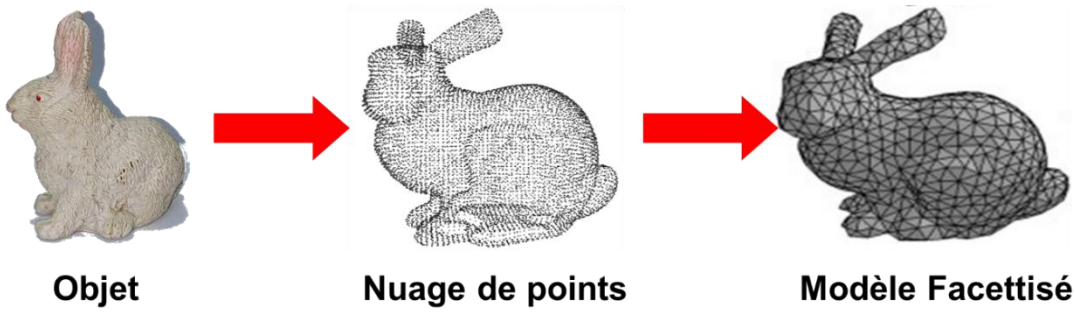


Figure 2 : Approche de Facettisation

De très nombreux algorithmes ou techniques ont ainsi été développés pour construire une triangulation ou en améliorer une ; mais, la triangulation la plus utilisée est celle de Delaunay qui est l'une des techniques de reconstruction de surface les plus anciennes et les plus fondamentales. Ainsi, la triangulation de Delaunay possède des propriétés géométriques remarquables et qui permet d'optimiser un certain nombre de critères.

La méthode de triangulation de Delaunay est générée généralement par deux algorithmes : « *Insertion incrémentale* » et « *Diviser pour régner* ». Généralement, l'algorithme *Insertion incrémentale* est le plus simple à mettre en œuvre ayant l'avantage de se généraliser à une dimension supérieure (trois et plus). Quant au deuxième algorithme, *Diviser pour régner*, il est conçu pour le parallélisme et nécessite des ordinateurs à processeurs multi-cœurs.

Les travaux de cette thèse de doctorat ont été réalisés dans le cadre de projet de recherche « *Usinage des surfaces gauches à partir de leurs représentations discrètes sur des fraiseuses à commande numérique à 03 axes* » au sein de l'équipe « *Conception et Fabrication Assistées par Ordinateur* (*CFAO*) » de la « *Division Productique et Robotique* (*DPR*) » du « *Centre de Développement des Technologies Avancées* (*CDTA*) ».

L'objectif de la thèse est de développer une nouvelle méthodologie de reconstruction de modèle d'un objet 3D à partir d'un nuage de points non-uniforme en un délai de traitement réduit. Autrement dit, étant donné un nuage de points aléatoire issu de la digitalisation d'un objet physique, il s'agit de proposer une démarche pour la reconstruction de cet objet 3D qui soit fidèle au modèle d'origine en utilisant des éléments géométriques élémentaires « *tétraèdres* ». Les objectifs assignés à cette thèse sont les suivants :

- Réaliser un état de l'art sur les approches réalisables et efficaces pour la modélisation en trois dimensions.
- Générer une triangulation 3D de pièce à partir d'un nuage de points irrégulier.
- Proposer une méthodologie permettant de réduire les temps de traitement.

Pour atteindre ces objectifs, nous avons proposé deux solutions en se basant sur l'utilisation de deux types d'algorithmes de la méthode de triangulation de Delaunay, à savoir « *Insertion incrémentale* » et « *Diviser pour régner* » :

- **Solution 1** : nous avons exploité l'algorithme « *Insertion incrémentale* » et proposé une solution en deux étapes :
  - Améliorer la génération de la triangulation de Delaunay tridimensionnelle,
  - Reconstruire les frontières de l'objet.

- **Solution 2** : nous avons exploité l'algorithme « *Diviser pour régner* » et proposé une solution en deux étapes :
    - Améliorer la subdivision du nuage de points en sous-nuages,
    - Proposer une nouvelle architecture de fusion tridimensionnelle qui spécifie l'ordre de fusion à chaque sous-nuage sans connaître la direction de partitionnement.

Organisation de la thèse

En plus de ce premier chapitre qui consiste en une introduction générale, la présente thèse est composée de deux parties ; chacune est organisée en deux chapitres :

- **Partie 1 :** cette partie s'intitule « R*econstruction des objets à partir de nuages de points* ». Elle est composée de deux chapitres :
    - **Chapitre 2** : ce chapitre est intitulé *« Approches de reconstruction d'objets à partir de nuages de points »*. Il commence par la description d'un état de l'art sur les algorithmes de reconstruction d'objets à partir d'un nuage de points. Chaque algorithme de reconstruction a ses propres avantages et inconvénients. À ce jour, la triangulation de Delaunay est toujours utilisée bien qu'elle soit une méthode ancienne [6]. Pour cela, les détails de la triangulation de Delaunay ainsi que les différentes techniques utilisées sont présentés dans ce chapitre.
    - **Chapitre 3** : ce chapitre s'intitule *« Description des approches développées* ». Il propose une méthodologie de reconstruction en passant par deux étapes : (*i*) générer une triangulation de Delaunay à partir d'un nuage de points non-structuré en utilisant trois algorithmes différents, et (*ii*) reconstruire des frontières de l'objet en utilisant la méthode de « *Alpha Shape* ». À la fin, ce chapitre valide la méthodologie proposée via plusieurs exemples de pièces.
- **Partie 2 :** cette deuxième partie est intitulée « *Triangulation de Delaunay parallèle* » et utilise le paradigme « *Diviser pour régner* ». Cette partie se compose de deux chapitres :
    - **Chapitre 4** : ce chapitre s'intitule *« Partitionnement du nuage de points en sous-nuages »*. Il présente une méthode de partitionnement du nuage de points en sous-nuage. Ce chapitre commence par l'élaboration d'un état de l'art sur les méthodes de partitionnement existant dans la littérature. Par la suite, le chapitre propose une nouvelle méthode de partitionnement. À la fin, la méthode proposée est validée en utilisant plusieurs exemples de pièces.
    - **Chapitre 5** : ce deuxième chapitre s'intitule *« Fusion parallèle des sous-triangulations* ». D'abord, il résume les principales approches de fusions des sous triangulations. Par la suite, le chapitre propose une nouvelle architecture de fusion tridimensionnelle qui spécifie l'ordre de fusion des sous-triangulations. Après, il détaille le processus de concaténation de deux sous-triangulations. Cette approche est testée et validée à la fin du chapitre sur plusieurs exemples de pièces.
- **Conclusion générale et perspectives** : pour achever ce travail, une conclusion générale et des éventuelles perspectives sont présentées.

Partie 1
Reconstruction des objets à partir de nuages de points

Introduction

La reconstruction d'un objet est la technique permettant d'obtenir une représentation en trois dimensions d'un objet à partir d'un nuage de points. Les coordonnées des points sont obtenues par la digitalisation de l'objet physique à l'aide de moyens spécifiques tels qu'une « *Machine à Mesurer Tridimensionnelle* (*MMT*) ». Le type de données peut être sous la forme de nuages de points structurés ou non-structurés. La précision du modèle reconstruit doit être prise en compte ; en effet, si le résultat est incorrect, il ne sera pas identique à la forme originale de l'objet. Par conséquent, les méthodes appropriées doivent être choisies en fonction des données utilisées. Des méthodes d'optimisations ont également été utilisées dans le domaine de la reconstruction. Cette partie met en évidence les recherches et méthodes utilisées dans le domaine de la reconstruction d'objets.

Chapitre 2
Approches de reconstruction d'objets à partir de nuages de points

1. Introduction

La reconstruction de surfaces est une autre façon de créer des objets tridimensionnels ; elle consiste à acquérir les données de l'objet physique réel. Cette acquisition peut être réalisée à l'aide de diverses techniques de numérisation ; le résultat est une collection de points décrivant la surface originale de l'objet (nuage de points). La détermination et l'exactitude du modèle final reconstruit dépendent fortement de la densité et de la distribution des points obtenus. Le véritable défi de la reconstruction de surface réside dans le fait que les informations du nuage de points fournies sur la surface inconnue sont très limitées. Chaque point est décrit par sa position dans l'espace ; aucune autre information telle que la topologie ou la connectivité n'est donnée. En outre, il faut tenir compte des ensembles de données dont la densité d'échantillonnage n'est pas uniforme.

Le type de données est un aspect important pour la reconstruction de surfaces car il permet de déterminer le type de techniques et d'approches qui seront utilisées pour reconstruire la surface de l'objet. Une fois la technique de reconstruction choisie, il faudrait peut-être effectuer certaines opérations de prétraitement (simplification des données, etc.) avant de commencer le processus de reconstruction. Cela dépend de la technique utilisée, car parfois certaines techniques ne peuvent pas traiter la grande densité de données. Par conséquent, certaines hypothèses doivent être faites et les limites doivent être précisées dans les résultats.

Ce chapitre va mettre en lumière les recherches et les méthodes de l'état de l'art utilisées dans le domaine de la reconstruction des objets.

2. Méthodes de reconstruction des modèles tridimensionnels

Les algorithmes de reconstruction de surfaces peuvent être divisés en plusieurs classes, pas nécessairement disjointes.

2.1.Classification selon la représentation des surfaces

Les méthodes de reconstruction de modèles tridimensionnels peuvent être regroupées selon la représentation des surfaces en trois classes (Figure 1), à savoir, (*i*) surface explicite, (*ii*) surface implicite et approches avec optimisation [7].

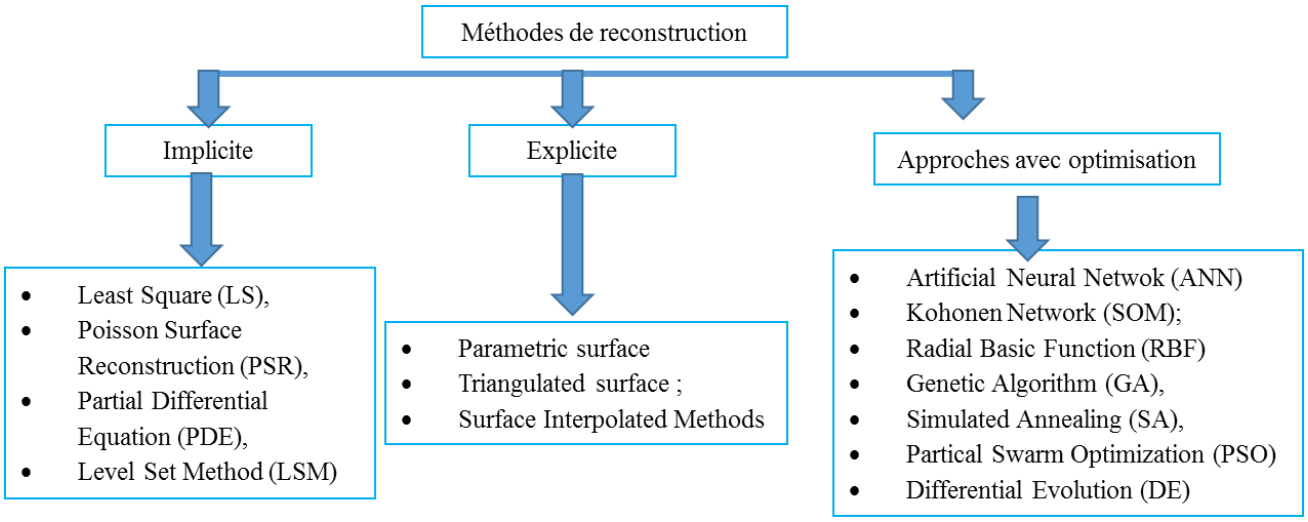


Figure 1 : Classification selon la représentation des surfaces

2.1.1. Surfaces explicites

Une surface explicite est donnée par l'équation (1) ; pour chaque valeur de X et Y correspond une et une seule valeur de Z :

$$Z = F(X, Y) \qquad (1)$$

Les surfaces explicites représentent fidèlement la forme exacte de la surface [8] [9] [10]. Pour ce type de surfaces, trois sous-classes de méthodes sont définies :

- **Surface paramétrique:** elle est représentée par morceaux (patchs), et décrite par une équation paramétrique. Plusieurs patchs peuvent être assemblés pour former une surface. Parmi les exemples de surface paramétriques, nous citons les surfaces B-spline, Bézier et NURBS [12-13].
- **Surface triangulée :** elle est représentée par des triangles, et est basée sur l'utilisation de diagramme de Voronoï et l'algorithme de Triangulation de Delaunay [14-16]. Pour cette classe, le nuage de points est supposé dense et sans bruit.
- **Méthode d'interpolation de surfaces** : les méthodes d'interpolation de surfaces utilisent une fonction mathématique spécifique pour interpoler les points échantillonnés [17]. Le maillage final des points échantillonnés est la surface triangulée à partir de la fonction interpolée. Comparée aux autres algorithmes existants, la méthode d'interpolation de surfaces peut être utilisée pour réduire l'effet du bruit dans les points d'échantillonnage (lisser la surface) ou simplifier la densité des points échantillonnés pour résoudre le problème de stockage.

2.1.2. Surfaces implicites

Une surface implicite est définie par une fonction F de trois variables (X, Y, Z) donnée par l'équation (2) :

$$F(X, Y, Z) = 0 \qquad (2)$$

Les surfaces implicites sont désignées par des approches de représentation volumétrique ou méthodes d'approximation [8] [11]. Elles sont réparties sur deux grandes sous-classes : (*i*) approximation globale et (*ii*) approximation locale. Dans ce qui suit, nous décrivons brièvement quelques méthodes utilisées :

- **Méthode des moindres carrés (Least Square):** la méthode des moindres carrés est l'une des méthodes d'approximation les plus courantes. Elle a fait l'objet de plusieurs travaux tels que ceux de [10], [18]. En général, cette méthode est combinée à une autre méthode d'optimisation selon les auteurs dans [19].
- **Reconstruction de surface de Poisson (Poisson Surface Reconstruction):** l'algorithme de reconstruction de surface de Poisson est une fonction implicite qui fusionne les schémas d'approximation globale et locale [20-22].
- **Équation Différentielle Partielle (Partial Differential Equation EDP) :** d'après [23], l'EDP a donné de très bons résultats dans le domaine de l'interpolation et de compression d'images. Cette méthode a été présentée dans les travaux de [24].
- **Méthode des ensembles de niveaux (Level Set Method):** la méthode des ensembles de niveaux a été introduite par Osher et Sethian [15]. Il s'agit d'une méthode numérique et théorique pour les surfaces implicites, comme mentionné dans [25]. En outre, elle peut être utilisée pour traiter facilement la topologie complexe et les données bruitées, la modélisation de surface et de nombreuses autres applications.

2.1.3. Approche avec optimisation

Plusieurs algorithmes d'optimisation sont utilisés afin de répondre à cette problématique ; nous citons quelques-uns :

- **Réseau Neuronal Artificiel (Artificial Neural Netwok) (ANN) :** les **ANN** s'inspirent du mode de fonctionnement des systèmes nerveux biologiques où le neurone artificiel constitue la base du réseau neuronal [26]. Sur cette base, plusieurs travaux ont été réalisés tels que ceux de [27-29].

- **Réseau Kohonen (Kohonen Network)** : le réseau de Kohonen, connu sous le nom de carte auto-organisée ou Self-Organized Map (SOM), est utilisé dans la reconstruction des surfaces. Cette méthode a été présentée dans les travaux de [49-50] [16] et [30].

- **Fonction de Base Radiale (Radial Basic Function (RBF)) :** la **RBF** est aussi utilisée dans la reconstruction des objets tels que dans [31-33]. Les travaux menés dans [34] montrent que cette méthode peut être associée à la méthode des moindres carrés et qu'elle permet de surmonter les problèmes d'approximation.
- **Algorithmes Génétiques (Genetic Algorithm (GA))** : la technique des algorithmes génétiques a été introduite par John Holland en 1975. Les AG ont été utilisées dans [19] [35-37].
- **Recuit Simulé (Simulated Annealing (SA))**: le SA a été proposé dans [17] comme méthode probabiliste. Dans les travaux de [39-41], les auteurs ont combiné l'algorithme **SA** et la méthode **RBF** pour surmonter les inconvénients de cette dernière.
- **Optimisation par Essaims de Particules (Partical Swarm Optimization (PSO))**: la méthode **PSO** a été développée par Kennedy et Eberhart en 1995. Il s'agit d'une technique d'optimisation stochastique basée sur la population, qui s'inspire du vol d'oiseaux et de la formation de bancs de poissons [42]. Dans le domaine de la reconstruction de surface, nous trouvons les travaux de [19] et [43] qui ont utilisé cet algorithme pour obtenir une paramétrisation correcte des points de données pour une surface de Bézier.
- **Evolution Différentielle (Differential Evolution (DE)):** la **DE** a été introduite comme méthode heuristique de recherche [44]. Il s'agit d'une stratégie de recherche basée sur la population et d'une méthode d'optimisation mathématique de fonctions multidimensionnelles **[**45**]**. Dans le travail de [46]**,** cet algorithme est meilleur que les autres méthodes dans l'optimisation des problèmes en raison de sa simple mise en œuvre et de ses caractéristiques de convergence rapide. Dans **[**47**]** et **[**48**],** cette technique a été utilisé pour minimiser l'écart entre les données mesurées et estimées. Il a été conclu que la méthode **DE** produit un meilleur résultat de reconstruction de formes et donne une erreur de reconstruction plus faible par rapport à la méthode **PSO**.

2.2.Classification selon l'approche volumétrique et l'approche géométrique

Les auteurs dans [51-52] ont proposé de regrouper les méthodes de reconstruction selon deux approches (Figure 2) : (*i*) approches géométriques computationnelles et (*ii*) approches volumétriques. Chacune de ces méthodes a des hypothèses pour fonctionner correctement.

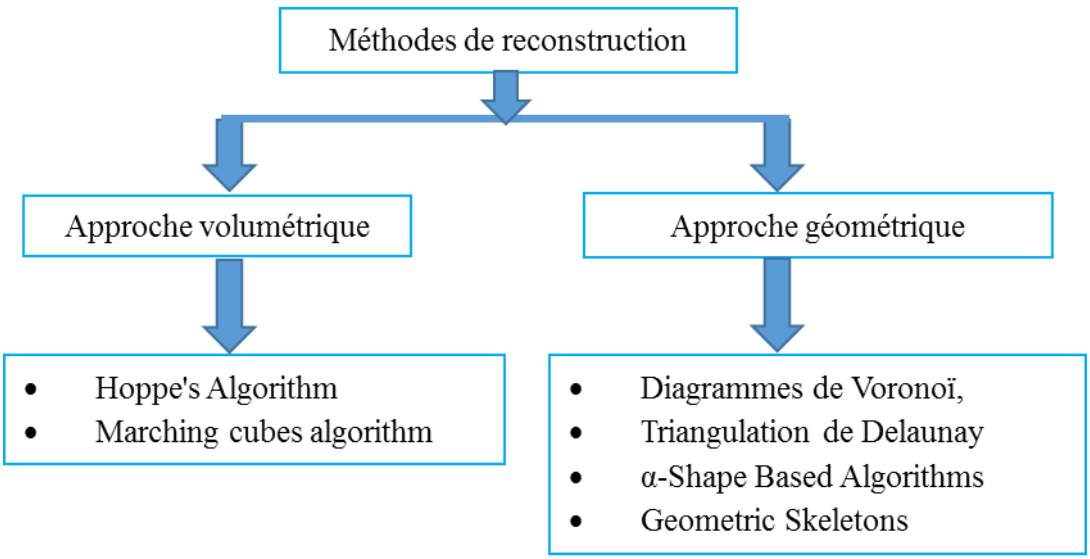


Figure 2 : Classification selon les approches volumétrique et géométrique

2.2.1. Approches géométriques computationnelles

Ces approches prennent en compte la structure de l'objet à reconstruire et sont basées sur des concepts de géométrie computationnelle tels que les diagrammes de Voronoï, la triangulation de Delaunay ou l'axe médian. Ainsi, ces approches construisent d'abord une triangulation initiale. À partir de cette triangulation, le processus de reconstruction sélectionne un sous-ensemble de triangles qui représente la surface reconstruite. Ces différentes approches sont discutées en détail dans ce qui suit :

- **Diagrammes de Voronoï (DV) :** le **DV** pour un ensemble *S* de points, appelés sites, dans l'espace euclidien R_d de dimension *d*, est la subdivision de cet espace en régions, nommées cellules de Voronoï (polyèdres), qui contiennent tous les points de l'espace près du site générateur de la région que des autres sites [53] [54]. Les polyèdres de Voronoï sont convexes, d'intérieurs non vide et forment un recouvrement de sorte que deux éléments distincts ont une intersection vide ou réduite à un point ou une arrête (Figure 3) [55].

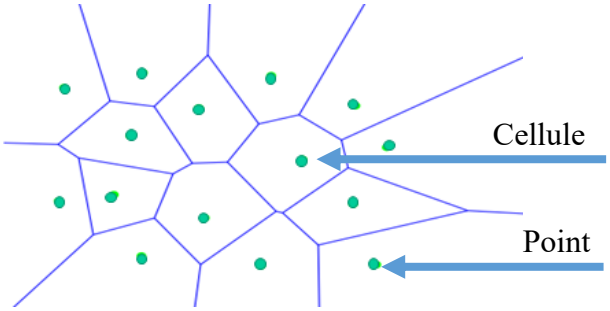
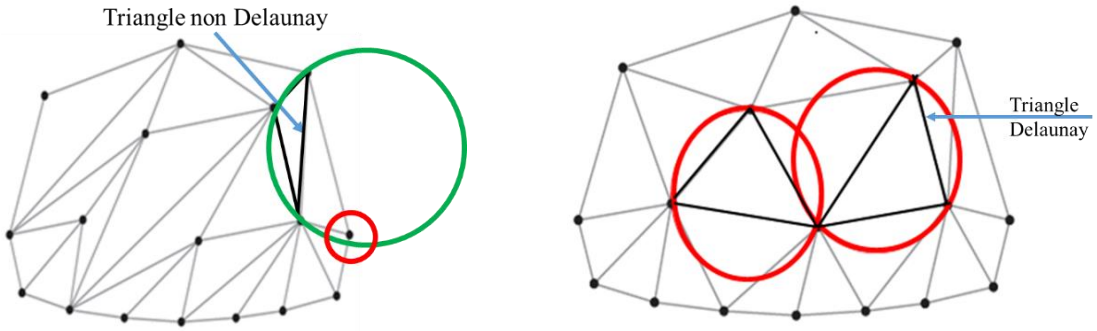


Figure 3 : Diagramme de Voronoï

- **Triangulation de Delaunay :** Soit *S* un ensemble de points en position générale. Nous appelons triangle de Delaunay de *S*, un triangle dont les sommets sont des points de *S* et qui est inscriptible dans un cercle vide ; c'est-à-dire un cercle qui ne contient aucun point de *S* en son intérieur. Une triangulation de *S* est Delaunay si tous les triangles sont des triangles de Delaunay de *S* (Figure 4). Ils existent plusieurs algorithmes qui génèrent la triangulation de Delaunay : algorithme incrémental, algorithme séquentiel et algorithme parallèle.



a. Triangulation quelconque b. Triangulation de Delaunay

Figure 4 : Triangulation de Delaunay vs. Triangulation quelconque

- **Algorithme Alpha forme (α-Shape)** : le principe de cet algorithme est basé sur l'utilisation de la triangulation de Delaunay à partir des points. La première étape consiste à créer une décomposition tétraédrique de l'enveloppe convexe à partir des points. Ensuite, la surface reconstruite est obtenue par la suppression des tétraèdres dont le rayon de la sphère est supérieur à α [55-56]. Un des problèmes de cet algorithme consiste à trouver une valeur α qui permet d'extraire la bonne surface correcte. Si α est trop petit, des lacunes apparaîtront et des

formes solides pourront être déconnectées. Une valeur α trop grande conduira au contraire à une perte de détails dans la forme finale. Par conséquent, α doit être choisi expérimentalement. La méthode est également instable en cas de densité non uniforme des points.

- **Algorithmes basés sur la triangulation de Delaunay et diagramme de Voronoï :** sur la base de deux algorithmes de triangulation de Delaunay et diagramme de Voronoï, plusieurs algorithmes ont été développés : **Crust, Cocone, Tight Cocone, Robust Cocone** et **PowerCrust** [58] et [60-61]. **« Crust »** est le premier algorithme à apparaitre [58]. Ensuite, une amélioration a été apporté sur cet algorithme dans [14] sous le nom « **Cocone** ». L'inconvénient majeur de ces deux algorithmes réside dans le trou laissé sur la surface reconstruite lorsque les nuages de points ne sont pas assez denses ; par conséquent, la surface reconstruite devient alors incomplète. Ultérieurement, d'autres auteurs [59] ont proposé l'algorithme « **Tight Cocone »** qui produit une surface sans trous malgré le fait qu'il ne parvient pas à reconstruire la surface lorsque les points sont bruités. Par la suite, l'algorithme « **Robust Cocone »** a été proposé pour reconstruire la surface à partir des points bruités. Enfin, l'algorithme « **Power Crust »** proposé dans [58] est un autre algorithme capable de reconstruire une surface sans trous à partir des points bien qu'il ne parvient pas à reconstruire la surface lorsque les nuages de points sont bruités.

2.2.2. Approches volumétriques

Les algorithmes volumétriques reposent sur l'idée de transformer la représentation de la surface en nuage de points en représentation volumétrique des objets [51]. Cette dernière peut être une grille 3D de points (uniforme ou non uniforme) qui décrivent une fonction de distance ou une sorte de description mathématique de cette fonction de distance. La fonction de distance est une fonction implicite décrivant l'approximation de la surface M. La surface de l'objet est extraite de la représentation volumétrique à l'aide des méthodes d'extraction d'iso-surface, généralement avec l'algorithme **« Marching cubes »**. Parmi les méthodes de cette sous-classe, nous pouvons citer :

- **Algorithme de Hoppe** : cet algorithme utilise une approximation du plan tangent de la surface pour définir la fonction de distance. Pour chaque point de données, un plan tangent orienté est estimé comme approximation linéaire locale de la surface. Les détails de cet algorithme sont présentés dans [62].
- **Axe médian combiné à la fonction de distance (Medial axis combined with distance function) :** cet algorithme est la combinaison de l'axe médian et de la fonction de distance. Il comporte trois phases : (*i*) incorporation de l'ensemble des points d'entrée dans un volume uniformément subdivisé, (*ii*) détermination de l'axe médian et (*iii*) reconstruction implicite de la surface. Le problème clé de cette méthode est la détermination de la taille de la grille. Une petite résolution entraîne une perte de détails ; une très grande résolution, par contre, peut changer la surface. Les détails de cet algorithme sont donnés dans [63].
- **Approche de Roth & Wibowoo :** cette approche est basée sur l'estimation des normales en chaque point comme l'approche de Hoppe. Les normales sont estimées en utilisant la grille de voxels. Cependant, l'orientation de la normale doit également être déterminée. Ensuite, la distance signée est calculée en utilisant les points de données ainsi que les normales estimées, en prenant la moyenne pondérée des distances signées de chaque point dans les cellules voisines. À la fin, l'algorithme **« Marching cubes »** est utilisé pour extraire la surface. Cet algorithme est décrit dans [64].

2.3.Analyse des techniques de reconstruction de surfaces

Nous avons présenté les techniques de reconstruction de surfaces, à savoir les techniques explicites, implicites, approche avec optimisation, approches géométriques computationnelles et approches volumétriques. Les observations tirées de cette partie sont données comme suit :

- Le choix d'algorithmes de reconstruction de surface à partir des points reste un problème ouvert dans la recherche.

- Certaines méthodes ne peuvent pas traiter de grands ensembles de données ; par conséquent, avant de choisir la méthode appropriée pour le processus de reconstruction, le type de données doit être identifié.
- Les méthodes implicites sont connues par leur temps de calcul élevé ; par exemple, le calcul de la surface des moindres carrés implique la manipulation de matrices.

- Les méthodes explicites consistent à déterminer la représentation mathématique de l'objet ; ceci augmente la complexité de l'algorithme.
- Les approches géométriques computationnelles sont très coûteuses en temps et en mémoire ; toutefois, ce sont les approches les plus fiables et les plus robustes.
- Les méthodes volumétriques souffrent du problème de représentation volumétrique. Une grille de volume uniforme est coûteuse car la taille de la grille doit être suffisamment grande pour capturer chaque détail du modèle. La représentation par fonction implicite nécessite certains paramètres ; par conséquent, elle est couteuse en temps et en mémoire.

Sur la base de cette analyse, chaque algorithme de reconstruction a ses avantages et ses inconvénients. En outre, la triangulation de Delaunay est l'une des techniques de reconstruction de surface la plus ancienne, la plus fondamentale et la plus utilisée dans toutes les nouvelles approches de reconstruction de surfaces à partir d'un nuage de points [65-68]. Tous les détails de la triangulation de Delaunay ainsi que les différentes techniques utilisées sont présentés dans ce qui suit.

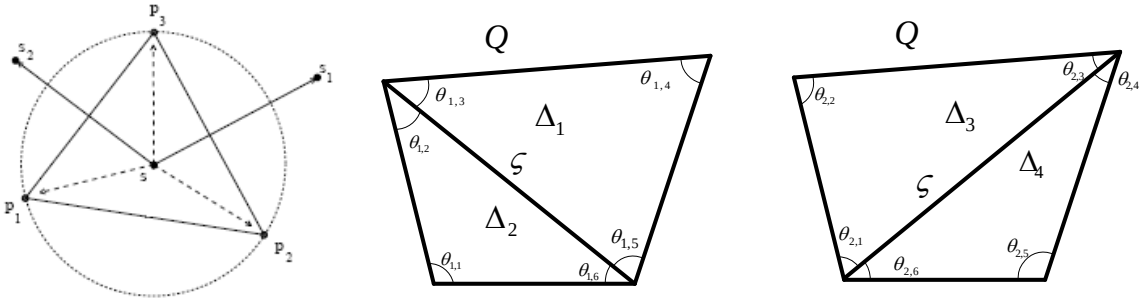
3. Triangulation de Delaunay

Cette triangulation ne tient compte que de la nature géométrique des points donnés, à savoir leurs coordonnées spatiales. Dans ce qui suit, la triangulation de Delaunay, qui est à ce titre une référence, est définie et ses diverses propriétés sont rappelées avec la présentation de quelques types d'algorithmes qui existent pour construire une triangulation de Delaunay [69-70]**.**

3.1.Propriétés de la triangulation dans le plan

La triangulation de Delaunay revêt de nombreuses propriétés géométriques très utiles. Nous citons les plus importantes :

- **Cercles circonscrits :** parmi toutes les propriétés, le cercle circonscrit à un triangle joue un rôle très important dans les algorithmes de triangulation. Ce cercle est défini par son centre et son rayon. Le calcul de ces deux paramètres peut se faire essentiellement de deux manières : soit en utilisant directement l'équation du cercle, soit en calculant le point d'intersection de deux médiatrices du triangle.
- **Maximum de l'angle minimum :** une des propriétés les plus classiques de la triangulation de Delaunay est la maximisation de l'angle minimum de ses triangles (Figure 5). L'utilité de cette propriété apparaît clairement dans les maillages pour la méthode des éléments finis ; dance cas, les triangles avec angles trop petits ne sont pas utiles dans certaines applications. Ceci explique en partie la forte utilisation de Delaunay comme triangulation de base pour engendrer des maillages triangulaires.



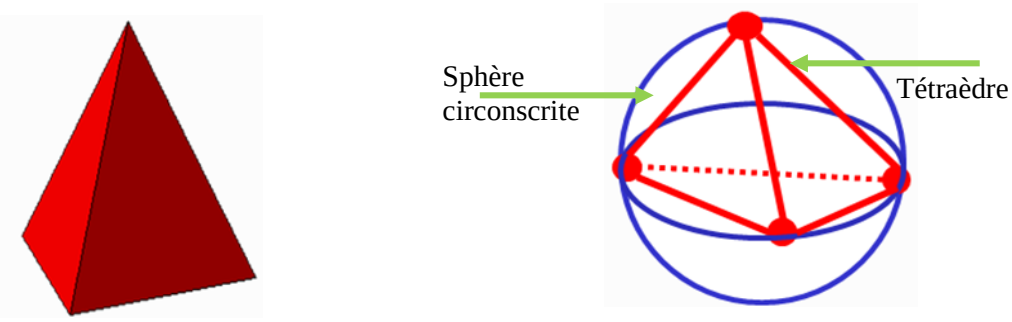
a. Cercles circonscrits b. Triangulations possibles d'un quadrilatère Q

Figure 5 : Angle minimum des triangles

- **Équiangularité** : une des propriétés essentielles est qu'elle est globalement équiangulaire. Ceci signifie que parmi toutes les triangulations possibles, celle de Delaunay donne des triangles les plus équilatéraux possibles. En pratique, ce résultat est essentiel, surtout pour l'utilisation de la triangulation de Delaunay dans la méthode des éléments finis.
- **Enveloppe convexe :** les arêtes extérieures de la triangulation de Delaunay d'un ensemble de points *S* constituent la frontière de l'enveloppe convexe de *S*.

3.2. Triangulation de Delaunay en trois dimensions

En 3D, une triangulation tridimensionnelle de Delaunay est appelée tétraédrisation. Elle revêt de nombreuses propriétés géométriques comme en 2D, sauf que le triangle est remplacé par un tétraèdre, et le cercle circonscrit est remplacé par une sphère circonscrite (Figure 6).



a. Tétraèdre b. Sphère circonscrite

Figure 6 : Triangulation de Delaunay en 3D

3.2.1. Algorithmes à insertion incrémentale

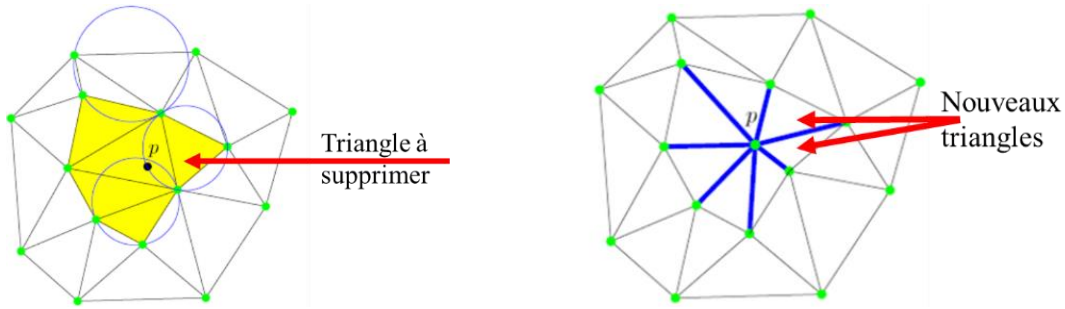
Ce sont les algorithmes les plus simples et ceux pour lesquels la bibliographie est la plus importante. Leur principe est d'ajouter les points les uns après les autres et de mettre à jour la triangulation après chaque insertion [71]. Les deux étapes d'un tel algorithme sont données ainsi :

- **Etape 1** : la localisation permet de trouver le triangle contenant le point à insérer en supposant disposer au départ d'un triangle infini qui contient tous les points du plan. Si le point se trouve en dehors de l'enveloppe convexe, il suffit de réactualiser celle-ci. Le problème est un peu différent lorsque le point est à l'intérieur de l'enveloppe convexe ; il faut alors rechercher le triangle qui le contient.
- **Etape 2** : la mise à jour peut être réalisée en échangeant les arêtes invalidées par l'insertion du nouveau point. Une arête peut être testée en appliquant le critère du cercle vide ; à chaque fois qu'elle est invalidée, elle est échangée avec l'autre diagonale du quadrilatère où elle était située, et de nouvelles arêtes sont ajoutées à la liste de celles qui doivent être testées.

Parmi les algorithmes développés dans la littérature, nous citons les plus importants dans ce qui suit :

3.2.1.1. Approche de Watson [72]

Au lieu de procéder par échange d'arêtes dans la phase de mise à jour, la méthode de Watson consiste à détruire tous les triangles en conflit avec le nouveau point. Ces triangles sont connus par le critère du cercle vide. Les triangles ainsi supprimés forment un polygone étoilé que l'on triangule en reliant le nouveau sommet à chaque arête de sa frontière . La Figure 7 illustre les deux cas qui peuvent se produire lors de l'insertion d'un point pour cette approche.



a. Triangles à supprimer b. Création de nouveaux triangles

Figure 7. Approche de Watson après insertion d'un nouveau point

3.2.1.2.Approche randomisée [73]

Dans cet algorithme, les points sont insérés dans un ordre aléatoire. Cette approche a été utilisée par Boissonnat et Teillaud. Ces derniers ont prouvé que le nombre moyen de swaps (basculement d'arêtes) est linéaire quelle que soit la distribution des points. Pour la phase de localisation, une structure d'arbre est utilisée, arbre de Delaunay, dans laquelle un nœud interne est un triangle qui a été détruit par l'insertion d'un point à un certain moment. Les feuilles de l'arbre contiennent la triangulation courante. Situer un point *p* revient alors à localiser *p* dans toutes les triangulations ayant existées successivement.

3.2.1.3. Algorithme par bascule de diagonales [74]

Etant donne une triangulation *T(S)* des points de *S*, nous pouvons obtenir une autre triangulation *T'(S)* en basculant une diagonale. Cette opération consiste à choisir deux triangles *pqr* et *rqs* adjacents dans *T(S)* tels que (Figure 8) :

- Le quadrilatère *pqrs* soit convexe,
- L'arête *qr* de la triangulation est supprimée,
- L'arête *ps* est ajoutée.

En utilisant suffisamment d'opérations de bascules, nous pouvons ainsi passer de n'importe quelle triangulation de *S* à n'importe quelle autre. Etant donné une triangulation quelconque, nous pouvons alors obtenir la triangulation de Delaunay par une suite de flip (bascule). En fait, si *pqr* et *rqs* sont deux triangules, l'arête *qr* est flippée (basculée) si et seulement si le cercle passant par *p*, *q* et *r* contient *s*. Si cette opération de flips est poursuivie jusqu'à ce que plus aucune arête de la triangulation ne corresponde au critère, une triangulation localement de Delaunay est obtenue et donc la triangulation de Delaunay. Lawson [74] a prouvé que quel que soit l'ordre des échanges considéré, en utilisant le critère du cercle vide, on converge toujours vers la triangulation de Delaunay (Figure 8). Pour construire la triangulation initiale, nous pouvons choisir par exemple la triangulation de poids minimum ou tout autre triangulation généralement obtenue en insérant les points les uns après les autres et en les reliant aux sommets visibles déjà présents.

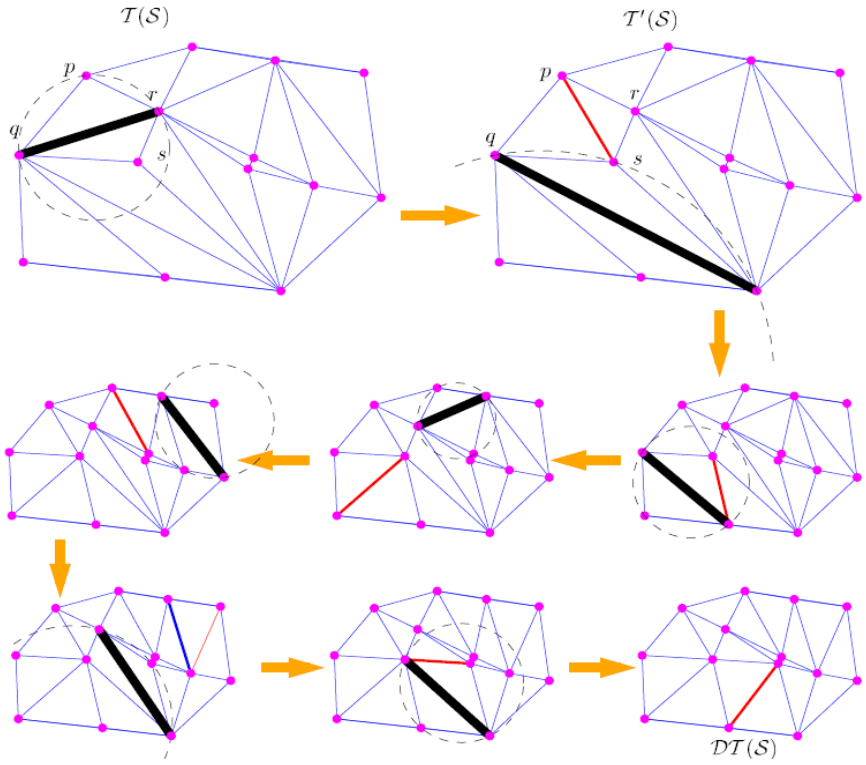


Figure 8. Mise à jour de triangulation après bascule de diagonales

4. Méthodes de reconstruction des frontières d'objets

Le passage des nuages de points vers un modèle facettisé est réalisé en utilisant des techniques de reconstruction d'objets basées sur le concept de la triangulation de Delaunay. Son résultat est l'enveloppe convexe de l'objet. Cette méthode seule ne détermine pas les surfaces frontières de l'objet. Pour remonter à la vraie forme de l'objet, il est indispensable d'utiliser d'autres techniques qui permettent, à partir de la triangulation obtenue, la reconstruction des frontières de l'objet. Plusieurs méthodes ont été développées pour y apporter une solution à ce problème qui s'avère une tâche ardue et couteuse en temps telles que : Graphe de Gabriel, Alpha Shape, Crust, etc.

4.1. Graphe de Gabriel [76]

Le graphe de Gabriel est un sous-graphe de la triangulation de Delaunay. Deux points de l'ensemble d'échantillonnage *S*, *p* et *q* définissent une arrête de graphe de Gabriel *G(S)* si et seulement si le cercle (sphère) du diamètre *pq* ne contient pas de point de *S* [75-76]. *G(S)* est clairement inclus dans la triangulation de Delaunay de *S DT(S)* ; il peut être extrait de *DT(S)* en temps linéaire.

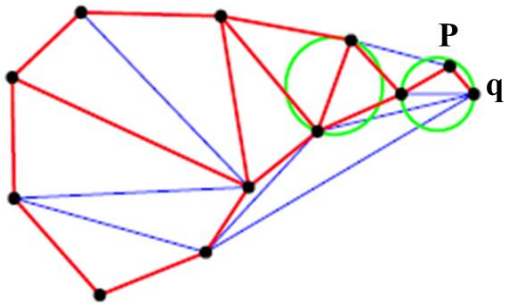
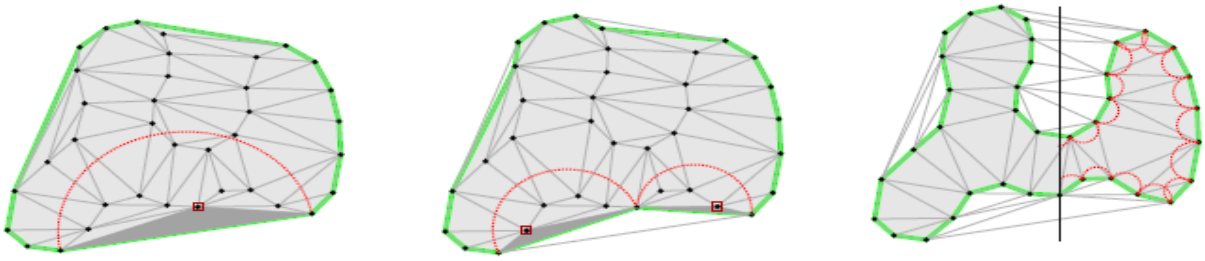


Figure 9 : Exemple d'un graphe de Gabriel

L'exemple ci-dessous illustre le processus de Graphe de Gabriel. Dans la Figure 10a, la courbe est initialisée sur l'enveloppe convexe de l'ensemble de points. L'arête courante, circonscrite par une demi-sphère de Gabriel non vide, forme un triangle de Delaunay (gris foncé) avec le point. Ce triangle

devient externe, la courbe est mise à jour en Figure 10b, et celle-ci continue de se contracter. Le résultat final apparaît en Figure 10c avec des demi-sphères de Gabriel vides.



(a) (b) (c)

Figure 10 : Processus du Graphe de Gabriel

4.2. CRUST [78-78]

Le CRUST est une méthode pour reconstruire une approximation du contour d'un objet 2D ou de la surface d'un objet en 3D à partir d'un ensemble de points. Étant donné un ensemble de points d'échantillonnage *P*, on détermine *T(P)* la triangulation de Delaunay de *P*. Puis, on calcule l'ensemble *C* des centres des cercles circonscrits aux triangles de *T(P)*. Après, on construit une triangulation de Delaunay TD (*P* et *C*) entre des points d'échantillonnage *P* plus l'ensemble des centres des cercles circonscrites *C*. Le CRUST est l'ensemble des arrêtes de triangulation de Delaunay formées uniquement de points d'échantillonnage *P*.

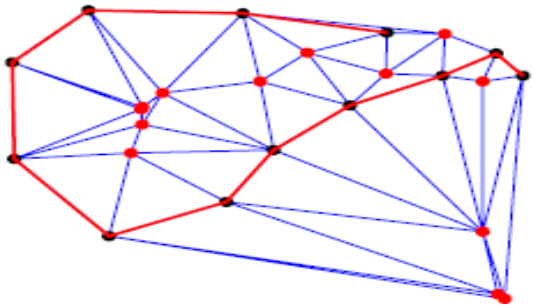


Figure 11 : Exemple de CRUST (les points *P* en noir et les points *C* en rouge)

4.1.1. Principe de la méthode de CRUST

Les principales étapes de la méthode de CRUST sont décrites dans ce qui suit :

- Acquisition d'un nuage de points de départ,
- Calcul du diagramme de Voronoï,
- Calcul des pôles des cellules de Voronoï,
- Construction de la triangulation de Delaunay de l'union des points du nuage et des pôles définis dans l'étape précédente,
- Garder uniquement les triangles dont les trois sommets sont des points de l'échantillon de départ.

4.1.2. Avantages et inconvénients de la méthode de CRUST

Les avantages de cette méthode peuvent s'énumérer comme suit :

- Garantie de converger vers une surface correcte,
- Obtention d'une reconstruction surfacique sans besoin de spécifier des informations sur le processus d'échantillonnage.

Cependant, les principaux inconvénients de cette approche sont donnés ainsi :

- La surface de sortie peut contenir des trous et des défauts au niveau de la reconstruction, si l'exigence de densité de l'échantillon n'est pas respectée,
- Le temps d'exécution de l'algorithme dépend de l'ordre des points d'entrée,
- Incapacité à gérer les virages serrés et les coins.

4.3. Alpha forme (Alpha-Shape) [79]

Soit $P = \{p_1, \ldots, p_N\} \subset R^3$ un ensemble fini de points. Intuitivement, la α-forme de P est constituée de l'ensemble des arêtes et des triangles, ayant pour sommets les points de P, et pour lesquels il existe une boule circonscrite dont l'intérieur ne contient aucun point de P.

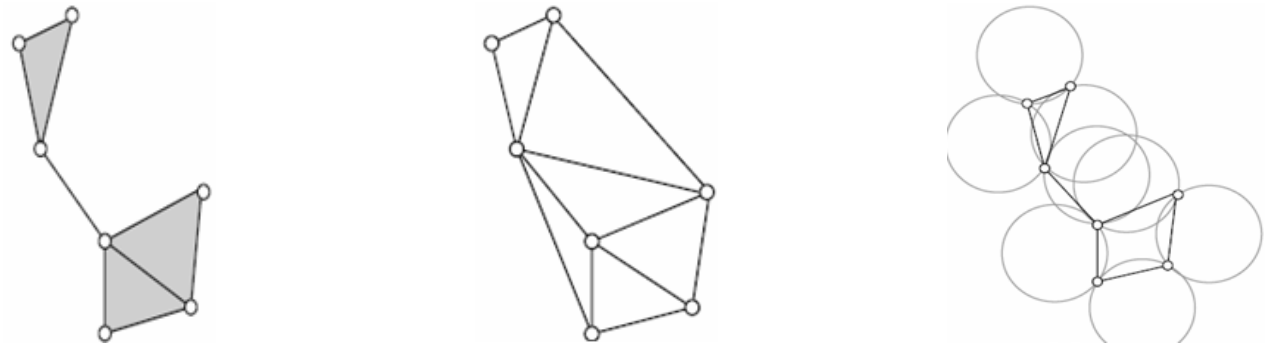
Pour une valeur donnée de $\alpha \in [0, \infty]$, les boules alpha sont des boules de rayon α autour des points de P. Le complexe alpha correspondant de P est la triangulation de Delaunay de P restreinte aux boules alpha. Un simplexe appartient au complexe alpha si les cellules de Voronoï de ses sommets ont une intersection commune non vide avec l'ensemble des boules alpha. Notons qu'à α = 0, le complexe alpha est constitué uniquement de l'ensemble P, et pour α suffisamment grand, le complexe alpha est la triangulation de Delaunay D(P) de P. Pour tout simplex σ ∈ D(P), soit α(σ) la valeur α à laquelle σ apparaît pour la première fois dans le complexe alpha. Le filtrage de la forme alpha est la séquence de complexes alphas obtenus en faisant croître α de zéro à l'infini.

4.3.1. Alpha forme tridimensionnelle (Alpha-Shape 3D) [80-82].

Alpha-Shape est une généralisation de l'enveloppe convexe d'un ensemble fini de points. Soit S un ensemble de points $\subset R^3$. Soit un réel $\alpha > 0$, trois points $p, q, r \in S$ forment une facette de Alpha-Shape de S si et seulement si il existe un cercle ouvert de rayon α vide de tout point de S et dont la frontière contient p, q et r

4.3.2. Alpha complexe [82]

Afin de calculer la forme, le α-complexe est utilisé. Le α-complexe $\boldsymbol{C\alpha}$ est un sous-complexe de la triangulation de Delaunay *DT(S)*. Le sous-ensemble de *DT(S)* est déterminé par $\boldsymbol{\alpha}$. Chaque triangle de $DT(S)$ est dans le α-complexe $C\alpha$ si le cercle circonscrit de T de rayon $r < \alpha$ est vide. Ensuite, la frontière $S\alpha$ se compose de tous les triangles de S qui sont exposés.



a. Complexe alpha Cα b. Triangulation de Delaunay DT(S) c. Frontière Sα

Figure 12 : α-complexe

La méthode alpha-Shape est exécutée selon les trois étapes suivantes :

- **Étape 1** : Calcul de la triangulation de Delaunay,
- **Étape 2** : Génération du complexe Alpha,
- **Étape 3** : Extraction de la frontière de la forme du complexe,
- Les étapes 1 et 2 peuvent être pré-calculées pour un *P* donné.

Figure 13 présente plusieurs approximations de surfaces selon les valeurs d'Alpha. Il est constaté qu'il y a des ensembles de points pour lesquels il n'y a pas de α satisfaisant. Ceci se traduit par le fait que tous les Alpha-Shape ne donnent pas intuitivement de bonnes approximations de la surface de l'objet. C'est le plus souvent le cas lorsque les points ne sont pas uniformément échantillonnés. Ce problème découle du fait que les Alpha-Shape sont basées sur les distances entre

Les points afin de décider quels points seront connectés par des triangles ou des lignes. Les points non-uniformes sont donc relativement inappropriés.

Alpha=100h Alpha=1.5h Alpha=0.5h Alpha=0

Figure 13 : Exemple de quelques Alpha Shape

5. Conclusion

Ce chapitre a présenté les techniques de reconstruction de surfaces les plus importantes, à savoir les techniques explicites, implicites, approches avec optimisation, approches géométriques computationnelles et approches volumétriques. Chaque algorithme de reconstruction a ses propres avantages et inconvénients.

La triangulation de Delaunay est l'une des techniques de reconstruction de surface la plus ancienne, la plus fondamentale et la plus utilisée dans toutes les nouvelles approches de reconstruction de surfaces à partir d'un nuage de points.

Le résultat de la triangulation de Delaunay est l'enveloppe convexe de la pièce. Cependant, ce résultat seul ne permet pas de déterminer les surfaces frontières de la pièce. Pour remonter à la forme réelle de la pièce, des méthodes de reconstruction des frontières d'objets ont été présenté.

En se basant sur l'état de l'art présenté, le chapitre suivant est consacré à la description de la solution proposée pour la reconstruction d'objets définis par des nuages de points non structurés. À cet effet, trois approches de la méthode de triangulation de Delaunay sont proposées et mises en œuvres. Enfin, une approche basée sur la méthode « Alpha Shape » est développée pour remonter à la forme réelle de l'objet.

Chapitre 3
Description des approches développées

1. Introduction

Ce chapitre est consacré à la description de la solution proposée pour la reconstruction d'objets définis par des nuages de points non-structurés. À cet effet, la solution est définie par deux étapes : (*i*) proposition et mise en œuvre de trois approches de la méthode de « *Triangulation de Delaunay* », et (*ii*), développement d'une approche basée sur la méthode « *Alpha Shape* » pour remonter à la forme réelle de l'objet. Les solutions proposées et développées dans le cadre de ce chapitre sont validées sur plusieurs exemples de pièces. Les résultats recueillis sont comparés avec ceux obtenus par d'autres méthodes de la littérature.

2. Solutions proposées pour la triangulation Delaunay 3D

Cette section décrit les méthodologies proposées pour convertir le nuage de points issus d'une digitalisation d'un objet 3D en tétraèdres en utilisant la triangulation de Delaunay tridimensionnelle (Figure 1).

La première étape concerne l'acquisition des données de l'objet scanné avec un fichier de données « .txt » qui contient les coordonnées (x, y, z) des points. Après cela, ces coordonnées sont sauvegardées dans une structure de données établie pour une manipulation plus facile dans les phases ultérieures. Une fois le nuage de points est structuré, la génération de la triangulation de Delaunay tridimensionnelle est réalisée par trois algorithmes :

- Algorithme destruction et construction inspiré de l'approche de Watson,
- Algorithme de Flip inspiré de l'algorithme par bascule de diagonales,
- Algorithme de Flip amélioré.

À la fin, les frontières de l'objet sont générées en utilisant l'algorithme Alpha-Shape.

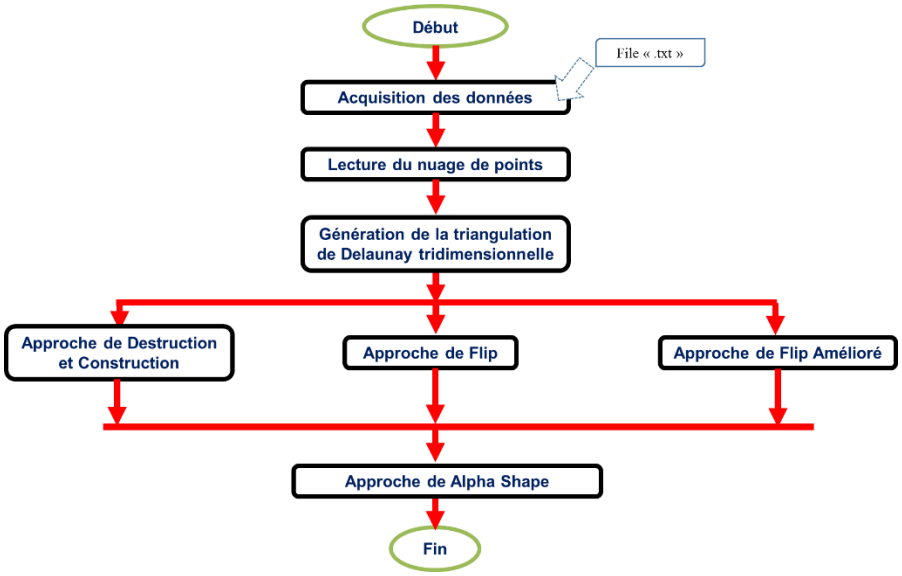


Figure 1 : Diagramme global des solutions proposées pour la triangulation Delaunay 3D

2.1. Approche de Destruction et construction

Cette solution permet de convertir le nuage de points issus d'une digitalisation d'un objet 3D en un ensemble de tétraèdres en utilisant l'approche de Watson de la triangulation de Delaunay 3D. Elle se compose des quatre étapes suivantes (Figure 2) :

- Filtrage du nuage de points,
- Génération du super-tétraèdre,
- Insertion de points et mise à jour de la triangulation,
- Génération de la triangulation finale.

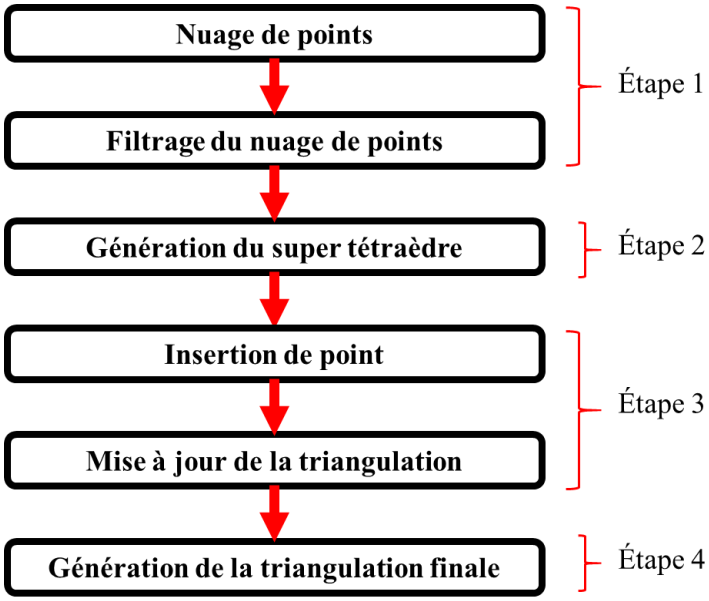
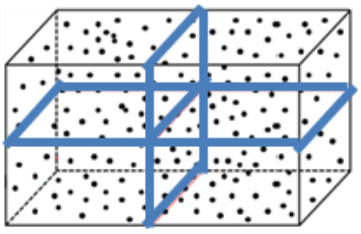


Figure 2 : Approche de Destruction et construction

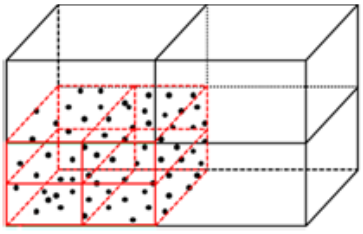
2.1.1. Filtrage des nuages de points

Dans le but d'éviter les cas dégénérés lors de la génération de la triangulation de Delaunay à partir d'un nuage de points non-structuré, le nuage de points est filtré. L'opération de filtrage consiste à déterminer les points redondants présents dans le nuage de points puis les éliminer (Figure 3). Pour faciliter la manipulation du nuage de points et réduire le temps de calcul, le nuage de points est simplifié par la spécification d'une distance minimale entre deux points. Cette simplification est nécessaire si le nuage de points est très dense. Pour cela, une sphère est créée à chaque point dont le rayon est égal à la distance spécifiée et le centre est le point en question. Par la suite, tout point dont sa distance par rapport à ce point est inférieure à la distance spécifiée est filtré. Après, les coordonnées des points extremums du brut (X_{max}, X_{min}, Y_{max}, Y_{min}, Z_{max}, Z_{min}) ainsi que ses dimensions (*longueur*, *hauteur* et *largeur*) sont calculées.

- **Création de cellules** : la pièce brute est subdivisée en blocs parallélépipédiques de même taille. Pour cela, il faut spécifier le nombre de cellules ou de pas le long des axes X, Y et Z. Ensuite, sur la base des dimensions des cellules et des coordonnées des points extremums, les points sont affectés aux cellules (Figure 3a).
- **Création de patchs** : pour réduire les temps de traitement, chaque cellule est subdivisée en petites cellules « patchs » après avoir spécifié le nombre de patchs ou d'étapes le long des axes X, Y et Z. Après, les points appartenant à chaque patch sont affectés aux cellules (Figure 3b). Enfin, les points appartenant à chaque patch sont identifiés (Figure 3b).
- **Filtrage des points** : une sphère de rayon égal à la distance spécifiée est créée pour chaque point et centrée sur ce point. Après, tous les points appartenant à cette sphère sont filtrés (Figure 3c). Cette tâche est effectuée en trois étapes dans cet ordre (Figure 3d) :
  - Filtrage des points pour chaque patch,
  - Filtrage des points pour les patchs adjacents dans la même cellule,
  - Filtrage des points pour les cellules adjacentes.



a. Création des cellules



b. Création des patchs

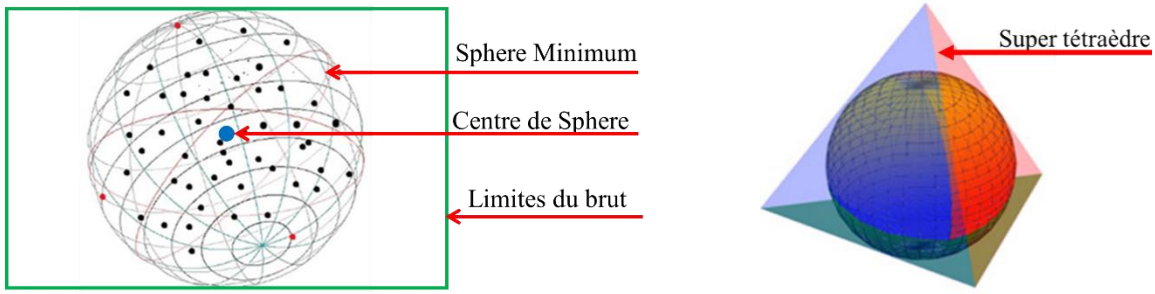
c. Suppression des points d. Nuages de points résultant

Figure 3 : Filtrage du nuage de points

2.1.2. Génération de super-tétraèdres

Pour s'assurer que chaque point inséré appartient à un tétraèdre, il faut créer un tétraèdre virtuel appelé « *supe-tétraèdre* ». Avant de commencer la triangulation de Delaunay, un super-tétraèdre qui contient tous les points filtrés est généré. Pour cela, ses quatre sommets virtuels sont calculés à partir du nuage de points. Leur détermination passe par les deux étapes suivantes :

- **Détermination de la super-sphère** : elle consiste à déterminer la sphère minimale contenant tous les points du nuage de points (Figure 4a). Son centre ***CE*** est le centre de la partie brute et son rayon est égal à la distance entre son centre et tout sommet de la partie brute.
- **Détermination des quatre sommets virtuels** : pour les déterminer, les quatre plans du super-tétraèdre doivent être tangents à la super-sphère (Figure 4b). Une fois le super-tétraèdre créé, son centre et son rayon sont calculés et utilisés dans la phase de triangulation.



(a) Super-sphère (b) Super-tétraèdre

Figure 4 : Création du super-tétraèdre et détermination de ses sommets

2.1.3. Génération de la triangulation 3D

La génération de la triangulation 3D de Delaunay et l'approche de destruction/construction suivent les étapes suivantes (Figure 5) :

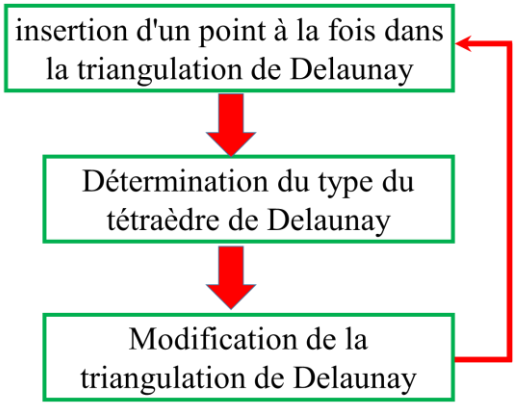
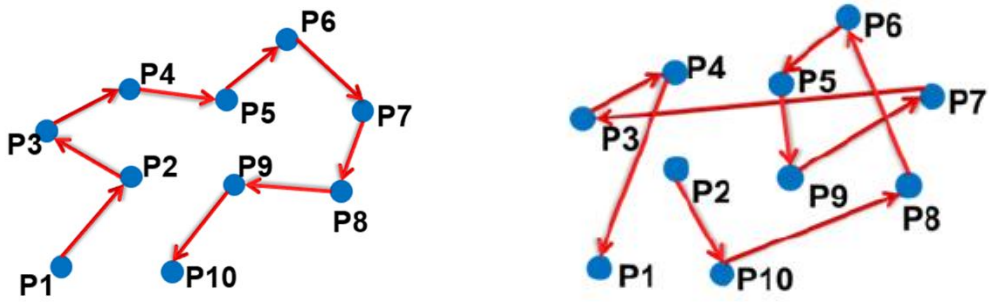


Figure 5 : Génération de la triangulation de Delaunay 3D

- **Insertion de points** : la génération de la triangulation 3D se fait par l'insertion d'un point à la fois. Deux modes d'insertion peuvent être utilisés : séquentiel (Figure 6a) ou aléatoire (Figure 6b). En pratique, l'insertion aléatoire est plus efficace que l'insertion séquentielle.



a. Mode séquentiel b. Mode aléatoire

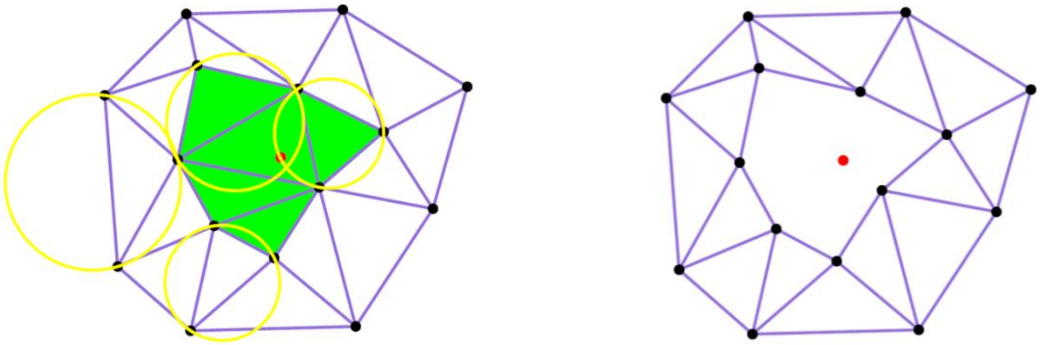
Figure 6 : Modes d'insertion des points

L'insertion du premier point dans le super-tétraèdre est un cas particulier puisque le super-tétraèdre est supprimé (Figure 7a) et remplacé par quatre nouveaux tétraèdres valides (Figure 7b).

a. Insertion du premier point b. Triangulation 3D initiale

Figure 7 : Initialisation de la triangulation

- **Types de tétraèdres de Delaunay** : lorsqu'un point est inséré, tous les tétraèdres dont les sphères circonscrites contiennent ce point inséré sont identifiés (Figure 8a). Ces tétraèdres sont stockés et ensuite supprimés (Figure 8b). Le type de tétraèdre est déterminé en fonction de la distance entre le point inséré et le centre de la sphère circonscrite du tétraèdre. Deux cas sont possibles :
  - Si la distance est inférieure au rayon, le tétraèdre est non-Delaunay,
  - Dans le cas contraire, le tétraèdre est Delaunay.



a. Tétraèdre non-Delaunay b. Suppression des tétraèdres non-Delaunay

Figure 8 : Identification et suppression des tétraèdres non Delaunay

- **Modification de la triangulation 3D** : la triangulation est mise à jour après chaque insertion d'un nouveau point. Pour chaque insertion, les faces des tétraèdres à supprimer sont stockées dans une liste sans répétition. Ensuite, ces tétraèdres sont supprimés. Sur la base de la liste des faces stockées et du point inséré, de nouveaux tétraèdres valides sont créés (Figure 9).

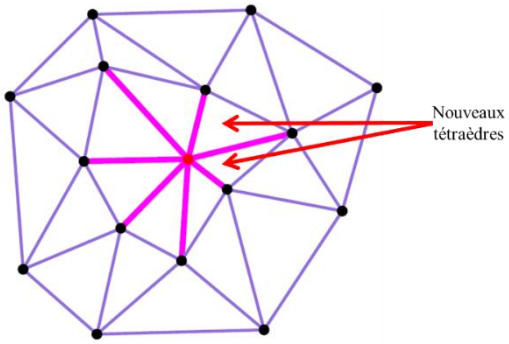
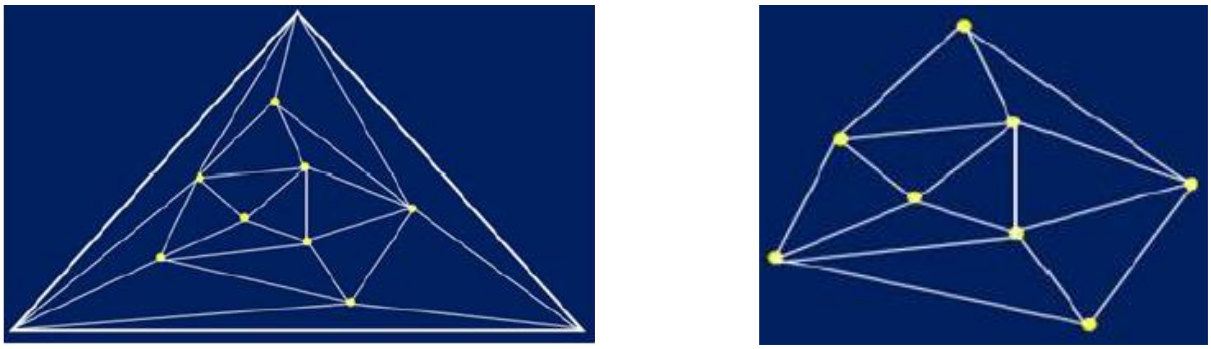


Figure 9 : Modification de la triangulation 3D

2.1.4. Génération de la triangulation finale

Après avoir inséré tous les points filtrés, chaque tétraèdre partageant au moins un sommet virtuel est retiré de la triangulation valide (Figure 10a). Les tétraèdres restants représentent la triangulation finale du nuage de points (Figure 10b).



a. Triangulation 3D avec des sommets virtuels b. Triangulation 3D finale

Figure 10 : Triangulation de Delaunay finale

2.2. Approche de FLIP

Dans cette section, une solution est proposée pour convertir le nuage de points en tétraèdres en utilisant la triangulation de Delaunay par FLIP. La démarche est composée de trois étapes illustrées dans la Figure 11 :

- Lecture du nuage de points,
- Décimation du nuage,
- Génération des tétraèdres.

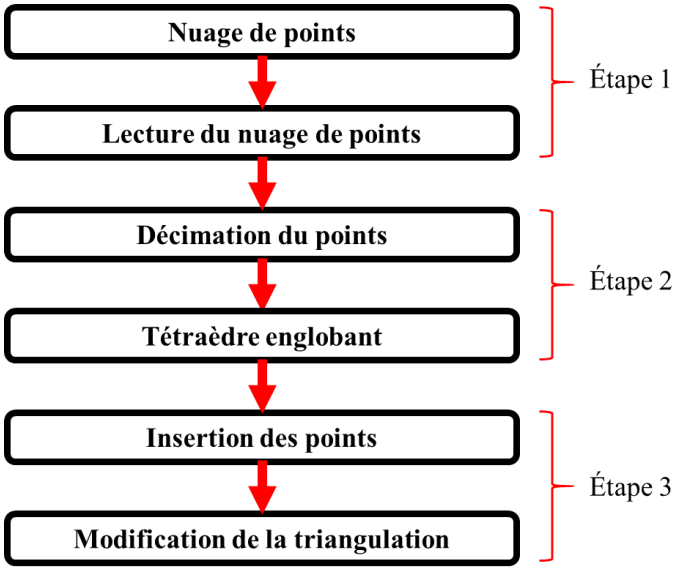


Figure 11 : Démarche de l'approche de FLIP

2.2.1. Lecture du nuage de points aléatoire

Cette étape consiste à vérifier la conformité syntaxique du fichier de base. Si le fichier n'est pas valide, le nuage de points n'est pas traité. Cette conformité se traduit par une représentation unique et correcte de l'objet digitalisé où chaque point est défini par ses coordonnées (x, y, z). Afin de faciliter la triangulation, les coordonnées des points du nuage non-structuré sont récupérées et stockées dans une liste de points.

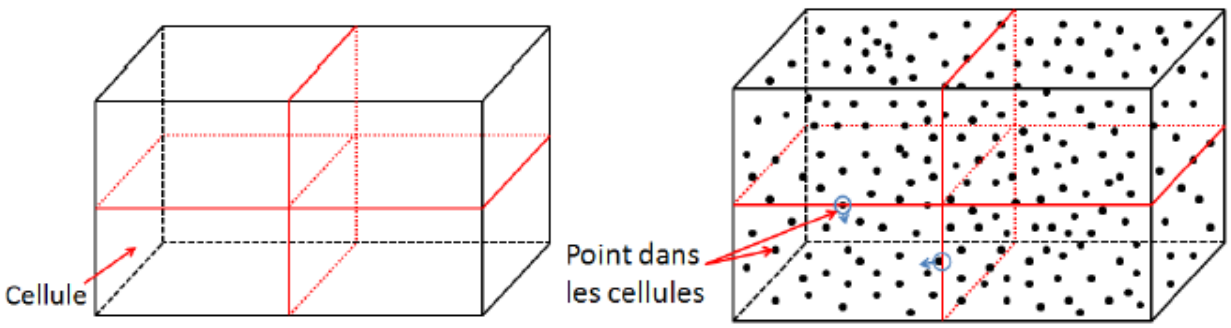
2.2.2. Décimation du nuage de points

Après lecture du fichier de points, celui-ci est simplifié pour faciliter sa manipulation et son traitement. L'objectif de cette étape est la réduction du temps de reconstruction des surfaces. Pour atteindre cet objectif, la solution suivante est proposée.

2.2.2.1. Création des cellules de points

Cette étape consiste à subdiviser le brut du nuage de points en cellules de points de même taille. A cet effet, les étapes suivantes sont suivies (Figure 12) :

- Calculer la hauteur, la longueur et la largeur du brut,
- Spécifier le nombre de cellules suivant les trois axes X, Y et Z,
- Créer et enregistrer les cellules dans un tableau dynamique à trois dimensions,
- Affecter les points aux cellules correspondantes.

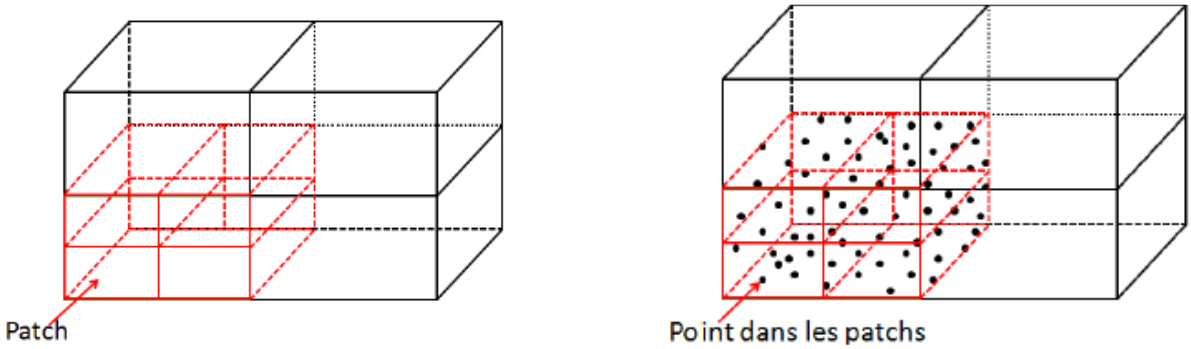


a. Cellules b. Points des cellules

Figure 12 : Création des cellules de points

2.2.2.2. Création des patchs des points

Cette opération permet de subdiviser chaque cellule en sous-cellules appelées patchs pour faciliter l'opération de simplification (Figure 13). La création des patchs et l'affectation des points à ces patchs est similaire à celle des cellules.



a. Patchs d'une cellule b. Points des patchs d'une cellule

Figure 13 : Création de patchs de points

2.2.2.3. Simplification du nuage de points

La simplification du nuage de points est réalisée en deux étapes. Dans la première étape, les points doubles sont déterminés et supprimés. Lors de la deuxième étape, les points dont la distance entre eux est inférieure à un seuil *R* de filtrage (spécifié par l'utilisateur) sont identifiés et supprimés (Figure 14). Cette simplification est effectuée pour chaque patch du modèle, chaque cellule, inter-patchs et inter-cellules.

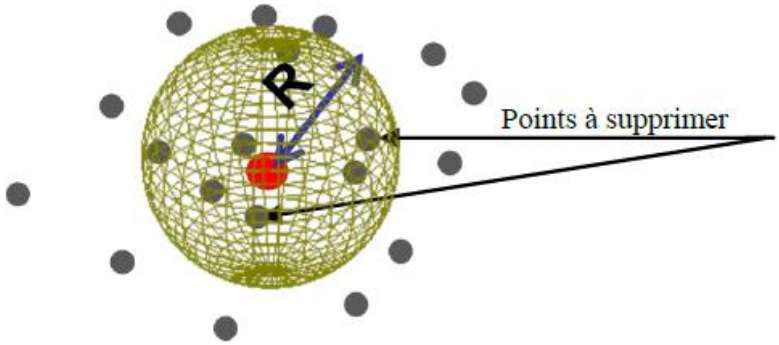


Figure 14 : Simplification de nuage de points

2.2.3. Génération des tétraèdres par la méthode de FLIP

Cette étape consiste à remplacer le nuage de points par des tétraèdres en passant par les étapes suivantes :

2.2.3.1. Génération du tétraèdre englobant

Le tétraèdre englobant le nuage de points est généré selon les étapes décrites par la Figure 15 ci-après :

- Déterminer le centre et le rayon de la sphère englobant les points du nuage,
- Déterminer quatre plans tangents à la sphère,
- Calculer les quatre intersections des quatre plans pour définir les sommets du tétraèdre englobant (sommets fictifs),
- Calculer le centre et le rayon de la sphère circonscrite du tétraèdre englobant.

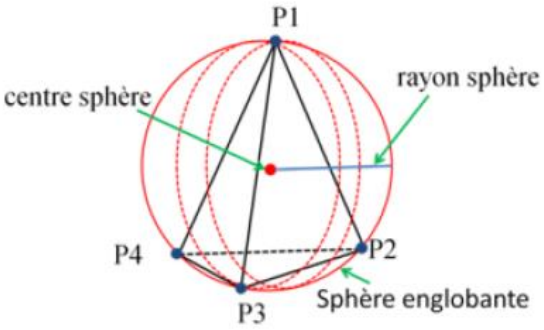


Figure 15 : Tétraèdre englobant

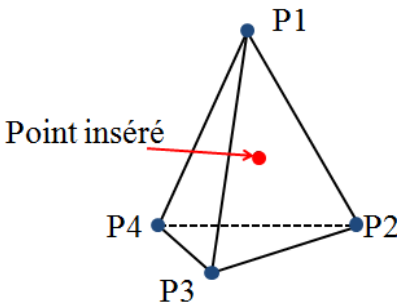
2.2.3.2. **Insertion d'un point**

Les points sont insérés suivant le mode séquentiel ou le mode aléatoire. Généralement, le dernier mode est utilisé puisqu'il réduit le temps de la triangulation.

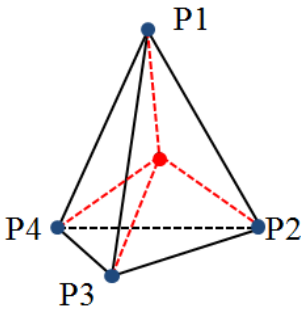
2.2.3.3. **Subdivision d'un tétraèdre**

Après sélection d'un point à ajouter, son insertion est réalisée en deux étapes comme suit (Figure 16) :

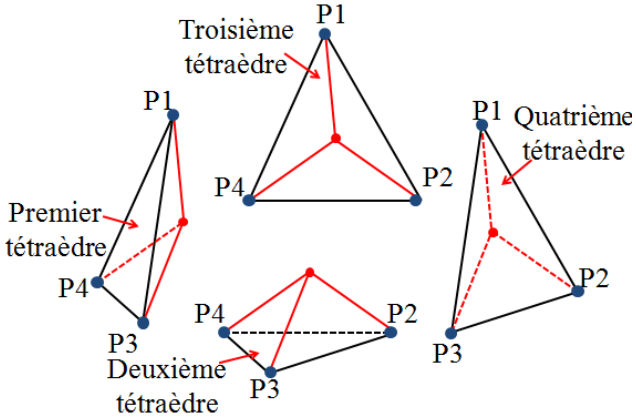
- Identification du tétraèdre *T* contenant ce point,
- Subdivision du tétraèdre *T* en le remplaçant par quatre nouveaux tétraèdres.



a. Point appartenant au tétraèdre



b. Création de quatre tétraèdres



c. Quatre tétraèdres créés

Figure 16 : Subdivision d'un tétraèdre

2.2.3.4. **Vérification du critère de Delaunay**

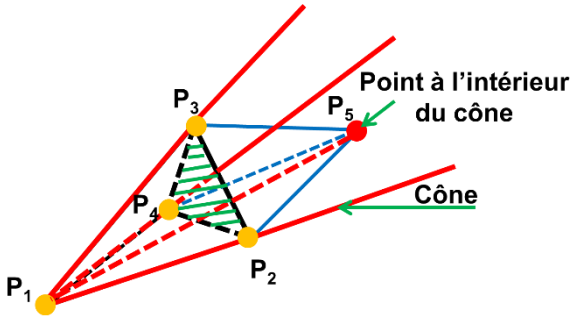
L'insertion d'un point aux tétraèdres peut engendrer des tétraèdres ne respectant pas le critère de Delaunay. Pour cela, une étape de vérification est indispensable. Elle se résume comme suit. Pour chaque tétraèdre, déterminer le rayon et le centre de la sphère circonscrite, et vérifier l'appartenance d'un point de ses voisins à la sphère circonscrite.

- Si oui, alors effectuer un « FLIP » entre ce tétraèdre et son voisin,
- Dans le cas contraire, valider ce tétraèdre et passer au tétraèdre suivant.

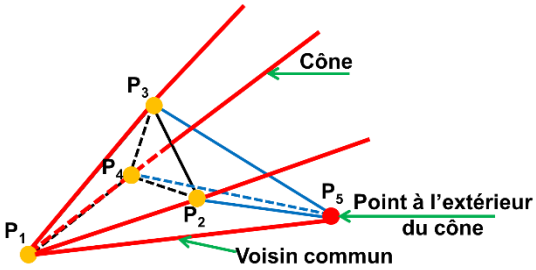
2.2.3.5. Réalisation de l'opération de FLIP

L'opération de FLIP est exécutée dans le cas où le critère de Delaunay n'est pas vérifié. Pour ce faire, trois modes de FLIP sont envisagés, à savoir « FLIP 2-3 », « FLIP 3-2 » et « FLIP 4-4 ». Pour déterminer quel mode de FLIP à utiliser, un test sur un cône formé à partir d'un tétraèdre est effectué selon les étapes suivantes :

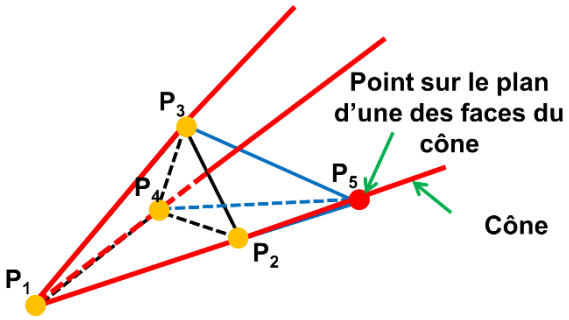
- Déterminer le cône pour ce tétraèdre tel que montré par la Figure 17,



a. Dans le cône



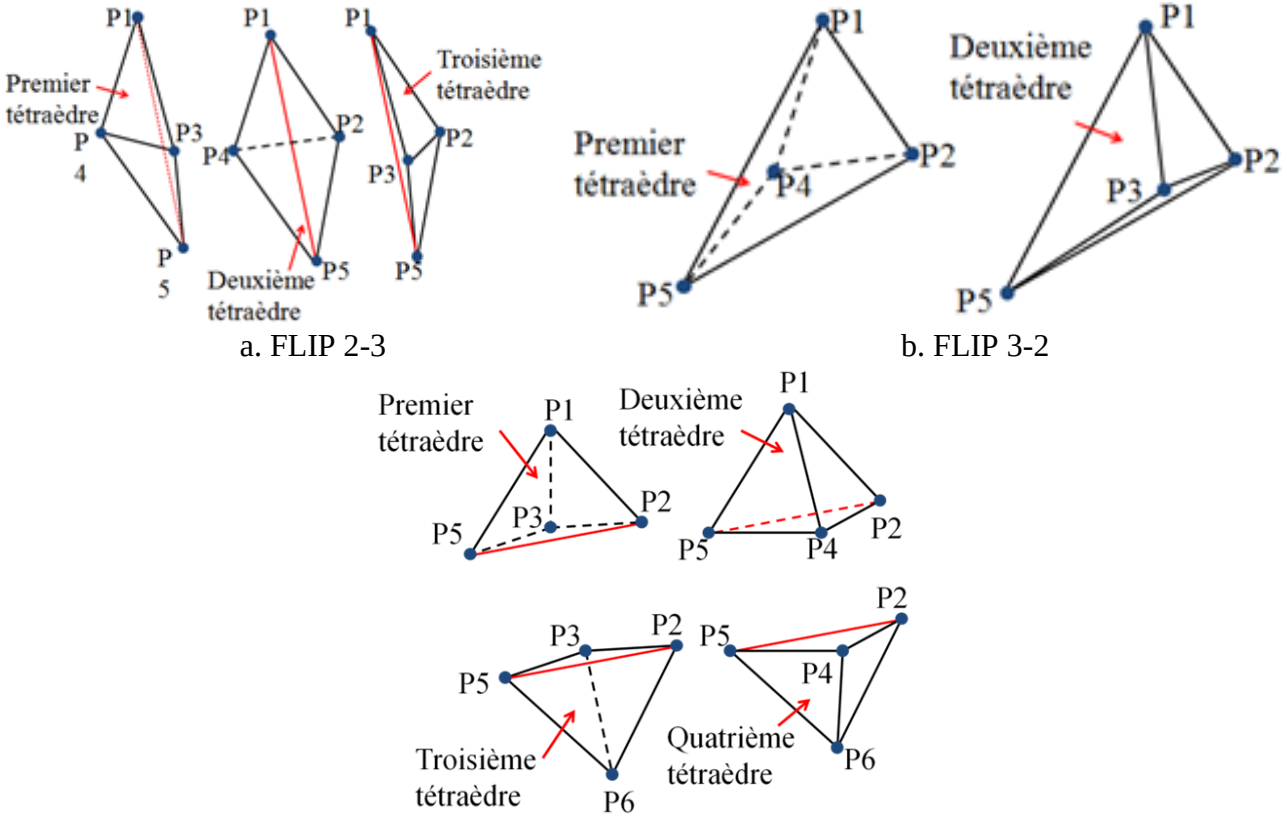
b. En dehors du cône



c. Sur le cône

Figure 17 : Cône et position du point p5

- Identifier le voisin de ce tétraèdre,
- Vérifier de quel côté se trouve le point P5 par rapport au cône,
- Si le point P5 est à l'intérieur du cône, le FLIP 2-3 est effectué ; par conséquent, les deux tétraèdres sont remplacés par trois nouveaux tétraèdres (Figure 18a),
- Si le point P5 est à l'extérieur du cône, une recherche d'un tétraèdre voisin entre les deux premiers tétraèdres est effectuée,
  - Si oui, le FLIP 3-2 est exécuté en remplaçant ces trois tétraèdres par deux nouveaux tétraèdres (Figure 18b).
  - Dans le cas contraire, le FLIP ne sera pas traité et d'autres FLIP vont régler ce problème.
- Si le point P5 est sur le plan d'une facette du tétraèdre, le FLIP 4-4 est exécuté comme montré par la Figure 18c :
  - Identifier le tétraèdre voisin T_j de cette facette,
  - Chercher un voisin commun entre le tétraèdre contenant le point P5 et le tétraèdre T_j,
    - Si un voisin existe :
      - ✓ Déterminer les points communs pour créer de nouveaux tétraèdres,
      - ✓ Créer quatre nouveaux tétraèdres à partir des quatre initiaux,
      - ✓ Supprimer les deux tétraèdres initiaux,
    - Autrement, ce cas ne sera pas traité et d'autres FLIP vont régler ce problème.



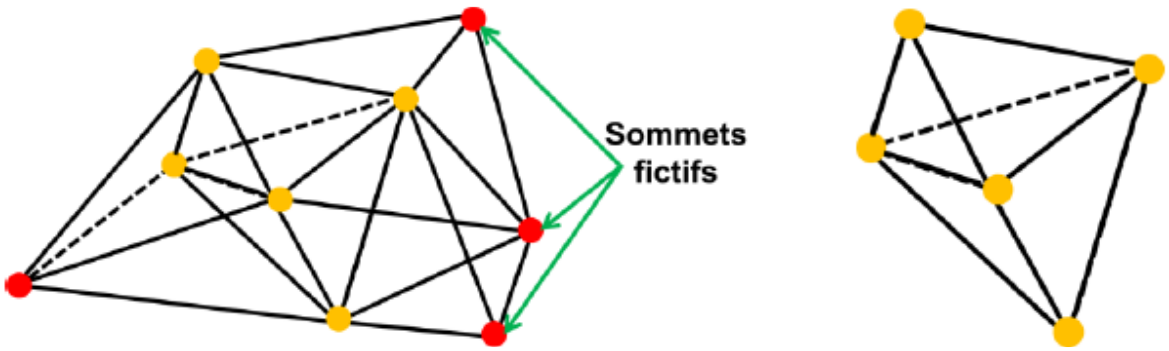
a. FLIP 2-3 b. FLIP 3-2

c. FLIP 4-4

Figure 18 : Différents modes de FLIP

2.2.3.6. **Suppression des points fictifs**

Après l'insertion de tous les points, l'ensemble des tétraèdres partageant un sommet fictif sont supprimés pour obtenir les tétraèdres finaux (tétraédrisation finale) telle que montrée par la Figure 19.



(a) Triangulation de Delaunay 3D (b) Triangulation finale

Figure 19 : Tétraédrisation finale

2.3. Approche de Flip amélioré

La génération de la triangulation de Delaunay à l'aide des flips est un processus lent. En effet, pour insérer un point dans la triangulation, tous ses tétraèdres sont consultés pour déterminer celui qui contient le point à ajouter. De plus, après l'identification des tétraèdres de Delaunay non valides, la modification par séquence de FLIP est lente. Pour pallier à cet inconvénient, nous avons proposé trois solutions :

- Calcul amélioré du prédicat In-Sphere (Critère de Delaunay),
- Recherche intelligente et rapide du tétraèdre cible contenant le point à ajouter,
- Mise à jour améliorée du voisinage pour accélérer la modification des tétraèdres non valides.

2.3.1. Calcul amélioré du prédicat In-Sphere

Tous les algorithmes de construction d'une triangulation de Delaunay sont basés sur des opérations rapides pour déterminer la position d'un point par rapport à un cercle circonscrit à un triangle. En 3D, cela consiste à déterminer si un point donné *e* se situe à l'intérieur, à l'extérieur ou sur la sphère circonscrite du tétraèdre défini par quatre points (a, b, c, d). Cette évaluation est effectuée à l'aide du prédicat *In-Sphere* qui calcule le signe du déterminant donné par l'équation (1) :

$$\text{In-Sphere}(a,b,c,d,e) = \begin{vmatrix} a_x & a_y & a_z & \|a\|^2 & 1 \\ b_x & b_y & b_z & \|b\|^2 & 1 \\ c_x & c_y & c_z & \|c\|^2 & 1 \\ d_x & d_y & d_z & \|d\|^2 & 1 \\ e_x & e_y & e_z & \|e\|^2 & 1 \end{vmatrix} \qquad (1)$$

Il faut mentionner que cette opération est gourmande en temps de calcul. Afin de la rendre plus efficace, nous proposons de calculer et de stocker les deux paramètres définissant le tétraèdre lors de sa création (*i*) centre P_{Centre} et (*ii*) rayon $R_{Sphère}$ de la sphère circonscrite. Ensuite, lors de la vérification du critère de triangulation de Delaunay, la distance est calculée pour vérifier l'appartenance du point à la sphère circonscrite (équation (2)). Certainement, le nombre d'opérations pour déterminer la position d'un point par rapport à la sphère circonscrite devient négligeable par rapport à l'utilisation du prédicat *In-Sphere*.

$$Distance = \sqrt{(X_e - X_{centre})^2 + (Y_e - Y_{centre})^2 + (Z_e - Z_{centre})^2} \qquad (2)$$

2.3.2. Recherche intelligente du tétraèdre cible

Pour améliorer la recherche du tétraèdre qui contient le point ajouté, un tétraèdre de départ est sélectionné comme le montre le schéma de la Figure 20. L'algorithme commence par la définition de la sphère entourant l'ensemble des points. Pour cela, le rayon de la zone R_{Zone}, qui représente la valeur maximale entre les dimensions de la partie brute, est introduit. Ensuite, l'enveloppe de la sphère est calculée. A ce moment, le tétraèdre de départ peut être identifié en testant le recouvrement des tétraèdres un par un avec l'enveloppe de la sphère. Le tétraèdre qui se chevauche sera considéré comme le tétraèdre de départ.

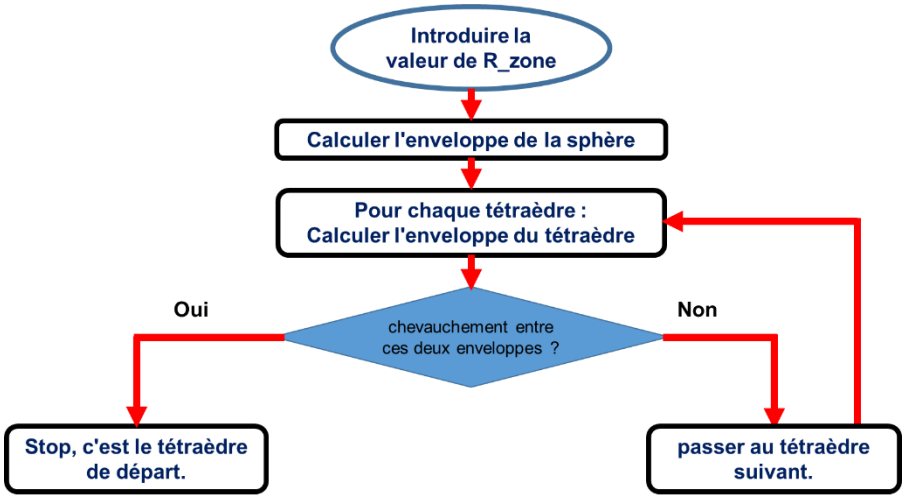


Figure 20 : Sélection du tétraèdre de départ

L'algorithme 1 donné par la Figure 21 est proposé pour identifier le tétraèdre contenant le point inséré sur la base du tétraèdre de départ. Les coordonnées du point inséré et du tétraèdre de

départ sont introduites. Ensuite, cinq vecteurs $\vec{V}$, $\overrightarrow{S_1}$, $\overrightarrow{S_2}$, $\overrightarrow{S_3}$ et $\overrightarrow{S_4}$ sont construits en utilisant ces données. Par la suite, quatre produits scalaires sont calculés et testés comme $\vec{V}.\overrightarrow{S_1}$, $\vec{V}.\overrightarrow{S_2}$, $\vec{V}.\overrightarrow{S_3}$ et $\vec{V}.\overrightarrow{S_4}$. Dans le cas où ces produits ont le même signe, ce tétraèdre est affecté comme tétraèdre cible. Autrement (un des produits a un signe différent), un autre tétraèdre est testé jusqu'à en trouver le bon.

Algorithme 1 : Recherche du tétraèdre cible

Introduire les coordonnées du point inséré ;
Introduire le tétraèdre de départ ;
Construire un vecteur $\vec{V}$ du centre du tétraèdre de départ au point inséré ;
Construire quatre vecteurs$\overrightarrow{S_1}$, $\overrightarrow{S_2}$, $\overrightarrow{S_3}$, $\overrightarrow{S_4}$ (chaque vecteur est le centre du tétraèdre de départ à une normale de sa face) ;
Calculer le produit scalaire entre $\vec{V}.\overrightarrow{S_1}$, $\vec{V}.\overrightarrow{S_2}$, $\vec{V}.\overrightarrow{S_3}$ et $\vec{V}.\overrightarrow{S_4}$;
Tester le signe des quatre produits scalaires ;
si tous les produits scalaires ont le même signe, ce tétraèdre est la cible ;
si un des produits scalaires a un signe différent, passer au tétraèdre suivant de la liste ;

Figure 21 : Algorithme de recherche du tétraèdre cible

2.3.3. Mise à jour améliorée des tétraèdres voisins

Lorsque les bascules des diagonales sont appliquées, certains tétraèdres sont supprimés et de nouveaux sont créés et ajoutés pour obtenir un maillage valide. Par conséquent, les tétraèdres voisins ont changé. Il est donc nécessaire de mettre à jour les voisins de chaque tétraèdre. Comme cette tâche est lente, elle est améliorée en localisant la zone de modification des voisins ; cette localisation est réalisée en deux niveaux (Figure 22) :

- **Premier niveau** : création d'un tableau de voisins qui contient les voisins de chaque tétraèdre créé,
- **Deuxième niveau** : création d'un tableau de zones pour identifier les voisins de chaque voisin.

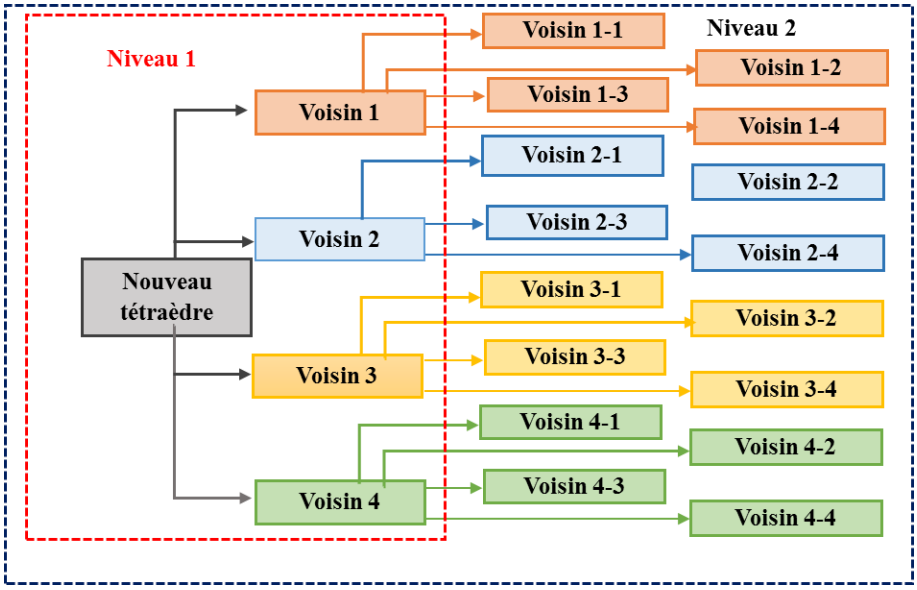


Figure 22 : Mise à jour des voisins du tétraèdre

2.4. Approche « Alpha-Shape »

Le résultat de la triangulation de Delaunay est l'enveloppe convexe de la pièce ; ce résultat seul ne détermine pas les surfaces frontières de la pièce. Pour remonter à la forme réelle de la pièce, une solution est proposée. Elle consiste à déterminer les surfaces frontières de la pièce à partir de la

triangulation de Delaunay obtenue en utilisant la méthode « Alpha-Shape ». La solution est composée de trois étapes (Figure 23) :

- Vérification et lecture de la triangulation de Delaunay,
- Détermination des tétraèdres et des facettes frontières,
- Application de la méthode « Alpha-Shape ».

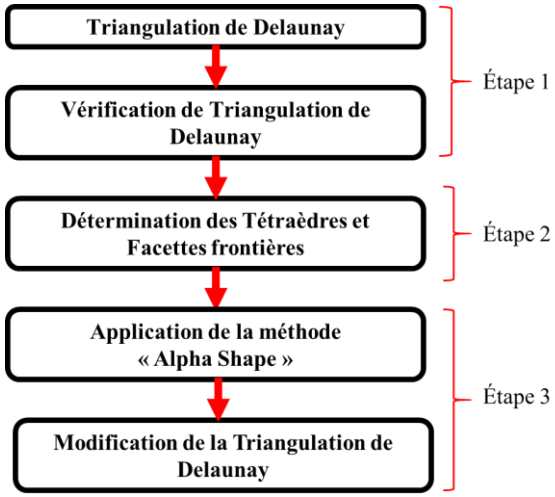
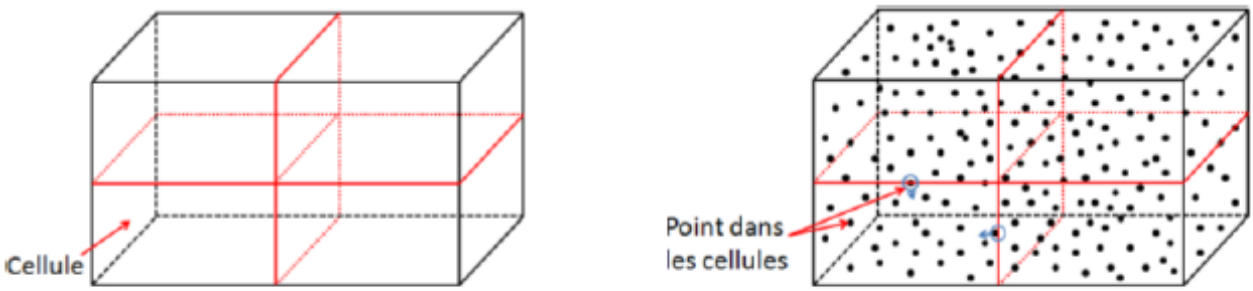


Figure 23 : Solution proposée pour la détermination des surfaces frontières de la pièce

2.4.1. Vérification du fichier et lecture de la triangulation 3D

La vérification du fichier de données consiste à examiner la conformité syntaxique du fichier de base. Par la suite, les points extrémums du brut de l'objet et ses dimensions minimales (*hauteur*, *longueur*, *largeur*) sont récupérés. Pour stocker les tétraèdres de la triangulation de Delaunay et éliminer la redondance des points, le brut est subdivisé en cellules rectangulaires identiques selon la démarche suivante (Figure 24) :

- Spécifier le nombre de cellules suivant les trois axes X, Y et Z,
- Créer et enregistrer les cellules dans un tableau dynamique à 3D,
- Stocker les tétraèdres un par un dans un tableau de tétraèdres,
- Affecter les points de chaque tétraèdre aux cellules adéquates,
- Affecter à chaque point les indices des tétraèdres le partageant.



a. Cellules b. Points des cellules

Figure 24 : Création des cellules de points

2.4.2. Détermination des facettes et des tétraèdres frontières

Le principe de la méthode « Alpha Shape » est basé sur les parties visibles de la triangulation de Delaunay. Pour cela, il est indispensable de déterminer les facettes triangulaires frontières de la triangulation de Delaunay en passant par les étapes suivantes :

2.4.2.1. Détermination des voisins des tétraèdres

Chaque tétraèdre possède un tétraèdre voisin au minimum et en possède quatre au maximum. De ce fait, la recherche de ses voisins est basée sur les indices attribués aux points de chaque tétraèdre. Pour atteindre cet objectif, les étapes données par la Figure 25 sont suivies :

- Pour chaque tétraèdre *A*, récupérer ses quatre facettes F_1, F_2, F_3 et F_4,
- Pour chaque facette F_i de *A*, récupérer ses points et comparer ses points avec les points de chaque facette des tétraèdres du modèle :
  - Dans le cas où la facette F_i de *A* a les mêmes points qu'une facette d'un tétraèdre *B*, ce dernier est le tétraèdre voisin de *A*,
  - Autrement, passer au tétraèdre suivant.

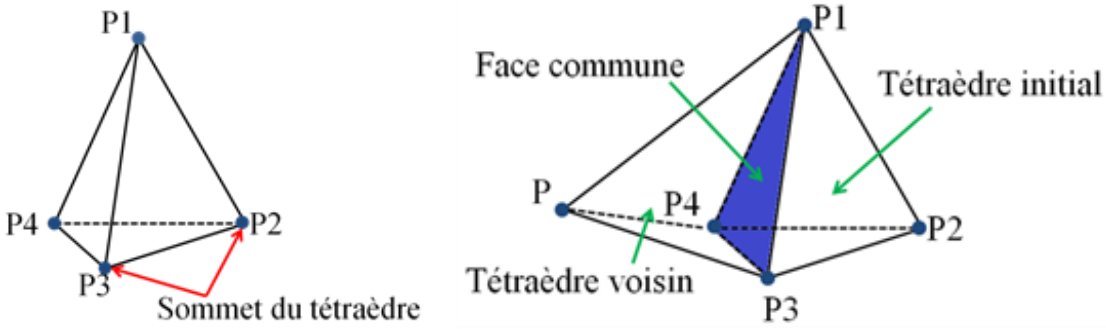


Figure 25 : Voisins d'un tétraèdre

2.4.2.2. Détermination des tétraèdres frontières

Un tétraèdre est dit frontière s'il possède au moins un voisin nul. De ce fait, après le calcul des voisins des tétraèdres, ceux ayant au moins un voisin nul sont stockés dans un tableau des tétraèdres frontières (Figure 26).

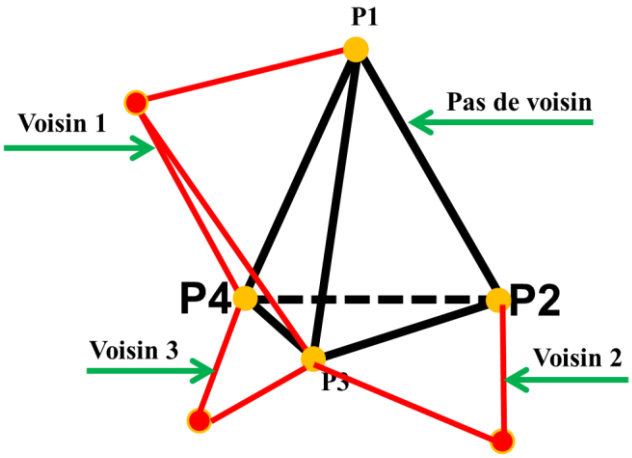


Figure 26 : Tétraèdre frontière

2.4.2.3. Détermination des facettes frontières

En se basant sur le tableau des tétraèdres frontières, les facettes frontières sont déterminées suivant les étapes données par la Figure 27 :

- Pour chaque tétraèdre frontière, tester ses quatre voisins V_1, V_2, V_3 et V_4 :
  - Si V_i est nul, cette facette est une facette frontière,
  - Dans le cas contraire, ce n'est pas une facette frontière.

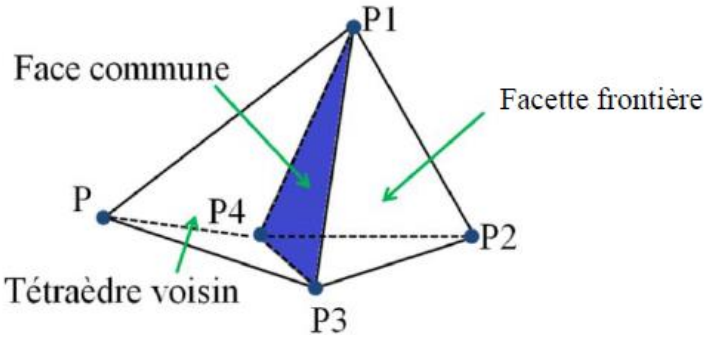


Figure 27 : Facette frontière

2.4.3. Application de l'algorithme « Alpha-Shape »

Pour se rapprocher de la forme réelle de l'objet digitalisé, la méthode « Alpha-Shape » est appliquée. Cette méthode suit les étapes suivantes :

2.4.3.1. Calcul du centre du cercle circonscrit de la facette frontière

Cette étape consiste à calculer pour chaque facette frontière le rayon et le centre du cercle circonscrit. Ces paramètres sont calculés comme suit :

- Calculer l'intersection des médiatrices des côtés du triangle de la facette ; soit C_0 le point de cette intersection.
- Soit $\vec{U}$ un vecteur de C_0 en un certain point sur le cercle et soit $\vec{V}$ un autre vecteur de C_0 à un autre point sur le cercle tel que $\vec{U}.\vec{V} = 0$.
- Les coordonnées du centre du cercle circonscrit sont données par l'équation (3) suivante :

$$\begin{cases} C_x = C_{x0} + \cos t \,.\, \vec{U}_x + \sin t \,.\, \vec{V}_x \\ C_y = C_{y0} + \cos t \,.\, \vec{U}_y + \sin t \,.\, \vec{V}_y \\ C_z = C_{z0} + \cos t \,.\, \vec{U}_z + \sin t \,.\, \vec{V}_z \end{cases} \qquad (3)$$

- Calculer le rayon du cercle circonscrit qui est la distance entre le centre du cercle circonscrit et un des sommets du triangle (Figure 28).

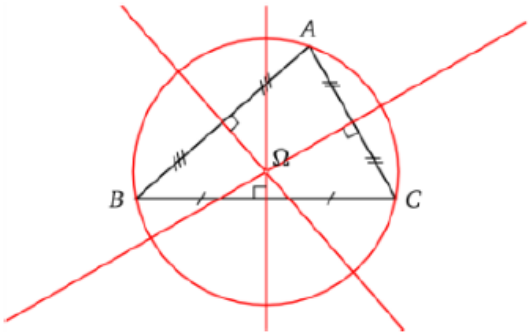


Figure 28 : Cercle circonscrit d'une facette

2.4.3.2. Détermination du centre de la sphère de la facette

Pour cette étape, le centre de la sphère de la facette frontière est calculé comme montré par la Figure 29 :

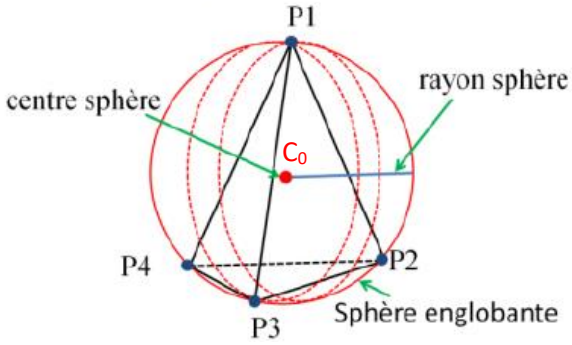


Figure 29 : Sphère circonscrite d'une facette

- Trouver la ligne perpendiculaire à la facette (parallèle à la normale de la facette) et passant par le point C_0,
- Sur cette ligne, trouver un point du côté de la normale et distant d'une valeur égale à « alpha » par rapport au point C_0. Les coordonnées de ce point sont données par l'équation (4) :

$$\begin{cases} M_x = C_0 + alpha.\vec{X} \\ M_y = C_0 + alpha.\vec{Y} \\ M_z = C_0 + alpha.\vec{Z} \end{cases} \quad (4)$$

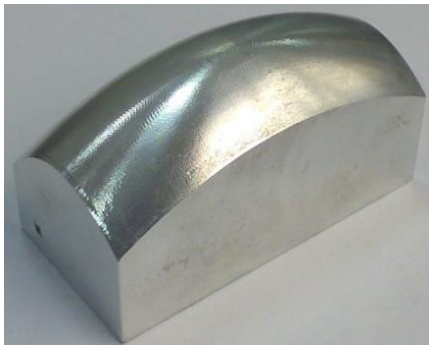
2.4.3.3. Mise à jour de la triangulation

Cette étape permet de mettre à jour la triangulation après la détermination de la sphère de la facette frontière. La procédure de modification de la triangulation de Delaunay est décrite ci-dessous :

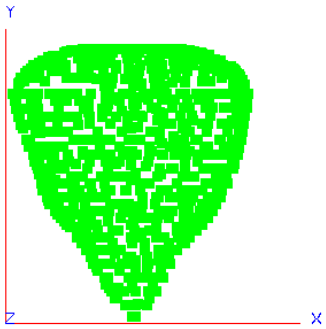
- Calculer la distance entre le centre de la sphère et les points du modèle,
  - Si le rayon de la sphère est inférieur à la distance calculée, supprimer la facette frontière et le tétraèdre qui la contient,
  - Vérifier si la sphère de la facette est vide :
    - Si oui, garder la facette frontière et le tétraèdre frontière,
    - Dans le cas contraire, supprimer le tétraèdre frontière et la facette frontière,
  - Si le rayon de la sphère est supérieur à la distance calculée, garder la facette et le tétraèdre frontière.

3. Validation et discussion des résultats obtenus

Les approches proposées sont implémentées en orienté objet sous Windows en utilisant le langage C++Builder et la bibliothèque graphique OpenGL. La validation de ces approches est réalisée sur plusieurs exemples de pièces. Ainsi, nous avons choisi trois modèles de pièces, à savoir, (*i*) *pièce convexe* (Figure 30a), (*ii*) *pièce statue* (Figure 30b) et (*iii*) *modèle de dent* (Figure 30c). Les tests de validation sont effectués sur un PC Intel Core i3 avec 6 Go de RAM et une résolution de 1366x768 pixels.


a. « *Pièce convexe* »


b. « *Pièce statue* »



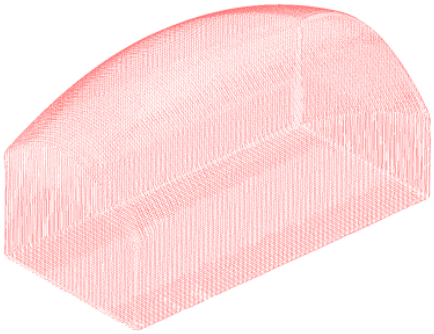
c. « *Modèle de dent* »

Figure 30 : Pièces à reconstruire leurs modèles

3.1.Approche de destruction et construction

L'approche est validée sur un nuage de points acquis à partir de la digitalisation de la « *pièce convexe* » (Figure 30a) sur une Machine à Mesurer Tridimensionnelles (MMT).

Le nuage de points initial est composé de 20788 points et les dimensions globales du brut de l'objet sont égales à 90.49×49.68×47.74mm (Figure 31a). Le filtrage de ce nuage est effectué en deux étapes : (*i*) élimination des points redondants et (*ii*) simplification. Pour cette dernière, la distance minimale entre deux points est considérée égale à 13mm. Cette opération réduit le nombre de points à 11114 (Figure 31b).


a. Avant filtrage

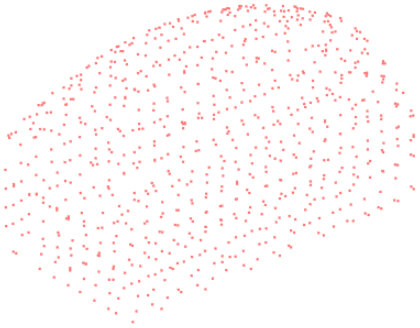
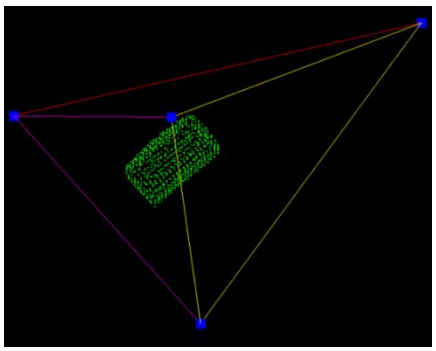

b. Après filtrage

Figure 31 : Filtrage du nuage de points de la « *pièce convexe* »

La création du super-tétraèdre s'appuie sur la génération de la super-sphère et des sommets fictifs. Par la suite, le centre et le rayon de la sphère circonscrite sont déterminés (Figure 32).


a. Super-tétraèdre

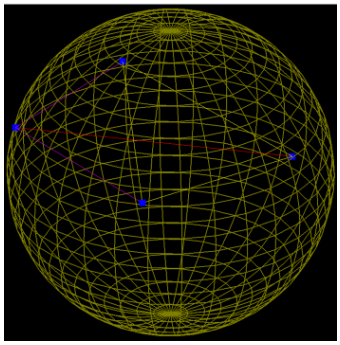
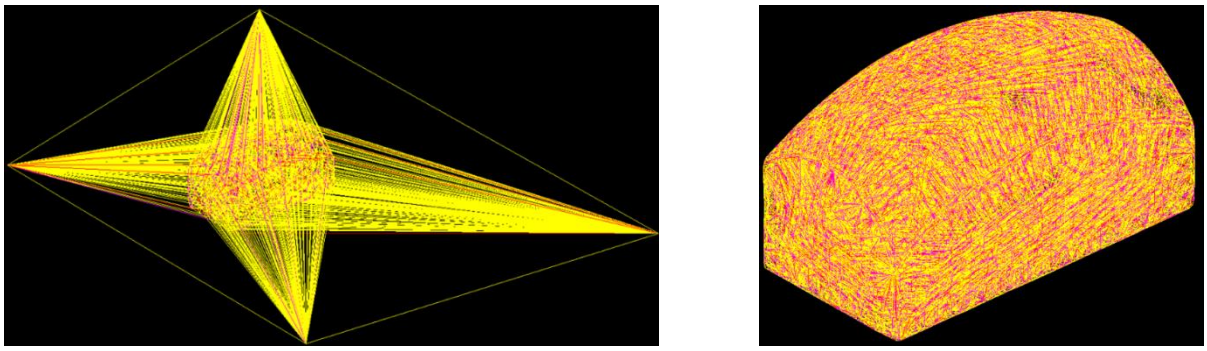

b. Sphère circonscrite

Figure 32 : Super-tétraèdre et sa sphère circonscrite de la « *pièce convexe* »

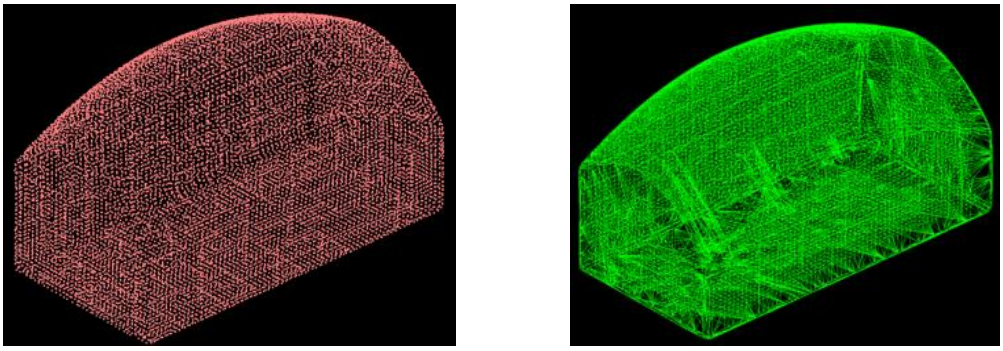
Pour la génération de la triangulation de Delaunay, les points sont insérés aléatoirement (Figure 33a). Une fois la triangulation est générée, les tétraèdres ayant des sommets fictifs sont supprimés (Figure 33b).



a. Tétraédrisation b. Tétraédrisation finale

Figure 33 : Génération de la triangulation de Delaunay 3D de la « *pièce convexe* »

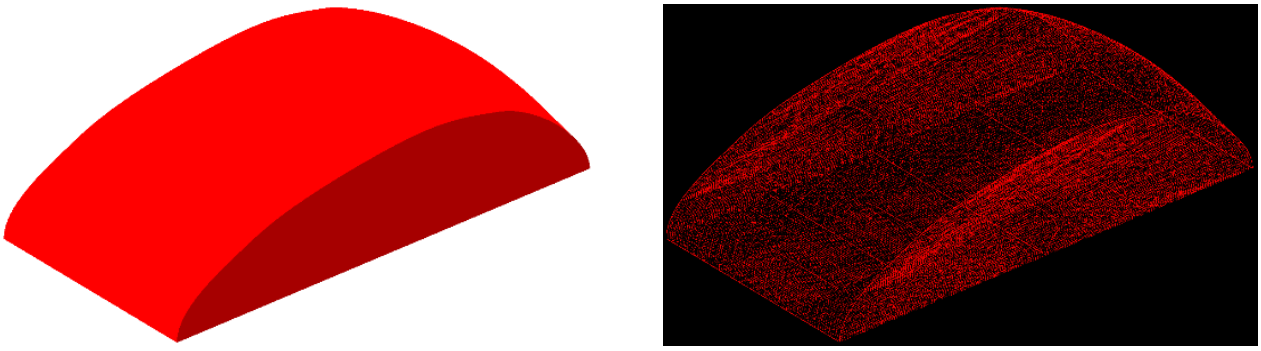
La distance minimale entre deux points est prise égale à 1mm, le processus utilisé génère un nuage de points plus dense. La Figure 34a montre le nuage de points et la triangulation associée dans la Figure 33b.



a. Nuage de points b. Triangulation de Delaunay

Figure 33 : Résultats de génération de la triangulation de Delaunay de la « *pièce convexe* »

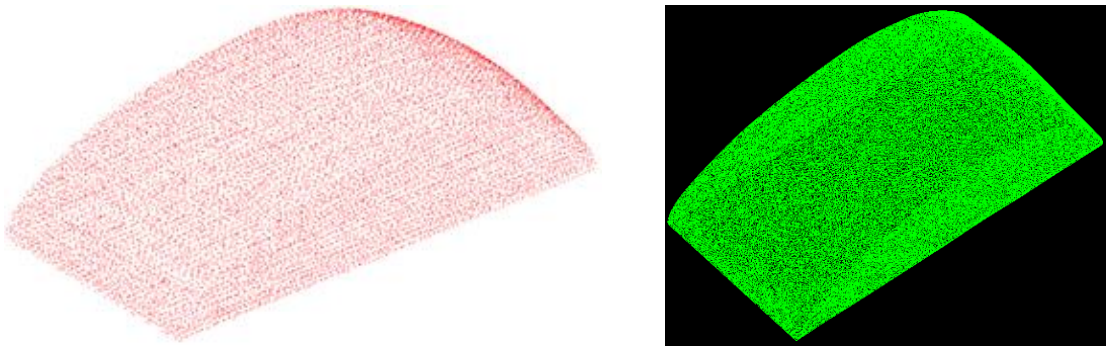
Les mêmes étapes sont appliquées sur le nouveau nuage de points composé de 357005 points de la « *pièce convexe* » (Figure 34).



a. Pièce de test b. Nuage de points correspondant

Figure 34 : Nuage de points de la « *pièce convexe* »

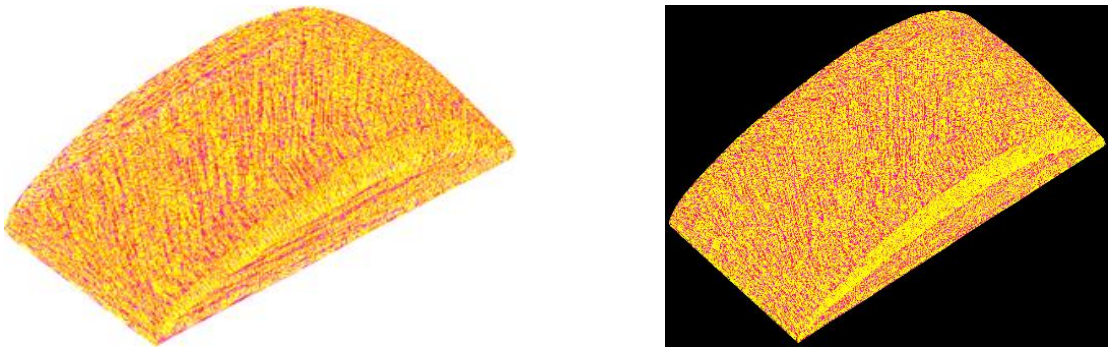
Le nuage de points est filtré en fixant la distance minimale entre deux points à 1mm puis à 0.5mm. Les nuages de points obtenus contiennent 11295 points (Figure 35a) et 34725 points (Figure 35b), respectivement.



a. Distance=1mm b. Distance=0.5mm

Figure 35 : Nuage de points après filtrage de la « *pièce convexe* »

Les triangulations 3D correspondantes à ces nuages de points sont représentées par la Figure 36a et Figure 36b, respectivement.



a. Distance=1mm b. Distance=0.5mm

Figure 36 : Triangulation de Delaunay des nuages de points de la « *pièce convexe* »

D'après les résultats obtenus, la forme de l'objet reconstruit est très proche de la forme réelle surtout en considérant un nuage de points plus dense. Par conséquent, il est très important de sélectionner judicieusement la distance minimale entre deux points puisque la précision de la forme de l'objet reconstruit dépend fortement de la densité du nuage de points.

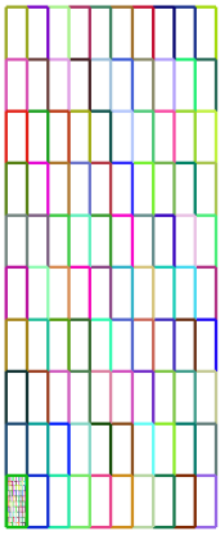
3.2.Triangulation par la méthode Flip

3.2.1. Validation sur la « *pièce Statue* »

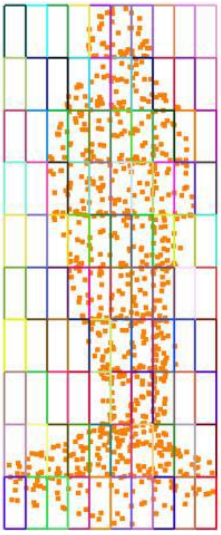
L'approche est validée sur un nuage de points acquis à partir de la digitalisation d'une pièce (Figure 30b). Le nuage de points est composé de 28692 points et les dimensions globales du brut de l'objet sont égales à 94.971x234.450x112.118mm (Figure 37).

Figure 37 : Nuage de points de la « *pièce Statue* »

Comme pour le cas précédent, l'opération de décimation du nuage de points est effectuée en deux étapes : (*i*) élimination des points redondants et (*ii*) simplification. Pour cette dernière, le nombre de cellules est pris égal à 10 suivant les trois directions (Figure 38a), le nombre de patchs est égal à 2, 1 et 1 suivant les directions X, Y et Z, respectivement, et la distance minimale entre deux points est prise égale à 10mm. Cette opération réduit le nombre de points à 864 (Figure 38b).



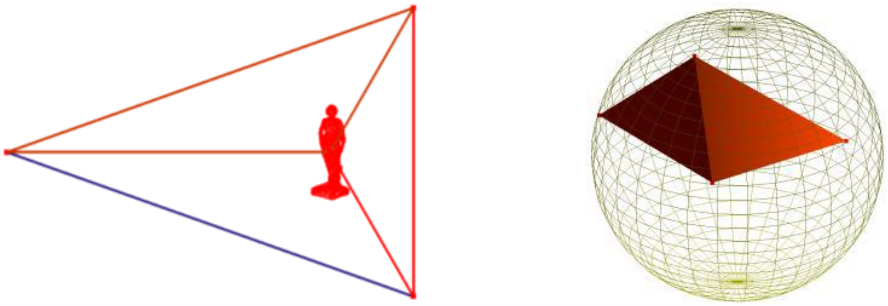
a. Cellules et patches



b. Nuage de points réduit

Figure 38 : Décimation du nuage de points de la « *pièce Statue* »

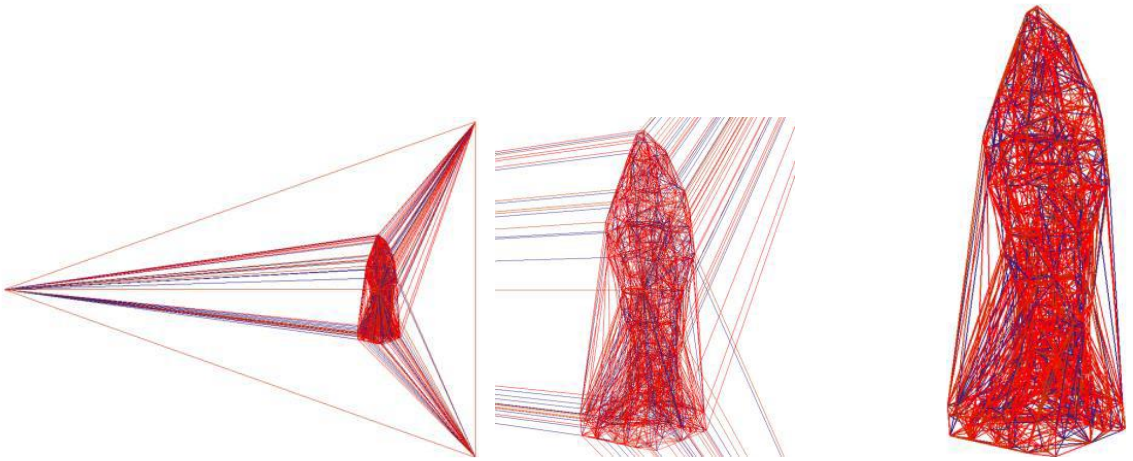
La création du tétraèdre englobant s'appuie sur la génération de la super-sphère et des sommets fictifs. Par la suite, le centre et le rayon de la sphère circonscrite sont déterminés tel que montré par Figure 39.



a. Tétraèdre englobant b. Sphère circonscrite

Figure 39 : Tétraèdre englobant et sphère circonscrite

Pour générer la triangulation de Delaunay, le mode d'insertion aléatoire est choisi. Une fois la triangulation est générée (Figures 40a), les tétraèdres ayant des sommets fictifs sont supprimés (Figure 40b).

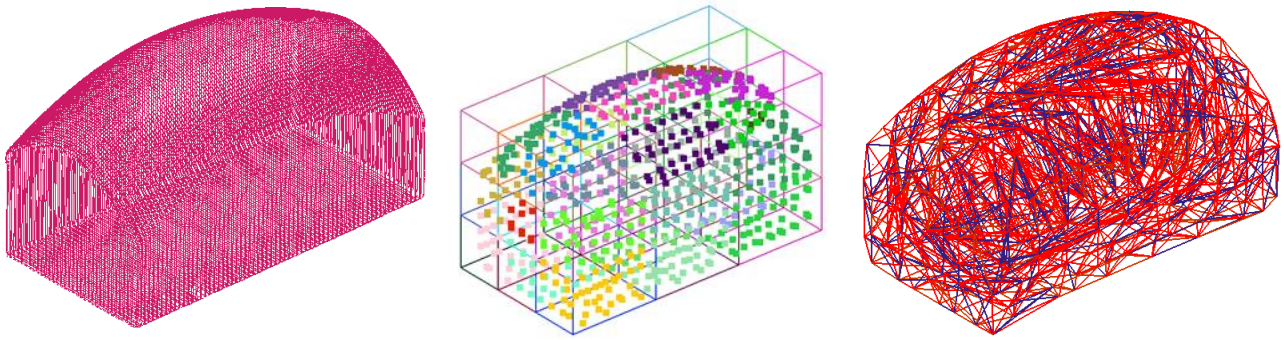


a. Tétraédrisation et zoom sur une partie de tétraédrisation b. Tétraédrisation finale

Figure 40 : Triangulation de Delaunay par FLIP pour la « *pièce Statue* »

3.2.2. Validation sur la « *pièce Convexe* »

Les mêmes étapes sont appliquées sur un nouveau nuage de points de la « *pièce convexe* ». Ce nuage est composé de 322787 points et les dimensions de la pièce brute sont 92.8x50.123x52.076mm (Figure 41a). Le nuage de points est filtré en fixant la distance minimale entre deux points à 18mm (Figure 41b) ; ainsi, le nuage de points obtenu contient 366 points. La triangulation 3D obtenue est présentée à la Figure 41c.



a. Nuage de points b. Décimation c. Tétraédrisation

Figure 41 : Triangulation de Delaunay par FLIP pour la « *pièce Convexe* »

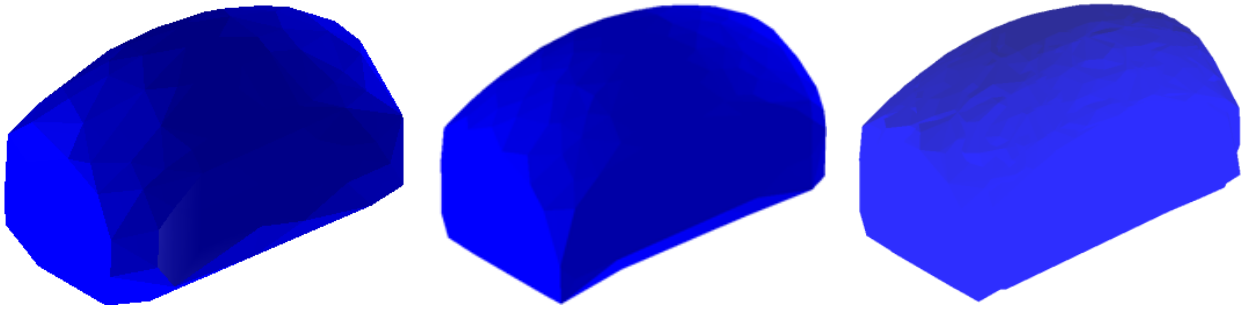
Une étude comparative entre trois résultats obtenus avec trois (03) distances de simplification appliquées sur un même nuage de points de 322787 points. Les résultats obtenus sont récapitulés dans le tableau suivant :

Distance de simplification	16 mm	18 mm	22 mm
Nombre de points	621 points	366 points	195 points
Temps d'exécution	1h et 3min	15 min et 6s	4 min

Tableau 1 : Temps de calcul de la triangulation avec trois distances de simplification pour la « *pièce Convexe* »

Le tableau 1 montre le temps de calcul de la triangulation de Delaunay pour trois distances de simplification différentes. Pour une distance égale à 22mm, le nombre de points est filtré à 195 points avec un temps de calcul de 4 min ; malheureusement, la forme finale obtenue de l'objet est loin de la forme réelle (Figure 42a). Avec une distance de 18mm, le nombre de points est plus important (366 points) et la forme est plus au moins proche de la forme réelle ; cependant, le temps de calcul est plus important (Figure 42b). La même chose est observée pour la distance de 16mm où le nombre de points et le temps de calcul sont plus élevés (Figure 42c).

Ces résultats montrent clairement que la forme finale de l'objet dépend fortement de la densité du nuage de points considérés. Aussi, comme attendu, le temps de calcul est plus important pour des distances de simplification plus petites.



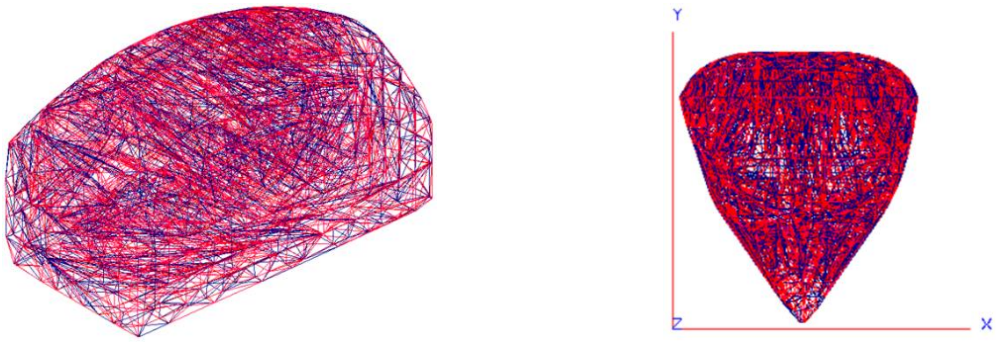
a. Distance=22mm b. Distance=18mm c. Distance=16mm

Figure 42 : Triangulation de Delaunay avec trois distances de simplification pour la « *pièce Convexe* »

3.3.Triangulation avec la méthode Flip amélioré

Les tests de validation ainsi que les performances de cette méthode sont démontrés sur plusieurs échantillons de deux modèles : « *Pièce convexe* » (Figure 30a) et « *Modèle de dent* » (Figure 30c).

La Figure 43 montre la triangulation du premier modèle « *Pièce convexe* » avec 900 points qui génèreront environ 10550 tétraèdres en 16s. Pour le second modèle, « *Modèle de dent* », le nombre de points et le nombre de tétraèdres générés sont d'environ 910 et 10112 en 14s, respectivement.

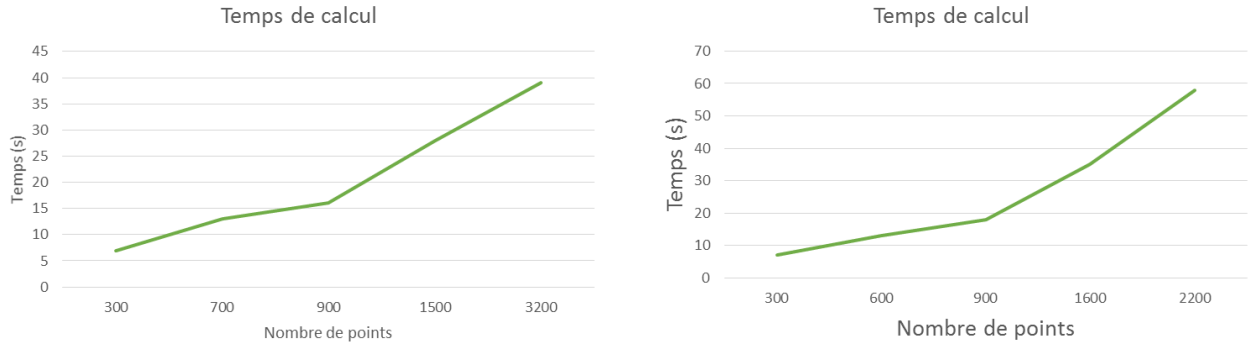


a. Triangulation de « *Pièce convexe* » b. Triangulation de « *Modèle de dent* »

Figure 43 : Triangulation de Delaunay de la « *Pièce convexe* » et du « *Modèle de dent* »

La méthodologie proposée a été évaluée en fonction de deux contraintes : (*i*) temps de calcul global de la triangulation de Delaunay et (*ii*) nombre de tétraèdres générés.

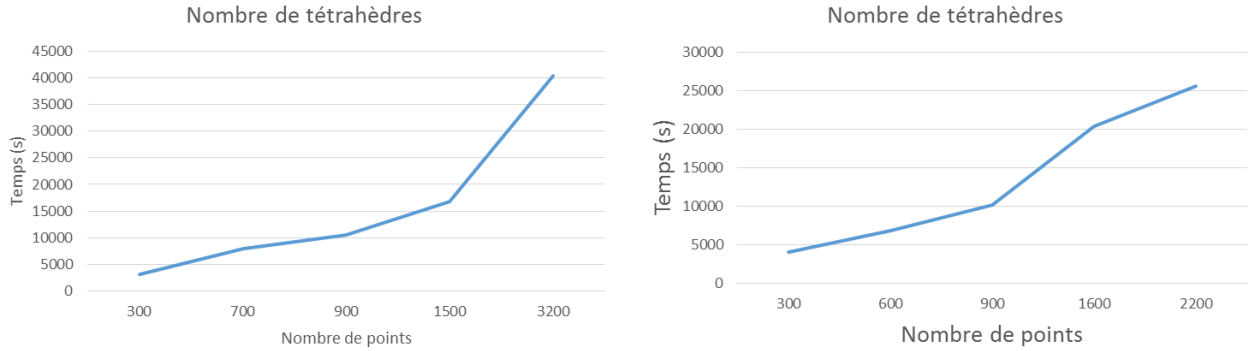
La Figure 44 montre l'évolution du temps de calcul de la triangulation de Delaunay en fonction du nombre de points pour les deux modèles.



a. Temps de calcul pour « *Pièce convexe* » b. Temps de calcul pour « *Modèle de dent* »

Figure 44 : Temps de calcul de la triangulation de Delaunay pour la « *Pièce convexe* » et du « *Modèle de dent* »

La Figure 45 illustre l'évolution du nombre de tétraèdres générés en fonction du nombre de points pour les deux modèles.



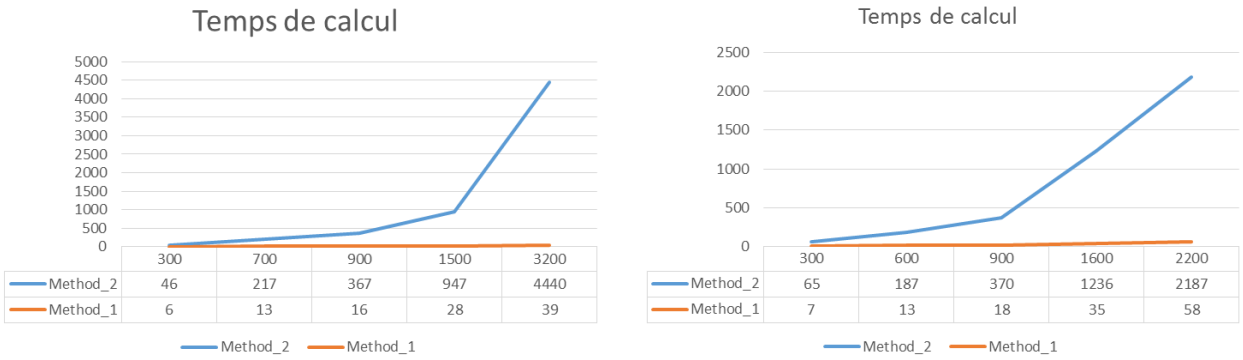
a. Nombre de tétraèdres pour « *Pièce convexe* » b. Nombre de tétraèdres pour « *Modèle de dent* »

Figure 45 : Nombre de tétraèdres générés pour les deux modèles « *Pièce convexe* » et « *Modèle de dent* »

3.4.Étude comparative

Pour évaluer les performances de l'approche « *Flip amélioré* », une étude comparative est réalisée entre cette méthode (méthode 1) et la méthode « *FLIP classique* » (méthode 2).

La Figure 46 illustre les temps de calcul pour générer une triangulation de Delaunay 3D pour ces deux algorithmes.



a. Temps de calcul pour « *Pièce convexe* » b. Temps de calcul pour « *Modèle de dent* »

Figure 46 : Temps de calcul de la triangulation de Delaunay pour « *FLIP amélioré* » et « *FLIP classique* » pour les deux pièces « *Pièce convexe* » et « *Modèle de dent* »

Les résultats révèlent que le temps de calcul est proportionnel au nombre de points pour la méthode de FLIP Amélioré. De plus, contrairement à la *FLIP classique*, ce temps reste faible et stable avec la méthode *FLIP amélioré* même si le nombre de points augmente.

Une autre étude comparative est réalisée concernant la localisation du point à insérer. Le Tableau 2 illustrent le nombre d'itérations exécutées pour déterminer le tétraèdre contenant un tel point pour les méthodes *FLIP amélioré* et *FLIP classique*, respectivement. Il montre clairement que le nombre d'itérations de la méthode *FLIP amélioré* est inférieur à celui de la méthode *FLIP classique*. Par exemple, pour insérer le 7ème point avec la première méthode (*FLIP amélioré*), quatre (04) itérations sont nécessaires pour déterminer le tétraèdre contenant ce point ; ceci donne le tétraèdre numéro 12 (parmi 120 tétraèdres), d'une part. D'autre part, la deuxième méthode (*FLIP classique*) effectue 22 itérations pour trouver le tétraèdre cible (indice 21) parmi les autres tétraèdres de la maille. Par conséquent, la méthode *FLIP amélioré* est meilleure puisque cette dernière effectue la recherche dans tous les tétraèdres de la maille (un par un). En effet, le nombre réduit d'itérations et la mise à jour des voisins dans une zone restreinte permettent de minimiser le temps requis pour le calcul de la triangulation de Delaunay malgré le fait que le nuage de points soit non-structuré et non-trié.

Méthode	**Index point**	1	2	3	4	5	6	7	8	9	10
FLIP amélioré	**Nombre d'itération**	2	3	4	0	5	2	4	0	5	75
	Index tétraèdre	2	3	15	1	13	38	34	2	56	66
FLIP classique	**Nombre d'itération**	3	3	6	10	13	1	22	24	16	41
	Index tétraèdre	2	2	5	9	12	0	21	23	15	40

Tableau 2 : Recherche du tétraèdre cible en utilisant la méthode « *FLIP amélioré* » et la méthode « *FLIP classique* »

3.5.**Reconstruction des frontières par la méthode « Alpha-Shape »**

L'approche est validée sur un exemple réel de la « *pièce statue* » acquise à partir d'une triangulation. Les résultats obtenus sont montrés par la Figure 47.

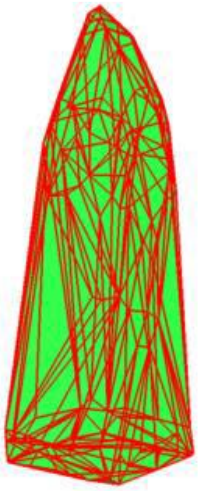
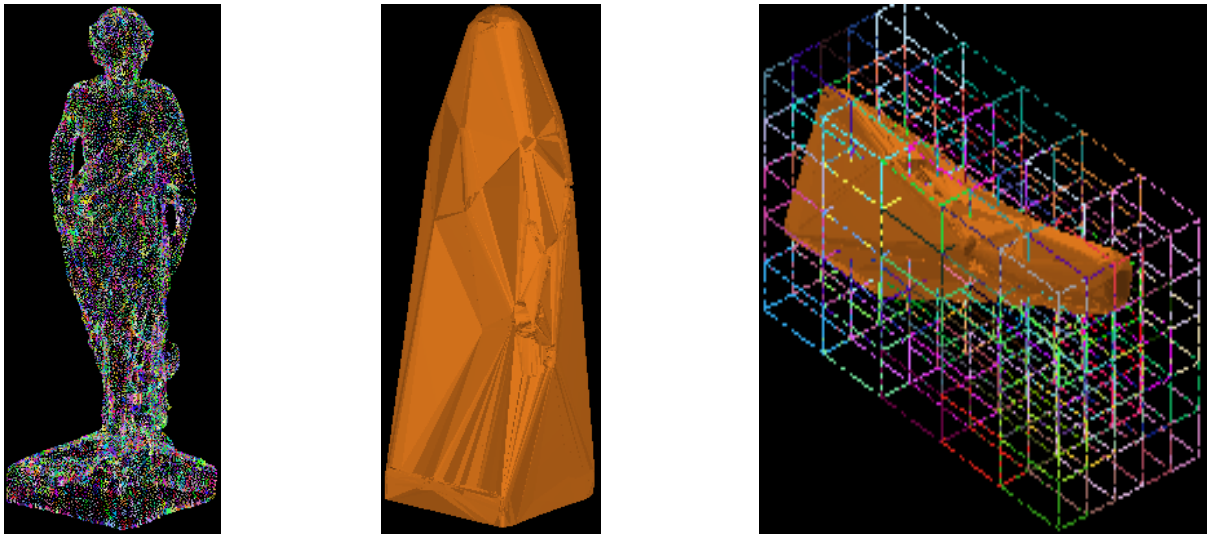


Figure 47 : Triangulation de Delaunay de la « *pièce statue* »

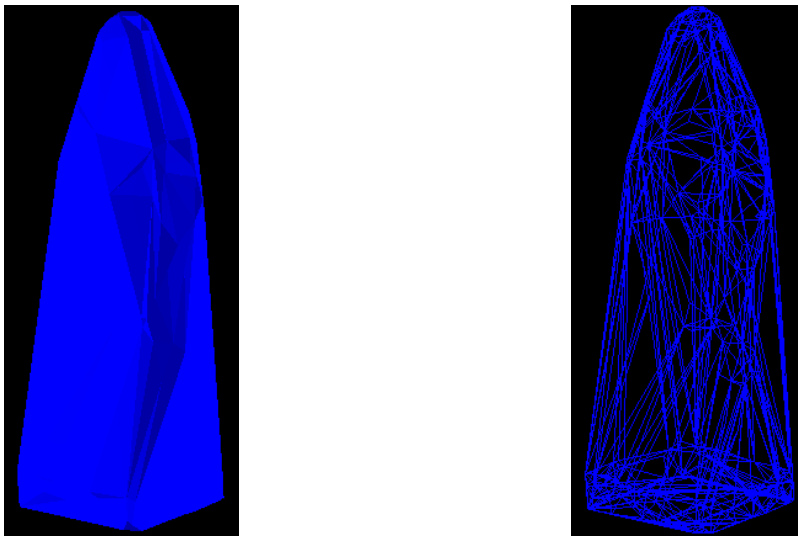
Pour la lecture de la triangulation de Delaunay, le nombre de cellules est égal à 5 suivant les trois axes X, Y et Z. La triangulation est composée de 193895 tétraèdres et de 28689 points. Les dimensions minimales du brut de cette pièce sont 92.971x232.150x10.118mm (Figure 46).



a. Nuage de points b. Tétraèdres du modèle c. Cellules de points

Figure 46 : Lecture de la triangulation de Delaunay pour la « *pièce statue* »

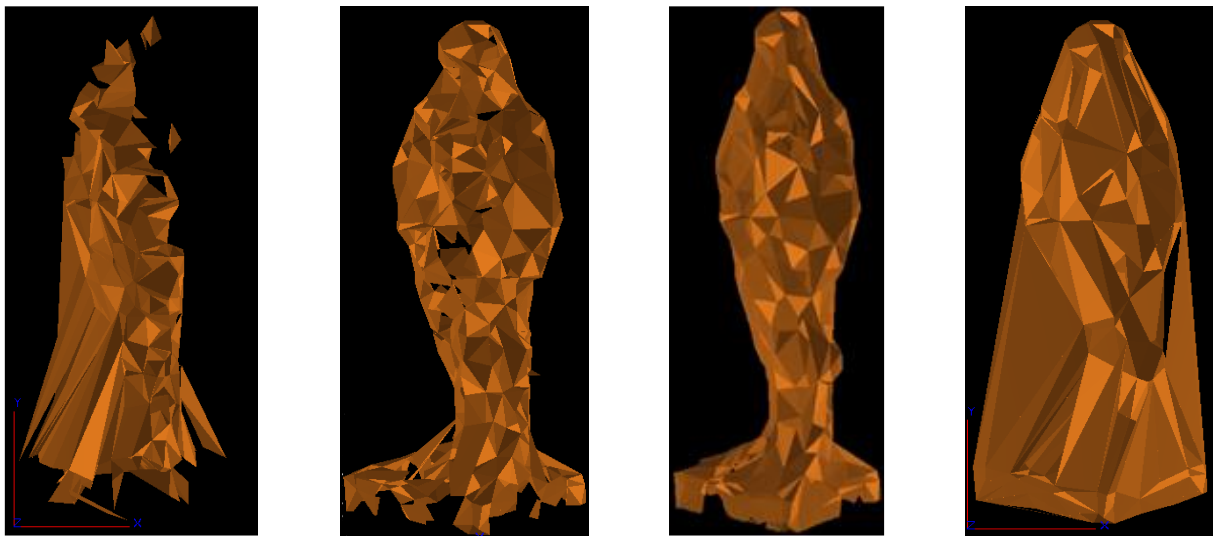
Par la suite, les tétraèdres frontières et les facettes frontières sont déterminés tels que montrés par la Figure 47.



a. Tétraèdres frontières b. Facettes frontières

Figure 47 : Tétraèdres et facettes frontières pour la « *pièce statue* »

Pour la validation de la méthode « Alpha-shape », plusieurs valeurs de « Alpha » ont été testées. Ainsi, les résultats obtenus sont montrés par la Figure 48. Globalement, la forme générale de l'objet est fidèlement reconstruite bien que le nuage de points considéré n'est pas dense. Pour cette pièce, la valeur de « Alpha » permettant de générer une frontière proche de la surface réelle est égale à 12.



a. Alpha=5 b. Alpha=9.5 c. Alpha=12 d. Alpha=20

Figure 48 : Modèle *statue* reconstruit avec différentes valeurs de « Alpha »

4. Conclusion

Ce chapitre a décrit et proposé plusieurs approches pour convertir le nuage de points issus d'une digitalisation d'un objet 3D en tétraèdres en utilisant la triangulation de Delaunay 3D et la méthode de « Alpha-shape ».

La première approche « Destruction et construction » consiste en la conversion d'un nuage de points non-structuré en un ensemble de tétraèdres en utilisant la triangulation de Delaunay 3D. La méthode développée est basée sur le principe de l'approche de Watson. Les différentes étapes de cette méthode sont données ainsi :

- Filtrage et simplification du nuage de points,
- Insertion des points,
- Génération de la triangulation.

L'approche a été validée sur une pièce réelle et les résultats obtenus sont encourageants du fait que le modèle obtenu est très proche du modèle réel en fonction de la densité du nuage de points.

La deuxième approche adoptée, *Approche FLIP*, est basée sur l'approche de bascule de diagonales. L'approche utilise comme données un nuage de points non-structuré pour fournir en sortie une tétraédrisation en passant par les mêmes étapes précédentes (filtrage et simplification du nuage de points, insertion des points, génération de la triangulation). L'approche a été validée sur un modèle de pièce et les résultats obtenus sont encourageants malgré le temps de réalisation très important.

La troisième approche, *FLIP amélioré*, est développée pour surmonter les inconvénients de l'approche FLIP. Son objectif principal est de proposer une approche rapide et efficace pour localiser le nouveau point à ajouter à la triangulation et mettre à jour les tétraèdres voisins. La méthodologie proposée a été testée sur différents échantillons et évaluée en fonction de deux critères : (*i*) le temps de calcul global de la triangulation de Delaunay et (*ii*) le nombre de tétraèdres générés. Son efficacité et ses performances pour localiser le nouveau point à insérer peuvent être observées grâce au nombre réduit d'itérations exécutées pour trouver le tétraèdre cible (contenant le point). De plus, les tétraèdres invalides (ne satisfaisant pas le critère de Delaunay) sont localisés en enregistrant le tétraèdre voisin pour mettre à jour la tétraédrisation. Le nombre d'itérations minimisé et la mise à jour des voisins dans une zone restreinte permettent d'améliorer le temps de triangulation, en particulier lors du traitement d'un nuage de points non-structuré.

La comparaison des résultats obtenus avec les deux approches précédentes a démontré la supériorité de la méthodologie proposée en termes des deux critères (temps de calcul global et nombre de tétraèdres générés).

La dernière approche consiste à déterminer les frontières d'un objet en utilisant la méthode « Alpha Shape ». L'approche utilise comme données une triangulation de Delaunay pour fournir en sortie un modèle proche de la forme réelle de l'objet en suivant les étapes suivantes :

- Vérification et lecture de la triangulation de Delaunay,
- Détermination des tétraèdres et des facettes frontières,
- Application de la méthode « Alpha Shape ».

L'approche a été validée sur un modèle réel ; aussi, les résultats obtenus sont encourageants du fait que le modèle obtenu est très proche du modèle réel en fonction de la densité du nuage de points.

Partie 2
Triangulation de Delaunay parallèle en utilisant la méthode « *Diviser pour régner* »

Introduction

Après une étude des principaux avantages et inconvénients des principaux algorithmes de reconstruction, la méthode de triangulation de Delaunay par l'approche « *Diviser pour régner* » ou « *Divide and Conquer* » est la plus utilisé et la plus efficace.

En général, l'algorithme « *Diviser pour régner* » est conçu pour le parallélisme et nécessite des ordinateurs à processeurs multi-cœurs. Il est réalisé en plusieurs étapes :

- La première divise récursivement le nuage de points en sous-nuages de points,
- Lors de la deuxième étape, chaque nuage de points est attribué à un processeur,
- Indépendamment, ces sous nuages de points sont triangulées simultanément,
- Enfin, les sous-triangulations sont fusionnées afin d'obtenir la triangulation globale du nuage de points.

Le premier chapitre de cette partie étudie le processus de partitionnement du nuage de points en plusieurs sous-nuages avec les différentes approches existantes. Par la suite, le chapitre propose une nouvelle approche de partitionnement spatiale. À la fin, l'approche développée est testée et validée sur plusieurs modèles d'objets.

Le deuxième chapitre est consacré à la fusion de toutes les sous-triangulations. Ce chapitre commence par étudier les différentes approches utilisées pour la fusion des sous-triangulations. Ensuite, il décrit une nouvelle approche de fusion. Enfin, le chapitre teste et valide l'approche proposée sur plusieurs modèles d'objets.

Chapitre 4
Partitionnement du nuage de points en sous-nuages

1. Introduction

Avec l'apparition des machines multi-cœurs qui fournissent des environnements de programmation parallèles, le processus de génération de la triangulation de Delaunay a été considérablement accéléré.

Les méthodes parallèles de triangulation de Delaunay permettent d'atteindre des vitesses de reconstruction très rapides. L'algorithme « *Diviser pour régner* » est parmi les algorithmes parallèles [83]. Cet algorithme est basé sur trois étapes : (*i*) le partitionnement du nuage de points en sous-nuages, (*ii*) la génération d'une sous-triangulation pour chaque processeur, et (*iii*) la fusion de toutes les sous-triangulations pour obtenir la triangulation finale. Ce chapitre s'intéresse au partitionnement du nuage de points en plusieurs sous-nuages.

2. État de l'art et problématique

Plusieurs méthodes de partitionnement ont été proposées dans la littérature. Dans [84-85] , les auteurs ont montré que le partitionnement des points peut être réalisé par la structure Octree. Selon [84], le partitionnement en octree devient très simple en utilisant un prédicat prédéfini $\kappa(P_i)$. L'octree est caractérisée par sa structuration hiérarchique et son accès rapide aux points. Par conséquent, la construction de l'arbre prend nettement moins de temps CPU selon [85].

Le partitionnement par les arêtes de Delaunay est une autre méthode utilisée par Lee et Park [86]. Elle permet d'éliminer l'étape de fusion nécessaire à l'intégration des sous-triangulations. Le principe utilisé consiste à diviser les points donnés en sous-régions selon des chemins constitués d'arêtes de Delaunay. Deux méthodes de partitionnement utilisant les chemins sont considérées. L'une divise une région en deux sous-régions selon un chemin vertical ; puis, chaque sous-région se divise selon un chemin horizontal. Cette méthode est répétée de manière récursive jusqu'à ce que le nombre de sous-régions soit égal à celui des processeurs. L'autre méthode divise une région en m sous-régions selon uniquement des chemins verticaux (ou horizontaux) où m représente le nombre de processeurs.

Récemment, Wu et al. [87] ont proposé une nouvelle méthode qui divise le nuage de points sur les cœurs du processeur en utilisant kd-tree pour améliorer l'équilibre de la charge sur les différents processeurs. Ils utilisent une stratégie d'ordonnancement dynamique basée sur un arbre kd pour supprimer les contraintes des implémentations parallèles traditionnelles de l'algorithme « *Diviser pour régner* ».

Chen et ses collègues dans [88] ont proposé de partitionner le nuage de points le long d'une seule dimension dans le but d'obtenir un nombre approximativement égal de points. Dans [88], une méthode de partitionnement parallèle de l'algorithme de recherche de la médiane de Hoare[89] est conçue pour partitionner l'ensemble de points. Dans [90-91], Lo a partitionné le nuage de points en zones. Le principe utilisé consiste d'abord à diviser les points en cellules ; chacune d'entre elles se voyant attribuer un nombre à peu près égal de points. Ensuite, les cellules peuvent être regroupées en zones. Pour renforcer cette dernière approche, l'auteur en [92] a spécifié le nombre de points pour chaque zone.

Plus tard, Liu et ses collègues dans [93] ont proposé un schéma d'insertion multi-grilles. Il consiste à appliquer récursivement l'insertion de grille régulière à une cellule de points, l'objectif visé est d'avoir un nombre de points presque égal dans toutes les cellules.

L'étude des travaux susmentionnés révèle que la méthode « *Diviser pour régner* » utilise des techniques de partitionnement par cellule uniforme. En outre, la meilleure approche de partitionnement d'un ensemble de points en sous-ensembles de taille appropriée n'a pas encore été déterminée. Par conséquent, l'efficacité du calcul parallèle n'est pas pleinement exploitée. Les principaux défis de ces techniques de partitionnement sont (*i*) le type de nuage de points (nuage de points non-structuré), (*ii*) l'espace mémoire requis, et (*iii*) le nombre déséquilibré de points dans les partitions trouvées. Par conséquent, les processeurs ne travaillent pas de manière égale et le calcul parallèle n'est pas pleinement exploité.

Pour faire face à ce problème, une contribution a été ajoutée [94]. Premièrement, nous proposons une méthodologie pour partitionner l'espace partiel représenté par un nuage de points en régions via le regroupement des points en sous-ensembles. Deuxièmement, nous introduisons la méthode *K-means basique* (*BK*) pour la première fois dans ce domaine de recherche. Troisièmement, ce travail est enrichi par l'utilisation d'une variante de la méthode BK, *K-means global rapide* (*FGK*), pour obtenir une solution globalement optimale. Cette solution est utilisée pour trouver un meilleur partitionnement qui améliore les performances de regroupement des points. Quatrièmement, une étude comparative de trois méthodes mises en œuvre sur un nuage de points non-structuré a été établie pour démontrer leurs performances et déterminer l'approche la plus appropriée pour créer des partitions (clusters) : (*i*) cellules, (*ii*) octree, (*iii*) BK et sa variante FGK. Enfin, nous sommes arrivés à la conclusion que FGK a obtenu de meilleures performances en termes de (*i*) homogénéité des partitions, (*ii*) charge de travail équilibrée sur les différents processeurs, et par conséquent sur (*iii*) les temps de calcul par rapport aux autres méthodes cellules, octree et BK.

3. Méthode K-means

Le partitionnement des données est une tâche importante en analyse de données ; en effet, elle permet de diviser un ensemble de données en plusieurs sous-ensembles appelés groupes ou clusters. Ces groupes sont caractérisés par une forte similarité à l'intérieur et une forte dissimilarité entre les membres de différents groupes [95]. L'usage de cette technique vise à identifier un résumé de la structure interne de ces données, sans aucune connaissance a priori sur les caractéristiques des données. Cela touche plusieurs domaines dont la reconnaissance des formes, l'imagerie, la bio-informatique et l'indexation des bases d'images. Dans ce cadre, plusieurs méthodes ont été développées, la plus populaire est celle des k-moyennes (K-means) ; elle doit sa popularité à sa simplicité et sa capacité à traiter de larges ensembles de données [96].

3.1 Principe de la méthode K-means

Le partitionnement en utilisant l'algorithme K-means de base (BK) est un des plus simples algorithmes de classification automatique des données [97]. Cet algorithme, donné par la Figure 1, a été proposé par MacQueen [97]. Étant donnés des points et un entier k, le problème est de diviser les points en k groupes, souvent appelés clusters, de façon à minimiser une certaine fonction. La distance d'un point à la moyenne des points de son cluster est considérée ; la fonction à minimiser est la somme des carrés de ces distances. Chaque point est affecté au centre le plus proche. Après l'affectation de tous les points, la moyenne de chaque groupe est calculée ; elle constitue les nouveaux représentants des groupes, lorsqu'aucun point ne change de groupe alors l'algorithme est arrêté.

Cet algorithme a été appliqué à de nombreux domaines tels que le traitement d'images [98-99], la génération de modèles TD [100], le partitionnement de pièces complexes par regroupement d'objets géométriques élémentaires dans le processus de CAO/FAO [101], etc. Les principaux avantages de l'algorithme BK sont (*i*) sa grande simplicité, (*ii*) sa tendance à réduire l'erreur quadratique, et (*iii*) son applicabilité aux larges ensembles de données.

Algorithme 1 : K-means
entrée
Ensemble de points : x ;
Nombre de clusters demandés : k ;
sortie
Partition de k clusters (C_1, C_2, C_3..,.C_k) ;
début
Initialiser aléatoirement les centres C_k ;
répéter
Affectation :
Générer une nouvelle partition en affectant chaque objet au groupe dont le centre est le plus proche : $x_i \in C_k \; if \; \forall j \; |x_i - \mu_k| = min|x_i - \mu_k|$, μ_k représente le centre de la classe k ;
Représentation :
Calculer les centres associés à la nouvelle partition $\mu_k = \frac{1}{N} \sum_{x_i \in C_k} x_i$;
jusqu'à la convergence de l'algorithme vers une partition stable ;
fin

Figure 1 : Algorithme K-means

3.1.1 Avantages et inconvénients

Parmi les avantages de cet algorithme, nous citons :

- Sa grande simplicité,
- Tend à réduire l'erreur quadratique,
- Applicable à larges ensembles de données.

Cette méthode présente plusieurs inconvénients, tels que :

- Le nombre de classes doit être fixé au départ,
- Ne détecte pas les données bruitées,
- Le résultat dépend du tirage initial des centres de classes,
- Les clusters sont construits par rapport à des objets inexistants (les milieux),
- N'est pas applicable en présence d'attributs qui ne sont pas du type intervalle.

3.1.2 Problème d'initialisation des centres

La principale limite de cette méthode est la dépendance des résultats des valeurs de départ (centres initiaux). À chaque initialisation correspond une solution différente (optimum local) qui peut, dans certain cas, être très loin de la solution optimale (optimum global). Une solution naïve à ce problème consiste à lancer l'algorithme k-means plusieurs fois avec différentes initialisations et retenir le meilleur regroupement trouvé. L'usage de cette solution reste limité du fait de son coût et de son incapacité à trouver une meilleure partition en une seule exécution. Pour résoudre ce problème, trois d'algorithmes dérivés de la méthode k-means ont été proposées : (*i*) K-means global (GK) [R], (*ii*) K-means global rapide (FGK) [R] et (*iii*) K-means incrémental [R].

3.2 Principe de la méthode k-means global

C'est une solution au problème d'initialisation de l'algorithme k-means de base ; elle est fondée sur les données et vise à atteindre une solution globalement optimale. La méthode k-means global consiste à effectuer un regroupement incrémental et à ajouter dynamiquement un nouveau centre suivi par l'application du k-means jusqu'à convergence. Cet algorithme est détaillé par la Figure 2.

Algorithme 2 : K-means global
entrée
Ensemble de points : P ;
Nombre de clusters demandés : k ;
sortie
Partition de k clusters (C_1, C_2, C_3, ..., C_k) ;
début
C_1=Centre de gravité de l'ensemble des données ;
répéter
Initialiser les (i-1) centres par le résultat de l'étape précédente ;
pour chaque donnée x de $P\{$ //Trouver le i^{ème} centre
Considérer x comme le i^{ème} centre ;
Affecter les données au centre le plus proche ;
Calculer l'erreur quadratique pour $C_i = x$;
fin pour
Conserver le centre $C_i = x$ qui minimise l'erreur quadratique ;
Appliquer K-means jusqu'à convergence ;
jusqu'à l'obtention d'une partition en k groupes ;
fin

Figure 2 : Algorithme K-means global

3.3 Principe de la variante k-means global rapide

La méthode K-means global nécessite un temps de calcul lourd à cause de la stratégie de choix du nouveau centre. Pour cela, l'algorithme k-means global rapide a été proposé avec une nouvelle stratégie permettant d'accélérer la méthode K-means global. Cette stratégie garde la même philosophie que sa précédente. En effet, toutes les données peuvent être candidates pour devenir un centre. Cependant, l'algorithme évite d'affecter les données aux centres les plus proches (centres déjà existant en plus du centre candidat) et de calculer l'erreur quadratique. L'algorithme de la Figure 3 donne les détails de cette méthode.

Algorithme 3 : K-means global rapide
entrée
Ensemble de points : P ;
Nombre de clusters demandés : k ;
sortie
Partition de k clusters (C_1, C_2, C_3, ..., C_k) ;
début
C_1=centre de gravité de l'ensemble P ;
répéter
Initialiser les (i-1) centres par le résultat de l'étape précédente ;
pour chaque donnée x de P // Trouver le i^{ème} centre :
Considérer x comme le i^{ème} centre ;
Calculer d_{k-1}^{i} ;
Calculer b_n pour $C_i = x$ en utilisant l'équation (2) ;
fin pour
conserver le centre $C_i = x$ qui minimise l'erreur quadratique ;
jusqu'à obtenir une partition en k groupes ;
Appliquer le K-means jusqu'à convergence ;
fin

Figure 3 : Algorithme K-means rapide global.

3.4 Principe de la méthode k-means incrémental

La méthode k-means incrémental (ou approche incrémental de classification) est similaire à celle de la méthode k-means global ; les différences entre elles résident dans les points suivants :

- Le nombre de points initiaux est égal à deux au lieu d'un seul dans la méthode k-means global ; ces deux points sont les plus loin parmi l'ensemble des éléments,
- La recherche du nouveau centre se limite à la recherche de l'élément le pire classé (celui qui possède la plus grande distance de son centre le plus proche) au lieu de tester toutes les données.

Algorithme 4 : K-means incrémental
entrée
Ensemble de points : P notés par x;
Nombre de clusters demandés : k ;
sortie
Partition de k clusters (C_1, C_2, C_3, ..., C_k) ;
début
$C_1=x_1$;
$C_2=x_2$;
$d(x_1, x_2) = \max_{i,j\in[1.N]}(d(x_1, x_2))$;
répéter
Initialiser les (i-1) centres par le résultat de l'étape précédente ;
Trouver le i[ème] centre C_i : $C_i = X : X = \max_{i\in[1,N]} d^i_{k-1}$;
Appliquer K-means jusqu'à convergence ;
jusqu'à obtenir une partition en k groupes ;
fin

Figure 4 : Algorithme K-means incrémental

4. Solution proposée

La solution proposée vise à partitionner le nuage de points non-structuré en plusieurs partitions. Puis, les partitions résultantes seront distribuées sur les cœurs du processeur pour améliorer l'équilibre de la charge de travail. Cela peut être accompli selon trois algorithmes : cellule, octree et BK avec sa variante FGK. La Figure 5 montre la structure générale de la solution proposée.

La première étape concerne l'acquisition de l'objet scanné avec un fichier de données « .txt », qui contient les coordonnées des points (x, y, z) de manière aléatoire. Après cela, ces coordonnées sont sauvegardées dans une structure de données établie pour une manipulation plus facile dans l'étape de partitionnement. Cette étape divise le nuage de points en plusieurs sous-nuages en utilisant trois méthodes de partition : Octree, K-means et Cellules. Enfin, les sous-nuages résultants sont répartis sur les cœurs du processeur. Enfin, le nombre de points pour chaque partition, le nombre de tétraèdres et le temps de calcul sont estimés.

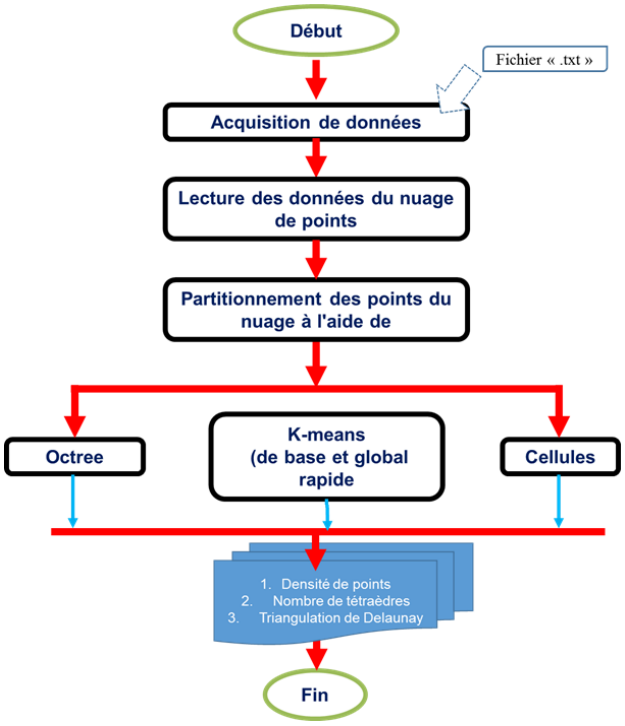
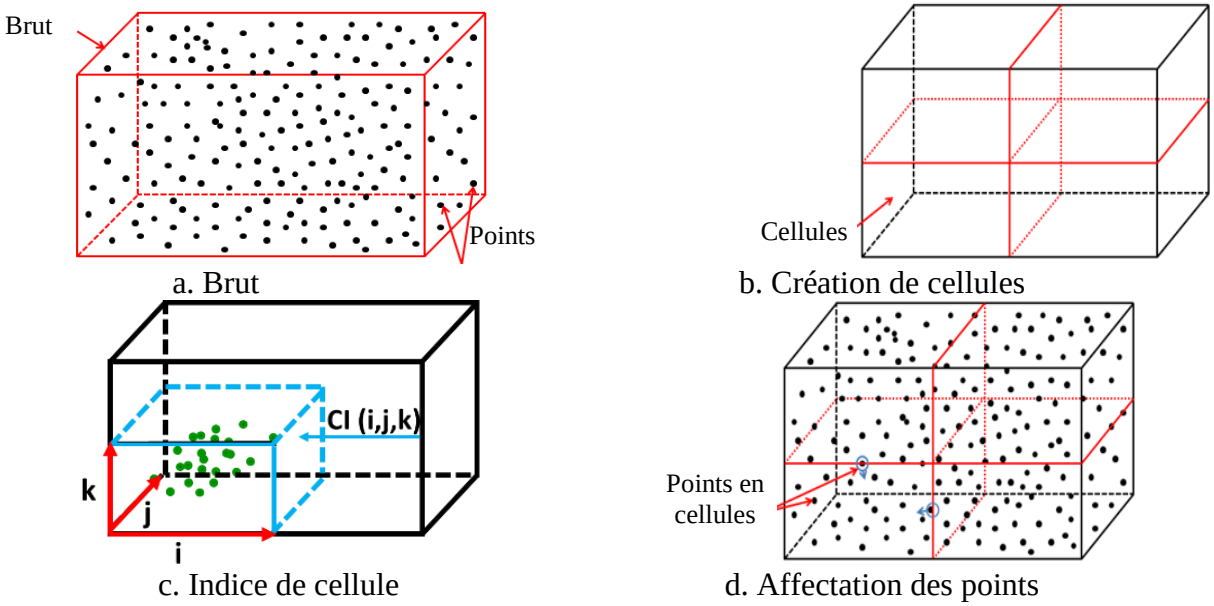


Figure 5 : Diagramme de la solution proposée

4.1 Partitionnement en cellules

Le nuage de points est subdivisé en plusieurs cellules de même taille selon les trois directions X, Y et Z. D'abord, la partie brute qui englobe l'ensemble des points est définie (Figure 6a). Ensuite, les cellules de même dimension sont déterminées (Figure 6b). Après, l'indice de cellule (*CI*) de chaque point est calculé (Figure 6c). Enfin, les points sont affectés à la cellule appropriée (Figure 6d).



a. Brut b. Création de cellules

c. Indice de cellule d. Affectation des points

Figure 6 : Partitionnement en cellules

L'algorithme 5, donné par la Figure 7, décrit le processus de partitionnement. Il commence par lire le nuage de points ; puis, le stocker dans un tableau de points. Chaque point est défini par des coordonnées (x, y, z). Ensuite, les dimensions du brut sont calculées. Une fois le nombre de cellules déterminé, les dimensions des cellules sont calculées. Après, les cellules sont obtenues en subdivisant le brut en plusieurs cellules de même taille. Enfin, les points sont affectés à leurs cellules appropriées

en utilisant leur *CI* ; cela permet d'affecter directement chaque point à la cellule appropriée sans avoir à tester ses coordonnées.

Algorithme 5 : Partitionnement des cellules
Entrée
Ensemble de points : P ;
Nombre de cellules : n_x, n_y, n_z ;
Sortie
Cellules ($cell_1$, $cell_2$, $cell_3$..,.$cell_k$) ;
Début
Calculer la limite de l'ensemble des points : X_{min}, X_{max}, Y_{min}, Y_{max}, Z_{min}, Z_{max} ;
Calculer les dimensions de la pièce brute : longueur, largeur et hauteur ;
Calculer la dimension d'une cellule : Pas_x, Pas_y et Pas_z ;
Créer des tableaux 3D : cellules le long des trois axes X, Y et Z ;
Pour chaque point
Calculer l'indice $i = \frac{x - X_{min}}{Pas_x}$;
Calculer l'indice $j = \frac{y - Y_{min}}{Pas_y}$;
Calculer l'indice $k = \frac{z - Z_{min}}{Pas_z}$;
Affecter le point $P(x, y, z)$ à la cellule $CI = (i, j, k)$;
Fin pour
Fin

Figure 7 : Algorithme de partitionnement en cellules

4.2 Partitionnement en Octree

L'octree est une structure de données de type arbre dans laquelle chaque nœud peut avoir jusqu'à huit enfants. Les octrees sont le plus souvent utilisés pour partitionner un espace 3D en le subdivisant récursivement en huit nœuds. L'octree est obtenu en trois étapes :

- **Étape 1** : Identifier les paramètres du brut englobant,
- **Étape 2** : Créer les boites (box),
- **Étape 3** : Créer l'octree.

4.2.1 Étape 1 : Identifier les paramètres de la partie brute englobante

Pour faciliter la manipulation des points lors de la création de l'octree, les dimensions *longueur*, *largeur* et *hauteur* de la partie brute sont calculées (Figure 8).

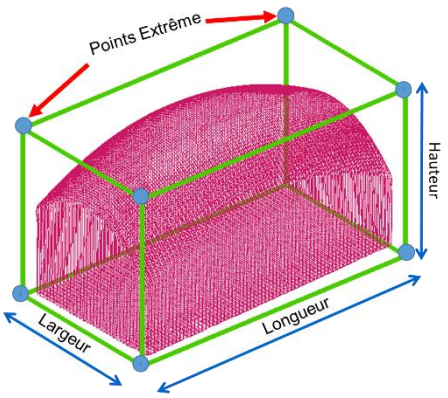
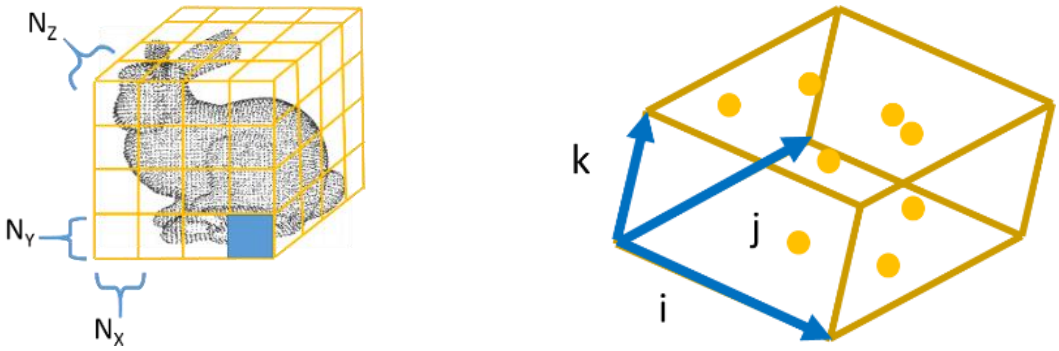


Figure 8 : Limite du nuage de points

4.2.2 Étape 2 : Créer les boites (boxes)

Dans cette étape, la pièce brute est subdivisée en petites cellules de même taille appelées boxes. Une fois la dimension de la partie brute calculée et le nombre de boxes introduit, les boxes sont créées (Figure 9a). Ensuite, chaque point est affecté au box en utilisant son indice (*BI*) défini par trois indices i, j, et k (Figure 9b).



a. Création de boxes b. Indice de box

Figure 9 : Création de boxes

L'algorithme 6 (Figure 10) résume le processus suivi pour créer les boxes. La première étape consiste à introduire les coordonnées des points et le nombre de boxes n_x, n_y et n_z. Ensuite, un tableau 3D de boxes est créé le long des trois directions X, Y et Z sur la base de deux calculs : (*i*) le nombre de boxes Nbr_x, Nbr_y et Nbr_z et (*ii*) les indices i, j et k. Enfin, chaque point est affecté au box approprié en fonction de son *BI*.

Algorithme 6 : Création de boxes
Entrée
Ensemble de points : P ;
Nombre de boxes : n_x, n_y, n_z ;
Sortie
Boxes (box_1, box_2, box_3...box_K) ;
Début
Calculer la limite de l'ensemble des points : X_{min}, X_{max}, Y_{min}, Y_{max}, Z_{min}, Z_{max} ;
Calculer les dimensions de la pièce brute : *longueur*, *largeur* et *hauteur* ;
Calculer le nombre de boxes : $Nbr\_x = 2^{nx}$, $Nbr\_y = 2^{ny}$ et $Nbr\_z = 2^{nz}$;
Calculer les dimensions du box : $pas\_X = \frac{Xmax-Xmin}{Nbr\_x}$, $pas\_Y = \frac{Ymax-Ymin}{Nbr\_y}$ et $pas\_Z = \frac{Zmax-Zmin}{Nbr\_z}$
Créer les tableaux 3D de boxes le long des trois axes : X, Y et Z ;
Pour chaque point
Calculer l'index $i = \frac{x-Xmin}{Pas\_X}$;
Calculer l'index $j = \frac{y-Ymin}{Pas\_Y}$;
Calculer l'index $k = \frac{z-Zmin}{Pas\_Z}$;
Affecter le point $P(x, y, z)$ au box $BI = (i, j, k)$;
Fin pour
Fin

Figure 10 : Algorithme de création de boxes

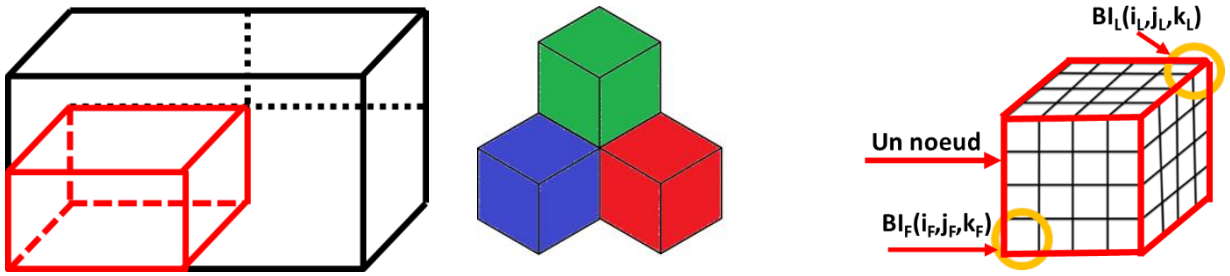
4.2.3 Étape 3 : Créer l'octree

Dans cette étape, le nuage de points est transformé en un octree. Pour minimiser l'espace de stockage, les nœuds père et fils sont représentés directement par leurs *BI* définis précédemment. L'algorithme 7, décrit par la Figure 11, récapitule la création de l'octree. Tout d'abord, le nombre maximal N de points dans chaque nœud de l'octree est introduit. Ensuite, des boxes sont utilisées pour créer l'octree. La racine de l'octree (père composé de toutes les boxes) et ses limites sont définies par les indices du premier et du dernier box ; elle est considérée comme étant le premier nœud. Pour chaque nœud, si le nombre de points est supérieur ou égal à N, une subdivision sur huit nœuds est appliquée ; sinon, aucune subdivision n'est opérée. Enfin, un tableau des nœuds obtenus est créé.

Algorithme 7 : Création d'un octree
Entrée
Boxes : box_1, box_2, box_3...box_K ;
N : nombre maximum de points dans chaque nœud de l'Octree ;
L'Octree racine (père) est composé de toutes les boxes ;
Le premier nœud est défini comme la racine de l'Octree ;
Sortie
Octree(Racine, $fils_1$, $fils_2$.,..,$fils_8$.,..) ;
Début
Pour chaque nœud de l'octree
Tant que (nombre de points dans chaque nœud ≥N)
Subdiviser l'octree ;
Fin pour
Créer un tableau à partir de l'arbre octree obtenu ;
Chaque octree est défini par ses fils, son père, ses trois voisins, sa position, le nombre de ses points et ses limites ;
Les limites d'un nœud de l'octree sont définies par l'index de la première et de la dernière boxe, qui appartient à ce nœud ;
Fin

Figure 11 : Algorithme de création de l'Octree

Chaque nœud de l'octree est défini par plusieurs informations : père et enfants (Figure 12a), trois voisins (Figure 12b), position, nombre de points et limites basées sur les indices des premier et dernier boxes (Figure 12c).



a. Père et ses fils b. Voisins d'un nœud c. Limite d'un Octree BI_F et BI_L

Figure 12 : Informations relatives à l'Octree

4.3 Partitionnement en utilisant K-means de base

L'algorithme 8, illustré par la Figure 13, décrit l'application de l'approche BK pour partitionner le nuage de points en partitions non-chevauchées. Il tente de décomposer directement le nuage de points en un ensemble de clusters disjoints (sous-ensembles de points). Une fois le nuage

de points connu, la première étape consiste à introduire le nombre de clusters demandés. Ensuite, l'approche conçoit les k centres $\mu_1, \ldots, \mu_k$ pour les k clusters parmi le nuage de points ; ces centres sont choisis aléatoirement. Le partitionnement se déroule en deux étapes :

- **Affectations** : le centre le plus proche est localisé pour chaque point non-centré. Ainsi, k clusters $C_1, \ldots, C_k$ sont définis, où $C_i = \{$l'ensemble des points les plus proches du centre $\mu_i\}$,
- **Représentation** : recalculer les centres pour les nouvelles partitions.

Algorithme 8 : K-means de base
Entrée
Ensemble de points : x ;
Nombre de clusters demandés : k ;
Sortie
Partition de k clusters (C_1, C_2, C_3, ..., C_k) ;
Début
Initialisation aléatoire des centres C_k ;
Répéter
Affectation : Générer une nouvelle partition en affectant chaque objet au groupe dont le centre est le plus proche : $x_i \in C_k \; si \; \forall \, j \; |x_i - \mu_k| = min|x_i - \mu_k|, \mu_k$ représente le centre de la classe k ;
Représentation : Calculer les centres associés à la nouvelle partition $\mu_k = \frac{1}{N} \sum_{x_i \in C_k} x_i$;
Jusqu'à la convergence de l'algorithme vers une partition stable ;
Fin

Figure 13 : Algorithme de partitionnement en utilisant K-means de base

4.4 Partitionnement en utilisant K-means global (GK)

L'algorithme K-means global (GK) est une variante du BK de base ; il s'agit d'une solution au problème d'initialisation de K-means. Il vise à obtenir une solution globalement optimale. Il consiste à effectuer un clustering incrémental et à ajouter dynamiquement un nouveau centre suivi de l'application de K-means jusqu'à la convergence.

Algorithme 9 : K-means global
Entrée
Ensemble de points : P ;
Nombre de clusters demandés : k ;
Sortie
Partition de k clusters (C_1, C_2, C_3, ..., C_k) ;
Début
C_1=Centre de gravité de l'ensemble des données ;
Répéter
Initialiser les (i-1) centres par le résultat de l'étape précédente ;
Pour chaque donnée x de $P\{$ //Trouver le i^{ème} centre
Considérer x comme le i^{ème} centre ;
Affecter les données au centre le plus proche ;
Calculer l'erreur quadratique pour $C_i = x$;
Fin pour
Conserver le centre $C_i = x$ qui minimise l'erreur quadratique ;
Appliquer K-means jusqu'à convergence ;
Jusqu'à l'obtention d'une partition en k groupes ;
Fin

Figure 14 : Partitionnement en utilisant K-means global

L'algorithme 9 (Figure 14) illustre ce concept. Une fois le nuage de points connu et le nombre de partitions introduit, la première étape calcule les centres choisis un par un ; le premier centre est le centre de gravité du nuage de points (résultat de l'application de K-means avec $k = 1$). Les autres centres sont sélectionnés à partir du nuage de points où chaque point est un candidat pour devenir un centre. Ensuite, il sera testé avec le reste de l'ensemble des points ; le meilleur candidat est celui qui minimise la fonction objective donnée par l'équation (1) :

$$j = \sum_{i=1}^{k} \sum_{x_{j \in c_j}} \left\| x_j - c_i \right\|^2 \quad (1)$$

4.5 Partitionnement K-means global rapide (FGK)

L'algorithme K-means global rapide (FGK) propose une nouvelle stratégie pour accélérer l'algorithme GK. Tous les points peuvent être candidats pour être un centre ; néanmoins, il évite d'assigner le point aux centres les plus proches et de calculer l'erreur quadratique b_n. Le nouveau centre est le candidat qui maximise l'équation (2) :

$$b_n = \sum_{i=1}^{N} max\left(d_{k-1}^{i} - \|x_n - x_i\|^2, 0\right) \quad (2)$$

où d_{k-1}^{i} représente la distance entre x_i et son centre le plus proche parmi les k-1 centres.

L'algorithme 10 (Figure 15) est mis en œuvre pour améliorer le temps d'exécution et garantir de bons résultats comme ceux fournis par la stratégie précédente.

Algorithme 10 : K-means global rapide
Entrée
Ensemble de points : P ;
Nombre de clusters demandés : k ;
Sortie
Partition de k clusters (C_1, C_2, C_3, ..., C_k) ;
Début
C_1=centre de gravité de l'ensemble P ;
Répéter
Initialiser les (i-1) centres par le résultat de l'étape précédente ;
Pour chaque donnée x de P // Trouver le i^{ème} centre
Considérer x comme le i^{ème} centre ;
Calculer d_{k-1}^{i} ;
Calculer b_n pour $C_i = x$ en utilisant l'équation (2) ;
Fin pour
Conserver le centre $C_i = x$ qui minimise l'erreur quadratique ;
Jusqu'à obtenir une partition en k groupes ;
Appliquer K-means jusqu'à convergence ;
Fin

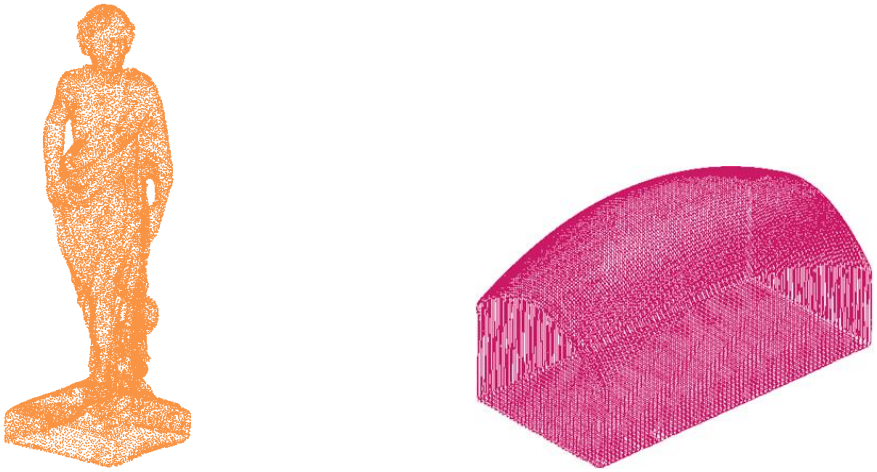
Figure 15 : Partitionnement en utilisant K-means global rapide

5. Validation et discussion des résultats obtenus

La méthodologie proposée a été mise en œuvre en utilisant le langage orienté objet C++Builder et la bibliothèque graphique OpenGL. Les algorithmes développés tournent sur un microprocesseur Intel Core i3 avec 6 Go de RAM et une résolution de 1366x768 pixels sous Windows 7.

5.1 Validation des approches développées

La validation de la méthodologie est effectuée via deux pièces complexes ; chaque pièce est représentée par son nuage de points. Ces pièces sont la statue « *asklepios_3d* » qui contient 28692 points (Figure 16a) et la « *pièce convexe* » avec 900 points (Figure 16b).

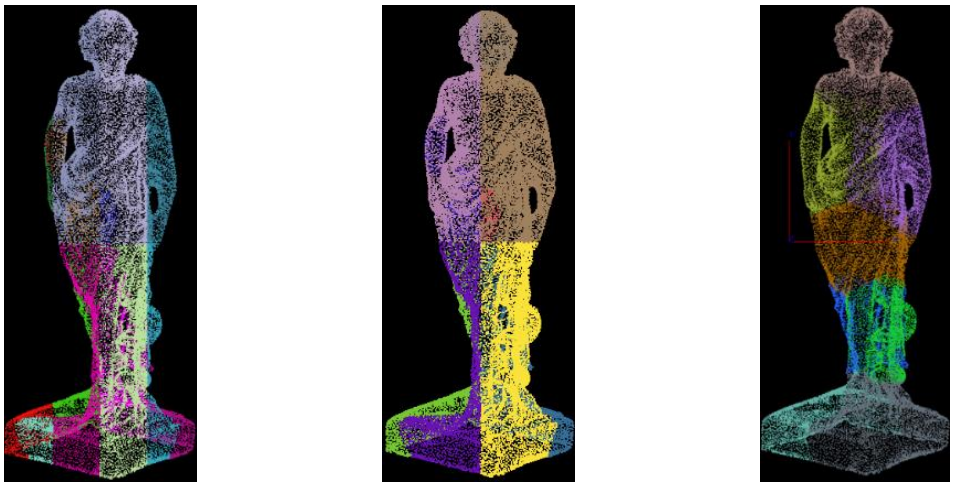


a. Statue « *asklepios_3d* » b. « *pièce convexe* »

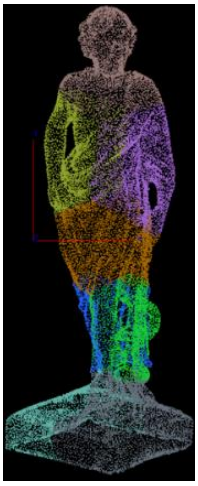
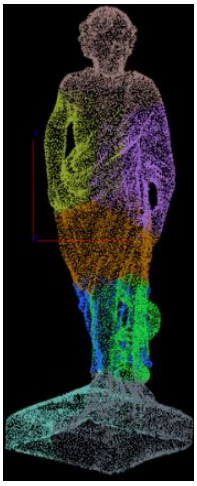
Figure 16 : Nuages de points des deux pièces de validation

5.1.1 Validation sur la pièce « *asklepios_3d* »

Le partitionnement du modèle de la pièce « *asklepios_3d* » par les différentes approches est décrit comme suit. La Figure 17a montre les résultats du partitionnement généré par la méthode Octree. La Figure 17b, quant à elle, montre les partitions générées par l'approche des cellules. La Figure 17c montre les partitions générées par l'algorithme K-means ; tandis que la Figure 17d par l'algorithme k-means global et Figure 17e l'algorithme k-means global rapide.



a. Octree b. Cellules c. K-means.

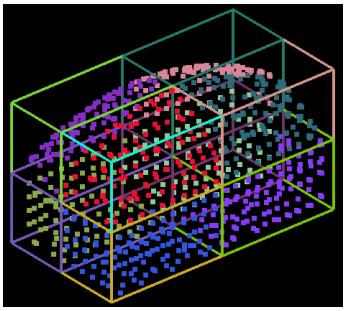
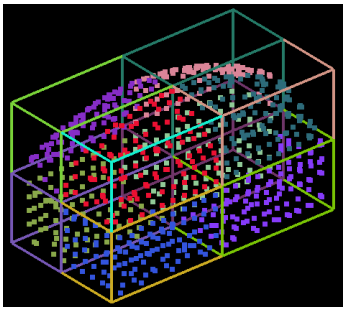
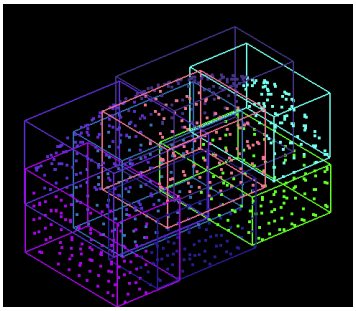



d. K-means global e. K-means global rapide

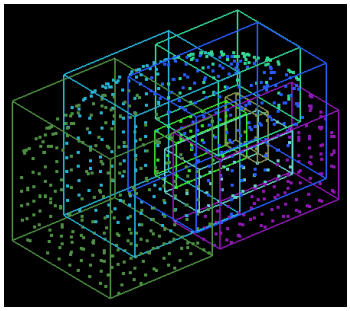
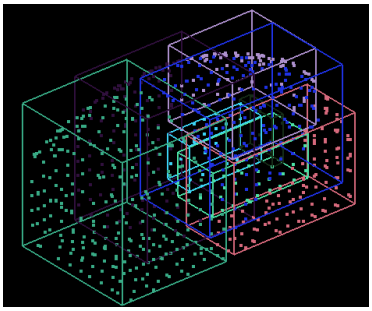
Figure 17 : Partitionnement du modèle de la pièce « *asklepios_3d* » par les différentes approches

5.1.2 Validation sur la « *pièce convexe* »

Le partitionnement du modèle de la « *pièce convexe* » par les différentes approches est présenté dans la Figure 18a (octree), Figure 18b (cellules), Figure 18c (K-means), Figure 18d (K-means global), et Figure 18d (K-means global rapide) :

a. Octree b. Cellules c. K-means

d. K-means global e. K-means global rapide

Figure 18 : Partitionnement du modèle de la « *pièce convexe* » par les différentes approches

5.2 Discussion des résultats obtenus

Les performances de chaque algorithme sont évaluées en termes de nombre de points pour chaque partition. La meilleure méthode de partitionnement est celle qui génère un nombre équilibré (plus ou moins égal) de points à l'intérieur des partitions trouvées ; ceci permet d'obtenir une charge de travail équilibrée sur les différents processeurs.

5.2.1 Comparaison des résultats pour la pièce « *asklepios_3d* »

Le tableau 1 illustre le nombre de points de l'espace de la pièce complexe « *asklepios_3d* » (avec 28692 points) attribué à chacun des huit clusters pour les quatre algorithmes mis en œuvre. La

dernière colonne donne le paramètre $\Delta point$, qui représente la différence entre le nombre maximum et minimum de points dans les partitions pour chaque méthode ($\Delta point = Max - Min$).

Pour la méthode octree, le nombre maximum (**) et minimum(*) de points à partir du tableau 1 pour les huit groupes est d'environ 7000 et 100, respectivement. Cela signifie qu'une différence significative existe concernant le nombre de points dans les partitions obtenues. Ces résultats sont similaires à ceux de l'approche de partitionnement des cellules. En revanche, pour le partitionnement BK, le nombre de points maximum et minimum dans tous les groupes est plus approprié (max=6000, min=2400). Enfin, la plus grande amélioration peut être observée sur la méthode de partitionnement FGK où max=4800 et min=2800. En effet, pour cette méthode, le nombre de points dans la quasi-totalité des partitions est d'environ 3000 points.

Méthode	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6	Group 7	Group 8	$\Delta point = Max - Min$
Octree	2859	3534	464	146*	2744	7124	4598	7223**	7077
Cellules	3062	563	4587	2746	7054**	3607	149*	6924	6905
BK	3565	4087	2411	4089	1597*	2368	4296	6279**	4356
FGK	3694	3333	3557	4876**	3713	2862*	3484	3173	1703

Tableau 1 : Nombre de points par groupe pour la pièce « *asklepios_3d* »

La Figure 19 montre clairement que l'approche BK offre le meilleur partitionnement (les groupes sont quasi-homogènes) par rapport aux autres méthodes octree et cellules.

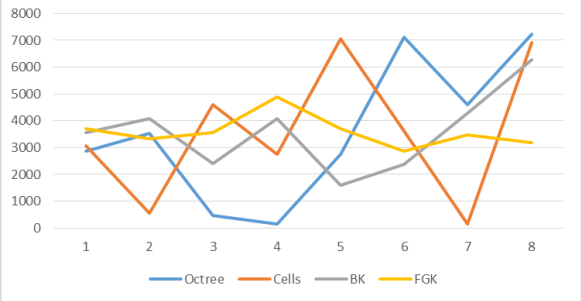


Figure 19 : Nombre de points par groupe pour la pièce « *asklepios_3d* »

Une autre évaluation a été réalisée entre tous les algorithmes en utilisant le paramètre $\Delta point$. La méthode FGK montre de meilleures performances de clustering par rapport à l'approche BK originale et aux autres méthodes (Figure 20) ; cependant, les résultats dépendent fortement de la densité du nuage de points.

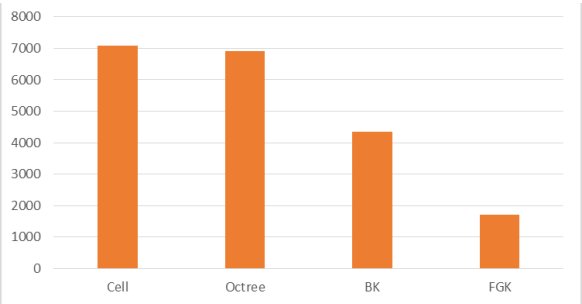


Figure 20 : Performances de partitionnement ($\Delta point$) pour la pièce « *asklepios_3d* »

Pour démontrer les performances de notre contribution, le nombre de points attribués pour chaque groupe est vérifié par rapport au nombre moyen de points. Dans ce cas, le nombre moyen de points à l'intérieur d'une partition est de 3586 points (28692 points sur 8 groupes). En utilisant

l'équation (3), les valeurs obtenues de V sont présentées dans le tableau 2. La dernière colonne représente la somme des valeurs V liées à la méthode de partitionnement.

$$V_i = |Nombre\ de\ points\ affectés\ au\ Groupe_i - Nombre\ moyen\ de\ points| \quad (3)$$

Le tableau 2 montre que les valeurs maximales de V sont de 3637 et 3468 pour les méthodes octree et cellules, respectivement. Pour les approches BK et FGK, les valeurs maximales de V sont de 2693 et 1290, respectivement. Quant aux valeurs minimales de V, elles sont de 52 et 21, pour les méthodes octree et cellules, respectivement. Pour les approches BK et FGK, les valeurs minimales de V sont de 21 et 29, respectivement. La dernière colonne du tableau 2 montre que la somme minimale des valeurs V est celle relative à l'algorithme FGK. Cela confirme que la méthode FGK présente de meilleures performances de clustering et surpasse les autres méthodes.

Méthode	**V_1**	**V_2**	**V_3**	**V_4**	**V_5**	**V_6**	**V_7**	**V_8**	**Somme**
Octree	727	52*	3122	3440	842	3538	1012	3637**	**16370****
Cellules	524	3023	1001	840	3468**	21*	3437	3338	**15652**
BK	21*	501	1175	503	1989	1218	710	2693**	**8810**
FGK	108	253	29*	1290**	127	724	102	413	**3046***

Tableau 2 : Nombre de points par groupe par rapport au nombre moyen de points pour la pièce « *asklepios_3d* »

5.2.2 Comparaison des résultats pour la « *pièce convexe* »

Les résultats obtenus pour la « *pièce convexe* », définie par 3447 points, en utilisant les algorithmes de partitionnement mises en œuvre (octree, cellule, BK et FGK) sont présentés dans le Tableau 3. Les résultats obtenus par chaque méthodes sont évalués selon (*i*) le nombre de points par groupe, (*ii*) le nombre maximum et minimum de points dans chaque partition et (*iii*) le paramètre $\Delta point$ défini précédemment.

Méthode	**Group 1**	**Group 2**	**Group 3**	**Group 4**	**Group 5**	**Group 6**	**Group 7**	**Group 8**	$\Delta point = Max - Min$
Octree	6903	6831	6409	6477*	4845	4978	4335	4620**	2568
Cellules	7354	4933	6470	7094	4893**	6407	4057*	4190	3297
BK	5947	5509	6434	6014	6425*	4696	4920	5453**	1738
FGK	5603	6340	6411**	5853	4698*	5728	5311	5453	1713

Tableau 3 : Nombre de points par groupe pour la pièce « *pièce convexe* »

Pour la méthode octree, une grande différence existe entre le nombre maximum et minimum de points dans les partitions obtenues. Cette différence est confirmé par le paramètre $\Delta point$=2568. Ces résultats sont similaires à ceux de l'approche de partitionnement des cellules ($\Delta point$ =3297).

Pour le partitionnement BK, le nombre de points maximum et minimum dans tous les groupes est plus approprié ($\Delta point$ = 1738). Enfin, la plus grande amélioration peut être observée pour la méthode FGK où ($\Delta point$ = 1713). En effet, pour cette méthode, le nombre de points dans la quasi-totalité des partitions est d'environ 5000 points.

La Figure 21 et Figure 22 montrent clairement que l'approche FGK offre le meilleur partitionnement (les groupes sont quasi-homogènes) par rapport aux autres méthodes octree et cellules.

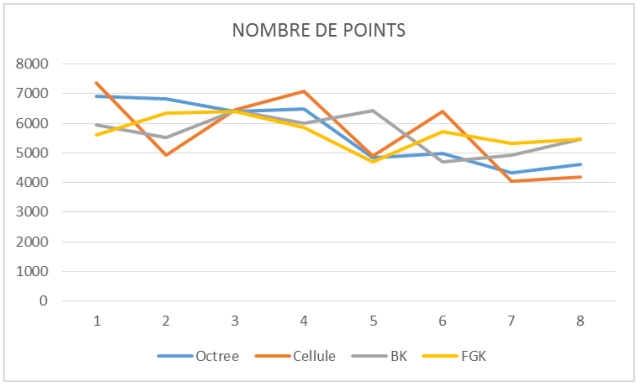


Figure 21 : Nombre de points par groupe pour la pièce « *pièce convexe* »

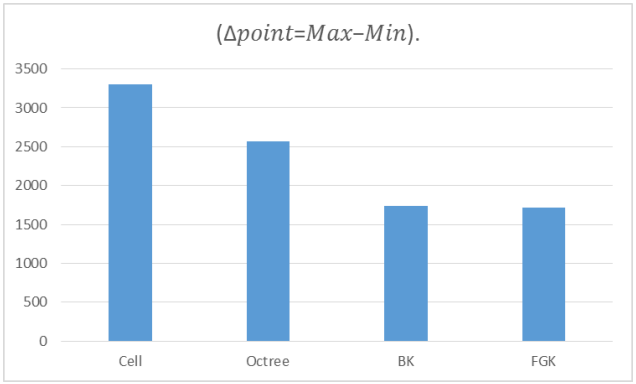


Figure 22 : Performances de partitionnement ($\Delta point$) pour la pièce « *pièce convexe* »

Pour cette pièce, le nombre moyen de points à l'intérieur d'une partition est de 5674 points (3447 points sur 8 groupes). En utilisant l'équation (3), les valeurs obtenues de V sont présentées dans le tableau 4. La dernière colonne représente la somme des valeurs V liées à la méthode de partitionnement. Elle montre que les sommes minimales de V est celles relatives aux algorithmes BK et FGK. Cela confirme que les méthodes BK et FGK offrent de meilleures performances de regroupement et surpassent les autres méthodes (octree et cellules).

Méthode	**V_1**	**V_2**	**V_3**	**V_4**	**V_5**	**V_6**	**V_7**	**V_8**	**Somme**
Octree	1128	1157*	735	803	829	696	1339	1054**	**7741****
Cellules	1680	741	796	1420	781*	733*	1617	1484	**9252**
BK	273*	165	760	340	751	978	754	221**	**4242**
FGK	108	253	29*	1290**	127	724	102	413	**3046***

Tableau 4 : Nombre de points par groupe par rapport au nombre moyen de points pour la « *pièce convexe* »

6. Conclusion

Ce chapitre a présenté une méthodologie efficace de division spatiale par regroupement de points. Son objectif principal consiste à utiliser les algorithmes K-means de base (BK) et son dérivé K-means global rapide (FGK) pour subdiviser le nuage de points en sous-nuages. La méthodologie développée a été testée et validée sur les modèles de la « *statue asklepios_3d* » et « *pièce convexe* ». En outre, pour évaluer la méthodologie proposée, une étude comparative des quatre méthodes (octree, cellules, BK et FGK) a été établie en termes de nombre de points par groupe. Les résultats obtenus ont démontré les hautes performances de la méthode FGK en ce qui concerne (*i*) l'homogénéité des partitions, et (*ii*) la charge de travail équilibrée sur les différents processeurs.

Les travaux futurs utiliseront d'autres dérivés de la méthode K-means, telles que les k-means incrémentaux.

Chapitre 5
Triangulation de Delaunay parallèle

1. Introduction

En général, l'algorithme « *Diviser pour régner* » est conçu pour le parallélisme. Il divise récursivement l'ensemble des points en sous-nuages de points. Indépendamment, ces régions sont ensuite triangulées simultanément.

Ce chapitre s'intéresse à la fusion des sous-triangulations afin d'obtenir la triangulation globale du nuage de points. Les détails de l'étape de fusion sont présentés dans ce qui suit.

2. Problématique et état de l'art

Plusieurs algorithmes parallèles sont développés en utilisant plus d'un processeur, comme dans [87] [91] [93] [102-104]. Ici, le nuage de points est divisé récursivement en sous-régions, chacune étant assignée à un processeur. Indépendamment, ces sous-régions sont ensuite triangulées et fusionnées en un seul domaine simultanément.

Cignoni et al.[105] ont proposé DeWall qui est une approche légèrement différente. En effet, au lieu de fusionner des triangulations partielles, une phase de division plus complexe est appliquée pour partitionner l'ensemble des points et construire d'abord la triangulation de fusion. Le résultat est ensuite utilisé pour construire la triangulation des deux sous-ensembles de points.

Les auteurs de[90-91] ont défini un algorithme parallèle basé sur le partitionnement du nuage de points selon une seule dimension. Après, chaque processeur effectue la sous-triangulation. A la fin, les sous-triangulations obtenues sont fusionnées dans l'ordre inverse du partitionnement de l'ensemble de points.

Comme il est difficile de fusionner les sous-triangulations, Chen [88] a introduit l'approche de la zone affectée, qui détermine la zone qui peut être modifiée lors de la fusion des sous-triangulations. Plus tard dans [109], la triangulation globale est obtenue en deux étapes : (*i*) la génération du premier tétraèdre et (*ii*) la génération du tétraèdre générique.

Cependant, ces méthodes contiennent des étapes de fusion compliquées. Pour cela, Wu et al. [87] ont présenté ParaStream, qui est un algorithme parallèle robuste. ParaStream utilise le k-tree pour distribuer les tâches de triangulation et de fusion entre les cœurs de processeur afin d'améliorer l'équilibre de la charge. De plus, cet algorithme met en œuvre une étape de suppression efficace pour réduire la mémoire utilisée en cours d'exécution.

De même, Lo [91] a présenté un schéma 2D pour la génération de la triangulation de Delaunay parallèle. Ce schéma est basé sur la partition des points en zones. Les points sont d'abord répartis en cellules approximativement en nombre égal de points. Les cellules sont regroupées en zones, dans lesquelles la triangulation de Delaunay est construite par insertion simultanée de points cellule par cellule dans chaque zone. Des triangles à la frontière entre les zones peuvent être créés en parallèle en ajoutant des limites de cellules pour chaque zone pour assurer le critère du cercle vide des triangles de frontière. Les triangles redondants à la frontière entre les zones peuvent être facilement éliminés. L'algorithme proposé est un algorithme générique qui pourrait être facilement étendu à des dimensions plus élevées. L'implémentation d'une version en 3D est présentée dans [92].

De nombreux algorithmes parallèles pour la triangulation de Delaunay sont proposés pour améliorer les performances et surmonter les limitations des techniques existantes. Par ailleurs, la triangulation de Delaunay parallèle 3D pour une distribution non-uniforme de points reste une question intéressante, qui pourrait être développée. Pour cela, certains types d'algorithmes sont combinés et présentés comme des méthodes hybrides pour rendre la reconstruction plus robuste.

Wu et al. [102] ont utilisé l'algorithme « *Diviser pour régner* » deux fois pour augmenter l'efficacité du traitement d'un nuage de points dense. Cet algorithme divise automatiquement le nuage de points en plusieurs sous-nuages le long des directions de l'axe des X et de l'axe des Y en alternance, en fonction du nombre de points et de leur répartition spatiale. Ensuite, l'algorithme de Guibas et Stolfi [110] est appliqué pour construire des sous-triangulations de Delaunay dans chaque sous-nuage. Enfin, les sous-triangulations sont fusionnées sur la base de l'arbre binaire. Les trois étapes séquentielles sont traitées à l'aide de la technologie parallèle multitâches. Les résultats obtenus

montrent que l'algorithme parallèle proposé est efficace pour construire la triangulation de Delaunay avec une bonne accélération. De même, Lin et ses collègues [103] ont choisi le cloud pour exécuter une nouvelle méthode hybride : L'insertion incrémentale et l'algorithme « *Diviser pour régner* ». Les deux travaux, [102-103], sont développés pour la triangulation de Delaunay en 2D. D'autres auteurs, comme [93], [111-113], ont considéré leurs algorithmes comme pratiques en 3D.

Compte tenu de la simplicité et de la linéarité de l'insertion de grille régulière, Lo dans [93], a proposé un schéma d'insertion multi-grilles pour la triangulation de Delaunay 3D de distributions de points non-uniformes. En effet, l'auteur a appliqué récursivement l'insertion de grille régulière à un sous-ensemble arbitraire de l'ensemble de points original.

Les auteurs de [111-113] ont proposé un nouvel algorithme basé sur la division du nuage de points en plusieurs partitions indépendantes qui peuvent ensuite être triangulées en parallèle. L'implémentation et l'évaluation ont été réalisées en utilisant MPI (Message Passing Interface) et C++ sur un cluster.

Dans [114], l'auteur présente un nouveau schéma de parallélisation évolutif pour générer la triangulation de Delaunay 3D d'un ensemble de points. Ce nouveau schéma de parallélisation est exprimé en deux parties. La première partie consiste à accélérer l'implémentation séquentielle de l'algorithme d'insertion incrémentale de Delaunay en utilisant une structure de données dédiée simple et un tri efficace des points. La deuxième partie est une application multithreads du noyau Delaunay ; elle permet d'insérer simultanément les points. Les coordonnées de la courbe de Moore [111] sont utilisées pour partitionner l'ensemble de points. Les conflits sont gérés en modifiant les partitions avec un simple redimensionnement de la courbe de remplissage d'espace.

L'étude des travaux susmentionnés montre que la génération de la triangulation à partir d'un nuage de points dense et non-structuré reste un obstacle pour les scientifiques traitant des nuages de points à grande échelle. Avec l'émergence des machines multi-cœurs, plusieurs algorithmes parallèles pour la triangulation de Delaunay sont proposés pour améliorer les performances et surmonter les limitations des techniques existantes.

Ce chapitre propose une nouvelle approche de fusion des sous-triangulations selon les trois directions X, Y et Z afin d'obtenir la triangulation de Delaunay en un temps de calcul réduit.

3. Avantages de la programmation parallèle

Certains algorithmes séquentiels ont un temps d'exécution très long. Ceci inclue par exemple les algorithmes de calcul des factoriels ou bien les algorithmes de recherche de graphes sur des grandes données à complexité temporelle et exponentielle ou bien factorielle, etc. Ces algorithmes nécessitent le recours aux architectures parallèles pour paralléliser les algorithmes et les rendre plus efficaces en termes de temps d'exécution, de performances de coût et d'exploitation des ressources.

Le parallélisme est omniprésent dans les ordinateurs d'aujourd'hui. Au niveau microscopique, les processeurs multiplient les unités arithmétiques pipelinées sur un même circuit intégré. Au niveau macroscopique, les stations de travail sont interconnectées en grappes pour construire des supercalculateurs à peu de frais. Dans les deux cas, l'algorithmique parallèle permet de comprendre et de maîtriser les concepts fondamentaux à mettre en œuvre pour l'utilisation de plateformes distribuées. Elle emprunte beaucoup à l'algorithmique classique dans sa problématique (conception, analyse, étude de complexité), mais s'enrichit d'une nouvelle dimension avec l'exploitation simultanée de plusieurs ressources. Les programmes parallèles sont écrits et sont exécutés sur plusieurs processeurs répartis géographiquement dans le monde, connectés via un réseau de télécommunication dans une architecture parallèle à mémoire distribuée. Dans ce cas, la communication entre les processus se fait par échange de messages. Aussi, les algorithmes parallèles peuvent s'exécuter sur une seule machine avec un processeur multi-cœurs dans une architecture parallèle à mémoire partagée où le parallélisme se fait par des threads s'exécutant chacun sur un cœur différent. La programmation parallèle est basée sur les algorithmes parallèles, dont le développeur doit considérer plusieurs facteurs tels que la partie du programme qui peut être traitée en parallèle, la manière de distribuer les données, les dépendances des données, la répartition de charges entre les

processeurs et les synchronisations entre les processeurs. Essentiellement, deux méthodes existent pour concevoir un algorithme parallèle. L'une consiste à détecter et à exploiter le parallélisme à l'intérieur d'un algorithme séquentiel déjà existant. L'autre consiste à inventer un nouvel algorithme dédié au problème donné.

4. Solution proposée

L'objectif de cette solution est de générer la triangulation de Delaunay 3D globale (Figure 1). Le nuage de points considéré dans cette solution est partitionné en plusieurs sous-nuages de points en utilisant les Octrees. Aussi, les sous-nuages de points sont triangulés simultanément sur plusieurs processeurs. L'approche proposée consiste à développer une nouvelle architecture de fusion tridimensionnelle, *Ordre de Fusion Amélioré* (*OFA*), pour fusionner les nœuds de l'octree sans tenir compte de la direction de partitionnement. À la fin, la triangulation globale est obtenue par la fusion des sous-triangulations.

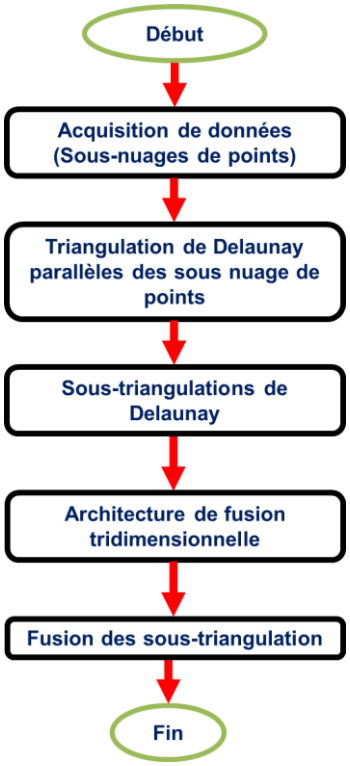


Figure 1 : Diagramme global de la solution proposée

4.1 Architecture de fusion tridimensionnelle

Une fois les sous-triangulations sont calculées en parallèles, leur fusion est nécessaire pour générer la triangulation globale. Pour y faire face, une nouvelle architecture de fusion, *OFA*, est proposée pour spécifier l'ordre de fusion pour chaque nœud de l'octree. L'architecture est réalisée dans trois directions X, Y et Z ; elle est décrite dans le diagramme de la Figure 2 ci-dessous.

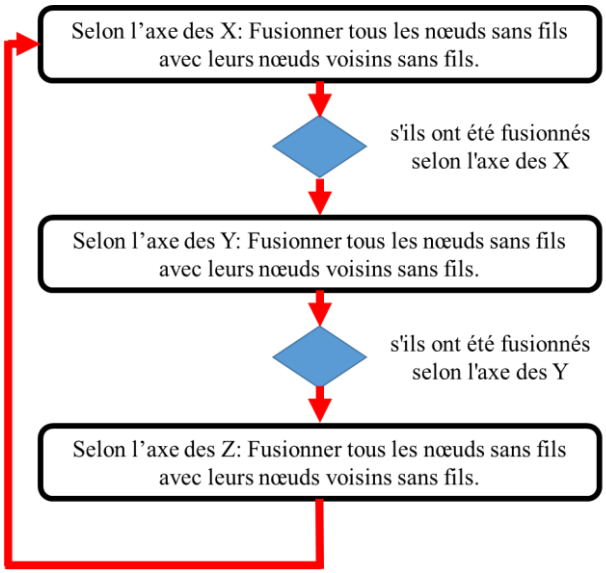
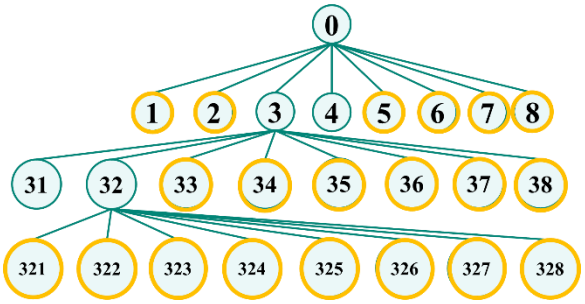
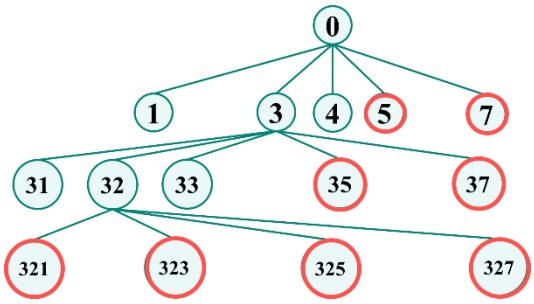


Figure 2 : Organigramme de la fusion tridimensionnelle

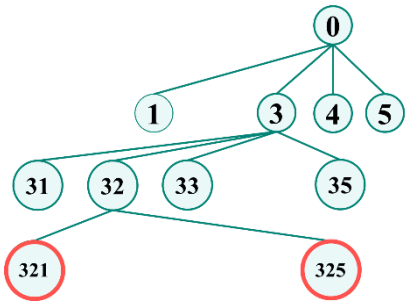
La génération de la triangulation globale commence par la détermination de tous les nœuds et voisins sans fils le long de l'axe X. S'ils existent, tous les nœuds trouvés sont fusionnés deux par deux (paires : nœud et son voisin). La même méthode est appliquée le long de l'axe des Y et de l'axe des Z. Ce processus de fusion le long des trois axes est répété jusqu'à ce que le nœud parent soit atteint. Tous les nœuds trouvés le long de l'axe X sont fusionnés, simultanément. La même opération est réalisée pour les nœuds trouvés le long de l'axe Y et ceux trouvés le long de l'axe Z (Figure 3).



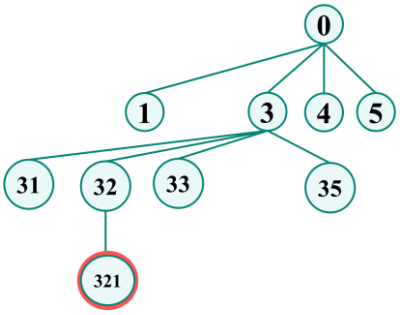
a. Nœuds de l'arbre



b. Fusion selon l'axe des X



c. Fusion selon l'axe des Y



d. Fusion selon l'axe des Z

Figure 3 : Architecture de fusion selon les trois axes X, Y et Z

L'algorithme de la Figure 4 résume cette approche.

Algorithme 1 : Architecture de fusion
début
Introduire les nœuds de l'Octree ;
répéter
Trouver tous les nœuds sans fils le long de l'axe X ;
pour chaque nœud
Déterminer ses nœuds voisins sans fils le long de l'axe X ;
Si oui, fusionner tous les nœuds deux par deux le long de l'axe X en parallèle (nouveau niveau de parallélisme) ;
fin pour
Déterminer tous les nœuds sans fils le long de l'axe Y ;
pour chaque nœud
Déterminer ses nœuds voisins sans fils le long de l'axe Y ;
Si oui, fusionner tous les nœuds deux par deux le long de l'axe Y en parallèle (nouveau niveau de parallélisme) ;
fin pour
Déterminer tous les nœuds sans fils le long de l'axe Z ;
pour chaque nœud
Déterminer ses nœuds voisins sans fils le long de l'axe Z ;
Si oui, fusionner tous les nœuds deux par deux le long de l'axe Z en parallèle (nouveau niveau de parallélisme) ;
fin pour
jusqu'à ce que (le nœud parent soit atteint) ;
fin

Figure 4 : Algorithme de l'architecture de fusion

4.2 Zone affectée

La zone affectée consiste en la région de la triangulation de chaque sous-nuage de points qui est susceptible d'être modifiée lors de la fusion des sous-triangulations. Ces régions représentent les portions de zones mitoyennes entre les différents sous-nuages dans les trois directions X, Y et Z tel que montré par la Figure 5.

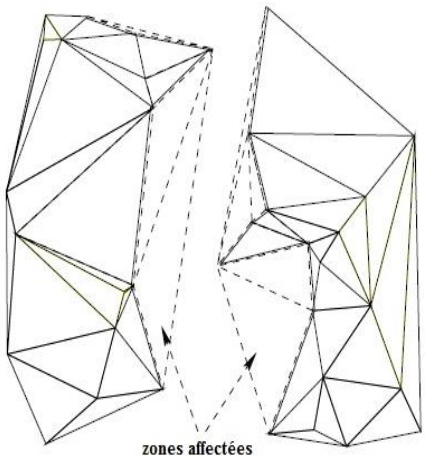


Figure 5 : Exemple de deux zones affectées

Pour accélérer cette étape et éviter le traitement de tous les tétraèdres des nœuds, seuls les tétraèdres dont la sphère circonscrite entre en contact avec les limites des nœuds voisins, selon les

trois directions X, Y et Z, seront considérés. Pour chaque nœuds, six (06) zones affectées doivent être déterminées : deux selon chaque axe. Les zones déterminées sont : zone affectée droite, zone affectée

gauche, zone affectée haut, zone affectée bas, zone affectée avant et zone affectée arrière. La Figure 6 résume le processus de création des zones affectées.

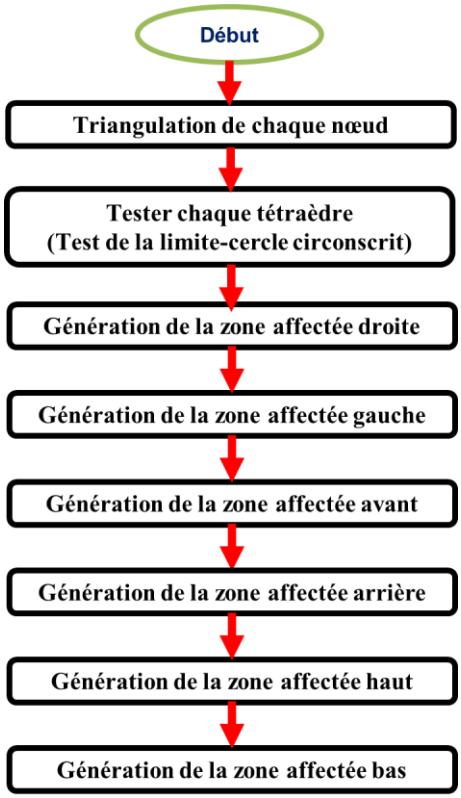
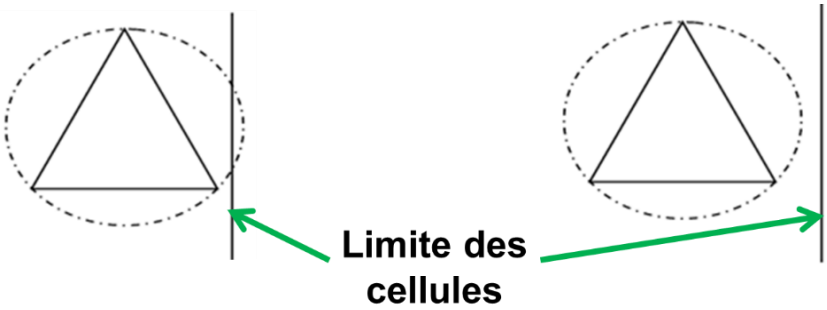


Figure 6 : Organigramme de la génération des zones affectées

4.2.1 Test de la limite-cercle circonscrit

Ce test est utilisé pour la génération des différentes zones affectées décrites précédemment. Le but étant de déterminer si la sphère circonscrite d'un tétraèdre entre en contact avec un plan représentant la limite de nœuds contenant le tétraèdre. Plusieurs cas peuvent apparaitre lors du test. Pour mieux comprendre la complexité de la définition des zones affectées, le processus est décrit à travers un exemple dans le plan où les tétraèdres seront remplacés par des triangles, les sphères par des cercles et le plan par une droite (Figure 7).



a. Triangle affecté b. Triangle non affecté

Figure 7 : Test du cercle circonscrit

Parfois, des cas existent où le test de la sphère circonscrite ne donne pas de résultat exact du fait que les limites des nœuds soient des plans et sont donc infinis. Ces cas sont mathématiquement corrects mais incohérents comme illustré par la Figure 8.

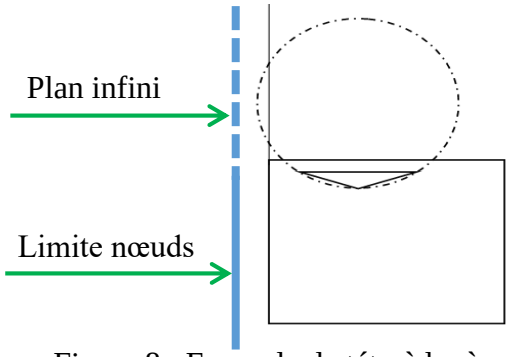


Figure 8 : Exemple de tétraèdre à ne pas considérer

D'autres cas existent et qui s'avèrent corrects par rapport au résultat souhaité tels que montrés par la Figure 9.

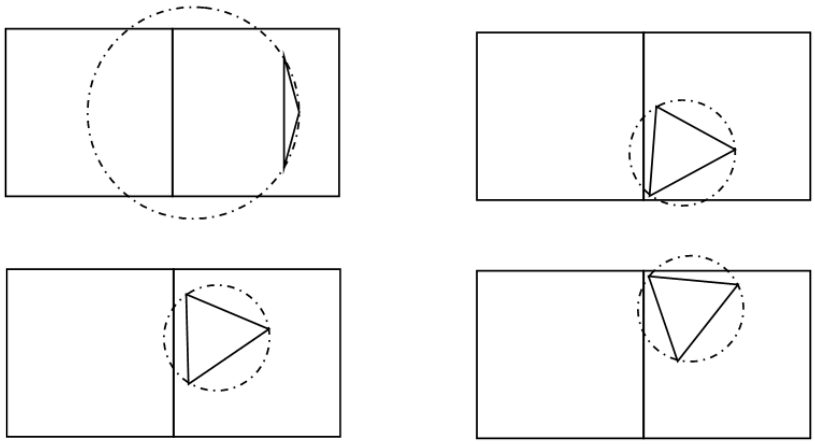


Figure 9 : Quelques exemples de cas rencontrés

Pour obtenir des résultats plus cohérents par rapport à l'approche développée et éviter les faux cas, il est recommandé d'appliquer les tests du cercle circonscrit plutôt que d'appliquer l'autre test de la sphère circonscrite. L'utilisation du test du cercle circonscrit est justifiée par le fait que l'intersection de la sphère d'un tétraèdre avec un plan (limite du nœud) est un cercle dont le rayon est facilement calculable (Figure 10).

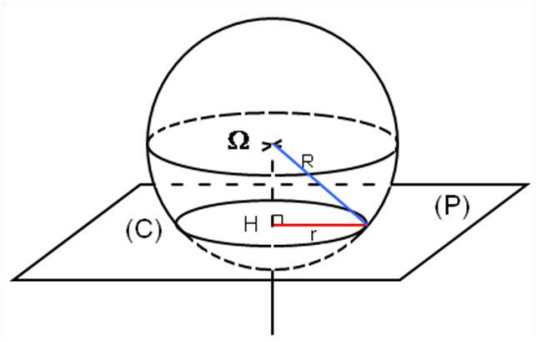


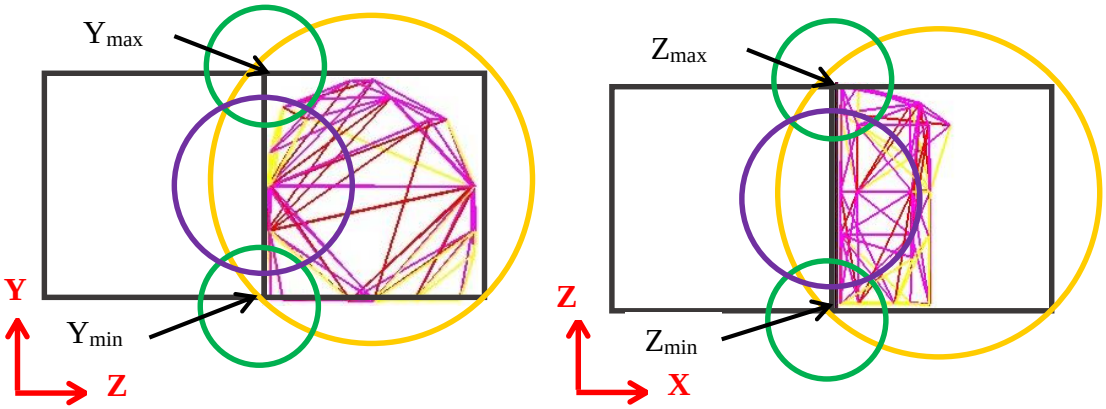
Figure 10 : Intersection d'une sphère avec un plan

4.2.2 Zone affectée droite (gauche)

La génération de la zone affectée droite (resp. gauche) suit le processus détaillé par les étapes suivantes et décrit par la Figure 11 :

- Vérifier l'intersection entre la sphère circonscrite et le plan X_{max} (resp. X_{min}),
- Si oui, calculer le rayon du cercle suite au calcul de l'intersection entre la sphère et le plan,

- Vérifier si le rayon est compris entre Z_{min} et Z_{max},
- Vérifier si le rayon est compris entre Y_{min} et Y_{max}.



a. Vérification selon l'axe Z b. Vérification selon l'axe Y

Figure 11 : Génération de la zone affectée droite (gauche)

4.2.3 Zone affectée avant (arrière)

La génération de la zone affectée arrière (resp. avant) suit le processus détaillé dans les étapes ci-après :

- Vérifier l'intersection entre la sphère circonscrite et le plan Y_{max} (resp. Y_{min}),
- Si oui, calculer le rayon du cercle d'intersection entre la sphère et le plan,
- Vérifier si le rayon est compris entre X_{min} et X_{max},
- Tester si le rayon est compris entre Z_{min} et Z_{max}.

4.2.4 Zone affectée haut (bas)

La génération de la zone affectée haut (resp. bas) est réalisée selon les étapes décrites comme suit :

- Vérifier si la sphère circonscrite s'intersecte avec le plan Z_{max} (resp. Z_{min}),
- Si oui, calculer le rayon du cercle d'intersection entre la sphère et le plan,
- Vérifier si le rayon est compris entre Y_{min} et Y_{max},
- Vérifier si le rayon est compris entre X_{min} et X_{max}.

Pour accélérer la fusion de deux sous-triangulations, les zones affectées des nœuds adjacents sont définies selon les trois directions comme suit :

- **Zones affectées selon l'axe X** : zone affectée droite et zone affectée gauche,
- **Zones affectées selon l'axe Y** : zone affectée avant et zone affectée arrière,
- **Zones affectées selon l'axe Z** : zone affectée haut et zone affectée bas.

En outre, pour éviter les calculs inutiles, la numérotation des nœuds indiquée sur la Figure 12 est utilisée. Par conséquent, le calcul nécessaire des zones affectées est donné comme suit :

- **Nombre impair** : zones affectées droite, arrière, haut et bas.
- **Nombre pair** : zones affectées gauche et avant.

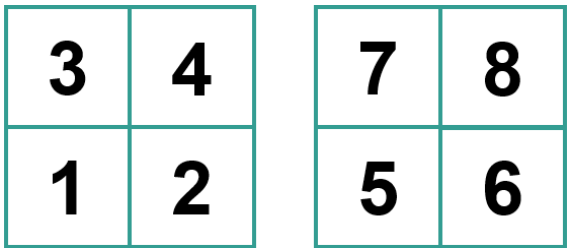


Figure 12 : Numérotation des nœuds

4.3 Fusion des sous-triangulations

La fusion est le cœur du processus de reconstruction du nuage de points subdivisé. Elle consiste à concaténer les sous-nuages deux à deux en utilisant les zones affectées des nœuds adjacents. Deux procédures de concaténation sont développées :

- La première procédure consiste à générer un premier tétraèdre qui servira de support pour la génération des autres tétraèdres [109].
- La deuxième procédure de concaténation consiste à générer trois sous-triangulations, la triangulation finale est obtenue par la fusion de ces trois sous triangulations [115].

4.3.1 Première procédure de fusion

4.3.1.1 Création du premier tétraèdre

La création du premier tétraèdre est basée essentiellement sur deux concepts tels que montré par la Figure 13 :

- Identifier les deux points les plus proches de chacune des deux zones affectées adjacentes,
- Tester la sphère circonscrite vide.

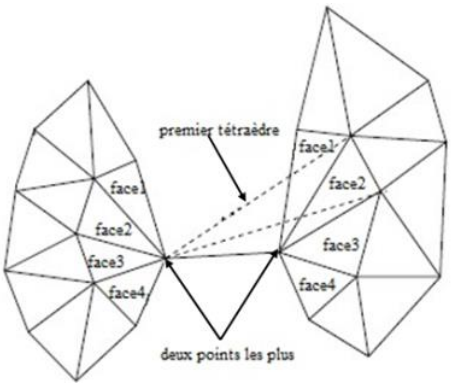


Figure 13 : Sélection des deux points et création du premier tétraèdre

La création du premier tétraèdre est décrite par l'organigramme suivant (Figure 14) :

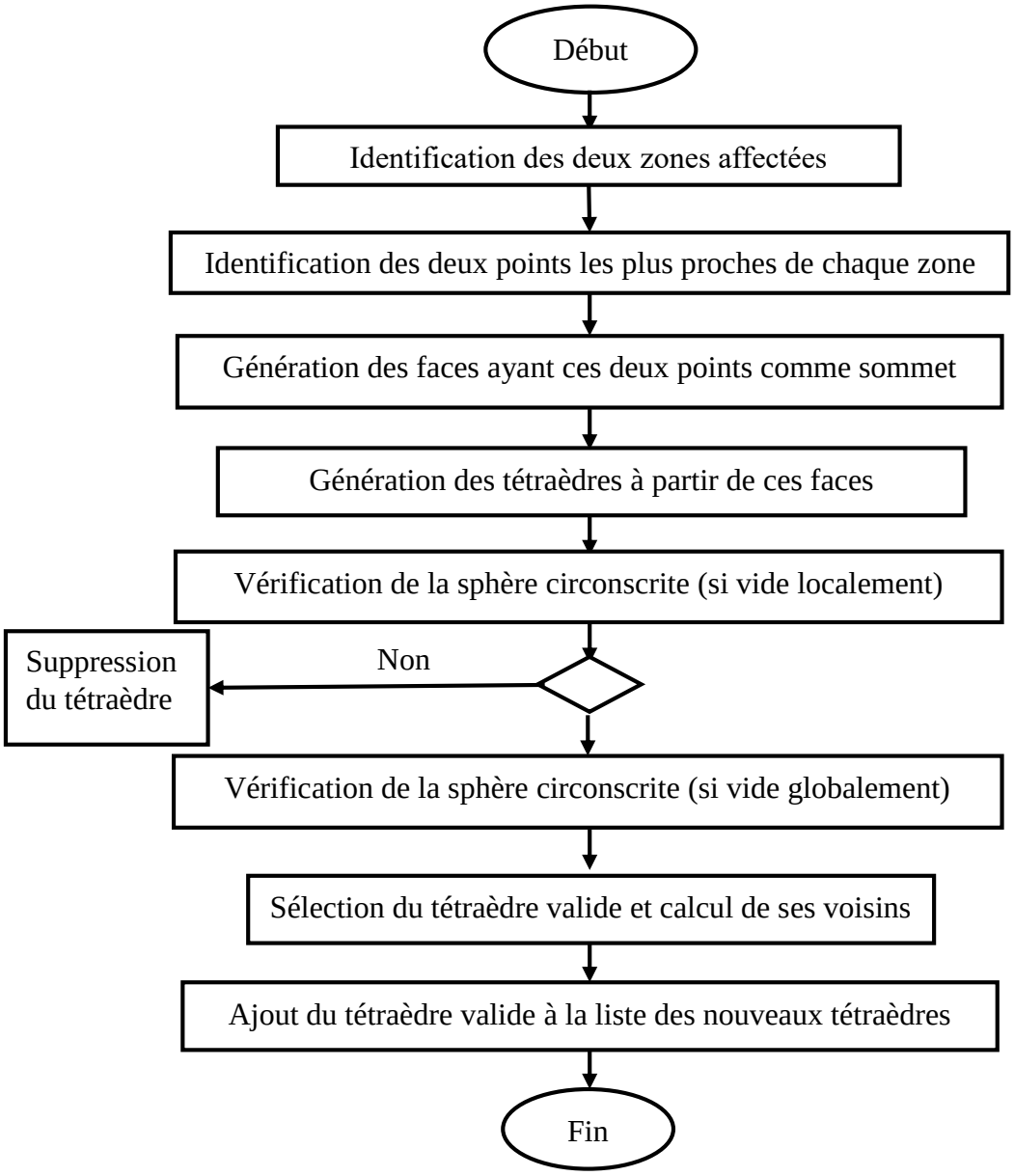


Figure 14 : Organigramme de création du premier tétraèdre

4.3.1.2 Création du tétraèdre générique

La création du tétraèdre générique est basée sur le test de la sphère circonscrite en deux étapes :

- **Test local de la sphère** : il consiste à tester la sphère avec les points de la zone affectée gauche si la face se trouve dans la zone affectée gauche et vice-versa.
- **Test global de la sphère** : ce test consiste à tester l'ensemble des points des deux zones affectées à fusionner.

Le processus de création du tétraèdre générique est illustré par les deux Figure 15 et Figure 16.

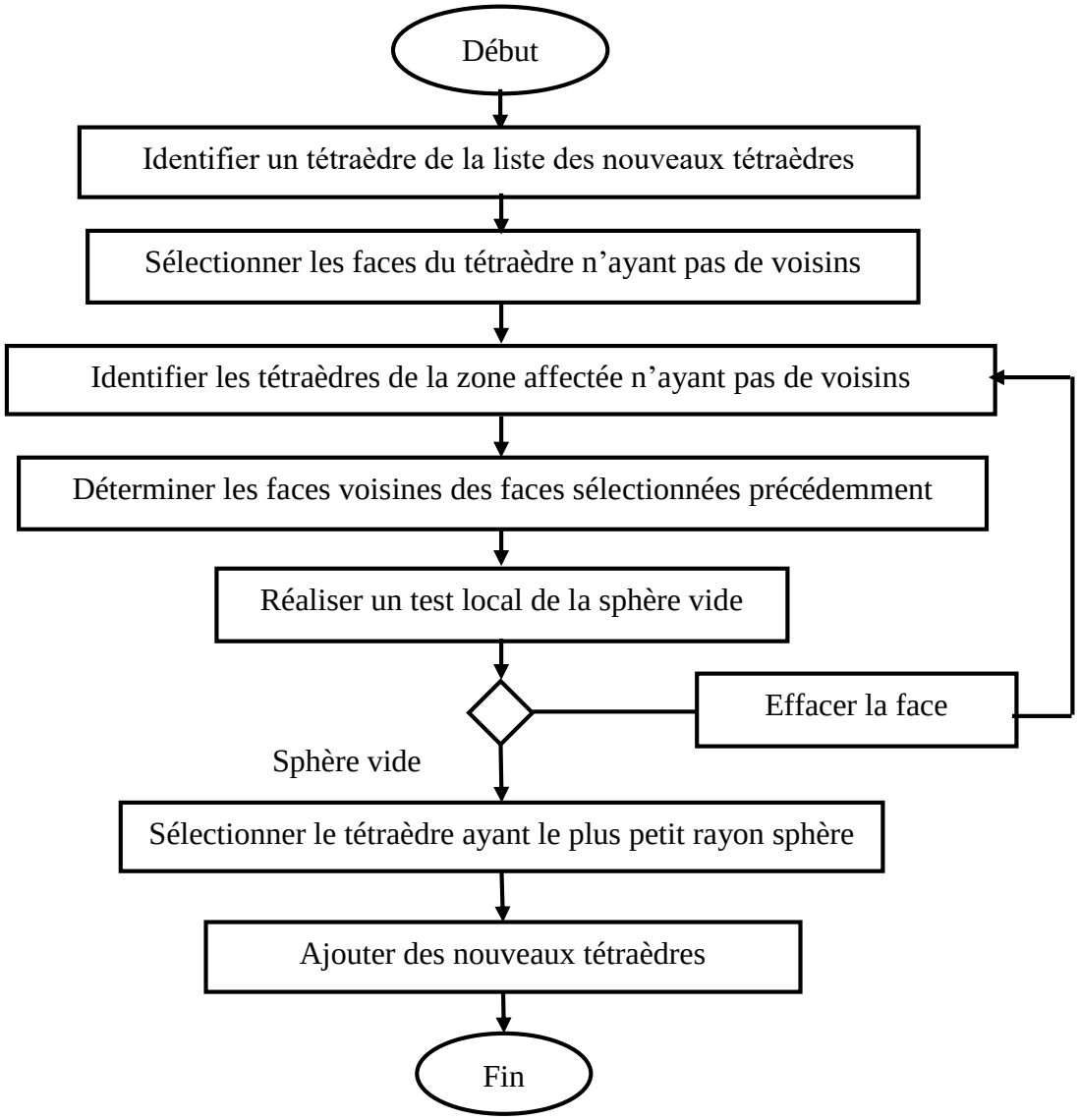


Figure 15 : Organigramme de création d'un tétraèdre générique

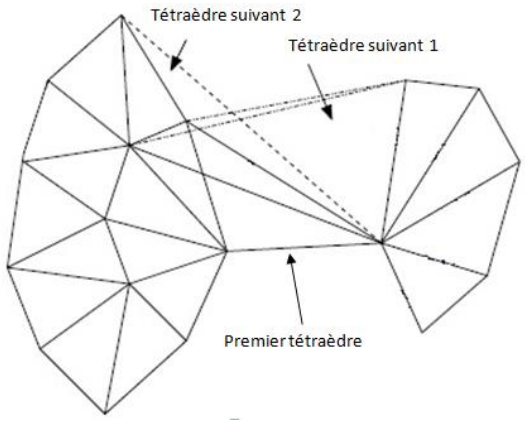


Figure 16 : Création du tétraèdre générique

La création du tétraèdre dit *« tétraèdre générique »* peut paraître simple à réaliser. En réalité, elle demande beaucoup de réflexion et impose le respect de plusieurs conditions, notamment lors des mises à jour après la suppression des faces (triangles) ou bien des arêtes. La création du tétraèdre

générique se fait en se basant sur le premier tétraèdre ce qui permet de recenser deux sortes de tétraèdres :

- **Premier cas :** il est décrit par la Figure 17. Ce cas du tétraèdre générique, consiste à réaliser le test de la sphère vide entre une des faces du premier tétraèdre sans voisins et les arêtes qui sont reliées à l'un de ses sommets. Si la sphère est vide, le tétraèdre est créé. Dans le cas contraire, l'arête est supprimée. Il est à noter que la suppression d'une arête peut engendrer la suppression d'un ou plusieurs tétraèdres de la zone affectée, ce qui rend la mise à jour de la zone affectée qui respecte les critères de Delaunay très délicate et difficile à réaliser[115].

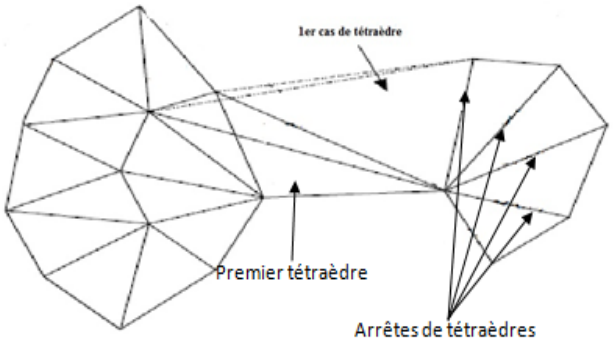


Figure 17 : Premier cas du tétraèdre générique

- **Deuxième cas :** le deuxième cas est décrit par la Figure 18. Dans le cas d'une existence simultanée de deux tétraèdres génériques appartenant aux deux côtés différents des deux zones affectées adjacentes et qui vérifient les conditions citées précédemment, le tétraèdre ayant la sphère avec le plus petit rayon est sélectionné. Ce dernier devient la base de création d'un autre tétraèdre générique.

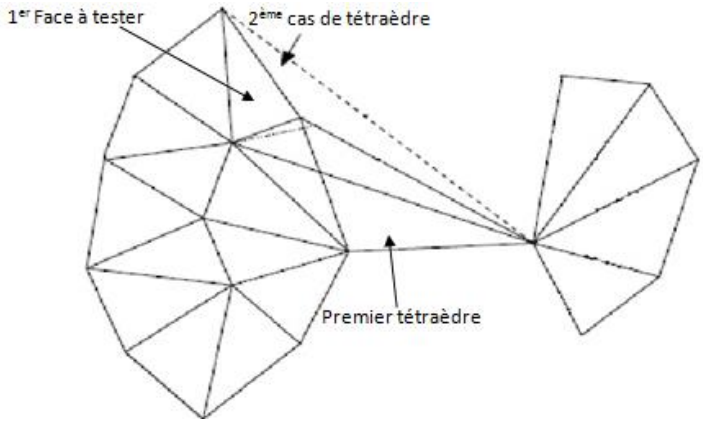


Figure 18 : Deuxième cas du tétraèdre générique

4.3.2 Deuxième procédure de fusion

Pour concaténer deux sous triangulations, l'algorithme 2 est mis en œuvre. Il commence par introduire les deux sous-triangulations et leurs zones affectées définies précédemment. Ensuite, un petit sous-ensemble est formé à partir des deux zones affectées ; il inclut tous les points définissant les tétraèdres sans répétition. Après, ce sous-ensemble est triangulé (triangulation de sous-ensembles) en utilisant l'approche de « Destruction et construction » [116]. Les tétraèdres finalement considérés sont (*i*) issus des deux sous-triangulations englobant tous les tétraèdres qui n'appartiennent pas aux zones affectées, (*ii*) issus de la triangulation du sous-ensemble incluant tous les tétraèdres dont les sommets proviennent des deux partitions, et tous les tétraèdres qui remplacent un tétraèdre précédemment trouvé dans les zones affectées. Cette procédure est décrite par l'algorithme de la Figure 19.

Algorithme 2 : Fusion de deux sous-triangulations
début
Introduire les sous-triangulations de deux nœuds T1 et T2 ;
Introduire les zones affectées définies de deux nœuds liste AZ1 et liste AZ2 ;
Créer un petit sous-ensemble B qui contient sans répétition
- Tous les points définissant les tétraèdres de la liste AZ1 ;
- Tous les points définissant les tétraèdres de la liste AZ2 ;

Trianguler le petit sous-ensemble B en T(B) ;
Trouver le tétraèdre final *Final_T* à partir de T1, T2 et T(B) ;
pour chaque tétraèdre de T1
si ce tétraèdre n'appartient pas à la liste AZ1
Ajouter ce tétraèdre à *Final_T* ;
sinon passer au tétraèdre suivant ;
fin pour
pour chaque tétraèdre de T2
si ce tétraèdre n'appartient pas à la liste AZ2
Ajouter ce tétraèdre à *Final_T* ;
sinon passer au tétraèdre suivant ;
fin pour
pour chaque tétraèdre de T(B)
si les points proviennent des deux partitions
Ajouter les points à *Final_T* ;
si les points sont dans une partition (T1 et T2) et le tétraèdre remplace un tétraèdre précédemment trouvé de la liste AZ1 ou de la liste AZ2
Ajouter les points à *Final_T* ;
fin pour
fin

Figure 19 : Algorithme de fusion de deux sous-triangulations

5. Validation et discussion des résultats obtenus

La méthodologie OFA proposée est mise en œuvre en C++Builder et la bibliothèque graphique OpenGL fonctionnant sous Windows 7. Les tests de validation sont effectués sur un PC Intel Core i3 avec 6 Go de RAM.

5.1 Validation de la méthodologie développée

L'efficacité et les performances de la méthodologie OFA sont montrées sur le « *Modèle de dent* » donné par la Figure 20 avec une résolution de 1366×768 pixels.

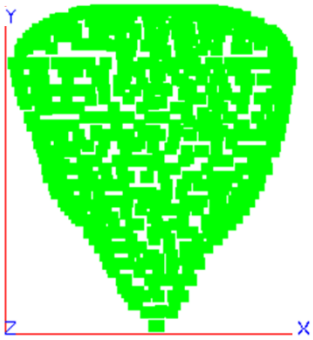
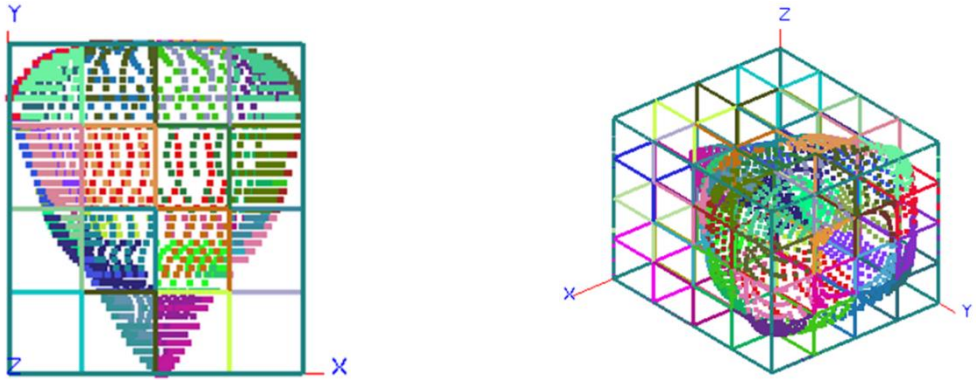


Figure 20 : Nuage de points pour le « *Modèle de dent* »

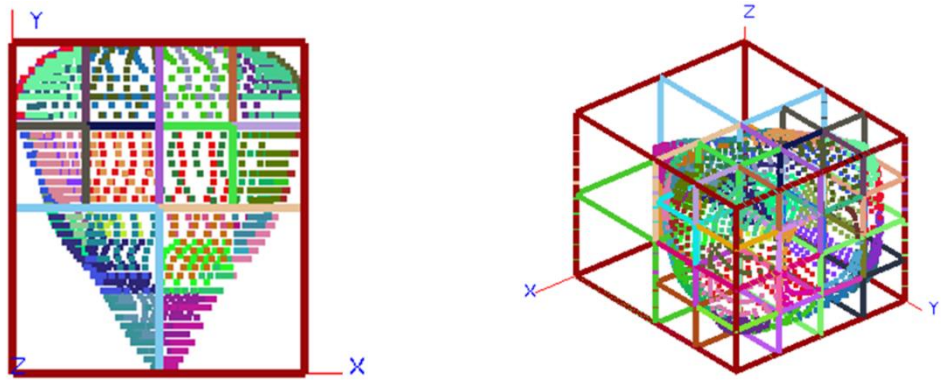
Le « *Modèle de dent* » de validation contient 4300 points ; le nombre de boxes dans chaque direction X, Y et Z est égal à 2^2. La Figure 21 montre les boxes créés.



a. Visualisation plan XY b. Visualisation tridimensionnelle

Figure 41 : Création des boxes

Le nombre maximum de points dans chaque nœud de l'octree est égal à 500 points. Ainsi, un octree est créé (Figure 22) contenant 40 nœuds, dont 36 nœuds sans fils.



a. Visualization plan XY b. Visualization 3D

Figure 22 : Création de l'octree du « *Modèle de dent* »

Les nœuds sans fils sont triangulés en parallèle en utilisant l'approche « *Destruction et construction* » [116]. Par conséquent, seuls 36 sont triangulés en parallèle. La figure 23 montre les sous-triangulations générées.

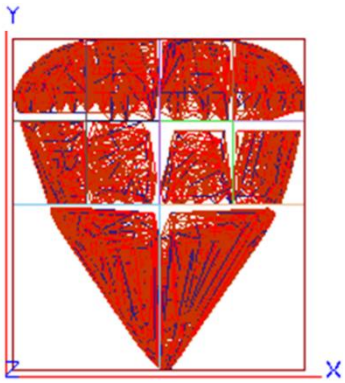


Figure 23 : Génération de sous-triangulations du « *Modèle de dent* »

Comme la triangulation est effectuée pour l'ensemble des points, les sous-triangulations sont fusionnées deux par deux jusqu'à trouver la triangulation finale. Pour fusionner ces nœuds le long des trois directions, l'algorithme 2 donne cinq niveaux de parallélisme :

- Le premier niveau le long de l'axe X : 18 paires sont fusionnées en parallèle,
- Le deuxième niveau le long de l'axe Y : 08 paires sont fusionnées en parallèle,
- Le troisième niveau le long de l'axe Z : 04 paires sont fusionnées en parallèle,
- La fusion des nœuds continue avec les résultats de la fusion précédente au quatrième niveau en fusionnant 2 paires le long de l'axe X en parallèle,
- Le cinquième niveau consiste à fusionner 2 paires supplémentaires le long de l'axe Y en parallèle.

Le tableau 1 illustre les cinq niveaux de parallélisme, la direction de la fusion et les nœuds concernés.

Niveau	**Niveau 1**	**Niveau 2**	**Niveau 3**	**Niveau 4**	**Niveau 5**
Direction	axe X	axe Y	axe Z	axe X	axe Y
Nœuds examinés	18 paires	8 paires	4 paires	2 paires	2 paires

Tableau 1 : Niveaux de parallélisme pour l'architecture *OFA*

Ces sous-triangulations sont fusionnées. La triangulation obtenue est très proche du modèle théorique (Figure 24).

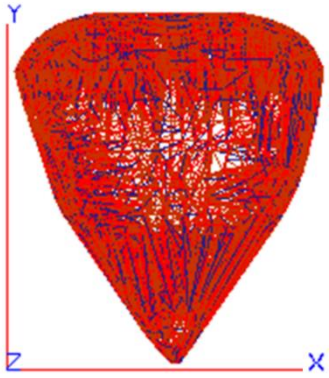


Figure 24 : Triangulation globale de Delaunay obtenue pour « *Modèle de dent* »

5.2 Étude comparative

Pour évaluer les performances de la méthodologie proposée, une étude comparative est réalisée entre l'approche proposée OFA et une autre approche, fusion des nœuds dans l'ordre inverse de leurs partitions, Ordre Inverse de Partitionnement (OIP), proposée dans [118]. Les tableaux 1 et 2 illustrent les niveaux des nœuds examinés et les directions de fusion pour les deux architectures, respectivement.

Selon le tableau 1, l'approche OFA donne cinq niveaux de parallélisme alors que le tableau 2 montre que l'approche OIP donne six niveaux de parallélisme.

Niveau	**Niveau 1**	**Niveau 2**	**Niveau 3**	**Niveau 4**	**Niveau 5**	**Niveau 6**
Directions	axe X	axe Y	axe Z	axe X	axe Y	axe Z
Nœuds examinés	16 paires	8 paires	4 paires	4 paires	2 paires	1 paire

Tableau 2 : Niveaux de parallélisme pour l'architecture OIP

Il faut mentionner que chaque direction représente un niveau de parallélisme. D'après les deux tableaux (1 et 2), le nombre de niveaux de parallélisme de l'architecture OFA est inférieur à celui de l'architecture OIP. Ainsi, une direction supplémentaire le long de l'axe Z est nécessaire pour l'architecture OIP afin d'obtenir la triangulation finale. En outre, concernant le nombre de paires par niveau, le nombre total de la nouvelle architecture OFA est inférieur à celui de l'architecture OIP.

L'approche OFA a réduit le nombre de directions ; par conséquent, elle a diminué le nombre de niveaux de parallélisme. De plus, l'architecture OFA a minimisé le nombre de paires ; elle a donc diminué les étapes de fusion des paires. Ces deux conséquences ont un effet positif sur le temps de calcul. On peut donc en déduire que la nouvelle architecture OFA est capable de fournir de meilleures solutions. Cette observation est d'autant plus visible avec une structure arborescente profonde et la disponibilité d'ordinateurs multi-cœurs pour le calcul parallèle.

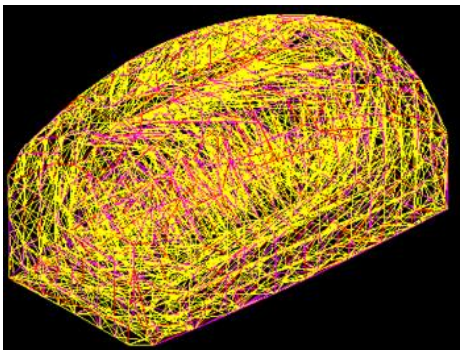
Une deuxième comparaison est réalisée concernant les temps de calcul de la triangulation de Delaunay 3D. Les résultats obtenus par l'approche OFA sont comparés à trois approches développées dans le chapitre 3 :

- Approche de Destruction et construction (ADC) [116],
- Approche de FLIP (AF) [117],
- Approche de FLIP Amélioré (AFA) [118].

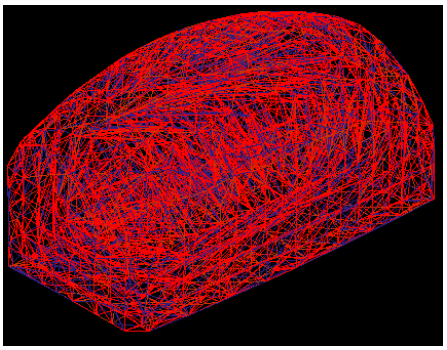
À cette fin, les résultats obtenus par les trois approches précédentes (ADC, AF, AFA) ont été comparés à ceux de OFA en utilisant le modèle « *pièce convexe* » (Figure 25), avec différents échantillons et densités. Le tableau 3 illustre les temps de calcul pour générer une triangulation de Delaunay 3D pour tous les algorithmes. Il montre clairement que le temps de calcul total de l'approche OFA (utilisant le calcul parallèle) est meilleur que celui des deux approches séquentielles (ADC, AF et AFA).

Densité / **Méthodologie**	**300**	**400**	**700**	**900**	**1500**	**3200**
OFA (s)	0.8	1.2	3.7	5.3	17	102
AF (s)	46	90	217	367	947	4440
ADC (s)	35	70	169	302	821	3861
AFA (s)	10	30	80	100	300	800

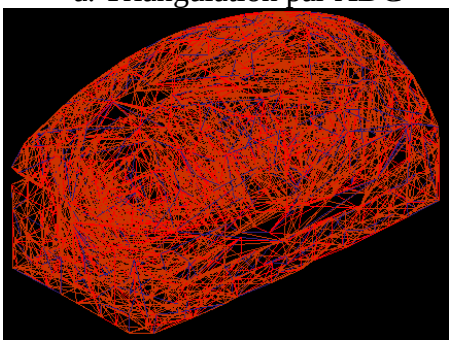
Tableau 3 : Temps de calcul des différentes approches *OFA*, *ADC*, *AF* et *AFA*



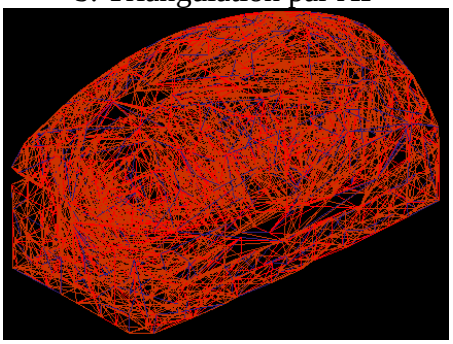
a. Triangulation par ADC



b. Triangulation par AF



c. Triangulation par AFA



d. Triangulation par OFA

Figure 25 : Triangulation de Delaunay de différentes approches *OFA*, *ADC*, *AF* et *AFA*

Les deux comparaisons révèlent que les triangulations générées pour toutes les approches (ADC, AF et OFA) sont très proches du modèle théorique avec un nuage de points plus dense. Cependant, les méthodes ADC et AF ont nécessité un temps de traitement plus important pour la triangulation de Delaunay. En revanche, la méthode OFA proposée a considérablement réduit les temps de calcul avec des points plus denses.

En résumé, l'étude comparative a démontré que les triangulations de Delaunay séquentielles donnent de bonnes formes avec des temps de calcul plus élevés. De plus, leurs triangulations de Delaunay sont proches du modèle théorique lorsque les nuages de points sont denses (où le calcul parallèle est recommandé). En outre, la comparaison des résultats obtenus avec l'architecture *OIP* a prouvé la supériorité de l'architecture OFA proposée en termes de nombre de niveaux et de nombre de nœuds de paires ; en particulier avec une structure arborescente profonde de l'octree (plusieurs niveaux de l'arbre) et l'utilisation d'ordinateurs multi-cœurs.

6. Conclusion

Ce chapitre a présenté la génération parallèle de triangulation de Delaunay 3D pour des nuages de points non-structurés. Son principal objectif consiste à proposer une nouvelle architecture de fusion des sous-triangulations. La méthode *Ordre de Fusion Amélioré* (*OFA*) utilise l'octree pour partitionner l'ensemble des points en sous-ensembles (nœuds) de densité quasi-égale. Pour fusionner les sous-triangulations obtenues, une nouvelle architecture de fusion est appliquée le long des axes X, Y et Z. L'approche développée a été testée sur deux modèles, « *Modèle de dent* » et « *Pièce convexe* ». La comparaison des résultats obtenus avec l'architecture de fusion par la méthode *Ordre Inverse de Partitionnement* (*OIP*), l'approche de *Destruction et Construction* (ADC), *Approche de Flip* (*FA*) et *Approche de Flip Amélioré* (*AFA*) ont démontré la supériorité de la méthodologie proposée en termes de (*i*) nombre de niveaux, (*ii*) nombre de nœuds de paires et surtout (*iii*) temps de calcul. En particulier, cette supériorité est confirmée lorsqu'il s'agit de nuages de points non-structurés, de l'utilisation d'une structure arborescente profonde de l'octree et de l'utilisation d'ordinateurs multi-cœurs.

Chapitre 6
Conclusions générales et perspectives

Conclusions générales et perspectives

La reconstruction d'objets est une discipline relativement jeune qui concerne un large éventail de domaines liés au génie mécanique, à la vision artificielle et à la robotique, ainsi qu'au génie médical, au génie civil et à l'environnement.

La réalisation des formes gauches de complexités diverses telles que les moules des ailes d'avions et des véhicules est basée sur la « *Conception et Fabrication Assistées par Ordinateurs* (*CFAO*) ». La conception de ces formes est réalisée principalement par deux classes d'approches : (*i*) approches classiques et (*ii*) approches de l'ingénierie inverse. Le temps de traitement très important et la modification successive de la forme sont parmi les inconvénients de la première classe. Cependant, l'ingénierie inverse est avantageuse par son temps réduit de développement de produits.

L'ingénierie inverse suit plusieurs étapes : numérisation de l'objet, recalage du nuage de points digitalisé, décimation, et enfin segmentation du nuage de points et approximation de surface. Il y a lieu de préciser que les deux dernières étapes présentent plusieurs difficultés de traitement (détermination de l'équation mathématique, utilisation du modèle B-Spline, etc.). Dans le but de rendre la conception des modèles plus souple et moins couteuse, *l'approche de Facettisation* a été utilisée.

Le travail présenté dans le cadre de cette thèse s'est concentré sur l'approche de *Facettisation*, où les points du nuage sont reliés par des formes géométriques simples (tétraèdres). La triangulation de Delaunay est l'une des techniques de reconstructions les plus utilisées. Elle est générée par l'algorithme « *Insertion incrémentale* » et « *Diviser pour régner* ».

L'objectif principal de cette thèse est de développer une nouvelle méthodologie de reconstruction de modèle d'objet 3D à partir d'un nuage de points non-uniforme en un délai de traitement réduit. Cet objectif a été divisé en plusieurs sous objectifs. Le premier but concerne la génération de la triangulation de Delaunay tridimensionnelle en utilisant pour la première fois l'approche de « *Destruction et construction* ». Cette approche a été mise en œuvre, testée et validée sur divers exemples de pièces. Les résultats obtenus prouvent que les modèles générés sont très proches des modèles réels.

L'approche *FLIP* (*AF*) a été développée pour générer une meilleure triangulation de Delaunay. Cette approche a été testée et validée sur plusieurs exemples de pièces. Les résultats sont satisfaisants malgré le temps de calcul très élevé. Ceci est lié principalement au temps nécessaire pour la recherche du tétraèdre contenant le point à insérer lors de la phase de subdivision et de la mise à jour des voisions lors de la phase de modification de la triangulation de Delaunay. Pour pallier à ces inconvénients, l'approche *FLIP Amélioré* (*AFA*) a été proposée en incluant deux concepts : (*i*) recherche rapide du point à insérer et (*ii*) mise à jour améliorée des voisins. Une accélération du processus de la triangulation de Delaunay tridimensionnelle est observée dans cette nouvelle approche.

Afin de se rapprocher de la forme réelle de l'objet, la reconstruction des frontières de l'objet est réalisée en utilisant l'algorithme « *Alpha Shape* ». Cet algorithme a été développé et testé puis validé sur plusieurs exemples de pièces. Les résultats sont satisfaisants bien que plusieurs tests ont été effectués pour retenir la valeur de « Alpha » qui donne la forme la plus proche.

Ultérieurement, le paradigme « *Diviser pour régner* » a été exploité. Ainsi, une méthodologie de partitionnement spatiale a été proposée. L'objectif principal consiste à utiliser les deux algorithmes K-means basique (BK) et son dérivé K-means global rapide (FGK). En effet, cet algorithme a été appliqué, pour la première fois, pour subdiviser les nuages de points en sous-nuages. La méthodologie développée a été testée sur plusieurs modèles de pièces. En outre, pour évaluer la méthodologie proposée, une étude comparative de quatre méthodes de partitionnement (octree, cellules, BK et FGK) a été établie en termes de nombre de points par groupe. Les résultats obtenus ont démontré les hautes performances de la méthode FGK en ce qui concerne l'homogénéité des partitions, et la charge de travail équilibrée sur les différents processeurs.

Par la suite, les sous-triangulations sont fusionnées selon une nouvelle architecture de fusion tridimensionnelle, *Ordre de Fusion Amélioré* (*OFA*). L'approche développée a été testée sur plusieurs

modèles de pièces. La comparaison des résultats obtenus avec l'architecture de fusion par *Ordre Inverse de Partitionnement* (*OIP*), *Approche de Construction et Destruction* (*ACD*), *Approche de Flip* (*AF*) et *Approche Flip Amélioré* (*AFA*) ont démontré la supériorité de la méthodologie proposée en termes de temps de calcul, en particulier lorsqu'il s'agit de nuages de points non-structurés et de l'utilisation d'ordinateurs multi-cœurs.

En termes de perspectives, nos travaux futurs concerneront principalement l'amélioration de l'approche FLIP en exploitant d'autres solutions liées à la localisation du point à insérer et la mise à jour de la triangulation après une opération de FLIP.

Dans le but de déterminer les frontières de l'objet, il sera envisagé de déterminer la valeur de « Alpha » directement sans avoir recours à plusieurs tests et utiliser d'autres méthodes de reconstruction des frontières de l'objet afin de réaliser une étude comparative, par la suite, entre les différentes méthodes.

Enfin, afin de mieux exploiter le domaine du parallélisme, le parallélisme du GPU sera étudié et d'autres modes de partitionnement seront examinés tels que les multi-cellules tridimensionnelles, les méthodes de regroupements, etc.

Références

[1] V. Raja et K. J. Fernandes, Reverse Engineering, an industrial perspective, Edition Springer, 2008.

[2] R. Ben Jemaa, Traitement de données 3D denses acquises sur des objets réels complexes, Thèse de Doctorat, 1998, Ecole Nationale Supérieure des Télécommunications, Paris, France.

[3] Cyril Brèque, Fabrice Brémand, Modélisation de forme 3D par méthode de moiré de projection et analyse par décalage de phases, Congrès Francophone de Vision par Ordinateur, ORASIS, 5-8 Juin 2001, Toulouse, France.

[4] T. Varady, R. R. Martin, J. Cox, Creating Geometric Models in Reverse Engineering, Fourth SIAM Conference on Geometric Design, 6-9 Novembre 1995, USA.

[5] Yan Zhou, Surface Interpolation and Approximation, Master Thesis, Michigan Technological University, Septembre 1998.

[6] Delaunay, Boris, Sur la sphère vide. Bulletin de l'Académie des Sciences de l'URSS, Classe des Sciences Mathématiques et Naturelles, 1934, 6: 793–800.

[7] S. P. Lim and H. Haron, "Surface reconstruction techniques: A review," Artif. Intell. Rev., vol. 42, no. 1, pp. 59–78, 2014.

[8] S. Boudjouad, N. Tafat-Bouzid, « Méthodes des plans parallèles pour l'usinage des surfaces gauches sur des fraiseuses à commande numérique à 3 axes », PFE, Université USTHB, 2004.

[9] J. C. Léon. « Modélisation et Conception de Surfaces pour la CFAO ». Edition Hermès, Paris, 1991.

[10] Wen PZ, Wu XJ, Zhu Y, Peng XW (2009) LS-RBF network based 3D surface reconstruction method. 2009 Chinese control and decision conference (CCDC 2009). pp 5785–5789

[11] C. K. Schene « Introduction to Computing with Geometry Notes ». Department of Computer Science ; Michigan Technological University, 2003.

[12] A. Saeedfar and K. Barkeshli, "Shape reconstruction of three-dimensional conducting curved plates using physical optics, NURBS modeling, and genetic algorithm," IEEE Trans. Antennas Propag., vol. 54, no. 9, pp. 2497–2507, 2006.

[13] Y. C. Tsai, C. Y. Huang, K. Y. Lin, J. Y. Lai, and W. Der Ueng, "Development of automatic surface reconstruction technique in reverse engineering," Int. J. Adv. Manuf. Technol., vol. 42, no. 1–2, pp. 152–167, 2009.

[14] N. Amenta, S. Choi, and R. K. Kolluri, "The power crust, unions of balls, and the medial axis transform," Comput. Geom. Theory Appl., vol. 19, no. 2–3, pp. 127–153, 2001.

[15] H. K. Zhao, S. Osher, and R. Fedkiw, "Fast surface reconstruction using the level set method," Proc. - IEEE Work. Var. Lev. Set Methods Comput. Vision, VLSM 2001, pp. 194–199, 2001.

[16] F. BOUDJEMAÏ, "Reconstruction De Surfaces D'Objets 3D a Partir De Nuages De Points Par Reseaux De Neurones 3D-Som," p. 116, 2006.

[17] R. Tang, S. Halim, and M. Zulkepli, "Surface Reconstruction Algorithms: Review and Comparison," no. 1984, p. 22, 2013.

[18] R. Goldenthal and M. Bercovier, "Design of Curves and Surfaces Using Multi-Objective Optimization," no. May, pp. 1–12, 2004.

[19] A. Gálvez, A. Iglesias, A. Cobo, J. Puig-Pey, and J. Espinola, "Bézier curve and surface fitting of 3D point clouds through genetic algorithms, functional networks and least-squares approximation," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 4706 LNCS, no. PART 2, pp. 680–693, 2007.

[20] M. Bolitho, M. Kazhdan, R. Burns, and H. Hoppe, "Parallel poisson surface reconstruction," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 5875 LNCS, no. PART 1, pp. 678–689, 2009.

[21] D. Müller, "Techniques informatiques efficaces pour la simulation de milieux granulaires par des méthodes d'éléments distincts," These EPFL, vol. 1545, 1996.

[22] X. Li, W. Wan, X. Cheng, and B. Cui, "An improved poisson surface reconstruction algorithm," ICALIP 2010 - 2010 Int. Conf. Audio, Lang. Image Process. Proc., pp. 1134–1138, 2010.

[23] E. Bae and J. Weickert, "Partial Differential Equations for Interpolation and," Mmcs 2008, vol. LNCS 5862, pp. 1–14, 2010.

[24] S. Osechinskiy and F. Kruggel, "PDE-based reconstruction of the cerebral cortex from MR images," 2010 Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. EMBC'10, pp. 4278–4283, 2010.

[25] H. Liu, X. Wang, and W. Qiang, "Implicit surface reconstruction from 3D scattered points based on variational level set method," 2008 2nd Int. Symp. Syst. Control Aerosp. Astronaut. ISSCAA 2008, no. January, 2008.

[26] Luger GF (2005) Artificial intelligence structures and strategies for complex problem solving fifth edition. Pearson Education Limited, London

[27] Chandrasekaran M, Muralidhar M, Murali Krishna C, Dixit US (2009) Application of soft computing tech- niques inmachining performance prediction and optimization: a literature review. Int JAdv ManufTechol 46:445–464

[28] Jaganathan B, Venkatesh S, Bhardwaj Y, Prakash CA (2011) Kohonen's self organizing map method of estimation of optimal parameters of a permanent magnet synchronous motor drive. India international conference on power electronics (IICPE). pp 1–6

[29] Yu Y (1999) Surface reconstruction from unorganized points using self-organizing neural networks. In: Pro- ceedings of the IEEE Visualization 99, Conference. pp 61–64

[30] Barhak J, Fischer A (2001) Parameterization and reconstruction from 3D scattered points based on neural network and PDE techniques. IEEE Trans Vis Comput Graph 7(1):1–16

[31] Wu HX, Dong HX, Su JQ (2008a) 3D reconstruction from section plane views based on self-adaptive neural network. Second international symposium on intelligent information technology application. pp 84–88

[32] Wu XM, Li GX, Zhao WM (2008b) Incomplete points cloud data surface reconstruction based on neural network. International conference on intelligent information hiding and multimedia signal processing. pp 913–316

[33] Wu XM, Li GX, Zhao WM (2008c) Sparseness points cloud data surface reconstruction based on radial basis function neural network (RBFNN) and simulated annealing arithmetic. 2007 International conference on computational intelligence and security workshops. pp 877–880.

[34] Wen PZ, Wu XJ, Zhu Y, Peng XW (2009) LS-RBF network based 3D surface reconstruction method. 2009 Chinese control and decision conference (CCDC 2009). pp 5785–5789

[35] Saeedfar A, Barkeshli K (2006) Shape reconstruction of three-dimensional conducting curved plates using physical optics, NURBS modeling, and genetic algorithm. IEEE Trans Antennas Propag 54(9):2497– 2507

[36] Bokhabrine Y, Fougerolle YD, Foufou S, Truchetet F (2007) Genetic algorithms for gielis surface recovery from 3D data sets. IEEE international conference on image processing, 2007. ICIP 2007, pp 549–552

[37] Wang S, Dhawan AP (2007) Shape-based reconstruction of skin lesion for multispectral nevoscope using genetic algorithm optimization. 4th IEEE international symposium on biomedical imaging: from nano to macro, 2007. ISBI 2007. pp 488–491.

[38] Bokhabrine Y, Fougerolle YD, Foufou S, Truchetet F (2007) Genetic algorithms for gielis surface recovery from 3D data sets. IEEE international conference on image processing, 2007. ICIP 2007, pp 549–552.

[39] Wu HX, Dong HX, Su JQ (2008a) 3D reconstruction from section plane views based on self-adaptive neural network. Second international symposium on intelligent information technology application. pp 84–88

[40] Wu XM, Li GX, Zhao WM (2008b) Incomplete points cloud data surface reconstruction based on neural network. International conference on intelligent information hiding and multimedia signal processing. pp 913–316

[41] Wu XM, Li GX, Zhao WM (2008c) Sparseness points cloud data surface reconstruction based on radial basis function neural network (RBFNN) and simulated annealing arithmetic. 2007 International conference on computational intelligence and security workshops. pp 877–880.

[42] Kennedy J, Eberhart R (1995) Particle swarm optimization. IEEE international conference on neural networks (ICNN'95), Perth, Australia. pp 1942–1948.

[43] 43Forkan F, Shamsuddin SM (2008) Kohonen-Swarm algorithm for unstructured data in surface reconstruction. Fifth international conference on computer graphics, imaging and visualization. pp 5–11

[44] 44ArdiaD, Boudt K, Carl P, MullenKM,PetersonBG (2011) Differential evolutionwith DEoptim an application to non-convex portfolio optimization. The R Journal 3/1:27–34.

[45] Weise T (2009) Global optimization algorithms—theory and application 2nd edn. [Online].Available at: http:// www.it-weise.de [Accessed 19 Aug 2011].

[46] Dehmollaian M (2011) Through-wall shape reconstruction and wall parameters estimation using differential evolution. IEEE Geosci Remote Sens Lett 8(2):201–205.

[47] Studholme C (2001) Deriving camera and point location from a series of photos using numerical optimiza- tion. [Online]. Available at http://www.cs.toronto.edu/~cvs/geometry/GeometryProject.pdf [Accessed 19 Aug 2011].

[48] Rekanos IT (2008) Shape reconstruction of a perfectly conducting scatterer using differential evolution and PSO. IEEE Trans Geosci Remote Sens 46(7):1967–1974.

[49] R. L. M. E. Do Rêgo and A. F. R. Araújo, "A surface reconstruction method based on self-organizing maps and intrinsic Delaunay triangulation," Proc. Int. Jt. Conf. Neural Networks, 2010.

[50] B. Gärtner and M. Hoffmann, "Chapter 6 Delaunay Triangulations," Comput. Geom. Lect. Notes, no. c, pp. 55–71, 2012.

[51] E. Strandell, "Computational Geometry and Surface Reconstruction from Unorganized Point Clouds," 2007.

[52] L. Papaleo, "Surface reconstruction : online mosaicing and modeling with uncertainty Dipartimento di Informatica e Scienze dell ' Informazione

Surface Reconstruction : Online Mosaicing and Modeling with Uncertainty by Laura Papaleo Theses Series DISI-TH-2004-XX," no. March, 2015.

[53] H. Ledoux, "Modelling Three-dimensional Fields in Geoscience with the Voronoi Diagram and its Dual," Sch. Comput., no. November, p. 188 pp., 2006.

[54] Céline ROUDET, Contribution de la reconstruction 3D à la compression de maillages surfaciques triangulaires, Rapport de stage – Master M2 Recherche Informatique, Université Claude Bernard Lyon 1,p. 13-14.

[55] Hermeline, F. (1982). Triangulation automatique d'un polyèdre en dimension $ N$. RAIRO-Analyse numérique, 16(3), 211-242.

[56] H. Edelsbrunner, "Alpha Shapes - a Survey," Tessellations Sci., pp. 1–25, 2010.

[57] H. Edelsbrunner and E. P. Mucke, "Three-dimensional Alpha Shapes 1 1 Introduction," ACM Trans. Graph., vol. 13, no. 1, pp. 43–72, 1994.

[58] N. Amenta, M. Bern, and M. Kamvysselis, "A new voronoi-based surface reconstruction algorithm," Proc. 25th Annu. Conf. Comput. Graph. Interact. Tech. SIGGRAPH 1998, pp. 415–422, 1998.

[59] T. K. Dey and S. Goswami, "Tight Cocone: A Water-tight Surface Reconstructor," J. Comput. Inf. Sci. Eng., vol. 3, no. 4, pp. 302–307, 2003.

[60] N. Amenta, S. Choi, R.K. Kolluri. The Power Crust. ACM Symposium on Solid and Physical Modeling archive Proceedings of the sixth ACM symposium on Solid modeling and applications. 2001.

[61] N. Amenta, S. Choi, R.K. Kolluri. The Power Crust, Unions of Balls, and the Medial Axis Transform. Computational Geometry, Volume 19, Number 2, July 2001 (pp. 127-153).

[62] Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W. (1992). Surface Reconstruction from Unorganized Point Clouds. Computer Graphics, 26(2), pp. 71-78.Proceeding of SIGGRAPH92

[63] Bittar E., Tsingos N., Gascuel M.-P. (1995). Automatic Reconstruction of unstructured 3D Data: Combining a Medial Axis and Implicit Surfaces. Computer Graphics Forum, 14(3), pp. 457-468.Proceedings of EUROGRAPHICS '95.

[64] Roth G., Wibowoo E. (1997). An Efficient Volumetric Method for Building Closed TriangularMeshes from 3-D Image and Point Data. Graphics Interface, pp. 173-180.

[65] C. Nguyen and P. J. Rhodes, "TIPP: Parallel delaunay triangulation for large-scale datasets∗," ACM Int. Conf. Proceeding Ser., 2018.

[66] A. Ben Makhlouf, B. Louhichi, M. A. Mahjoub, and G. Subsol, "Approach for CAD model reconstruction from a deformed mesh," Proc. IEEE/ACS Int. Conf. Comput. Syst. Appl. AICCSA, vol. 2017-Octob, no. October, pp. 327–333, 2018.

[67] G. Mei, J. Zhang, N. Xu, and K. Zhao, "A sample implementation for parallelizing Divide-and-Conquer algorithms on the GPU," Heliyon, vol. 4, no. 1, 2018.

[68] S. H. Lo, "3D Delaunay triangulation of 1 billion points on a PC," Finite Elem. Anal. Des., vol. 102–103, pp. 65–73, 2015.

[69] S. Fortune, Handbook of Discrete and Computational Geometry, Third Edition. Chapman and Hall/CRC, 2017.

[70] S. W. Cheng, T. K. Dey, and J. R. Shewchuk, "Delaunay mesh generation," Delaunay Mesh Generation. pp. 1–386, 2012

[71] P.L.George, H. Bourouchaki. Triangulation de Delaunay et maillage, application aux éléments finis. Editions Hermes, Paris 199

[72] WATSON, D. F. 1981. Computing the n-dimensional Delaunay tessellation with ap- plication to Voronoi polytopes. Computer Journal 24, 2, 167–172.

[73] J. D. Boissonnat and M. Teillaud, "On the randomized construction of the Delaunay tree," Theor. Comput. Sci., vol. 112, no. 2, pp. 339–354, 1993

[74] C. L. Lawson, "Transforming triangulations," Discrete Math., vol. 3, no. 4, pp. 365–372, 1972.

[75] MATULA, David W. et SOKAL, Robert R. Properties of Gabriel graphs relevant to geographic variation research and the clustering of points in the plane. Geographical analysis,

[76] L. U. Claude and B. Lyon, "L'université claude bernard – lyon 1," 2006.

[77] P. Alliez, "1 Iteration de Lloyd en 2D 3 Diagramme de Voronoi borné 2D," pp. 1–5.

[78] S. Linux, "Reconstruction & G . A . Crust - Courbes / Surfaces isovaleur," pp. 2011–2012.

[79] EDELSBRUNNER H., KIRKPATRICK D. G., SEIDEL R.: On the shape of a set of points in the plane. IEEE Trans. Inform. Theory IT.29 (1983), 551.559.

[80] EDELSBRUNNER, Herbert et MÜCKE, Ernst P. Three-dimensional alpha shapes. ACM Transactions on Graphics (TOG), 1994, vol. 13, no 1, p. 43-72.

[81] K. Fischer, "Introduction to Alpha Shapes," Dep. Inf. Comput. Sci. Fac. Sci. Utr. Univ., pp. 1–17, 2000.

[82] H. Edelsbrunner and P. M. Ernst, "Three-Dimensional α Shapes," vol. 13, no. September, 2005.

[83] M. I. Shamos and D. Hoey, "Closest-point problems," pp. 151–162.

[84] T. Boubekeur, P. Reuter, C. Schlick, L. Inria, and F. Universit, "Reconstruction locale et visualisation de nuages de points par surfaces de subdivision," 2008.

[85] C. Lemaire and J. M. Moreau, "A probabilistic result on multi-dimensional Delaunay triangulations, and its application to the 2D case," *Comput. Geom. Theory Appl.*, vol. 17, no. 1–2, pp. 69–96, 2000.

[86] S. Lee, C. I. Park, and C. M. Park, "An improved parallel algorithm for Delaunay triangulation on distributed memory parallel computers," *Parallel Process. Lett.*, vol. 11, no. 2–3, pp. 341–352, 2001.

[87] H. Wu, X. Guan, and J. Gong, "ParaStream: A parallel streaming Delaunay triangulation algorithm for LiDAR points on multicore architectures," *Comput. Geosci.*, vol. 37, no. 9, pp. 1355–1363, 2011.

[88] M. Bin Chen, T. R. Chuang, and J. J. Wu, "Parallel divide-and-conquer scheme for 2D Delaunay triangulation," *Concurr. Comput. Pract. Exp.*, vol. 18, no. 12, pp. 1595–1612, 2006.

[89] C. A. R. Hoare, "Algorithm 63: Partition," *Communications of the ACM*, vol. 4, no. 7. p. 321, 1961.

[90] S. H. Lo, "Parallel Delaunay triangulation-application to two dimensions," *Finite Elem. Anal. Des.*, vol. 62, pp. 37–48, 2012.

[91] S. H. Lo, "Parallel Delaunay triangulation in three dimensions," *Comput. Methods Appl. Mech. Eng.*, vol. 237–240, pp. 88–106, 2012.

[92] S. H. Lo, "3D Delaunay triangulation of 1 billion points on a PC," *Finite Elem. Anal. Des.*, vol. 102–103, pp. 65–73, 2015.

[93] S. H. Lo, "3D Delaunay triangulation of non-uniform point distributions," *Finite Elem. Anal. Des.*, vol. 90, pp. 113–130, 2014.

[94] Z. Tchantchane, K. Bouhadja, O. Azouaoui, N. Ghoualmi-Zine, "Enhanced unstructured points cloud subdivision applied for parallel Delaunay triangulation" Cluster Computing journal

[95] D. T. Larose, *An Introduction to Data Mining The CRISP-DM*. 1999.

[96] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognit.*, vol. 36, no. 2, pp. 451–461, 2003.

[97] M. James and others, "Some methods for classification and analysis of multivariate observations," *Proc. fifth Berkeley Symp. Math. Stat. Probab.*, vol. 1, no. 14, pp. 281–297, 1967.

[98] O. LEZORAY and C. CHARRIER, "Segmentation d'images couleur par coalescence non supervisée d'histogrammes 2D et fusion de régions selon la théorie de Dempster-Shafer," *TS. Trait. du signal*, vol. 21, no. 6, pp. 605–621, 2004.

[99] B. Mokhtari, K. E. Melkemi, and D. Michelucci, "Dynamic Clustering-Based Method for Shape Recognition and Dynamic Clustering-Based Method for Shape Recognition and," no. February 2015, 2014

[100] K. Bouhadja, M. Bey, K. Sebti, H. Moulay, and M. Bouaziz, "Volume modeling of complex mechanical parts via triple dexel," *Lect. Notes Eng. Comput. Sci.*, vol. 2224, no. December 2017, pp. 796–801, 2016

[101] K. Bouhadja, A. Boubekeur, and M. Bouaziz, "Complex Parts Partitioning by Elementary Geometric Objects Clustering in CAD/CAM Process," *J. Inst. Eng. Ser. C*, vol. 101, no. 2, pp. 229–240, 2020.

[102] W. Wu, Y. Rui, F. Su, L. Cheng, and J. Wang, "Novel parallel algorithm for constructing Delaunay triangulation based on a twofold-divide-and-conquer scheme," *GIScience Remote Sens.*, vol. 51, no. 5, pp. 537–554, 2014.

[103] J. Lin, R. Chen, L. Wu, Z. Shu, and C. Yang, "Adaptive parallel Delaunay triangulation construction with dynamic pruned binary tree model in Cloud," *Concurr. Comput. Pract. Exp.*, vol. 29, no. 24, 2017.

[104] T. Su, W. Wang, Z. Lv, W. Wu, and X. Li, "Rapid Delaunay triangulation for randomly distributed point cloud data using adaptive Hilbert curve," *Comput. Graph.*, vol. 54, pp. 65–74, 2016

[105] P. Cignoni, C. Montani, and R. Scopigno, "DeWall: A fast divide and conquer Delaunay triangulation algorithm in Ed," *CAD Comput. Aided Des.*, vol. 30, no. 5, pp. 333–341, 1998.

[106] M. Bin Chen, "A parallel 3D Delaunay triangulation method," *Proc. - 9th IEEE Int. Symp. Parallel Distrib. Process. with Appl. ISPA 2011*, pp. 52–56, 2011.
[107] M. Bin Chen, T. R. Chuang, and J. J. Wu, "A parallel divide-and-conquer scheme for Delaunay triangulation," *Proc. Int. Conf. Parallel Distrib. Syst. - ICPADS*, vol. 2002-Janua, pp. 571–576, 2002.
[108] M. Bin Chen, "The merge phase of parallel divide-and-conquer scheme for 3D delaunay triangulation," *Proc. - Int. Symp. Parallel Distrib. Process. with Appl. ISPA 2010*, pp. 224–230, 2010
[109] L. Guibas and J. Stolfi, "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi," *ACM Trans. Graph.*, vol. 4, no. 2, pp. 74–123, 1985.
[110] C. Nguyen and P. J. Rhodes, "TIPP: Parallel delaunay triangulation for large-scale datasets∗," *ACM Int. Conf. Proceeding Ser.*, 2018.
[111] C. Marot, J. Pellerin, J. Lambrechts, and J. Remacle, "Toward one billion tetrahedra per minute," *26Th*, vol. 00, no. September, p. 5, 2017.
[112] C. Marot, J. Pellerin, and J. F. Remacle, "One machine, one minute, three billion tetrahedra," *Int. J. Numer. Methods Eng.*, vol. 117, no. 9, pp. 967–990, 2019.
[113] J. Remacle, C. Marot, J. Remacle, and C. Marot, "Multi-threaded mesh generation To cite this version : HAL Id : hal-01908910," no. May 2017, 2018
[114] D. Funke and P. Sanders, "Parallel d-D delaunay triangulations in shared and distributed memory," *Proc. Work. Algorithm Eng. Exp.*, vol. 0, pp. 207–217, 2017.
[115] Z. Tchantchane, M. Bey, K. Azouaoui, and H. Bendifallah, "STL Model of Sculptured Surfaces from 3D Unstructured Cloud of Points," *11èmes Journées de Mécanique de l'EMP (JM'11–EMP)*, pp. 3–7, 2018.
[116] Z. Tchantchane, M. Bey, O. Azouaoui, and H. Bendifallah,, " Triangulation D'un Nuage De Points 3D Non Structure Par La Méthode De Flip," International Conference On Advanced Mechanics And Renewable Energies « ICAMRE ». 28&29 Novembre 2018, Boumerdes – Algerie.
[117] Z. Tchantchane, M. Bey, O. Azouaoui, and H. Bendifallah,, " Reconstruction Des Surfaces Frontieres D'un Objet À Partir D'un Nuage De Points 3D," International Conference On Advanced Mechanics And Renewable Energies « ICAMRE ». 28&29 Novembre 2018, Boumerdes – Algerie.
[118] Z. Tchantchane,O. Azouaoui, N. Ghoualmi-Zine, and M. Bey, "Enhanced Flip in 3D Delaunay Triangulation", International Conference on Cyber Security, Artificial Intelligence and Theoretical Computer Science "ICCSAITCS 2022", 27-28 Novembre 2022, Boumerdes, Algerie
[119] Z. Tchantchane,O. Azouaoui, N. Ghoualmi-Zine, and M. Bey, "Enhanced merging order: A novel architecture for merging sub-triangulations", International Conference on Cyber Security, Artificial Intelligence and Theoretical Computer Science "ICCSAITCS 2022", 27-28 Novembre 2022, Boumerdes, Algerie