REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE SAAD DAHLEB DE BLIDA
FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE

CDTA

MEMOIRE DE FIN D'ETUDES

Pour l'obtention

D'un Diplôme de Master en Informatique

Option : Ingénierie de logiciel

THÈME :

Conception et Développement d'une Approche Optimale Pour la Génération du modèle Triple-Dexels de Pièces de Formes Complexes

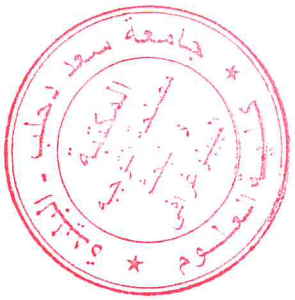
Réalisé par:

Mr. BOUMENIR Sidahmed
Mr. DEBIEB Mohamed Chems Eddine

Soutenue le : 27-06-2018 à 11h, devant :

Mr. HAMMOUDA	**Président**
Mme. MANCER	**Examinateur**
Mr. Kameche Abdallah Hicham	**Promoteur**
Mme. Bouhadja Khadija	**Encadreur**
Mr. Bey Mohamed	**Encadreur**

Année Universitaire 2017/2018


جامعة سعد دحلب
البليدة

ملخـــص:

إن القطع ذات أشكال معقدة موجودة في كل مكان في مختلف الصناعات (التغليف ، القوالب ، السيارات ، الطيران ، ... إلخ). يستخدم برنامج CAD / CAM في عملية الإنتاج الخاصة بهم. محاكاة المكننة هي أداة مهمة في هذه العملية ، وهي مبنية على النمذجة الهندسية للجزء والأداة ومسار الأداة. النماذج المستخدمة هي مستمرة أو منفصلة (مجموعة من النقاط ، الأسطح ، الأحجام).

إن استخدام النماذج المنفصلة أمر سهل التنفيذ مقارنة بالنماذج المستمرة ولكنه يتطلب تفرغا دقيقا لتمثيل النماذج النظرية بأمانة مما يزيد من وقت الحساب. الهدف من هذا العمل هو تطوير وحدة برمجية رسومية وتفاعلية تحت نظام ويندوز تسمح بنمذجة الحجم للأشكال المعقدة ، التي تم تحديدها من خلال نماذج STL الخاصة بهم ، باستخدام تقنية "Triple-Dexel" والإجابة على التنازلات الدقة ، التكلفة.

الكلمات المفتاحية: نموذج معقد ، محاكاة ، K-means ، نموذج STL ، نموذج "Triple-Dexel".

Résumé :

Les pièces de formes complexes sont omniprésentes dans diverses industries (emballage, moules, automobile, aéronautique, …etc.). Les logiciels de CAO / FAO sont utilisés dans leur processus de production. La simulation d'usinage est un outil important dans ce processus, elle est basée sur la modélisation géométrique de la pièce, de l'outil et du trajet d'outil. Les modèles utilisés sont continus ou discrets (ensemble de points, de surfaces, de volumes).

L'utilisation de modèles discrets est simple à implémenter par rapport aux modèles continus mais nécessite une discrétisation fine pour représenter fidèlement les formes théoriques ce qui augmente le temps de calcul. Le but de ce travail est le développement d'un module logiciel graphique et interactif sous Windows permettant la modélisation volumique des pièces de formes complexes, définies par leurs modèles STL, en utilisant la technique des « Triple-Dexels » et en repondant au compromis précision, coût.

Mots-Clés : Forme Complexe, Simulation, K-means, Modèle STL, Modèle « TripleDexels ».

ABSTRACT :

Pieces of complex shapes are ubiquitous in various industries (packaging, molds, automobile, aeronautics…etc.). CAD / CAM software is used in their production process. Machining simulation is an important tool in this process, it is based on geometric modeling of the part, the tool and the toolpath. The models used are continuous or discrete (set of points, surfaces and volumes).

The use of discrete models is simple to implement compared to continuous models but requires a fine discretization to faithfully represent the theoretical forms which increases the calculation time. The goal of this work is the development of a graphic and interactive software module under Windows allowing the dynamic modeling of parts of complex shapes, defined by their STL models, using the "Triple-Dexel" technique and answering the precision compromise, cost.

Keywords: Complex Form, Simulation, K-means, STL model, "Triple-Dexel" model.

REMERCIEMENTS

Nous remercions ALLAH Seigneur du monde de nous avoir donné l'inspiration et la patience pour mener à bien ce travail.

Nous tenons en premier lieu, à exprimer nos reconnaissances envers nos encadreurs ***Mme BOUHADJA Khadîdja, M BEY Mohamed*** *pour nous avoir dirigées, aidées et soutenues afin de mener à bien ce Modeste travail. Qu'elle ait l'expression de nos remerciements les plus vifs.*

Nous n'oublierons pas de remercier ***M KAMECHE Abdallah Hicham*** *et* ***Mlle Belkasmi,*** *pour ses orientations et ses précieux conseils qui nous ont aidés à structurer notre travail.*

Nous remercions le président de jury, ***M HAMMOUDA*** *ainsi que les membres du jury,* ***Mme MANCER****, pour l'honneur qu'ils nous ont fait d'avoir acceptés d'examiner notre travail.*

Nous adressons, également nos remerciements les plus sincères à ***toute l'équipe CFAO du CDTA*** *pour nous avoir aidées et conseillées, pour leurs précieuses orientations concernant l'aspect méthodologique.*

Nous exprimons notre gratitude à l'ensemble du corps enseignant, technique et administratif du département d'informatique à l'Université de Blida.

DEDICACE

Je dédie ce modeste travail :

Mes chers parents, aucune dédicace ne saurait exprimer mon respect, mon amour éternel et ma considération pour les sacrifices que vous avez consenti pour mon instruction et mon bien être. Je vous remercie pour tout le soutien et l'amour que vous me portez depuis mon enfance et j'espère que votre bénédiction m'accompagne toujours. Que ce modeste travail soit l'exaucement de vos vœux tant Formulés, le fruit de vos innombrables sacrifices, bien que je ne vous en acquitterai jamais assez. Puisse Dieu, le Très Haut, vous accorde santé, bonheur et longue vie et faire en sorte que jamais je ne vous déçoive.

À ma chère sœur Icherak, et mes frères mouloud et houcine, je voudrais vous exprimer toute mon affection et admiration☺.

A toute ma famille en particulier mes cousins et cousines.

Une pensée émue pour mon binome Mohamed et sahbi Abderahmene et tous mes amis, ainsi qu'à tous mes camarades de l'équipe CFAO : Nour El Houda, Sofia, Islam, Fares, Lilia, Asmaa, chahinez, Abde Elkader, Dhia Eddine, Akram, Abde Erahmane et Sabrina l'😈bstacle.

Je voudrais, finalement assurer ma reconnaissance et mes remerciements les plus Distingués à tous ceux qui m'ont apportée leur soutien et leur aide dans l'accomplissement de
Cette étude et a toutes les personnes qui ont contribué de près comme de loin à l'élaboration de ce travail, je leur exprime mes plus profonds respect

Sid Ahmed

DEDICACE

Toutes les lettres ne sauraient trouver les mots qu'il faut... Tous les mots ne sauraient exprimer la gratitude, L'amour, le respect, la reconnaissance... Aussi, c'est tout simplement que Je dédie cette thèse ...

A MA GRAND MERE Qui m'a accompagné par ses prières, sa douceur, puisse Dieu lui prêter longue vie et bcp de santé et de bonheur dans les deux vies.

À MES CHERS PARENTS Aucune dédicace ne saurait exprimer mon respect, mon amour éternel et ma considération pour les sacrifices que vous avez consenti pour mon instruction et mon bien être. Je vous remercie pour tout le soutien et l'amour que vous me portez depuis mon enfance et j'espère que votre bénédiction m'accompagne toujours. Que ce modeste travail soit l'exaucement de vos vœux tant formulés, le fruit de vos innombrables sacrifices, bien que je ne vous en acquitterai jamais assez. Puisse Dieu, le Très Haut, vous accorder santé, bonheur et longue vie et faire en sorte que jamais je ne vous déçoive.

À MES CHERS SOEURS et ses PETIT enfants RAOUF, RAMY, ADEM Aucune dédicace ne saurait exprimer tout l'amour que j'ai pour vous, Votre joie et votre gaieté me comblent de bonheur. Puisse Dieu vous garder, éclairer votre route et vous aider à réaliser à votre tour vos vœux les plus chers.

À MES CHERS ONCLES, TANTES A MES CHERS COUSINS COUSINES

À MES AMIS DE TOUJOURS : ADEL, HAMZA, SIDAHMLED, HOUSSEM, HAMZA, SOFIENE, KADER, DIAE, FARES, ISSLEM ...

À TOUTES LES PERSONNES QUI ONT PARTICIPÉ A L'ÉLABORATION DE CE TRAVAIL À TOUS CEUX QUE J'AI OMIS DE CITER

MOHAMED.

Table des matières

CHAPITRE 2

CHAPITRE 3

Liste des tableaux

Liste des figures

CHAPITRE 1

CHAPITRE 2

CHAPITRE 3

Liste des algorithmes

Introduction Générale

INTRODUCTION GENERALE

1. Contexte :

Le présent travail s'inscrit dans le cadre du programme de recherche triennal 2018-2020 de l'équipe CFAO intitulé « Production Automatisée des Pièces Complexes sur des Fraiseuses Multiaxes» au niveau du Centre de Développement des Technologies Avancées « CDTA ». Il nous a été demandé de proposer et de concevoir une approche optimale pour la modélisation volumique des pièces de forme complexe afin de l'employer dans la simulation d'enlèvement de matière lors du processus d'usinage des surfaces complexes sur des fraiseuses multiaxes.

2. Problématique :

Les pièces de formes complexes sont omniprésentes dans diverses industries (emballage, moules, automobile, aéronautique, etc.). Les logiciels de CAO « Conception Assistée par Ordinateur » et FAO « Fabrication Assistée par Ordinateur » sont utilisés dans leur processus de production. La simulation d'usinage est une étape obligatoire avant de procéder à l'usinage réel sur machine. Elle permet de vérifier la trajectoire d'outil, de détecter les collisions et de prédire les efforts de coupe et la topographie de la surface finie. Elle peut être une simulation géométrique ou bien une simulation physique. Comme la simulation physique est basée sur la simulation géométrique, trois modèles de représentation géométrique doivent être considérés : modèle de la pièce, modèle de l'outil et modèle du trajet d'outil. Ces modèles peuvent être continus ou discrets (points, triangles « STL », volumes, …etc.). Les modèles volumiques tels que « Voxel », « Dexel », « Triple-Dexel », etc. sont largement utilisés dans le processus de simulation d'enlèvement de matière.

L'utilisation de modèles discrets (STL, Dexels, etc.) est simple à mettre en œuvre par rapport aux modèles continus. L'inconvénient majeur c'est la nécessité d'une discrétisation poussée ou très fine pour représenter fidèlement les formes théoriques ce qui augmente considérablement le temps de traitement.

Le présent projet est une extension d'un travail antérieur réalisé en 2015 par Khaled Sebti et Hamid Moulay dans le cadre des projets de fin d'étude « PFE » proposés et encadrés par l'équipe CFAO. Dans ce travail, une modélisation volumique de la pièce à usiner en Triple-

Dexels a été générée. Afin de minimiser le temps de calcul, une approche intéressante a été proposée et implémentée. Elle consiste à regrouper les triangles du modèle STL dans des ensembles par la méthode de K-means. Sa principale limitation est liée à l'initialisation des centres de Clusters. Chaque initialisation du processus correspond à une solution différente qui peut dans certains cas être très éloignée de la solution optimale. Dans cet ordre d'idées et pour surmonter les limites de cette approche, le présent travail sera développé.

3. Objectif

L'objectif de ce travail est de proposer et d'implémenter une approche optimale permettant la modélisation volumique des pièces de formes complexes, définies par leurs modèles STL, en utilisant la technique des « Triple-Dexels ». Le but de ce travail est le développement d'un module logiciel graphique et interactif sous Windows répondant à notre problématique et surmontant le compromis précision et coût.

4. Structuration du mémoire :

Le présent mémoire est composé des chapitres suivants :

- Le premier chapitre est consacré à la présentation du processus de CFAO et au format d'échange de données « STL ». Un état de l'art sur les différentes méthodes de simulation est également présenté dans ce chapitre.
- L'étude conceptuelle de notre application logicielle est menée au chapitre deux avec description des algorithmes utilisés.
- Le dernier chapitre présente l'application logicielle développée, les tests et la validation des résultats.

Ce mémoire sera clôturé par une conclusion générale et des perspectives à donner à ce travail pour l'enrichir davantage.

Chapitre I :

Généralités et état de l'art

Introduction :

Les pièces mécaniques de formes complexes sont largement utilisées dans diverses industries (emballage, moules, automobile, aéronautique, …etc.). Elles sont usinées sur des fraiseuses numériques multiaxes de 03-axes à 05-axes en raison de leurs formes géométriques très complexes. Ces pièces doivent répondre à des exigences fonctionnelles et/ou de style, ce qui impose une attention particulière dans leur mise en production.

Dans ce chapitre, nous commençons par introduire le processus de fabrication de pièces de formes complexes. Puis, nous présentons l'outil CFAO ainsi que le format d'échange de données STL. Par la suite, un état de l'art sur les différentes méthodes de simulation est exposé pour aboutir finalement à la méthode Triple-Dexels objet de notre travail.

1. Fabrication de pièces :

Parmi les techniques de fabrication de pièces mécaniques, l'usinage par enlèvement de matière est le plus utilisé dans l'industrie. Son principe est d'enlever la matière de façon à donner à la pièce brute la forme et les dimensions voulues avec une bonne précision à l'aide d'une machine dédiée. Lors de l'usinage d'une pièce, l'enlèvement de matière est réalisé par la conjonction de deux mouvements relatifs entre la pièce et l'outil : le mouvement de coupe (vitesse de coupe) et le mouvement d'avance (vitesse d'avance). De nos jours, des machines-outils à commande numérique « MOCN », c'est-à-dire des machines asservies par un système informatique, permettent d'automatiser partiellement ou totalement la procédure [1].

1.1. Usinage des pièces avec le système informatisé :

À chaque phase du processus de fabrication (Figure I.1), le concepteur et/ou l'operateur choisit le type d'usinage à réaliser, la machine, l'outil ainsi que le support de pièce permettant l'obtention de tous les éléments de cotation de la surface considérée. D'une manière générale, les formes des surfaces usinées peuvent être planes ou de révolution. Les principaux usinages sont le fraisage (surfaces planes) et le tournage (surfaces de révolution). Avec l'apparition de la commande numérique et les outils de Conception et de Fabrication Assistés par Ordinateur « CFAO », il est désormais possible d'usiner une multitude de surfaces courbes. Toutefois, il convient de noter que les outils utilisés sont sensiblement les mêmes que pour les machines traditionnelles et que leurs trajectoires sont constituées de segments de droites et d'arcs de cercles [1].

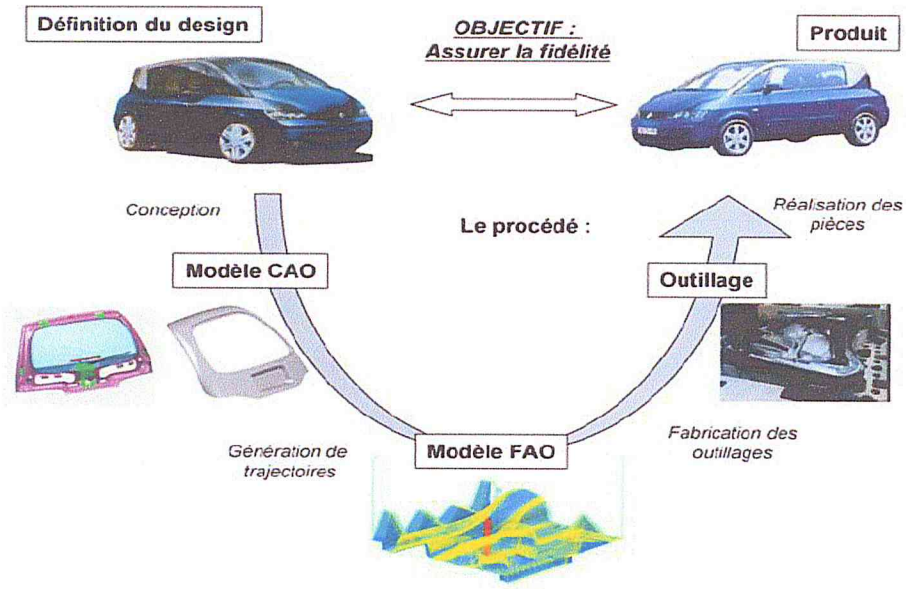


Figure I.1 : Processus de fabrication d'une pièce mécanique [2].

1.2. Forme libre :

Les formes libres des pièces mécaniques (Figure I.2) sont conçues à partir d'une image de la forme que l'ingénieur concepteur veut obtenir. Elles ne sont pas décrites par des équations analytiques. Donc, il est nécessaire d'avoir un système de conception qui permet de concevoir des formes très complexes sans avoir à donner les équations de ces surfaces. Ces dernières sont appelées « surfaces gauches » ou « surfaces de formes libres » [3]. Les polynômes, les fonctions trigonométriques, les fonctions exponentielles et les fonctions logarithmiques sont autant de formes mathématiques permettant la description des courbes et des surfaces. En outre, les techniques de modélisation basée sur les polynômes représentent la possibilité de représenter des fonctions continues sur un intervalle [a, b] donné avec une précision arbitrairement fixée. Ces surfaces peuvent être représentées également par des modèles discrets tels que le modèles STL d'écrit dans la section 2.5.

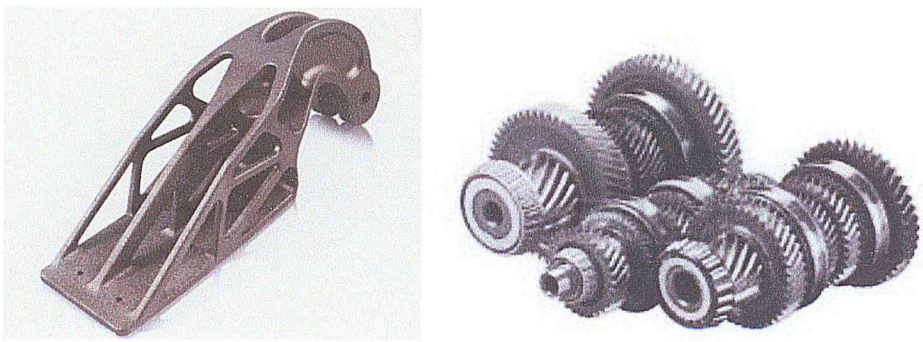


Figure I.2 : Pièces mécaniques complexes.

2. Généralités sur la CFAO :

2.1. Définition

L'idée générale de la Conception et Fabrication Assistées par Ordinateur « CFAO » est d'utiliser les capacités de l'ordinateur afin de concevoir la pièce pour ensuite la fabriquer grâce à une machine connectée directement à l'ordinateur. Les objectifs de la « CFAO » sont d'obtenir une extrême précision, de réaliser un gain de temps et de minimiser l'intervention humaine. La « CFAO » est la combinaison de la « CAO » (Conception Assistée par Ordinateur) et de la « FAO » (Fabrication assistée par Ordinateur). C'est un procédé qui permet de concevoir et de programmer les formes d'une pièce à usiner et d'en définir les différentes opérations d'usinage et agencer les phases successives menant à la réalisation en final d'une pièce conforme aux exigences exprimées par le designer. L'évolution des systèmes de la « CFAO » doivent répondre aux critères suivants :

- Très haute rentabilité.
- Automatisation complète de la fabrication.
- Communication et échange de données.
- Perfection et finition des surfaces.
- Contrôle des éventuels points de collisions.
- Vérification des performances et de la productivité globale : applicables à des formes diverses et complexes.
- Applicables à divers modèles de surfaces.
- Traitement des pièces géométriques différentes.
- Respect de la qualité imposée et de la validité de la pièce.
- Contrôle de l'efficacité de l'usinage.

2.2. Processus de CFAO

La CFAO composé de deux principaux modules : le premier est dédié à la conception assistée par ordinateur « CAO » et le second est dédié à la fabrication assistée par ordinateur « FAO ». Le passage d'un module à l'autre se fait systématiquement dans le cas où la « CAO » est intégrée à la « FAO » (Figure I.3). Dans le cas découplé ou interfacé, le passage se fait par l'utilisation de formats d'échanges de données tel que : IGES, STEP, VRML, DWG, DXF et STL. Notre travail se focalise sur le format STL.

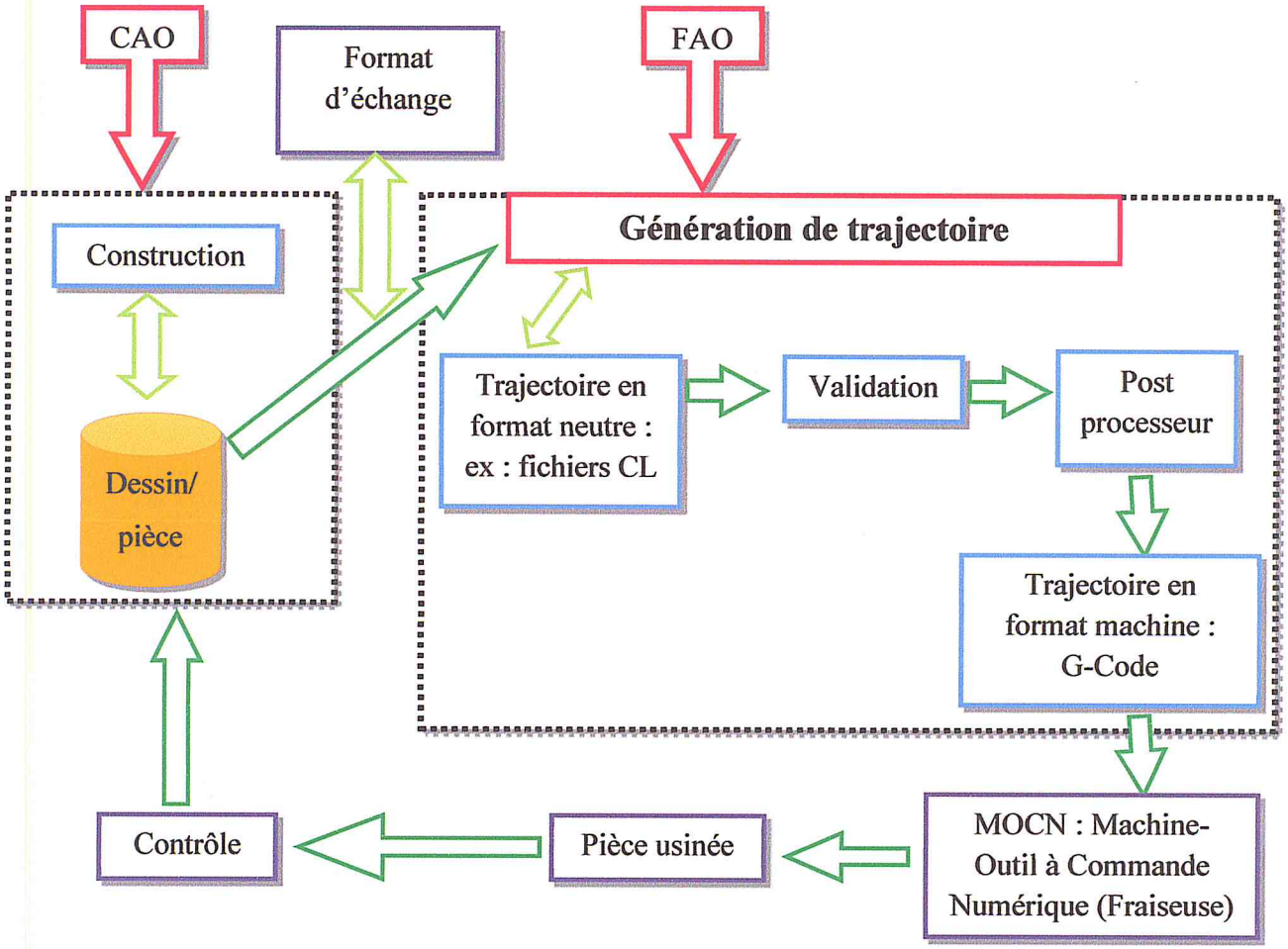


Figure I.3 : Flux d'informations sur le système de CAO/FAO découplé ou Interface [4].

2.3. Conception assistée par ordinateur « CAO » :

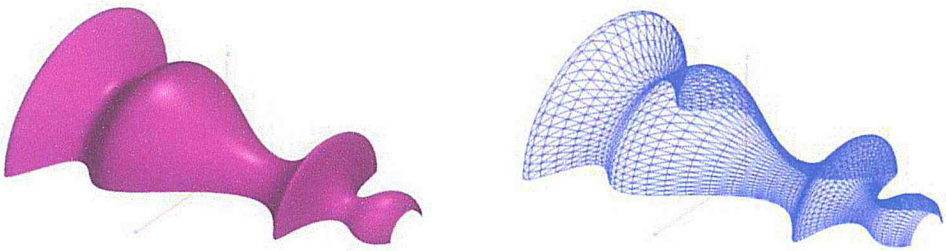
La CAO a modifié profondément la conception des pièces. Les idées conceptionnelles peuvent être reprises et développées plus facilement. L'ingénieur en charge de la conception possède des informations provenant de calculs ou mesures matérialisées par des points, des courbes, des surfaces, appelées données géométriques fonctionnelles. Elles servent de canevas pour la reconstruction de la surface dans un environnement CAO. En sus de ces données, l'ingénieur prendra en compte les contraintes fonctionnelles de conception ne pouvant être exprimées sous forme géométrique (points, droites, etc.) d'une base de données CAO. Suivant le cas, nous pouvons être confrontés à ces types de contraintes : masses, volumes, surfaces ou directions de tangence et de concavité.

2.4. Fabrication assistée par ordinateur « FAO » :

Après avoir numérisé la pièce par des données géométriques fonctionnelles à l'aide de la CAO, la production des programmes d'usinage qui contiennent les parcours d'outils et les fonctions annexes (mise en route de la broche, le changement d'outil, etc.) est réalisée par la FAO. Ces logiciels ne sont pas entièrement automatiques, laissant ainsi à l'utilisateur de fixer le type de trajectoires à utiliser en fonction de l'entité choisie (profil, poche, etc.).

2.5. Format d'échange de données STL :

Le format STL est un format dédié à la Stéréo-lithographie introduit par la société 3D Systems en 1987. Il permet de décrire un objet sous la forme d'un polyèdre à facettes triangulaires. Il offre l'avantage d'être facilement généré par des outils CAO [5]. Le modèle STL permet de représenter la peau extérieure des objets avec des facettes triangulaires dont le nombre et la taille dépendent de la précision demandée (Figure I.4).



a. Surface de base. b. Modèle STL.

Figure I.4 : Format STL d'une surface théorique [6].

Donc, ce modèle est composé d'une liste de triangles où chaque triangle est défini par sa normale unitaire N dirigée vers l'extérieur et les coordonnées de ses trois sommets orientés P1(X1, Y1, Z1), P2 (X2, Y2, Z2) et P3(X3, Y3, Z3) (Figure I.5 et Figure I.6).

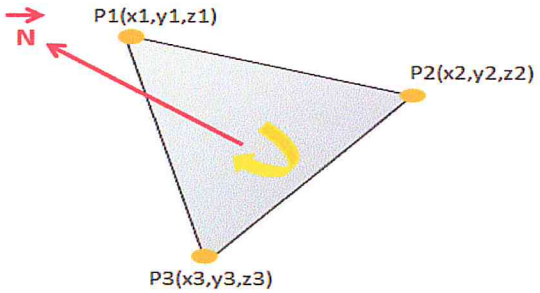


Figure I.5 : Paramètres d'un triangle.

```
facet normal  -0.39361165  -0.40610725  0.82471011
   outer loop
     vertex  1.8181818  0.45454545  1.1168503
     vertex  2.2727273  0.45454545  1.3337925
     vertex  2.2727273  0.90909091  1.5576217
   endloop
 endfacet
```

Figure I.6 : Structure d'un triangle du fichier STL.

3. Simulation d'usinage :

La simulation d'usinage (Figure I.7) est une représentation virtuelle du programme qui sera exécuté sur la machine où chaque mouvement est décomposé et réparti sur les différents axes de la machine [7]. C'est une opération qui s'effectue en FAO avant de passer à l'usinage réel sur machine-outil.

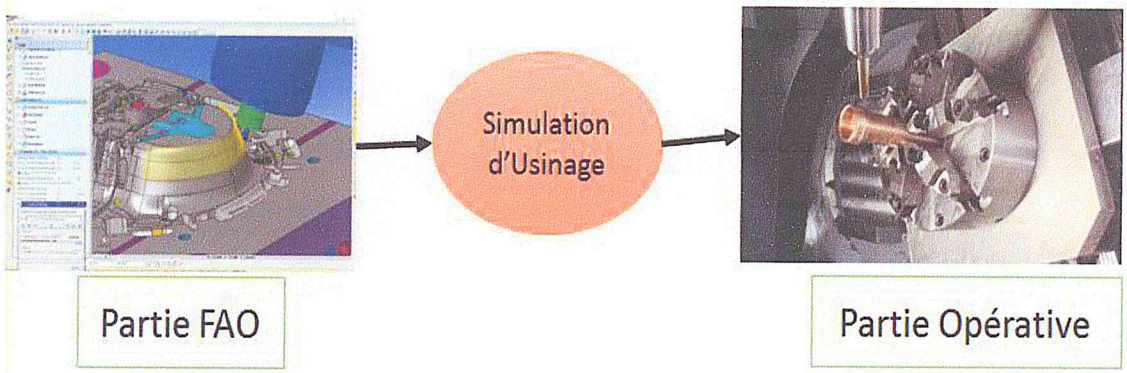


Figure I.7 : Simulation d'usinage entre FAO et machine.

3.1. Catégories de simulation :

La simulation de l'usinage est distinguée en deux catégories: simulation géométrique et simulation physique (Figure I.8).

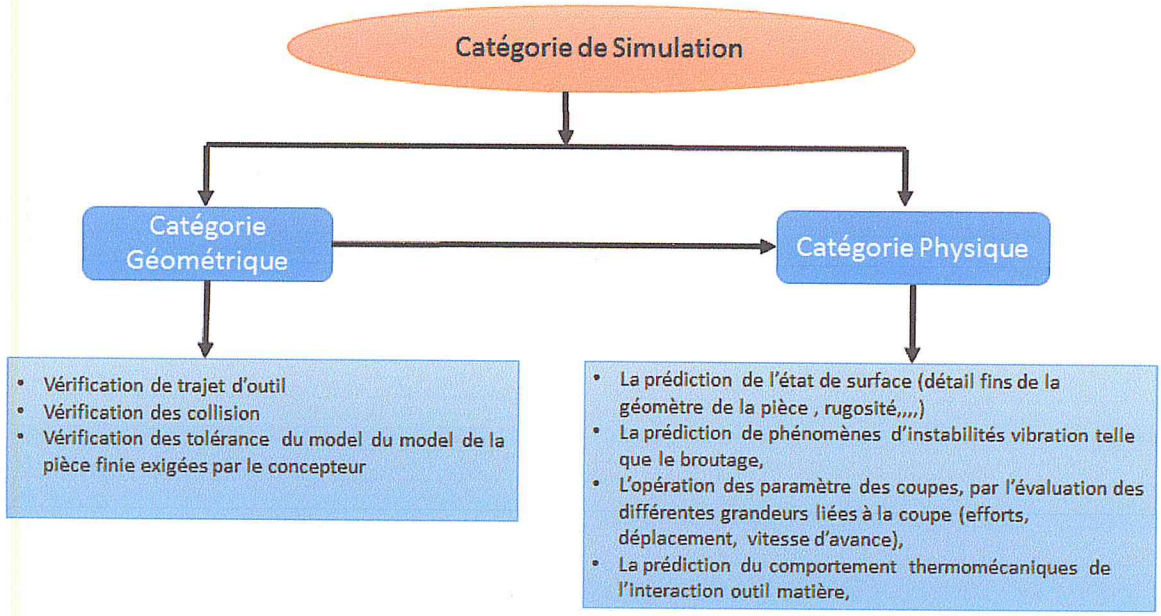


Figure I.8 : Catégories de simulation [7].

3.1.1. Simulation Géométrique :

La simulation géométrique est utilisée pour la vérification du trajet d'outil, les collisions, la tolérance du modèle de la pièce finie exigée par le concepteur. En outre, elle peut fournir des informations géométriques telles que l'angle d'attaque de l'outil à la simulation physique.

3.1.2. Simulation physique :

La simulation physique d'un processus d'usinage est utilisée pour la prédiction de l'état de surface, de phénomènes d'instabilités, de vibrations,...etc. Elle est basée sur la simulation géométrique et le choix du matériau de l'outil de coupe.

3.2. Echelles de simulation :

L'étude de l'usinage est souvent abordée à l'aide d'une approche multi-échelles. Ceci permet de séparer les difficultés en limitant le nombre de phénomènes à prendre en compte et la taille du modèle à une échelle donnée. Trois échelles d'analyse peuvent être considérées (Figure I.9) : humaine, macroscopique et microscopique.

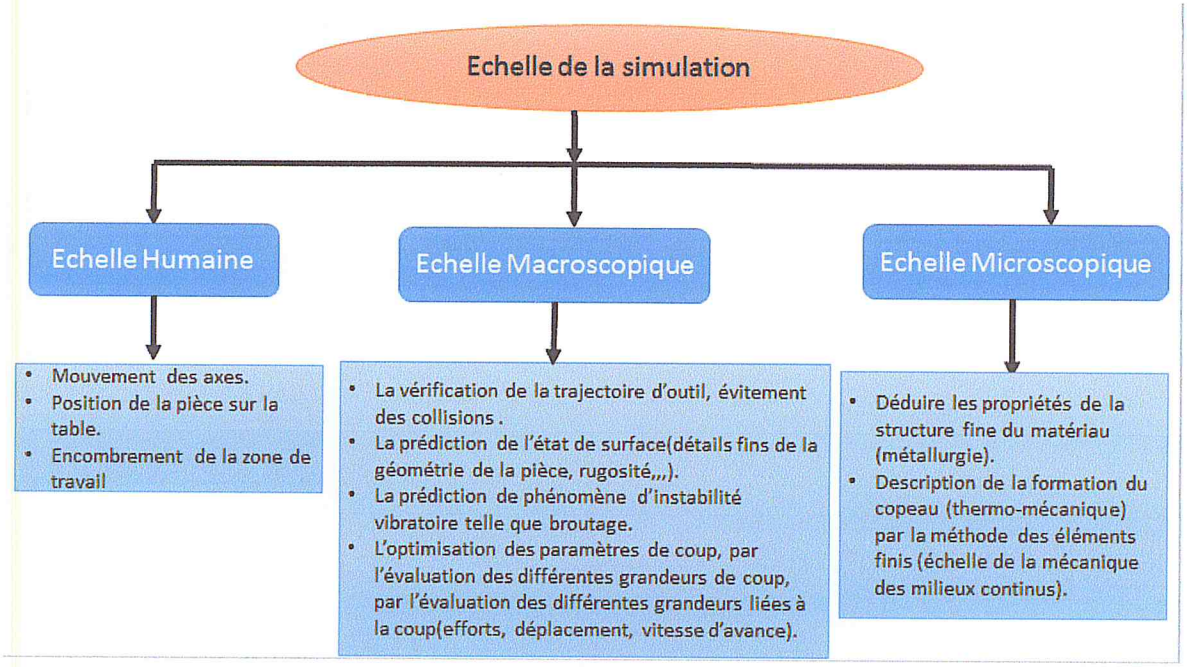


Figure I.9 : Echelles de simulation [7].

3.2.1. Echelle humaine :

Est une simulation globale de l'environnement de l'usinage. Son objectif est de s'imprégner du comportement du moyen de production afin de préparer l'usinage :

- Mouvements des axes.
- Position de la pièce sur la table.
- Encombrement de la zone de travail.

Cette étape est indispensable dans le cas où le moyen de production est très complexe et induit des mouvements relatifs de la pièce par rapport à l'outil difficiles à anticiper (machine multiaxes, robot d'usinage, etc.). Elle permet de détecter d'éventuelles collisions pendant le processus. La plupart du temps, ce type de simulation est intégré dans les logiciels de FAO. Il existe cependant des logiciels de simulation indépendants spécialisés dans l'interprétation des codes ISO issus des logiciels de FAO.

3.2.2. Echelle macroscopique :

Le but de cette simulation est de déterminer globalement le volume de matière enlevé pour chacune des phases d'usinage de la pièce. A cette échelle, les techniques de simulation permettent, entre autre, de visualiser et d'anticiper les défauts de surface liés purement à la stratégie programmée ou à la cinématique de la machine. Dans la littérature, différents sortes de travaux sont menés. Ceux qui s'appuient sur la représentation de la

pièce à usiner. D'autres travaux se focalisent sur la génération du volume de matière balayé par l'outil de coupe. La difficulté à ce niveau-là est prononcée par la cinématique de la machine 05 axes où l'outil subit un mouvement de rotation et de translation simultanés. Pour une haute précision, d'autres travaux proposent la théorie de la cinématique des systèmes multi-corps.

3.2.3. Echelle microscopique :

La simulation à cette échelle est celle de l'étude des matériaux. Il s'agit de déduire certaines propriétés à partir de la structure fine du matériau. Ces propriétés sont, par exemple, les lois de comportement du matériau utilisé. A une échelle un plus supérieure, nous trouvons l'échelle Microscopique. A cette échelle, la description de la formation du copeau est étudiée. En se basant sur une description thermomécanique faisant intervenir les différents phénomènes physiques et métallurgiques, mais à une échelle qui est celle de la mécanique des milieux continus, à cette échelle la simulation est souvent abordée par la méthode des éléments finis.

3.3. Modèle de simulation

La simulation de l'usinage est distinguée en deux modèles : modèle géométrique et modèle physique (Figure I.10).

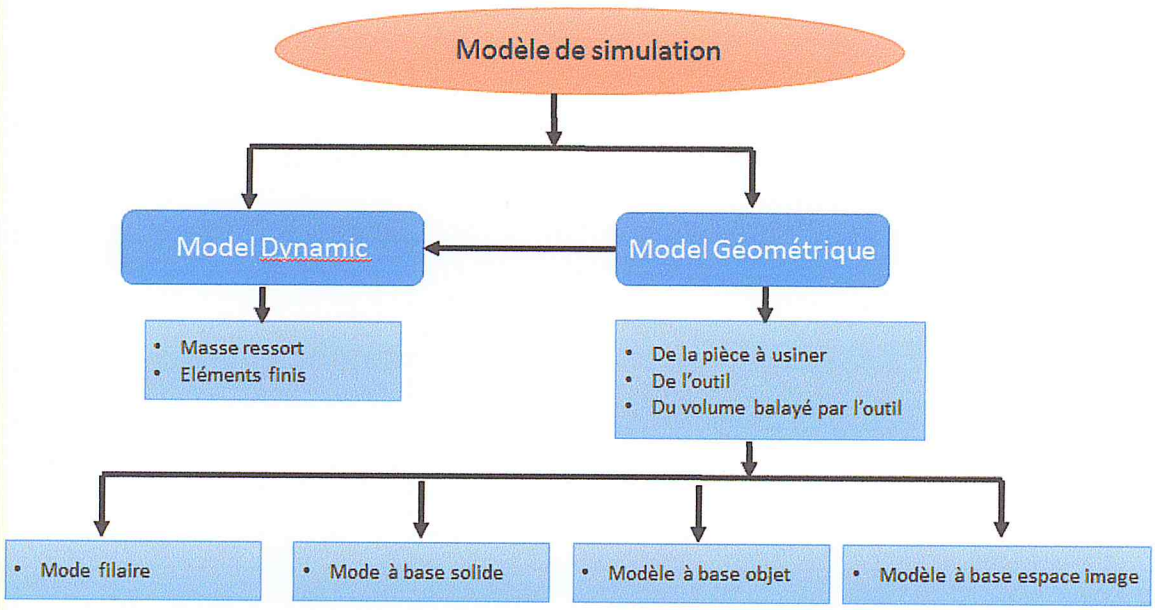


Figure I.10 : Modèles de simulation [7].

3.3.1. Modèle dynamique :

Le modèle dynamique peut être un modèle simple masse ressort ou un modèle complexe élément finis. La modélisation par éléments finis permet une discrétisation spatiale beaucoup plus fine et plus souple. Elle permet aussi d'obtenir des modes de vibrations beaucoup plus réalistes et d'aborder le cas où la pièce et/ou l'outil sont déformables dans la zone de travail.

3.3.2. Modèle géométrique :

Trois modèles de représentation géométrique sont à considérer à savoir : modèles de la pièce, modèle de l'enveloppe de l'outil et modèle du volume balayé par l'outil. Les modèles géométrique utilisés dans ce domaine peuvent aller du plus simple, une série de points, au plus complexe, description facettées ou volumique. Nous distinguons quatre grandes familles de modèles: filaire, à base solide, à base objet et à base espace image. Dans ce dernier, se retrouve les techniques de représentation volumique telles que le Dexel, le Voxel et le Triple-Dexel. Notre travail se focalise sur la simulation géométrique à l'échelle macroscopique en utilisant la modélisation volumique des pièces de formes complexes par des Triple-Dexels à partir de leurs modèles STL.

3.4. Simulation géométrique à l'échelle macroscopique :

La littérature montre qu'ils existent différentes façons de classification pour la représentation géométrique en simulation à l'échelle macroscopique. Les méthodes utilisées sont classées comme suit : mode filaire, mode à base solide, mode à base objet et mode espace image [8].

3.4.1. Mode filaire :

Dans ce mode de simulation, la trajectoire et la forme de la pièce à usiner sont affichées sous forme de fil de fer. Ce modèle a une structure de données simple et rapide. Il a été largement appliqué au début de la simulation de l'usinage. L'utilisation de ce modèle reste applicable aux pièces de géométrie simple [9].

3.4.2. Mode à base solide :

La simulation basée sur le mode solide est une représentation en volume 3D. Il est utilisé pour la géométrie et les simulations physiques. Ce modèle permet une représentation géométrique très précise mais coûteuse. Les deux modèles existants pour ce cas sont basés sur CSG et sur B-Rep.

➢ ***Model CSG :*** il définit la forme constructive d'un modèle 3D en utilisant des volumes primitifs tels que cylindres, sphères, etc. Bien que, les opérations booléennes et la cohérence sont simples, la visualisation ou l'analyse des données peut nécessiter une transformation en Modèle B-Rep.

➢ ***Model B-Rep :*** ce modèle est adapté à la visualisation. Contrairement au modèle CSG, le modèle B-Rep explicitement définit le volume par une liste de surfaces, arêtes et sommets. Le calcul du coût est très élevé en termes de temps, de stockage des données et de complexité. Pour n mouvements d'outil, le coût de la simulation est estimé à O(n).

3.4.3. Modèle à base objet :

Dans une simulation d'usinage à base d'espace objet, les pièces sont représentées par un ensemble de points discrets avec des vecteurs ou des surfaces avec des vecteurs ou certains éléments de volume. Ils existent trois grandes méthodes de décomposition pour les modèles de simulation d'usinage espace objet :

➢ ***Méthode de Z-map :*** elle consiste à décomposer le modèle de la pièce en plusieurs vecteurs 3D (Figure I.11). Chaque vecteur commence par la valeur de la hauteur de la partie brute. Pendant le processus de simulation, les hauteurs des vecteurs 3D sont mises-à-jour pour chaque mouvement d'outil. Dans ce cas, les Opérations booléennes ont une seule dimension et donc la simulation est très rapide [7].

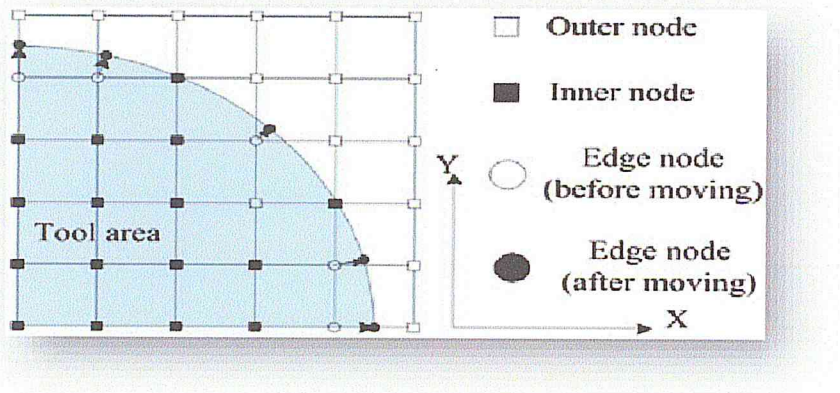


Figure I.11 : Modèle Z-map [7].

➢ ***Méthode des vecteurs :*** cette méthode implique [7] la discrétisation de la surface selon des méthodes spécifiques pour obtenir un ensemble de points (Figure I.12). Pour chaque point, un vecteur est associé à des limites entre la surface nominale et la partie brute. Ses vecteurs peuvent être orientés de deux façons :

- Suivant la normale à la surface (exacte) : dans ce cas, chaque vecteur est indépendant linéairement des autres vecteurs.
- Suivant l'axe Z de l'outil (simplifiée) : dans ce cas, tous les vecteurs sont parallèles à l'axe Z. Ce cas s'adapte à l'usinage 03 axes.

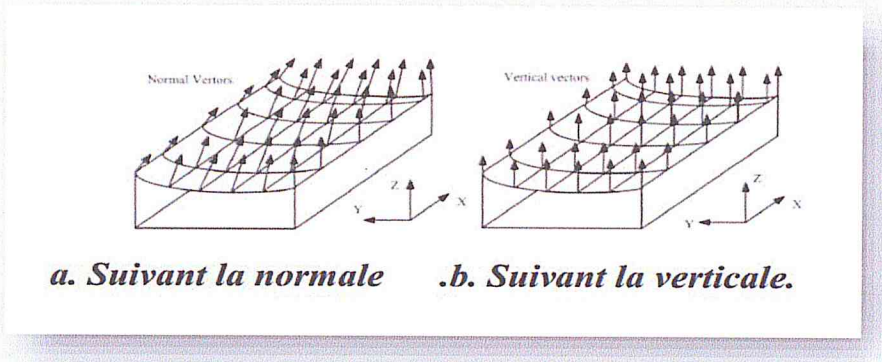


Figure I.12 : Orientation des vecteurs [7].

➢ ***Méthode d'Octree :*** un Octree [7] (Figure I.13) est une structure de données de type arbre dans laquelle chaque nœud peut compter jusqu'à huit enfants. Les octrees sont le plus souvent utilisés pour partitionner un espace tridimensionnel en le subdivisant récursivement en huit octants. Cette représentation en octree hiérarchique offre à la simulation d'usinage une simplicité de calcul des opérations booléennes même lorsque la zone de coupe locale est complexe.

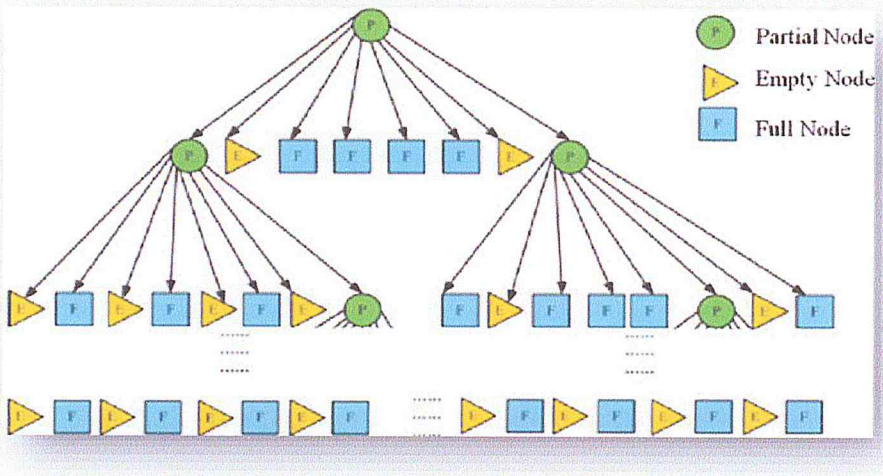


Figure I.13 : Modèle Octree [7].

3.4.4. Modèle à base espace image :

Dans la simulation d'usinage en espace image, les pièces sont représentées par des profondeurs et des pixels (Dexel). Ce modèle est l'extension du Z-Buffer qui est utilisé dans l'élimination des parties cachées en infographie. Ce modèle est particulièrement performant puisqu'il est issu des méthodes de représentation de l'imagerie et de rendu réaliste [10].

4. Techniques de représentation volumique :

La simulation d'usinage permet de vérifier le trajet d'outil, la tolérance d'usinage, détecter les collisions et prédire la qualité de surface. Pour cette dernière, le modèle géométrique à prendre en considération est le modèle volumique. Dans la littérature plusieurs modèles volumiques sont utilisés dans la simulation d'enlèvement de matière tels que les Voxels, les Dexels et les Triple-Dexels.

4.1. Voxels :

Les solides sont représentés par une liste de Voxels (éléments de volume). Les Voxels (Figure I.14) sont des cellules spatiales occupées par le solide. Ce sont généralement des cubes de taille fixe répartis selon une grille. En général, le solide est défini par la liste des coordonnées des centres des cellules [11]. Les avantages du Voxel sont :

- Facilité de validation.
- Simplicité de l'accès à un point donné.
- Unicité spatiale assurée.

Par contre, ses inconvénients qu'il nécessite énormément de ressources, tant pour le stockage que pour le rendu qui ne bénéficie pas d'accélération matérielle.

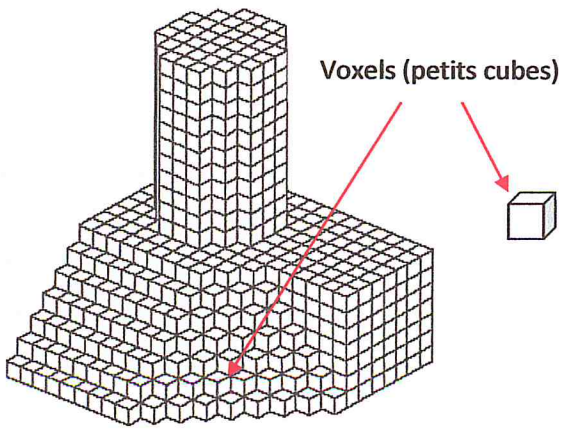


Figure I.14 : Représentation de volume par des Voxels.

4.2. Dexels :

Les Dexels peuvent modéliser des pièces de forme complexe ayant des trous ou des surface courbes, ondulées, etc. La Figure I.15 montre quelques exemples en 2D de pièces modélisées par des Dexels.

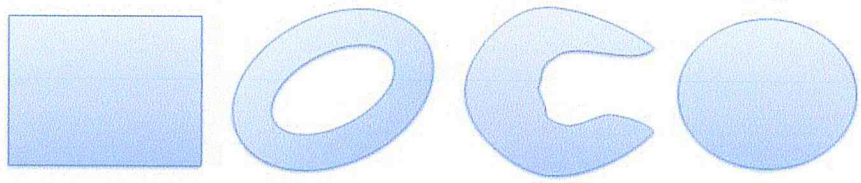


Figure I.15 : Différents domaines modélisé par des Dexels en 2D [12].

Le modèle en Dexels permet d'approximer un solide par un ensemble de parallélépipèdes appelé Dexels (Figure I.16) orientés selon les trois directions principales (X, Y et Z) dont le nombre et la taille dépendent des précisions imposées. Alors qu'un Dexel est un pixel avec profondeur.

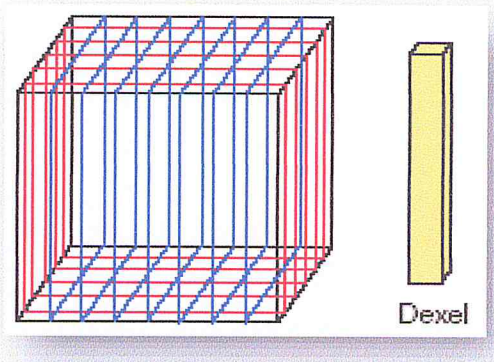


Figure I.16 : Modèle en Dexels [13].

Chaque Dexel est défini le long de l'axe Z est une colonne de matière parallèle à cet axe (Figure I.17), qui est défini par le centre (X0, Y0) de hauteur H (Z min et Z max) de base carrée ou rectangulaire (ΔX et ΔY) (Figure I.17). Les valeurs de base (ΔX, ΔY et ΔZ) sont les paramètres qui déterminent un modèle précis. Selon la forme de l'objet à modéliser, plusieurs Dexels peuvent être créés sur la même ligne droite.

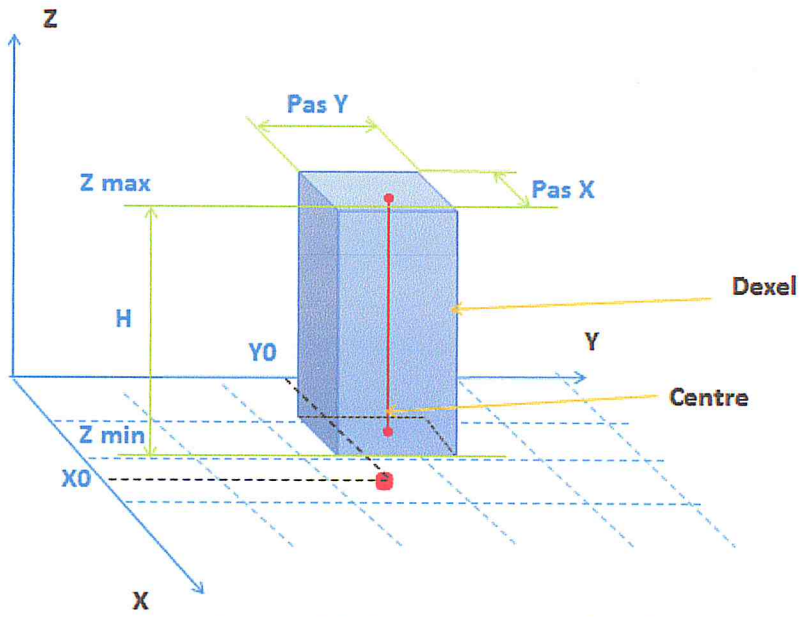
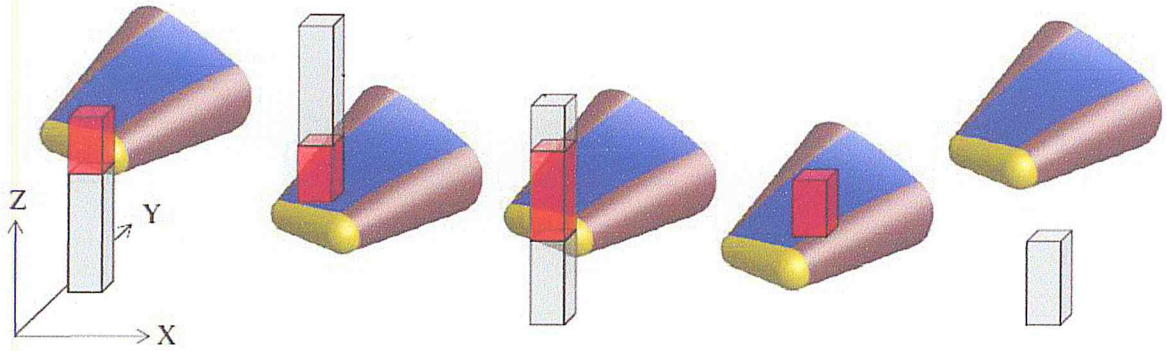


Figure I.17 : Paramètres d'un Dexel.



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

Figure I.18 : Opérations de coupe élémentaires 1D le long d'un axe.

(a), (b) Dexel est coupé en haut et en bas, (c) Dexel est divisé, (d) Dexel de type vide est supprimé, (e) pas d'interaction entre la droite et la pièce [14].

4.3. Triple Dexel :

La modélisation Triple-Dexels (Figure I.19) est une méthode de représentation volumique de la géométrie d'un solide. Elle est obtenue par un calcul d'intersection entre un modèle surfacique d'un solide et des rayons orientés selon les trois directions orthogonales X, Y et Z (Figure I.20). En raison de ses opérations booléennes rapides, de sa structure de données simple et de sa mise en œuvre facile, la modélisation Triple-Dexels est très adaptée aux applications de simulation graphiques en temps réel telles que la vérification de l'usinage numérique et la sculpture virtuelle [15].

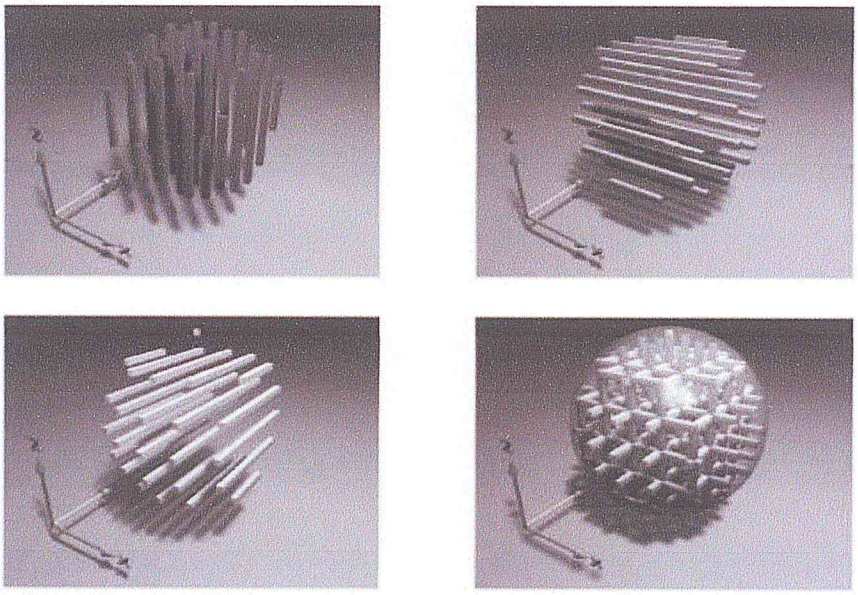


Figure I.19 : Représentation du modèle Triple-Dexels [12].

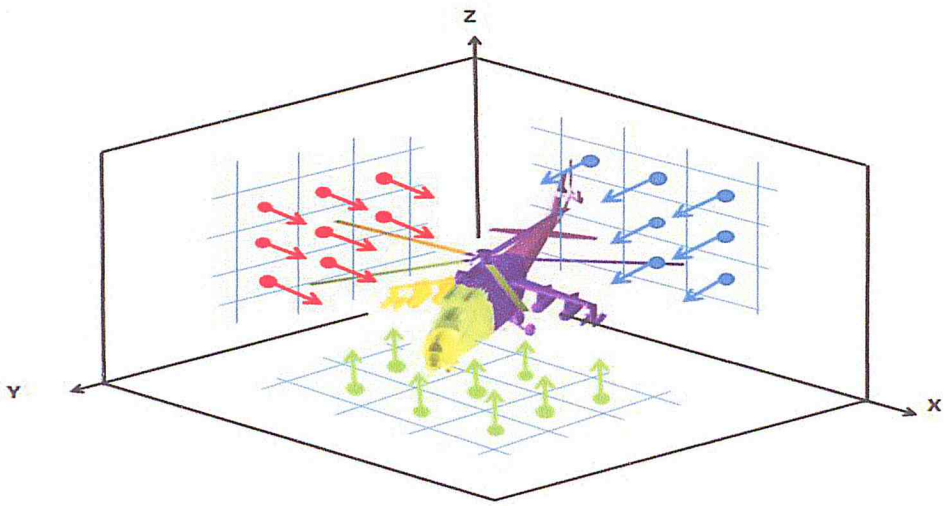


Figure I.20 : Paramètres d'un Triple-Dexel.

4.3.1. Création du modèle Triple-Dexels :

La création des Dexels dans les directions X et Y est réalisée en suivant le même processus que pour la direction Z (Figure I.21). Le modèle de l'objet en Triple-Dexels est obtenu en combinant les Dexels créés dans les trois directions.

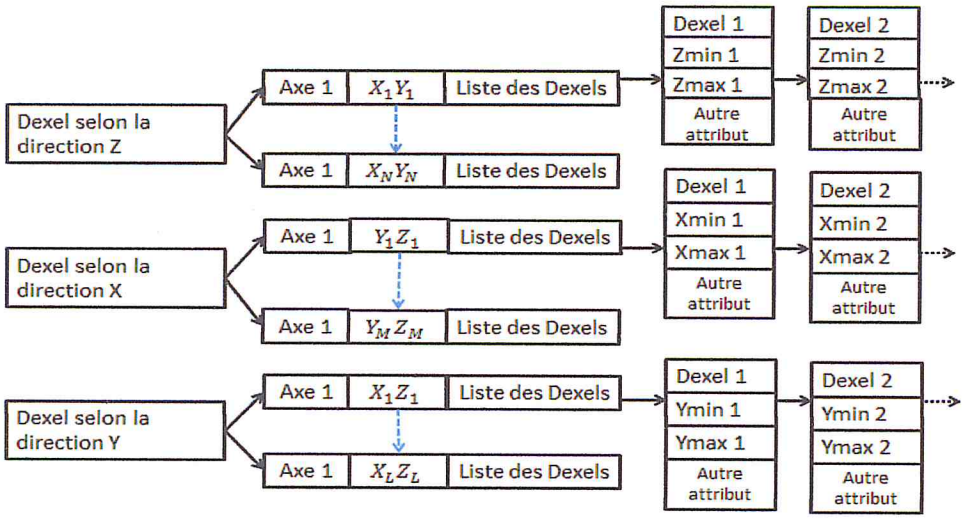


Figure I.21 : Création du modèle Triple-Dexels [3].

4.3.2. Avantages :

Le modèle Dexel est limité dans les régions où les surfaces et les normales sont presque perpendiculaires à la direction des rayons. La modélisation Triple-Dexel est utilisée pour résoudre ce problème. Les auteurs de [12] montrent clairement (Figure I.22) que la surface générée à partir des données Triple-Dexels est plus précise que la surface reconstruite à partir des données simples Dexel et la modélisation Triple-Dexel est parfaitement adaptée pour les applications de simulation à base graphique en temps réel tels que la simulation d'usinage et que les Triple Dexels sont plus rapides est plus précises que les Voxels [16].

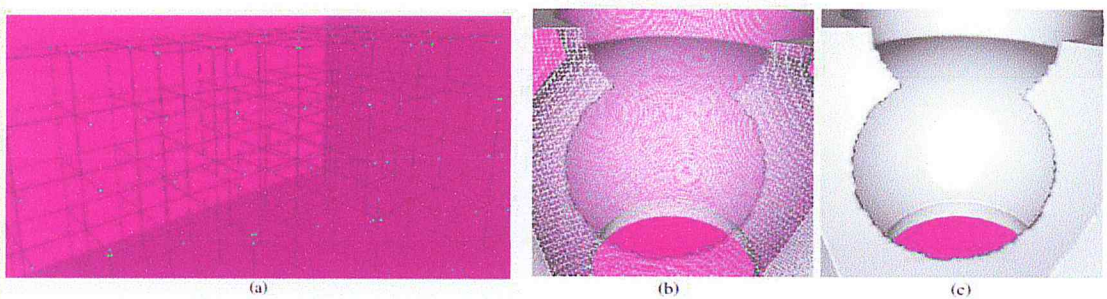


Figure I.22 : Représentation de surface à l'aide d'un modèle orthogonal Triple-Dexel. (a) entrées et sorties (points verts) (b) squelette topologique: segments de ligne (c) peau rendue: triangulation [14].

4.3.3. Limite des Triple-Dexels :

Le modèle Triple-Dexel est limité dans le cas où le rayon passe par un sommet ou bien un segment d'un triangle. La Figure I.23 montre un polygone sur le plan 2D afin de mettre en évidence les problèmes de concavité et de convexité (Figure I.24). Comme tous

modèles discrets, il faut affiner la discrétisation pour représenter fidèlement la pièce à usiner. Dans le cas des Triples-Dexels, il faut que les dimensions de la base des Dexels (pas au carré) soient très petites. Par conséquent, le temps de calcul augmente énormément.

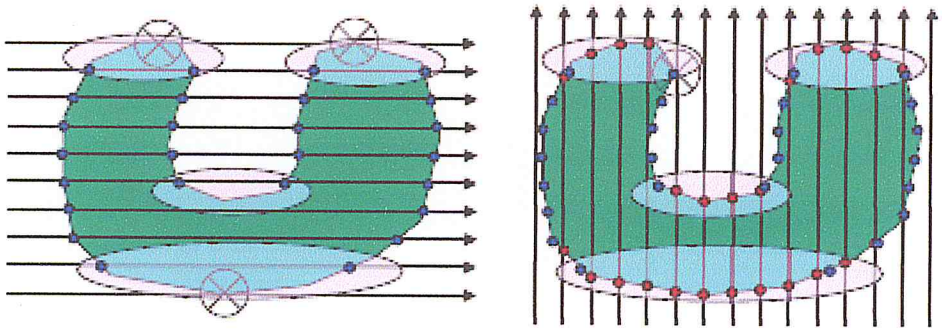
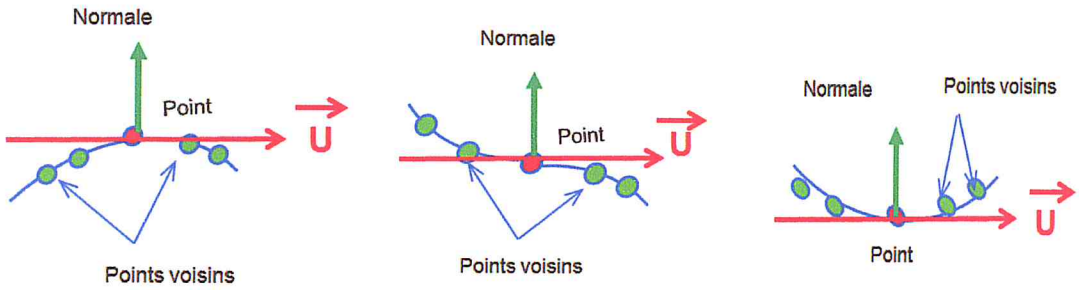


Figure I.23 : Polygone sur le plan 2D [14].



a. Point convexe. b. Point en selle de cheval. c. Point concave.

Figure I.24 : Différents cas où le problème de Triple-Dexel se pose.

Conclusion :

Dans ce chapitre et en première partie, nous avons introduit le processus de fabrication des pièces de forme libre et nous avons présenté d'une manière générale l'outil CFAO ainsi que le format d'échange STL. En deuxième partie, nous avons exposé les différentes techniques de simulation et nous avons accentué sur la simulation géométrique à l'échelle macroscopique des pièces complexes usinées sur des fraiseuses multiaxes. Par la suite, nous avons enchaîné la simulation d'enlèvement de matière qui nécessite la représentation volumique des pièces à usiner, nous avons également étudié les techniques de Voxel, Dexel et Triple-Dexel. Finalement, nous avons opté pour le modèle Triple Dexels.

Dans le chapitre qui suit, nous allons procéder à une conception d'une approche optimale permettant la représentation volumique de pièces de forme libre par des Triple-Dexels à partir de leurs modèles STL en surmontant les limites de cette technique présentées dans la section précédente.

Chapitre II :

Etude Conceptuelle

Introduction :

Dans le précédent chapitre, nous avons présenté une situation de l'état de l'art et des généralités indispensables pour la compréhension de notre travail. Dans ce chapitre, nous posons la problématique et l'approche proposée pour la résoudre. Ensuite, nous présentons la démarche conceptuelle détaillée de l'application développée selon un enchaînement logique des différents algorithmes utilisés. Pour cela, nous avons exploité le formalisme UML (Unified Modeling Language) qui offre une flexibilité marquante et qui s'exprime par l'utilisation des diagrammes.

1. Analyse des besoins :

Ce travail s'insère dans le cadre de développement de modules logiciels pour la production des surfaces de formes complexes initié par l'équipe Conception et Fabrication Assistées par Ordinateur « CFAO » de la Division Productique et Robotique « DPR » du Centre de Développement des Technologies Avancées « CDTA ». Dans ce projet, nous nous intéressons à la modélisation volumique des pièces de formes complexes, définies par leurs modèles STL, en utilisant la technique des « Triple-Dexels ». Le travail consiste à proposer une méthodologie de discrétisation du volume d'un objet de forme quelconque en introduisant la notion de matière. La discrétisation générée doit garantir un compromis entre précision et temps de traitement.

1.1. Problématique :

Comme indiqué dans le chapitre précédent, le modèle Triple-Dexel est limité dans le cas de pièces présentant de fortes concavités ou de fortes convexités. Comme tout modèle discret, il faut affiner la discrétisation pour représenter fidèlement la pièce à usiner. Dans le cas des Triples-Dexels, il est impératif que les dimensions de la base des Dexels (pas suivant les trois axes X, Y et Z) soient très petits. Ceci augmente considérablement le temps de calcul.

1.2. Solution proposé :

Afin de remédier à cette problématique, les auteurs [khaled, hamid] ont proposé et implémenté une approche qui consiste à générer le modèle Triple-Dexel en fonction de la soustraction booléenne qui dépend du temps d'exécution, de la précision et de la mémoire. Afin de minimiser le temps de traitement, une approche intéressante a été proposée et implémentée. Elle consiste à regrouper les triangles du modèle STL dans des ensembles en utilisant la méthode de K-Means. Sa principale limitation est liée à l'initialisation des centres

de Clusters. Chaque initialisation du processus correspond à une solution différente qui peut dans certains cas être très éloignée de la solution optimale. Dans cet ordre d'idées et pour surmonter cette contrainte, ce sujet est proposé.

Le but de ce travail est la conception et le développement d'une approche optimale pour la génération du modèle Triple-Dexels de pièces de formes complexes en répondant au compromis précision et coût. Notre solution se résume en les points suivants :

- Résoudre le problème d'initialisation des centres de Clusters.
- Proposer d'autres méthodes de groupement et les comparer avec les K-Means.
- Générer les modèles en « Dexels » suivant les trois axes X, Y et Z simultanément en proposant une manière générique.
- Résoudre le problème des formes avec de fortes concavités et convexités.
- Comparer l'approche développée avec l'ancienne approche.

2. Architecture générale de l'application :

L'architecture générale de l'application logicielle à mettre au point est composée de sept parties:

1. Récupération du modèle CAO de la pièce en modèle STL.
2. Choisir une méthode de classification du modèle STL.
3. Création d'une grille pour les trois plans XY, XZ et YZ.
4. Identifier chaque centre de cellule de la grille par une droite à une direction.
5. Calculer les intersections entre la droite et les triangles du modèle STL.
6. Détecter les zones matières et le représenter par un Dexel.
7. Intégrer le calcul parallèle pour réduire le temps de calcul.

3. Démarche de l'application :

3.1. Lecture du fichier STL :

La lecture du fichier STL sert à récupérer le modèle de la pièce à discrétiser .Ce modèle est composé d'un ensemble de triangles où chaque triangle est défini par les coordonnées de ses trois (03) sommets et de sa normale unitaire sortante.

Après la lecture du fichier STL, les limites de la pièce sont calculées et le brut (enveloppe) correspondant est défini par sa largeur, sa longueur et sa hauteur (Figure II.1). Les coordonnées de ses huit (08) points extrémums sont données par :

A= minX, minY, minZ	B= minX, maxY, minZ
C=maxX, minY, minZ	D=maxX, maxY, minZ
E= minX, minY, maxZ	F= minX, maxY, maxZ
G= maxX, minY, maxZ	H= maxX, maxY, maxZ

Avec :

- minX (minY, minZ) est la valeur minimale de la coordonnée X (Y, Z) des sommets.
- maxX (maxY, maxZ) est la valeur maximale de la coordonnée X (Y, Z) des sommets.

Les dimensions du brut sont données par :

$$Longueur = C(X_{max}) - A(X_{min})$$

$$Largeur = B(Y_{max}) - A(Y_{min})$$

$$Hauteur = E(Z_{max}) - A(Z_{min})$$

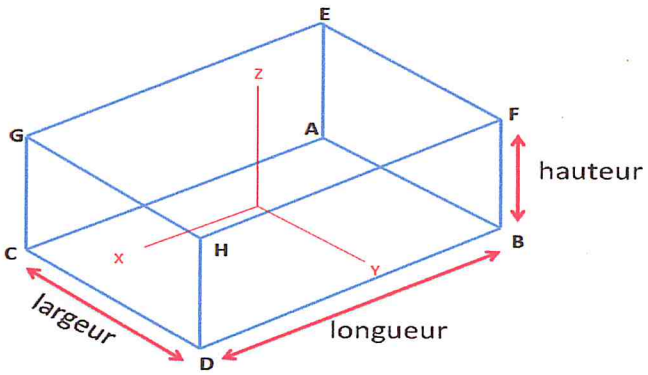


Figure II.1. Brut de la pièce.

3.2. Classification des triangles :

Pour la représentation volumique des pièces de n'importe quelle forme géométrique d'une manière précise et rapide, il est indispensable de grouper (Clustering) des triangles pour faciliter et pour réduire les temps de traitement global de l'approche.

➢ ***Clustering :*** il consiste au groupement d'un ensemble d'objets dans des groupes (Clusters) sous certaines contraintes géométriques (distance, aire, similitude, …etc.). Pour cela, deux méthodes de Clustering sont adoptées à savoir « Cellules » et « Cluster ».

3.2.1. Groupement par « Cellules » :

➢ ***Création des cellules :*** à partir du brut de la pièce à discrétiser, une matrice de « Cellules » « blocs parallélépipédiques » sont créées. Pour cela, le nombre de cellules suivant les trois axes X, Y et Z doivent être spécifiés. A partir de ces données, les dimensions et les coordonnées des points extrémums de toutes les cellules sont calculées (Figure II.2).

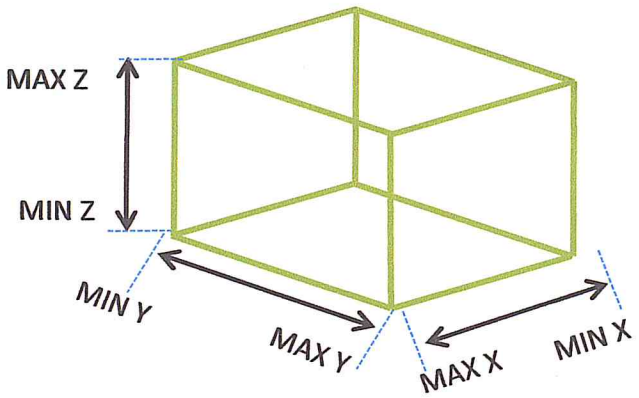


Figure II.2. Dimensions une cellule.

Les dimensions de la cellule (D_X, D_y, D_Z) sont calculées par :

D_X= Longueur du brut / nombre cellules selon X

D_y= Largeur du brut / nombre cellules selon Y

D_Z= Hauteur du brut / nombre cellules selon Z

Les extrémités de la première cellule suivant les trois axes sont données par :

minX_cellule = minX + I * Dx.

maxX_cellule = minX_cellule + Dx.

minY_cellule = minY + J * Dy.

maxY_cellule = minY_cellule + Dy.

minZ_cellule = minZ + k * Dz.

maxZ_cellule = minZ_cellule + D_Z.

Les paramètres I, J et K sont égales à 0 puisque c'est la première cellule. La démarche de création de toutes les cellules est illustrée par l'Algorithme II.1. La Figure II.3 montre une matrice des cellules.

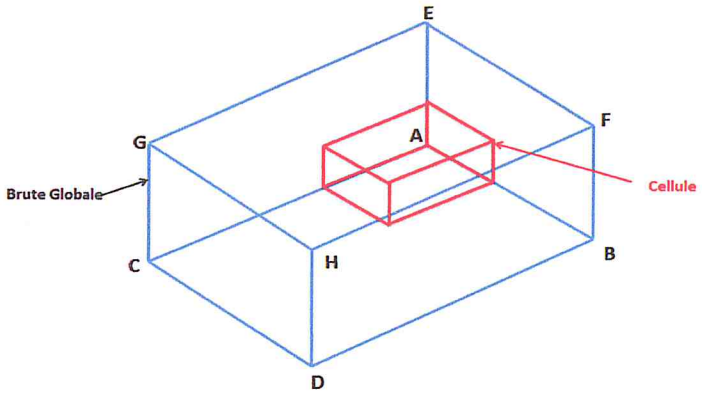


Figure II.3. Création de la première cellule.

Algorithme : Création des cellules
Entrée
Ensemble des extrémités de bute longueur, largeur, hauteur ;
Nombre de cellule de création selon X, Y, Z ;
Sortie
Début

1) D_X= Longueur de Brute / nombre cellule selon X données ;
2) D_y= Largeur de Brute / nombre cellule selon Y ;
3) D_Z= Hauteur de Brute / nombre cellule selon Z ;

Pour I de 0 jusqu'à D_X, **pas 1 Faire**
Pour J de 0 jusqu'à D_y, **pas 1 Faire**
Pour K de 0 jusqu'à Dz , pas 1 Faire

minX_cellule = A + I * D_X ;
maxX_cellule = minX_cellule + D_X ;

minY_cellule = C + J * D_y ;
maxY_cellule = minY_cellule + D_y ;

minZ_cellule =E + k * D_Z ;
maxZ_cellule = minZ_cellule + D_Z ;

Fin Faire
Fin Faire
Fin Faire
Fin

Algorithme II.1. Création des cellules.

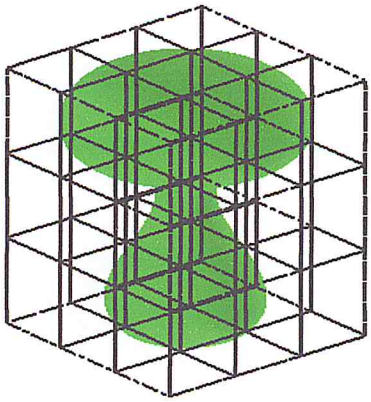


Figure II.4. Matrice des cellules de la pièce.

➢ ***Affectation des triangles aux cellules :*** c'est l'étape de détermination des triangles appartenant entièrement à une cellule donnée. Pour un triangle partiellement contenu dans une cellule, il est automatiquement affecté aux cellules qui se chevauchent avec son enveloppe (Figure II.5).

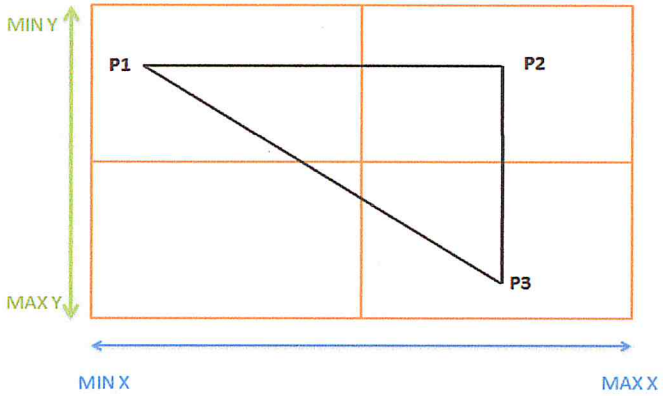


Figure II.5. Affectation d'un triangle en 2D à plusieurs cellules.

Pour cela, les coordonnées minimales et maximales de ses sommets P1, P2 et P3 suivant les axes X, Y et Z sont calculées par :

minX =min ($P1_x$, min ($P2_x$, $P3_x$) maxX =max (P1x, max (P2x, P3x).

minY =min (P1y, min (P2y, P3y) maxX =max (P1y, max (P2y, P3y).

minZ =min (P1z, min (P2z, P3z) maxZ =max ($P1_z$, max ($P2_z$, $P3_z$).

Par la suite, le triangle est affecté à chaque cellule qui a des extrémités se trouvant dans ce nouveau intervalle ([minX, maxX] , [minY, maxY] , [minZ, maxZ]).

L'Algorithme II.2 illustre la procédure d'affectation des triangles aux cellules.

Algorithme : Affectation triangle cellule

Entrée

Tableau_triangle = Tableau contient ensemble des Triangle de la pièce et chaque triangle contient des sommets P1, P2, P3 ;

A= X_{min} représente le point minimum de brute selon axe X;

C= Y_{min} représente le point minimum de brute selon axe Y;

E= Z_{min} représente le point minimum de brute selon axe Z;

Tableau_triangle_de_la_cellules = Tableau d'une cellule [i] [j] [k] contient ensemble des Triangles qui sont affecter à la cellule (dans 1er exécution le tableau est vide)

Sortie

Début

Lire Nombre de cellules effectué par utilisateur (nbr_cellule_X, nbr_cellule_Y, nbr_cellule_Z) ;

Récupérer les pas_X, pas_Y, pas_Z de la cellule ;

Pour T de 0 jusqu'à la fin de Tableau_triangle **pas** 1 **Faire**

```
i1 = (Partie entier) ((P1.get_X () - A) /  pas_X) ;
If (i1 == nbr_cellule_X) i1=i1-1;

j1= (Partie entier)  ((P1.get_Y () - C) / pas_Y);
If (j1==nbr_cellule_Y) j1=j1-1;

k1= (Partie entier)  ((P1.get_Z ()  - E)/ pas_Z);
If (k1==nbr_cellule_Z) k1=k1-1;

i2 = (Partie entier) ((P2.get_X () - A) /  pas_X) ;
If (i1 == nbr_cellule_X) i2=i2-1;

j2= (Partie entier)  ((P2.get_Y () - C) / pas_Y);
If (j2==nbr_cellule_Y) j2=j2-1;

k2= (Partie entier)  ((P2.get_Z ()  - E)/ pas_Z);
If (k2==nbr_cellule_Z) k2=k2-1;

i3 = (Partie entier) ((P3.get_X () - A) /  pas_X) ;
If (i3 == nbr_cellule_X) i3=i3-1;

J3= (Partie entier)  ((P3.get_Y () - C) / pas_Y);
If (j3==nbr_cellule_Y) j3=j3-1;

K3= (Partie entier)  ((P3.get_Z ()  - E)/ pas_Z);
If (k3==nbr_cellule_Z) k3=k3-1;
```

```
imin=min(i1,min(i2,i3));      imax=max(i1,max(i2,i3));
jmin=min(j1,min(j2,j3));      jmax=max(j1,max(j2,j3));
kmin=min(k1,min(k2,k3));      kmax=max(k1,max(k2,k3));

Pour i de imin jusqu'à imax pas 1 Faire
  Pour j de jmin jusqu'à jmax pas 1 Faire
    Pour k de kmin jusqu'à kmax pas 1 Faire
Tableau_celulul [i] [j] [k].Tableau_triangle_de_la_cellule. Ajouter(Tableau_triangle[T].index)
Fin Faire
```

Algorithme II.2. Affectation des triangles aux cellules.

3.2.2. Groupement par K_Means :

Pour le groupent par K_Means, l'utilisateur doit spécifier les paramètres suivants :

- Critères d'arrêt : stabilité et nombre d'itérations maximales.
- Nombre de Clusters.
- Variantes de K-Means.

Les différentes variants de K-Means considérées sont : K-Means initial, K-Means Global, Fast Global K-Means et Incremental K-Means.

➢ ***K Means Initial :*** c'est un simple algorithme de classification automatique des données. L'idée principale est de choisir aléatoirement un ensemble de centres fixés a priori (centres des Clusters) et de chercher itérativement la partition optimale (Clusters » par l'affectation de chaque individu au centre le plus proche. A la fin, la moyenne de chaque groupe est calculée et va constituer les nouveaux centres. Ce processus itératif est répété jusqu'à sa convergence [20]. Son principe est donné par l'Algorithme II.3.

Algorithme : K_Means initial
Entrée
Ensemble de N données, noté par x
Nombre de groupes souhaité, noté par k
Sortie
Une partition de K groupes $\{C_1, C_2,...C_k\}$
Début
1) Initialisation aléatoire des centres Ck ;
Répéter
2) Affectation : générer une nouvelle partition en assignant chaque objet au groupe dont le centre est le plus proche ;

$$x_i \in C_k \; si \; \forall_j \; |x_i - \mu_k| = \min|x_i - \mu_k|$$

Avec μk le centre de la classe K ;
3) Représentation : Calculer les centres associe à la nouvelle partition ;
Jusqu'à convergence de l'algorithme vers une partition stable ;

Fin.

Algorithme II.3. K-Means initial [20].

➢ ***K-Means Global :*** c'est une solution au problème d'initialisation du K-Means [21]. Elle est basée sur les données et vise à atteindre une solution globalement optimale. Elle consiste à effectuer un Clustering incrémental et à ajouter dynamiquement un nouveau centre suivi par l'application du K-Means jusqu'à la convergence. Les centres sont choisis un par un de la façon suivante : le premier centre est le centre de gravité de l'ensemble des données, les autres centres sont tirés de l'ensemble de données où chaque donnée est une candidate pour devenir un centre. Cette dernière sera testée avec le reste de l'ensemble. Le meilleur candidat est celui qui minimise la fonction objectif. L'Algorithme II.4. illustre son principe.

➢ ***Fast Global K-Means:*** il permet d'accélérer K-Means Global [21]. Cette stratégie garde la même philosophie que la précédente (toutes les données peuvent être candidates pour devenir un centre), mais évite d'affecter les données aux centres les plus proches (centres déjà existant en plus du centre candidat) et de calculer l'erreur quadratique. Cette erreur quadratique diminue en fonction du nombre de centres par un taux b_n et le nouveau centre sera le candidat qui maximise ce taux. L'Algorithme II.5. illustre son principe.

Algorithme 2 : Global K-Means
Entrée
Ensemble de N données, notés par x ;
Nombre de groupes souhaiter, noté par k ;
Sortie
Une partition de K groupes $\{C_1, C_2,...C_k\}$
Début
1) C_1 = Centre de gravité de l'ensemble des données ;
Répéter
2) Initialiser les centres i-1 par le résultat de l'étape précédente ;
3) Trouver l'iéme centre ;
Pour chaque donnée x faire
3.1) Considère x comme étant le iéme centre ;
3.2) Affecter les données aux plus proche centre ;
3.3) Calculer l'erreur quadratique pour Ci=x ;
Fin faire
3.4) Garder le centre Ci = x qui minimise l'erreur quadratique ;
4) Appliquer le k-means jusqu'à la convergence ;
Jusqu'à obtenir une partition en k groupes ;
Fin.

Algorithme II.4. Global K-Means [21].

Algorithme 3: Fast Global K-Means
Entrée
Ensemble de N données, notés par x ;
Nombre de groupes souhaiter, noté par k ;
Sortie
Une partition de K groupes $\{C1, C2,...Ck\}$
Début
4) $C1$ = Centre de gravité de l'ensemble des données ;
Répéter
2) Initialiser les centres i-1 par le résultat de l'étape précédente ;
3) Trouver l'iéme centre ;
Pour chaque donnée x **faire**
3.1) Considère x comme étant le $i^{éme}$ centre ;
3.2) Calculer b_n pour Ci=x tel que

$$b_n = \sum_{i=1}^{N} \max(d_{k-1}^{i} - \|x - x\|^2, 0)$$

Avec d_{k-1}^{i} la distance entre xi et son plus proche centre parmi les k-1 centres.
Fin faire
Jusqu'à obtenir une partition en k groupes ;
4) Appliquer le k-means jusqu'à la convergence ;
Fin.

Algorithme II.5. Fast Global K-Means [21].

➢ ***Incremental K-Means :*** cette approche incrémentale de classification [20] est similaire à celle Globale K-Means et la différence réside dans les points suivants :

- Le nombre de points initiaux sont deux au lieu d'un seul dans Global K-Means.
- La recherche du nouveau centre se limite à la recherche de l'élément le mal classé au lieu de tester toutes les données.

L'Algorithme II.6. illustre son principe.

Algorithme 4 : K_Means Incrémental
Entrée
Ensemble de N données, notés par x ;
Nombre de groupes souhaiter, noté par k ;
Sortie
Une partition de K groupes $\{C1, C2,...Ck\}$
Début
$C_1 = x_1$, $C_2 = x_2$ $avec$ $d(x_1, x_2) = \max_{\substack{i,j \in [1..N] \\ i \neq j}} (d(x_j, x_i))$
Répéter
2) Initialiser les centres i-1 par le résultat de l'étape précédente ;
3) Trouver l'iéme centre C_i :

$$C_i = x \; tal \; que \; x = \max_{i \in [i,N]} (d^i_{k-1})$$

Avec d^i_{k-1} la distance entre xi et son plus proche centre parmi les k-1 centres.

4) Appliquer le k-means jusqu'à la convergence ;
Jusqu'à obtenir une partition en k groupes ;
Fin

Algorithme II.6. Incremental K-Means[20].

3.3. Création des Grilles :

Cette étape consiste à générer trois grilles de cellules dans les trois plans XY, XZ et YZ. La première grille est générée dans un plan parallèle au plan XY et passant par la face horizontale du brut de coordonnée ZMIN. La deuxième grille est générée dans un plan parallèle au plan XZ et passant par la face verticale du brut de coordonnée YMIN. La troisième grille est générée dans un plan parallèle au plan YZ et passant par la face verticale du brut de coordonnée XMIN [3] (Figure II.6). La création de ces grilles nécessite la spécification des pas de discrétisation introduits par l'utilisateur Pas_X, Pas_Y et Pas_Z suivant les trois axes X, Y et Z. La création de la grille ne doit pas dépasser les extrémités du brut. A partir des dimensions minimales du brut, les pas des cellules suivant les trois axes X, Y et Z sont calculés par :

nombre_cellul_X = Longueur / Pas_X

nombre_cellul_Y = Largeur / Pas_Y

nombre_cellul_Z = Hauteur / Pas_Z

Les nombres de cellules doivent être des entiers. Ils sont calculés par :

nouveau_nombre_cellul_X = (Partie entier) nombre_cellul_X +1

nouveau_nombre_cellul_Y = (Partie entier) nombre_cellul_Y +1

nouveau_nombre_cellul_Z = (Partie entier) nombre_cellul_Z +1

Les nouveaux pas sont donnés par :

Nv_pas_X= Longueur / nouveau_nombre_cellul_X

Nv_pas_Y= Largeur / nouveau_nombre_cellul_Y

Nv_pas_Z= Hauteur / nouveau_nombre_cellul_Z

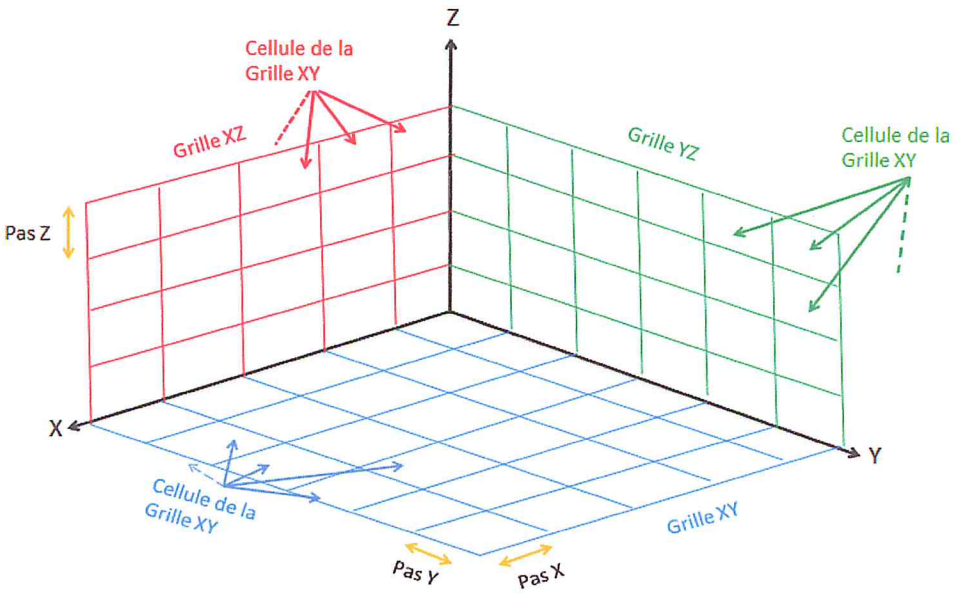


Figure II.6. Création des trois grilles de cellules.

Ces cellules représentent les bases des Dexels à créer. Le nombre de cellules suivant chaque axe dépend des pas de discrétisation et des dimensions du brut. Chaque cellule est caractérisée pas ses points limites, son centre et son axe qui est perpendiculaire au plan de la cellule et dirigé vers l'intérieur du brut. Ces paramètres sont calculés en fonction des dimensions du brut et des pas de discrétisation. La Figure II.7 représente un exemple de dimension d'une cellule sur le plan XY et la droite orientée vers la direction Z. L'Algorithme 7 permet d'illustrer le principe de création de la grille avec les droites.

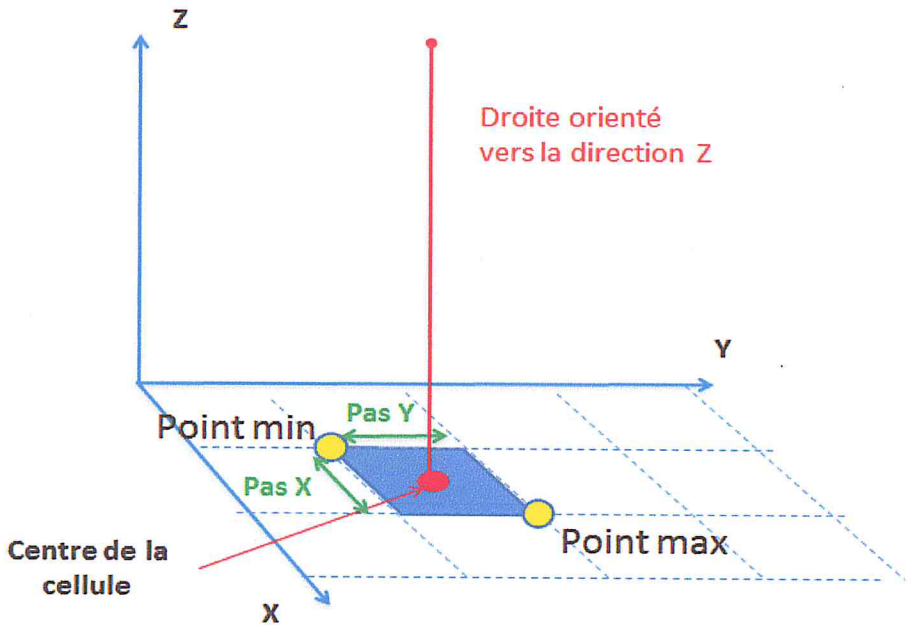


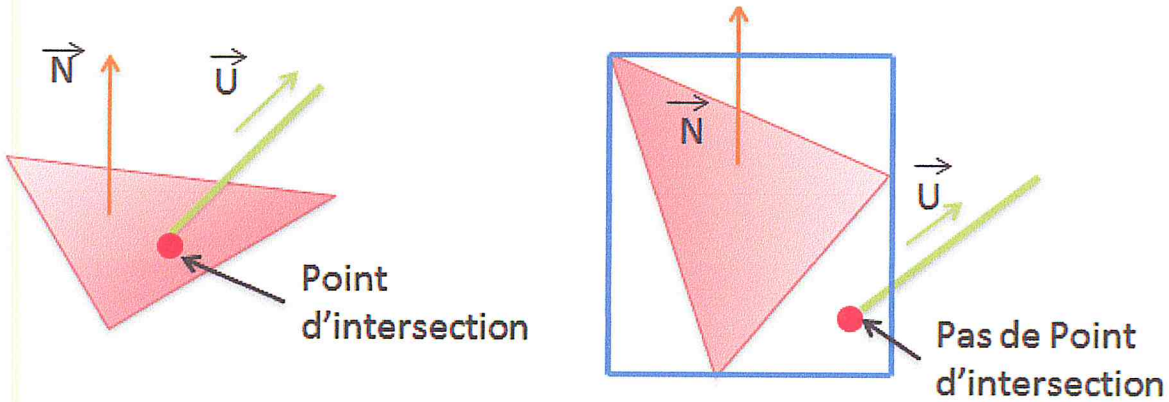
Figure II.7. Paramètres d'une cellule.

3.4. Calcul des points d'intersection :

Pour chaque droite de la grille créée, les points d'intersection de cette droite avec les triangles du modèle STL. Pour cela, deux cas sont à considérer.

3.4.1. Non Colinéaire:

La normale du triangle et la droite ne sont pas perpendiculaires. Dans ce cas, la droite et le triangle ne sont pas parallèles. Par conséquent, il existe un seul point d'intersection entre la droite de vecteur directeur et le plan passant par le triangle de vecteur normal (Figure II.8.a). Le point calculé n'est valide que s'il appartient au triangle (Figure II.8.b) [3].



a. Intersection triangle et droite en 3D. b. Appartenance d'un point au triangle

Figure II.8. Intersection entre droite et triangle non colinéaires.

```
Algorithme : Création grille
Entrée
        Nouveau calcule des pas de la cellule Nv_pas_X, Nv_pas_Y, Nv_pas_Z ;
        Le nombre de cellule nv_nombre_cellul_X, nv_nombre_cellul_Y, nv_nombre_cellul_Z
        Récupérer les limite de Brute minX, maxX, minY, maxY, minZ, maxZ ;
Sortie
Début // création une grille selon XY
        Pour i de 0 jusqu'à nv_nombre_cellul_X Faire
          Pour j de 0 jusqu'à nv_nombre_cellul_Y Faire

              min1_cellule = minX + i* Nv_pas_X;
              max1_cellule = min1_cellule + Nv_pas_X;

              min2_cellule=j* Nv_pas_Y + minY;
              max2_cellule= min2_cellule + Nv_pas_Y;

              Calculer le centre de la cellule ;
              Cree une droite qui caractériser par deux point :
                (Point 1(Centre de hauteur minZ), Point 2 (centre de hauteur maxZ) ;
          Fin Faire
        Fin Faire
        // Création une grille selon XZ
        Pour i de 0 jusqu'à nv_nombre_cellul_X Faire
          Pour j de 0 jusqu'à nv_nombre_cellul_Z Faire

              min1_cellule = minX + i* Nv_pas_X;
              max1_cellule = min1_cellule + Nv_pas_X;

              min2_cellule=j* Nv_pas_Z + minZ;
              max2_cellule= min2_cellule + Nv_pas_Z;

              Calculer le centre de la cellule ;
              Cree une droite qui caractériser par deux point :
                (Point 1(Centre de hauteur minY), Point 2 (centre de hauteur maxY)
          Fin Faire
        Fin Faire
        // Création une grille selon YZ
        Pour i de 0 jusqu'à nv_nombre_cellul_Y Faire
          Pour j de 0 jusqu'à nv_nombre_cellul_Z Faire

              min1_cellule = minY + i* Nv_pas_Y;
              max1_cellule = min1_cellule + Nv_pas_Y;

              min2_cellule=j* Nv_pas_Z + minZ;
              max2_cellule= min2_cellule + Nv_pas_Z;

              Calculer le centre de la cellule ;
              Cree une droite qui caractériser par deux point :
                (Point 1(Centre de hauteur minX), Point 2 (centre de hauteur maxX)
          Fin Faire
        Fin Faire
Fin.
```

Algorithme II.7. Création de la grille.

D'abord la première partie, le point d'intersection est calculé de la façon suivante :

La droite est définie par ses deux points, un point début représenté par le point A et un point fin représenté par B (Figure II.9). Le point d'intersection M entre la droite et le triangle est calculé à partir des étapes suivantes :

$$\overrightarrow{AM} = \alpha\overrightarrow{AB}$$

La projection de cette équation vectorielle sur les trois axes donne :

$$X_M = \alpha\,(X_B - X_A) + X_A$$

$$Y_M = \alpha\,(Y_B - Y_A) + Y_A$$

$$Z_M = \alpha\,(Z_B - Z_A) + Z_A$$

Le point d'intersection M n'est pas connu par ses coordonnées puisque α n'est pas encore calculé. Pour cela, l'équation du plan de triangle est utilisée :

$$AX + BY + CZ + D = 0$$

En remplaçant les coordonnées du point M dans l'équation du plan :

$$A\,(\alpha\,(X_B - X_A) + X_A) + B\,(Y_M = \alpha\,(Y_B - Y_A) + Y_A) + C\,(\alpha\,(Z_B - Z_A) + Z_A) + D = 0$$

La valeur de α est donnée par :

$$\alpha = \frac{-(A\,X_A + B\,Y_A + C\,Z_A + D)}{A(X_B - X_A) + B(Y_B - Y_A) + C(Z_B - Z_A)}$$

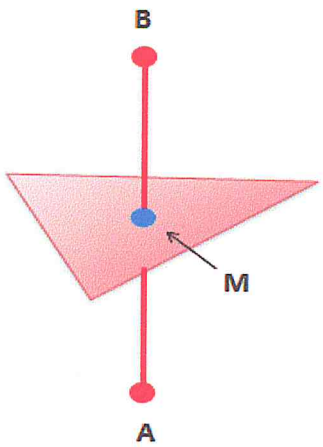


Figure II.9. Intersection entre droite et triangle.

Par la suite, les cordonnées du point d'inertsection M (X_M, Y_M, Z_M) sont calculées.

Dans la deuxième partie, il faut vérifier l'appartenance du point d'intersection M au triangle. Les deux premières vérifications sont :

- Vérifier si le point d'intersection est confondu avec un des sommets du triangle.
- Vérifier si le point d'intersection appartient aux côtés du triangle.

Dans le cas contraire, il faut vérifier l'appartenance du point d'intersection à l'intérieur du triangle. A cet effet, plusieurs méthodes sont utilisées.

➢ ***Méthode 1 :*** le point d'intersection M appartient au triangle si les trois conditions suivantes sont vérifiées (Figure II.10).

$$\begin{cases} (\overrightarrow{P_1P_2} \wedge \overrightarrow{P_1M}).(\overrightarrow{P_1M} \wedge \overrightarrow{P_1P_3}) \geq 0 \\ (\overrightarrow{P_2P_1} \wedge \overrightarrow{P_2M}).(\overrightarrow{P_2M} \wedge \overrightarrow{P_2P_3}) \geq 0 \\ (\overrightarrow{P_3P_1} \wedge \overrightarrow{P_3M}).(\overrightarrow{P_3M} \wedge \overrightarrow{P_3P_2}) \geq 0 \end{cases}$$

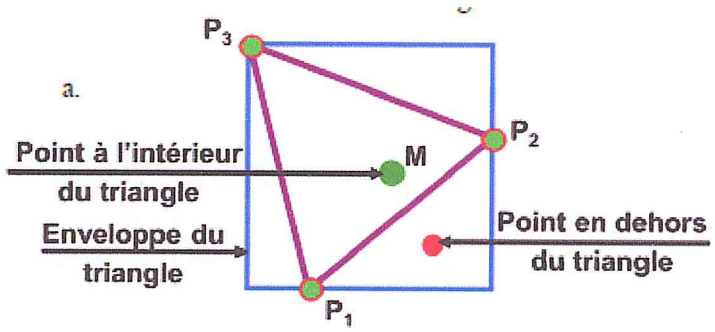


Figure II.10. Appartenance d'un point au triangle (1ère méthode) [7].

➢ ***Méthode 2 :*** lorsqu'on considère un repère (O, $\vec{U},\vec{V}$) du plan (Figure II.11), le couple de vecteurs $\vec{U}$, et $\vec{V}$, notée ($\vec{U}$, $\vec{V}$) est une base du plan de triangle [22]. Les coordonnées U et V dans ce repère sont données par :

$$U = \frac{\overrightarrow{AB}.\overrightarrow{AB} * \overrightarrow{AM}.\overrightarrow{AC} - \overrightarrow{AB}.\overrightarrow{AC} * \overrightarrow{AM}.\overrightarrow{AB}}{\overrightarrow{AC}.\overrightarrow{AC} * \overrightarrow{AB}.\overrightarrow{AB} - \overrightarrow{AC}.\overrightarrow{AB} * \overrightarrow{AB}.\overrightarrow{AC}}$$

$$V = \frac{\overrightarrow{AC}.\overrightarrow{AC} * \overrightarrow{AM}.\overrightarrow{AB} - \overrightarrow{AC}.\overrightarrow{AB} * \overrightarrow{AM}.\overrightarrow{AC}}{\overrightarrow{AC}.\overrightarrow{AC} * \overrightarrow{AB}.\overrightarrow{AB} - \overrightarrow{AC}.\overrightarrow{AB} * \overrightarrow{AB}.\overrightarrow{AC}}$$

Le point d'intersection M appartient au triangle si les trois conditions suivantes sont vérifiées :

$$0 \leq U \leq 1$$

$$0 \leq V \leq 1$$

$$0 \leq U+V \leq 1$$

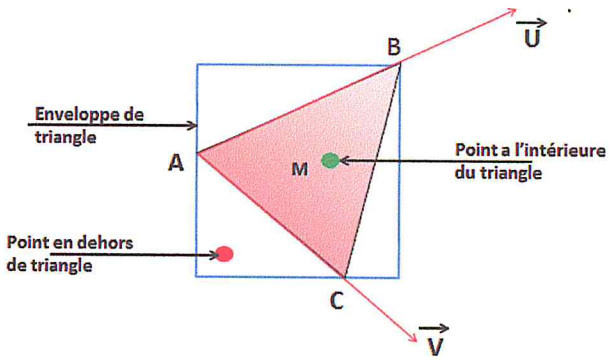


Figure II.11. Appartenance d'un point à un triangle (2ème méthode).

➢ ***Méthode 3 (somme des aires) :*** le principe de cette méthode est de calculer la somme des aires des trois triangles et la comparer à l'aire totale du triangle (Figure II.12). L'aire totale S du triangle et les aires S1, S2 et S3 des trois triangles sont données par :

$$S = \frac{1}{2} \| (A - B) * (A - C) \|$$

$$S1 = \frac{1}{2} \| (B - P) * (C - P) \|$$

$$S2 = \frac{1}{2} \| (C - P) * (A - P) \|$$

$$S3 = \frac{1}{2} \| (A - P) * (B - P) \|$$

Le point d'intersection M appartient au triangle si la condition suivante est vérifiée :

$$S1 + S2 + S3 = S$$

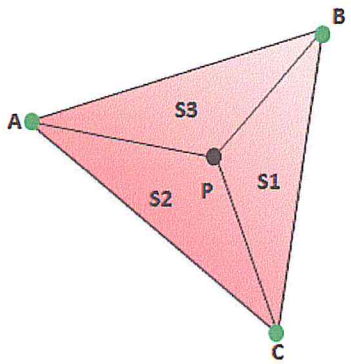


Figure II.12. Appartenance d'un point à un triangle (3ème méthode).

➢ ***Méthode 4 :*** consiste à calculer les paramètres α, β et δ pour vérifier l'appartenance (Figure II.12). Comme pour la méthode 3, les aires S, S1, S2 et S3 sont calculées. Par la suite, les paramètres α, β et δ sont calculés par :

$$\alpha = \frac{S1}{S} \qquad \beta = \frac{S2}{S} \qquad \delta = \frac{S3}{S}$$

Le point d'intersection M appartient au triangle si les conditions suivantes sont vérifiées :

$$0 \leq \alpha \leq 1 \qquad 0 \leq \beta \leq 1 \qquad 0 \leq \delta \leq 1 \qquad 0 \leq \alpha + \beta + \delta \leq 1$$

➢ ***Méthode 5 :*** cette méthode passe par la résolution du système d'équation suivant en fonction de α, β et δ en utilisant la méthode de Gauss [23] (Figure II.13) :

$$Ax * \alpha + Bx * \beta + Cx * \delta = Px.$$

$$Ay * \alpha + By * \beta + Cy * \delta = Py.$$

$$Az * \alpha + Bz * \beta + Cz * \delta = Pz.$$

$$\begin{bmatrix} Ax & Bx & Cx & Px \\ Ay & By & Cy & Py \\ Az & Bz & Cz & Pz \end{bmatrix}$$

Le point d'intersection M appartient au triangle si les conditions suivantes sont vérifiées :

$$0 \leq \alpha \leq 1 \qquad 0 \leq \beta \leq 1 \qquad 0 \leq \delta \leq 1 \qquad \alpha + \beta + \delta = 1$$

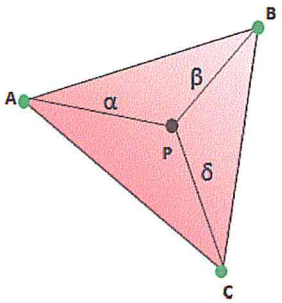


Figure II.13. Appartenance d'un point à un triangle (4ème méthode).

➢ ***Méthode 6 :*** cette méthode permette de crée une demi-droite qui est parallèle à un segment P1P2 puis on calcule s'il existe une intersection avec les deux autres segments comme défini dans la Figure II.14 :

Pour déterminer les points d'intersection on doit utiliser la méthode de cramer [24] tel que :
Exemple de calcule avec le segment P2P3 :

$$\overrightarrow{P2D} = \alpha\, \overrightarrow{P2P3} \iff \overrightarrow{MD} = \beta\, \overrightarrow{MC}$$

Après le calcule avec la méthode de cramer on déduit α et β, pour valider le point M. il s'agit que $\beta >= 0$ et $0 <= \alpha < 1$
Après avoir déterminé le même calcule avec segment P1P3, on vérifie s'il y a un point d'intersection dans les deux segments ce qui explique que l'appartenance n'est pas vérifiée,

sinon il existe un point d'intersection dans un seul segment cela veut dire que l'appartenance est vérifié.

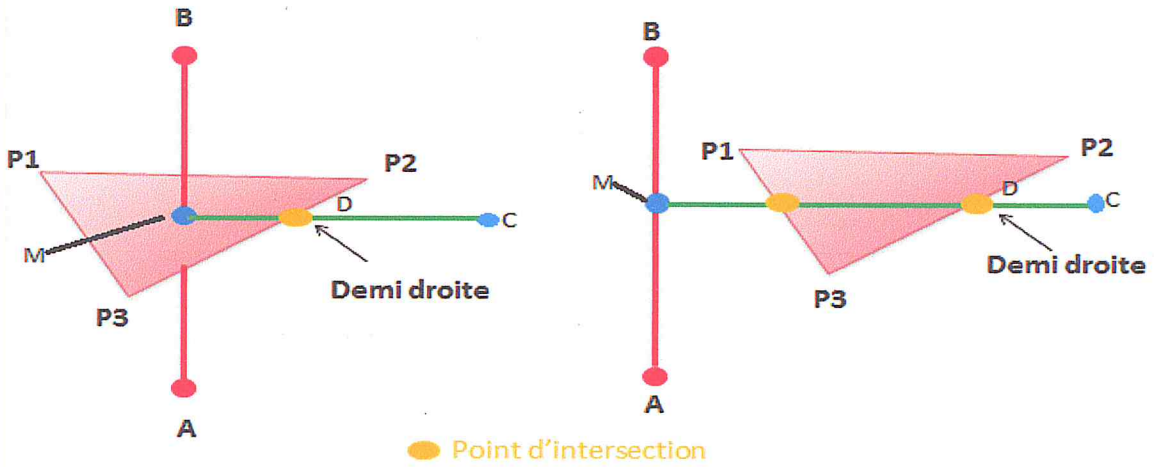


Figure II.14. Appartenance d'un point à un triangle (5ème méthode).

3.4.2. Colinéaire:

La normale du triangle et la droite sont perpendiculaires [3]. Autrement dit, la droite et le triangle sont parallèles. Dans ce cas, la droite et le triangle peuvent être disjoints ou coïncidents.

- S'ils sont disjoints, alors pas de points d'intersection.
- S'ils sont coïncidents, alors le calcul des points d'intersection revient à calculer les points d'intersection entre la droite de la cellule et les trois droites passant par les segments du triangle.

Les différents cas possibles d'intersection entre une droite et un triangle sont illustrés par la Figure II.15.

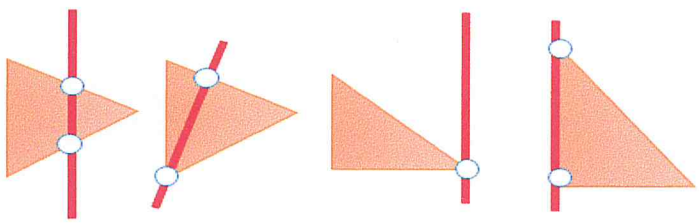


Figure II.15. Différentes configurations d'intersection entre une droite et un triangle.

Pour vérifier que la droite passe par un triangle il faut:

- Vérifier que la droite appartient au plan du triangle.
- Calculer le point d'intersection entre la droite et chaque segment du triangle (Figure II.16) en utilisant la méthode de Cramer [24].

$$\overrightarrow{P1M} = \alpha \overrightarrow{P1P2} \iff \overrightarrow{AM} = \beta \overrightarrow{AB}$$

$$\overrightarrow{P1M} = \alpha \overrightarrow{P1P3} \iff \overrightarrow{AM} = \beta \overrightarrow{AB}$$

$$\overrightarrow{P2M} = \alpha \overrightarrow{P2P3} \iff \overrightarrow{AM} = \beta \overrightarrow{AB}$$

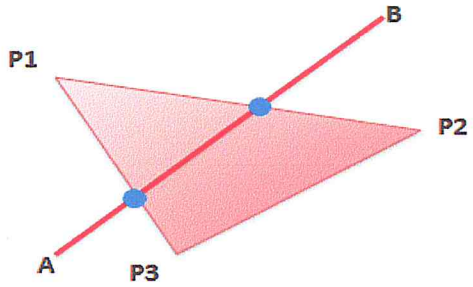


Figure II.16. Intersection droite avec segment de triangle.

Après les calculs, le point d'intersection est validé si la condition suivante est vérifiée :

$$0 <= \alpha <= 1$$

3.5. Création des Dexels :

La création des Dexels passe par l'identification des types de segments de la droite (matière ou vide) (Figure II.17). Le processus de création des Dexels passe par les étapes suivantes :

- Tri de la liste des points d'intersection suivant le sens de la droite de la cellule.
- Initialisation de liste des segments de chaque droite qui contient des points d'intersection triés et création des segments en prenant les points deux à deux dans le sens du tri.
- Identification du type de chaque segment

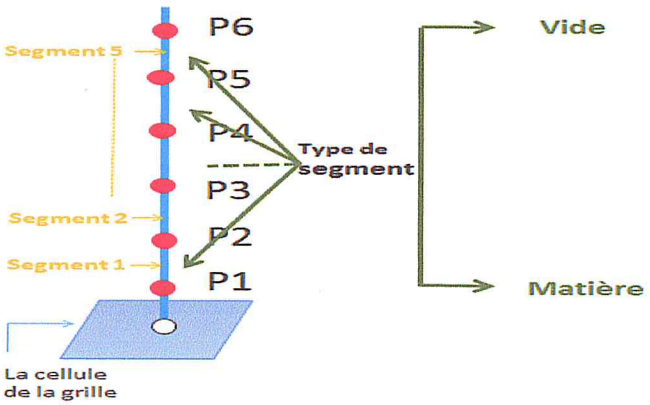


Figure II.17. Identification du type du segment.

L'Algorithme II.8 illustre comment le type d'un segment est identifié.

Algorithme : identifier le type de segment de chaque droite

Début

If (la droite ne passe ni par sommet ni par segment)
pour i de 0 **jusqu'à** la fin du tableau point d'intersection **pas** 2 **Faire**
- Créer un segment en matière [i] [i+1] ;

Fin Faire
Else (la droite passe par un sommet ou segment)
pour j **de** 0 **jusqu'à** la fin du tableau point d'intersection **pas** 1 **Faire**
If (le point d'intersection [j] et le point d'intersection [j+1] appartient à le même triangle)
- Créer un segment en matière [j] [j+1]

Else (point d'intersection [j] et point d'intersection [j+1] n'appartient pas à le même triangle)
- Calculer le centre entre point intersection [j] et point intersection [j+1]
- Créer une demi droite passe par le centre qu'on a calculé à n'importe quelle direction
- Nbr = On calcule le nombre de point d'intersection de demi droite avec le modèle STL
  If (Nbr modulo 2 == 1)
  - Créer un segment en matière [j] [j+1]

  Fin if

Fin if
Fin Faire
FIN

Algorithme II.8. Identification du type des segments d'une droite.

Un Dexel est un parallélépipède caractérisé par sa section représentée par sa base rectangulaire de la forme de la cellule défini par le pas selon X, le pas selon Y et le pas selon Z. Sa hauteur est représentée par la longueur du segment de type « matière ». Donc, les segments de types « matière » forment les Dexels de la pièce dans la direction de la droite de la cellule (Figure II.18). A partir de la section et de la longueur d'un Dexel, le volume du Dexel est calculé. Le volume total de l'objet est la somme des volumes de tous les Dexels. Selon [25], la création des Dexels dans les directions X et Y est réalisée en suivant le même processus que pour la direction Z (Figure II.18). Le modèle de l'objet en Triple-Dexels est obtenu en combinant les Dexels créés dans les trois directions.

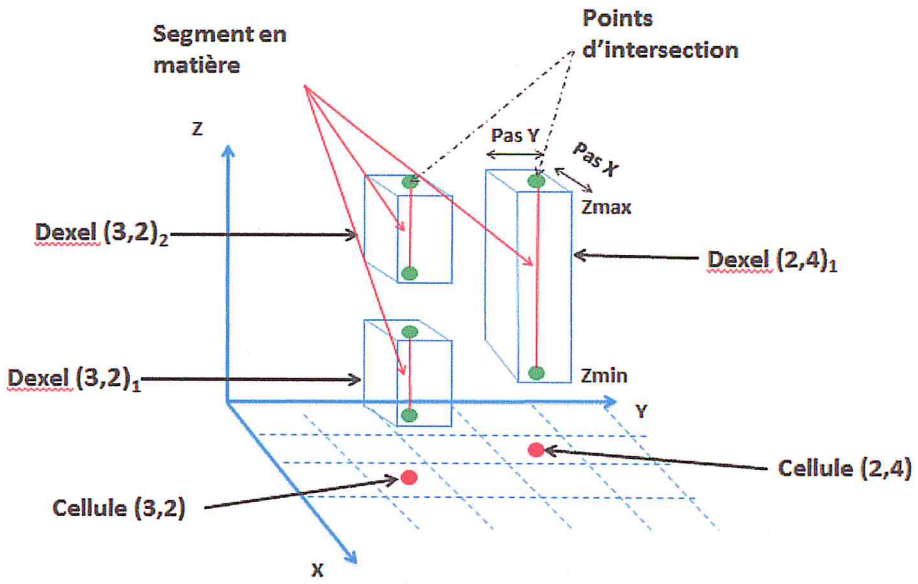


Figure II.18. Création des Dexels selon l'axe Z.

4. Modélisation de l'application avec l'UML :

Chaque application et chaque système avant d'être réalisé doit passer par une étape de conception avec une méthode ou bien un langage de modélisation. Cette étape permet de décrire les fonctionnalités du système, son comportement et tout le détail nécessaire pour la réalisation de ce système et son déroulement. Le Langage de Modélisation Unifié « UML » (Unified Modeling Language), est un langage de modélisation graphique conçu pour fournir une méthode normalisée pour visualiser la conception d'un système. Il est couramment utilisé en développement logiciel et en conception orientée objet [27]. L'UML :

- Offre un standard de modélisation pour représenter l'architecture logicielle.
- Facilite l'analyse, la compréhension et réduit la complexité d'un système.
- Pense objet dès le départ.

Dans notre projet, nous avons utilisé les diagrammes suivants :

- Diagramme de cas d'utilisation.
- Diagramme de classe.

4.1. Diagrammes de cas d'utilisation :

Les cas d'utilisations permettent de recueillir, d'analyser et d'organiser les besoins et de recenser les grandes fonctionnalités du système. Les figures suivantes montrent les différents diagrammes des différentes tâches.

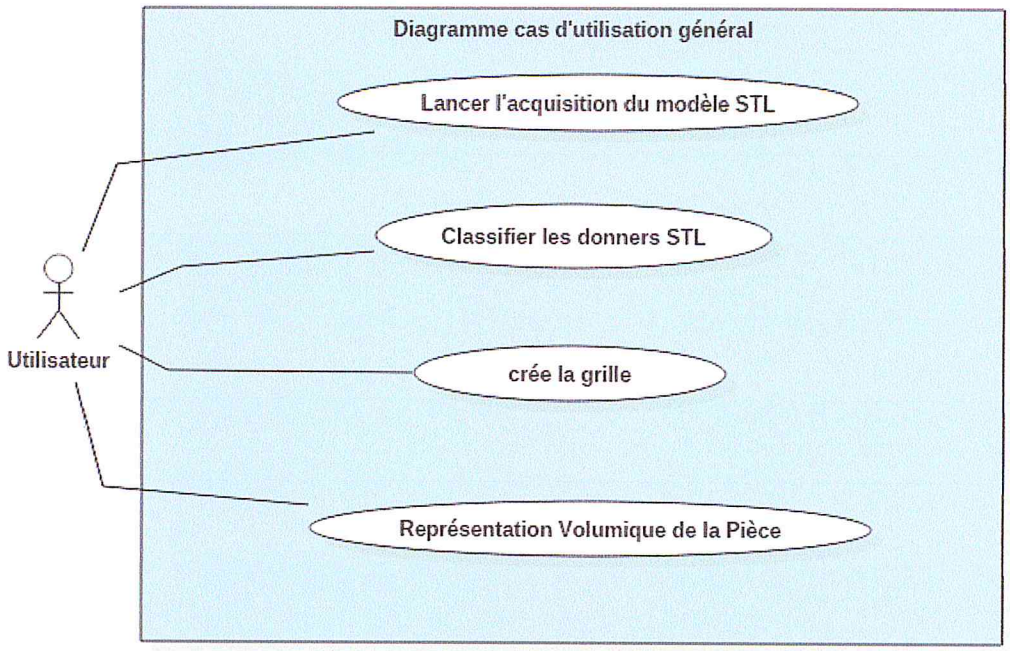


Figure II.19. Diagramme cas d'utilisation générale.

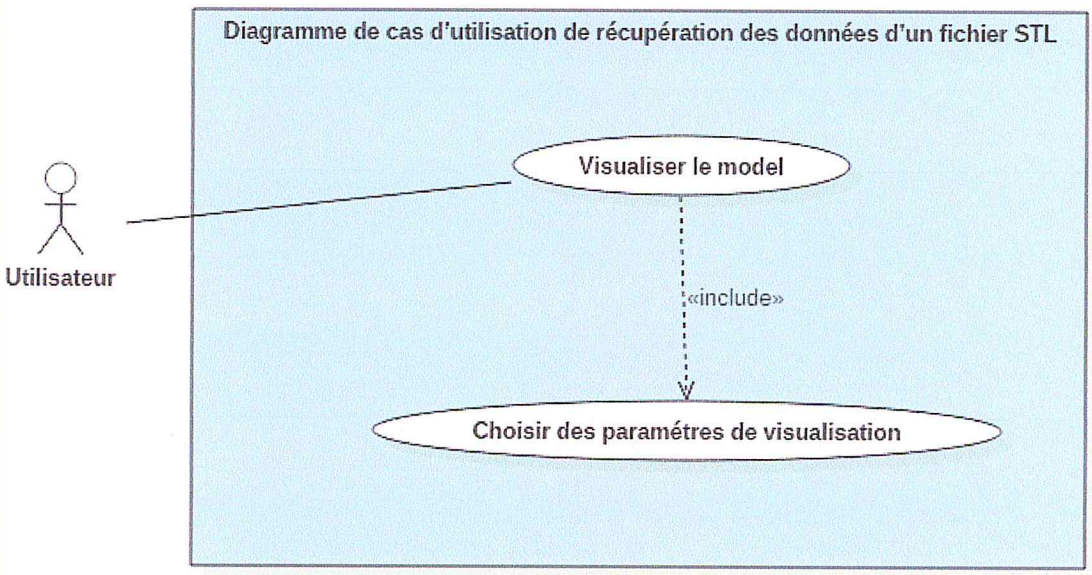


Figure II.20. Diagramme de cas d'utilisation récupération des données d'un fichier STL.

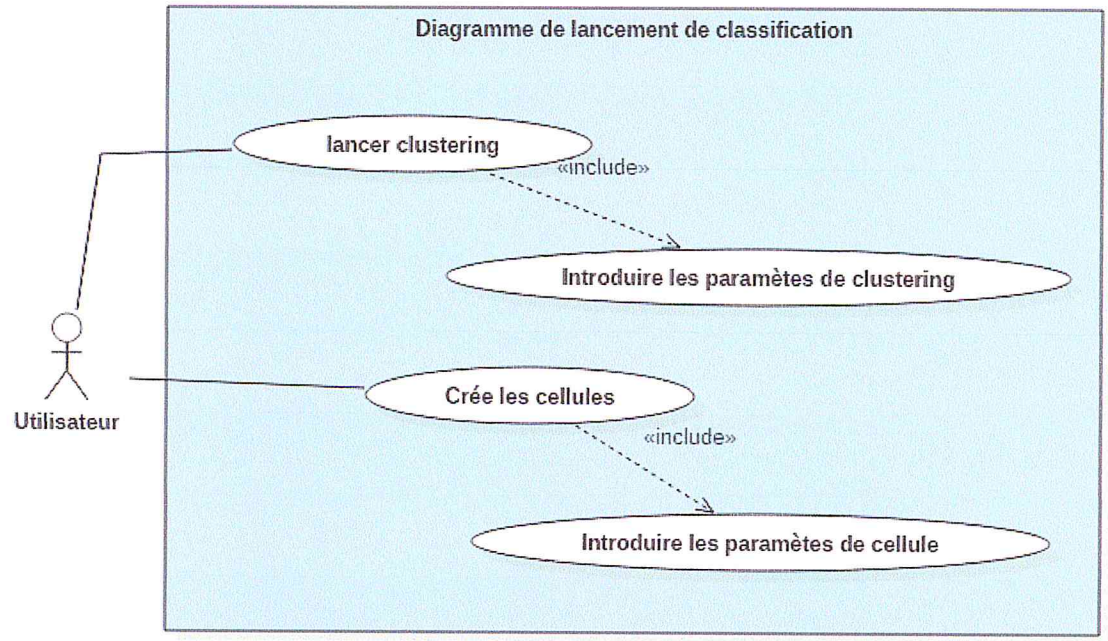


Figure II.21. Diagramme de lancement de la classification.

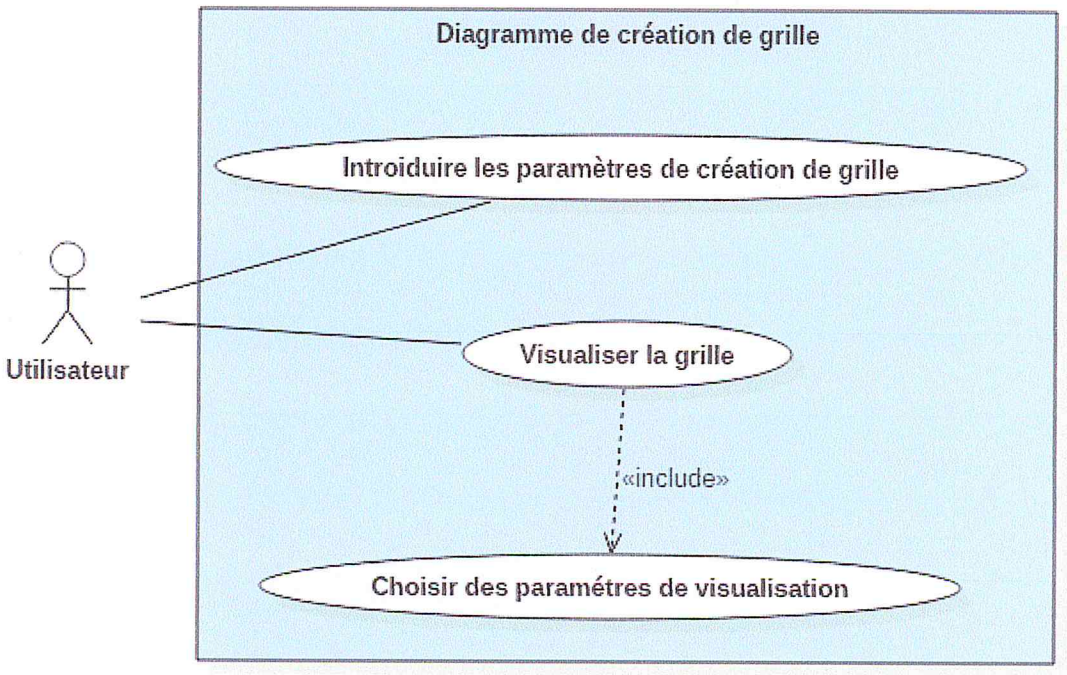


Figure II.22. Diagramme de création de la grille.

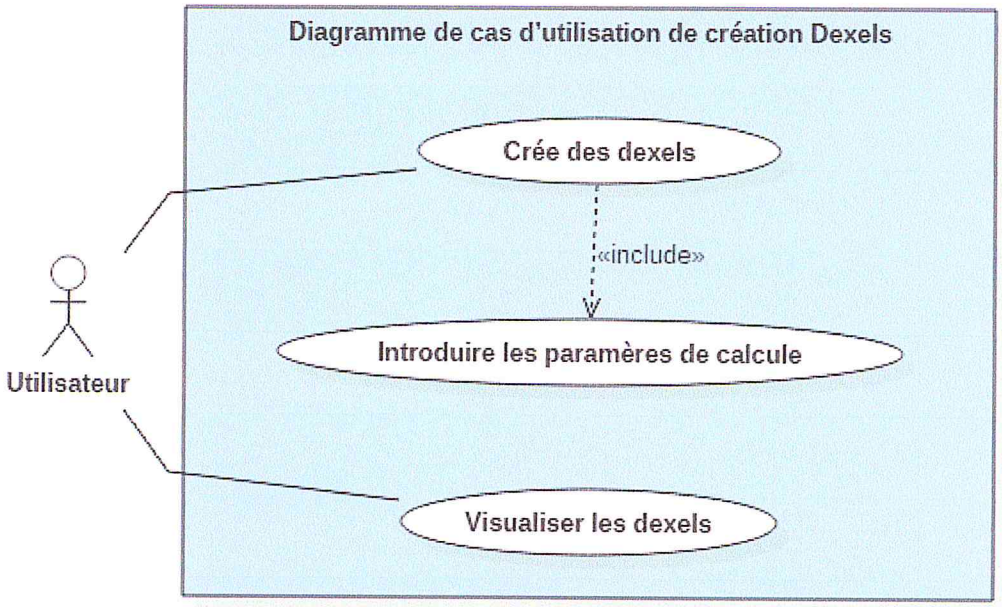


Figure II.23. Diagramme de cas d'utilisation de création Dexels.

4.2. Diagramme de classe :

Les diagrammes de classes décrivent les types d'objets composant un système et les différents types de relations statiques qui existent entre eux. Les diagrammes de classes font abstraction du comportement du système (Figure II.24).

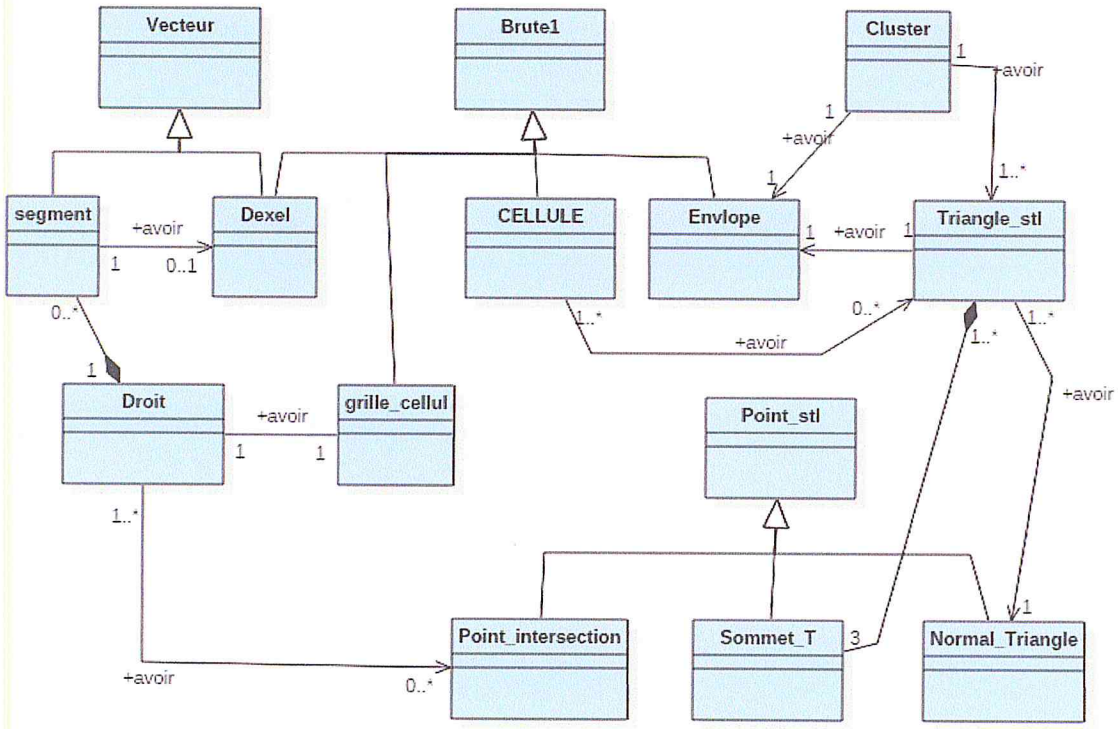


Figure II.24. Diagramme de classes.

4.3.1 Représentation des classes :

➢ ***Classe Droite :*** cette classe représente la droite de la grille de cellule utilisée pour déterminer les points d'intersections entre la droite et les triangles de la pièce afin de créer les Dexels (Figure II.25).

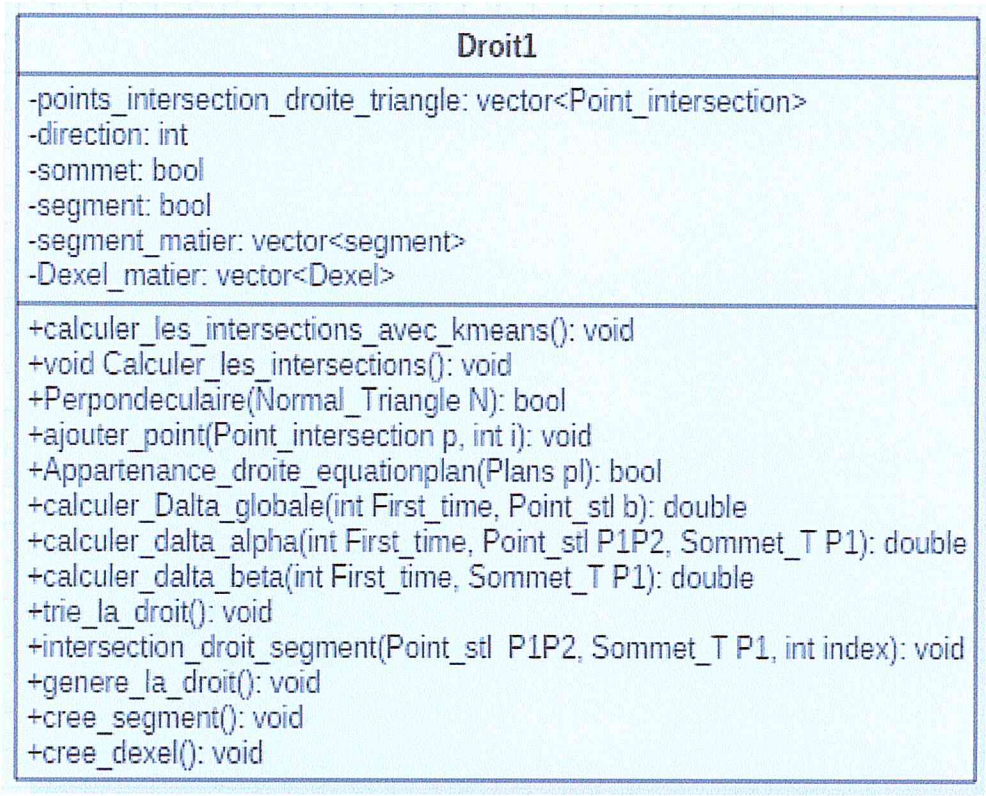

Droit1
-points_intersection_droite_triangle: vector<Point_intersection>
-direction: int
-sommet: bool
-segment: bool
-segment_matier: vector
-Dexel_matier: vector<Dexel>
+calculer_les_intersections_avec_kmeans(): void
+void Calculer_les_intersections(): void
+Perpondeculaire(Normal_Triangle N): bool
+ajouter_point(Point_intersection p, int i): void
+Appartenance_droite_equationplan(Plans pl): bool
+calculer_Dalta_globale(int First_time, Point_stl b): double
+calculer_dalta_alpha(int First_time, Point_stl P1P2, Sommet_T P1): double
+calculer_dalta_beta(int First_time, Sommet_T P1): double
+trie_la_droit(): void
+intersection_droit_segment(Point_stl P1P2, Sommet_T P1, int index): void
+genere_la_droit(): void
+cree_segment(): void
+cree_dexel(): void

Figure II.25. Classe Droit.

- **calculer_les_intersections_avec_kmeans() :** la méthode qui calcule l'intersection entre la droite et les éléments des clusters (classification par k_means).
- **Calculer_les_intersections() :** la méthode qui calcule l'intersection entre la droite et les éléments des cellules (classification par cellule).
- **Perpondeculaire(Normal_Triangle N) :** c'est la méthode qui détermine la position de la droite par rapport au triangle (colinéaire ou bien non colinéaire).
- **calculer_Dalta_globale(Point_stl)**
- **calculer_dalta_alpha(Point_stl,Sommet_T)**
- **calculer_dalta_beta(Sommet_T P1)**

} Sont des méthodes qui permettent de calculer la méthode de Cramer (10)

- **Intersection_droit_segment(Point_stl P1P2,Sommet_T P1,int index) :** calcule l'intersection entre la droite et un des segments du triangle.
- **trie_la_droit() :** c'est une méthode pour le tri des points d'intersection le long d'une droite.

- **cree_dexel() :** cette méthode permet de créer les Dexels à partir de ses segments de type matière.

➢ ***Classe Triangle :*** représente les triangles du fichier STL (Figure II.26).

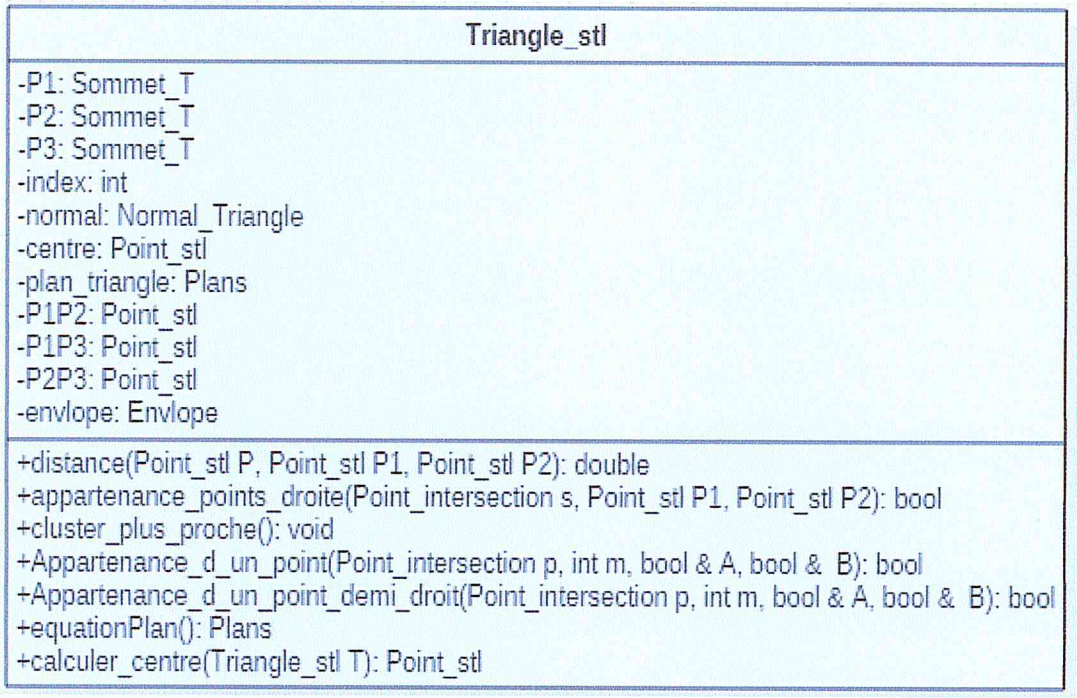


Figure II.26. Classe Triangle.

- **Appartenance_d_un_point (Point_intersection, int, bool&, bool&) :** vérifie si le point d'intersection appartient au triangle.
- **equationPlan() :** cette méthode permet de retourner l'équation du triangle en 3D.
- **calculer_centre (Triangle_stl) :** cette méthode calcule le centre du triangle.
- **appartenance_points_droite (Point_intersection, Point_stl, Point_stl) :** cette fonction vérifie si un point d'intersection appartient à un segment du triangle.
- **cluster_plus_proche () :** affecte le triangle au Cluster le plus proche.

Conclusion :

Dans ce chapitre, nous avons défini la conception de notre application en présentant la problématique, les différents objectifs à atteindre. Ensuite, nous avons présenté la structure de notre application et son déroulement en utilisant le modèle orienté objet et le langage de modélisation UML avec le diagramme d'état, les diagrammes de cas d'utilisation, les diagrammes de séquence et le diagramme de classe. Dans le chapitre suivant, nous allons montrer les fonctionnalités de l'application logicielle et sa validation à travers des exemples.

Chapitre III :

Implémentation & Résultats

Introduction :

Après avoir présenté la conception de notre application dans le chapitre précédent, nous allons présenter dans ce dernier chapitre l'implémentation et les résultats du travail réalisé. Nous commençons par définir les outils de développement utilisés dans notre application. Par la suite, nous présentons les fenêtres de l'application. A la fin, nous procédons aux tests sur différentes pièces complexes.

1. Présentation du projet :

Ce travail s'insère dans le cadre de développement de modules logiciels pour la production des surfaces de formes complexes initié par l'équipe Conception et Fabrication Assistées par Ordinateur « CFAO » de la Division Productique et Robotique « DPR » du Centre de Développement des Technologies Avancées « CDTA ». Les outils utilisés lors de développement et la mise en œuvre de l'application logicielle sont C++, OpenGL et PPL.

1.1 Définition du C++ :

C++ [27] est un langage de programmation compilé permettant la programmation sous de multiples paradigmes tels que la programmation procédurale, orientée objet ou générique. Ses bonnes performances et sa compatibilité avec le C en font un des langages de programmation les plus utilisés dans les applications où la performance est critique. Il existe de nombreuses bibliothèques C++ en plus de la bibliothèque standard du C++ (C++ Standard Library) qui est incluse dans la norme. Par ailleurs, C++ permet l'utilisation de l'ensemble des bibliothèques C existantes.

1.2 Présentation OpenGL :

OpenGL[28] (Open Graphics Library) est une spécification qui définit une API multiplateforme pour la conception d'applications générant des images 3D et également 2D. Elle utilise en interne les représentations de la géométrie projective pour éviter toute situation faisant intervenir des infinis. L'interface regroupe environ 250 fonctions différentes. Elles peuvent être utilisées pour afficher des scènes tridimensionnelles complexes à partir de simples primitives géométriques. Du fait de son ouverture, de sa souplesse d'utilisation et de sa disponibilité sur toutes les plateformes, elle est utilisée par la majorité des applications scientifiques, industrielles ou artistiques 3D et certaines applications 2D vectorielles. Par ailleurs, OpenGL

est portable et multiplateforme. En théorie, un même code OpenGL fonctionne sur tous les systèmes d'exploitation (Windows, Linux, et Mac) et sur toutes les plateformes.

1.3 Présentation de Embarcadero Builder C++ :

Builder C++ [29] est un logiciel de développement rapide d'applications conçu par Borland. Tout d'abord C++ est un outil RAD, c'est-à-dire orienté vers le développement rapide d'applications (Rapid Application Development) sous Windows. Builder C++ permet de réaliser de façon très simple l'interface des applications et de relier aisément le code utilisateur aux événements Windows. Le but de Builder C++ est de créer des applications rapidement sous Windows grâce à ses bibliothèques variées ainsi qu'à une interface graphique avec son éditeur de ressources.

1.4 Bibliothèque de programmation parallèle « PPL » :

La bibliothèque de programmation parallèle « Parallel Programming Library » « PPL » permet d'exécuter des tâches en parallèle en tirant parti de l'utilisation de plusieurs CPU de périphériques et d'ordinateurs. La « PPL » inclut des fonctionnalités avancées pour l'exécution de tâches, la jointure de tâches, l'attente de groupes de tâches, etc., à traiter. Pour effectuer toutes ces opérations, le pool de threads se règle automatiquement en fonction de la charge sur les CPU de sorte que vous n'avez pas à gérer la création ou la gestion des threads. Pour utiliser cette bibliothèque, il est impératif d'inclure le fichier « System.Threading » dans nos applications. Cette unité est constituée de plusieurs fonctionnalités pouvant être incluses dans des projets nouveaux ou existants. L'unité inclut également des arguments surchargés qui lui permettent de s'adapter à C++. En utilisant la « PPL » [30], les applications peuvent aisément :

- Accélérer la boucle en utilisant « TParallel.For ».
- Exécuter plusieurs tâches en parallèle en utilisant « TTask » et « ITask ».
- Exécuter un processus pendant l'exécution d'autres tâches, en permettant d'obtenir le résultat au moment choisi. « IFuture » permet de définir la priorité d'exécution des blocs de codes et d'obtenir les résultats au moment choisi.

Dans notre travail, c'est « TParallel.For » qui est utilisée pour gérer toutes les droites en même temps.

➢ ***TParallel.For :*** elle permet d'exécuter au moins deux événements d'itération en parallèle, c'est-à-dire simultanément, au lieu de les exécuter séquentiellement l'un après l'autre, ce qui est le cas avec la boucle for habituellement utilisée.

1.5 Matériel utilisé :

Les algorithmes développés fonctionnent sur un microprocesseur Intel Core i5 avec une RAM de 6Go et Windows 7 64 bits.

2. Présentation de l'application logicielle :

Notre Application est un module logiciel graphique et interactif sous Windows permettant la représentation volumique des pièces de n'importe quelles formes géométriques d'une manière précise et rapide. Dans cette phase nous décrivons les différentes tâches implémentées dans le cadre de notre projet.

2.1 Fenêtre principale :

La fenêtre principale de notre application est composée de cinq onglets (Figure III.1) :

- ***1er onglet « Model STL » :*** cet onglet permet de lire la pièce en format de fichier STL.
- ***2ème onglet « Classification » :*** cette partie permet de choisir la méthode de classification « K-means » ou « Cellules ».
- ***3ème onglet « Grille » :*** c'est la partie de création des grilles.
- ***4ème onglet « Triple-Dexel » :*** c'est la partie principale de notre application qui permet de calculer les intersections entre la droite est les triangles afin de créer le les Triple-Dexels.
- ***5ème onglet « Comparaison » :*** c'est la partie de comparaison des temps de calcul des approches séquentielle et parallèle.

Figure III.1. Onglets de l'application développée.

Les différentes fonctionnalités de ces onglets de sont détaillées dans ce qui suit.

2.1.1. Onglet « Model STL » :

L'onglet « Model STL » (Figure III.2) permet d'effectuer les tâches suivantes :

- Ouvrir la pièce à modéliser sous format de fichier STL.
- Déterminer la taille de la pièce suite au calcul des limites du brut.

- Visualiser les points de la pièce.
- Visualiser les triangles de la pièce en filaire ou en rendu.
- Visualiser l'enveloppe de la pièce en filaire ou en rendu.

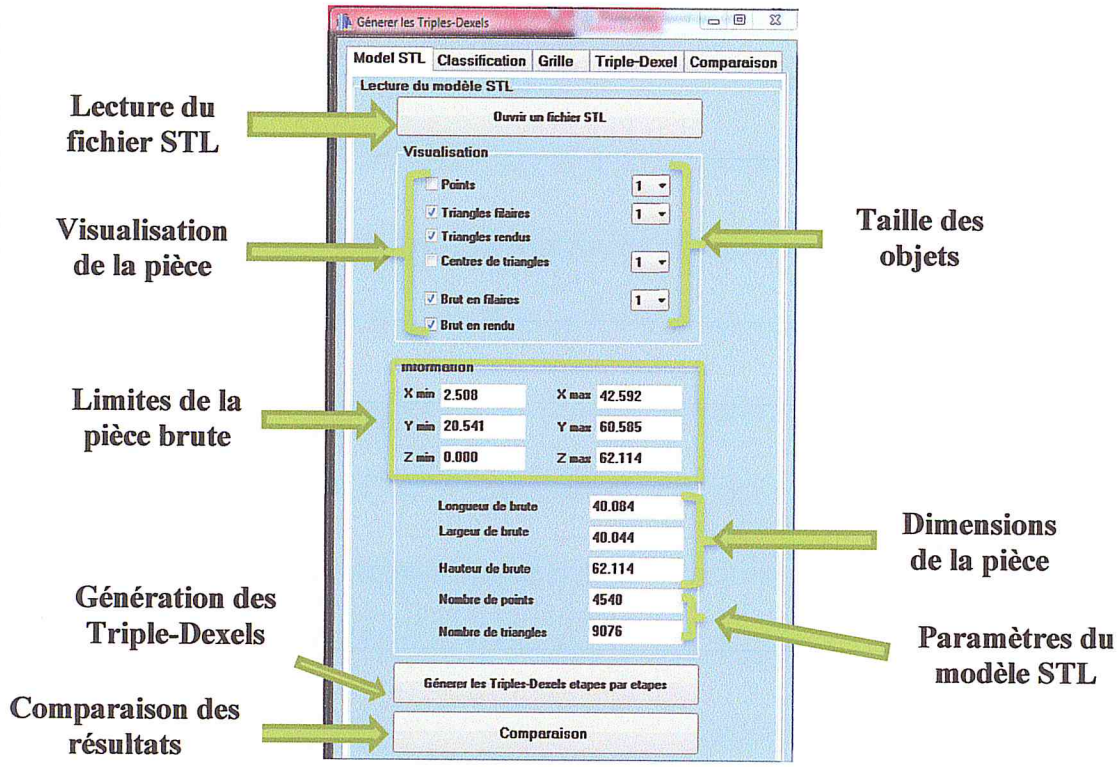


Figure III.2. Onglet « Model STL ».

2.1.2. Onglet « Classification » :

Après lecture du fichier STL, une méthode de classification des triangles doit être choisie pour limiter les calculs des points d'intersection entre la droite et les triangles. Pour cela, deux méthodes de classification par « Cellules » et par « K-means » sont proposées (Figure III.3). L'utilisateur choisit la méthode de classification la plus adéquate.

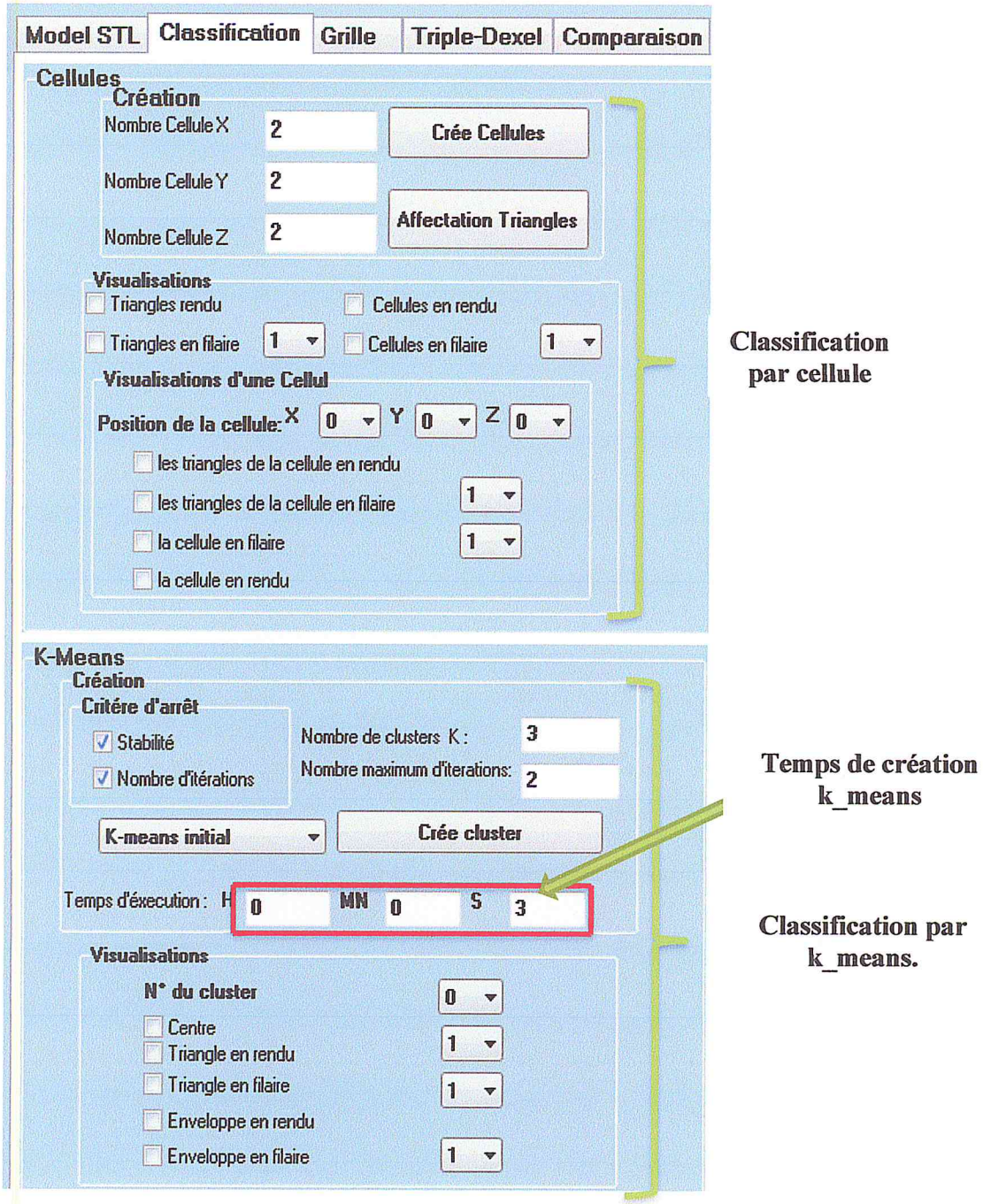


Figure III.3. Onglet « Classification ».

➢ ***Classification par « Cellules » :*** elle permet d'affecter les triangles à des cellules une fois que l'utilisateur introduit le nombre de cellules suivant les axes X, Y et Z (Figure III.4). Cette partie englobe les tâches suivantes :

- Créer les cellules selon le nombre de cellules introduit par l'utilisateur.
- Affecter les triangles aux cellules.
- Visualiser les cellules en mode filaire et rendu (Figure III.5).

- Visualiser les triangles en mode filaire et rendu.
- Visualiser cellule par cellule et leurs triangles en filaire et en rendu (Figure III.6).

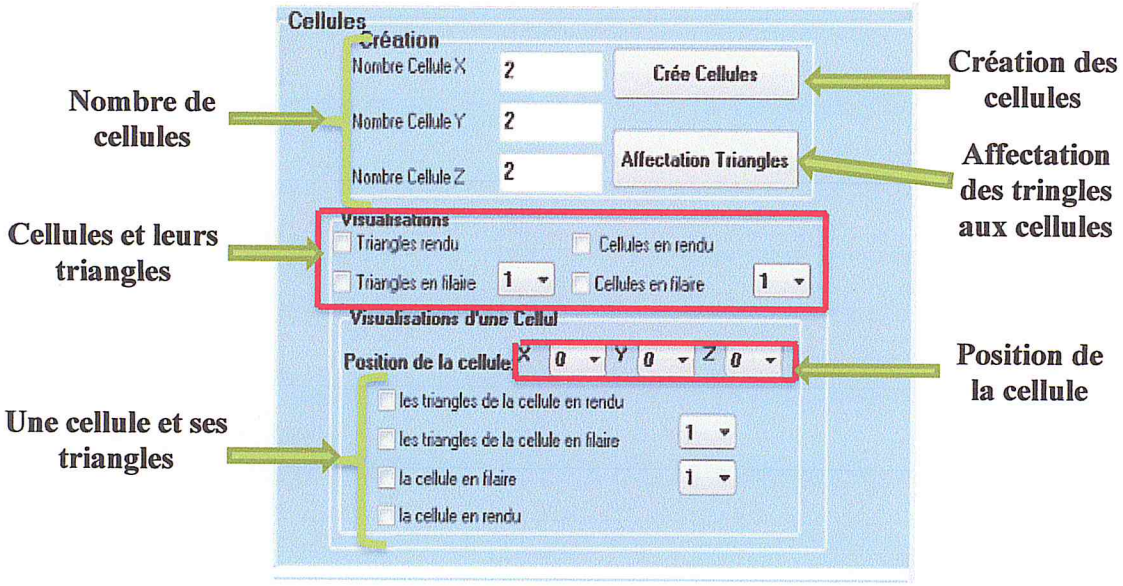


Figure III.4. Onglet « Classification par cellules ».

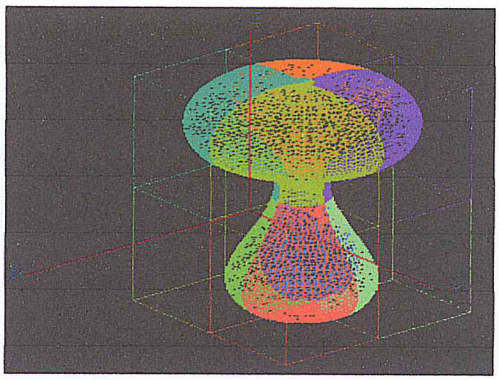


Figure III.5. Visualisation des cellules.

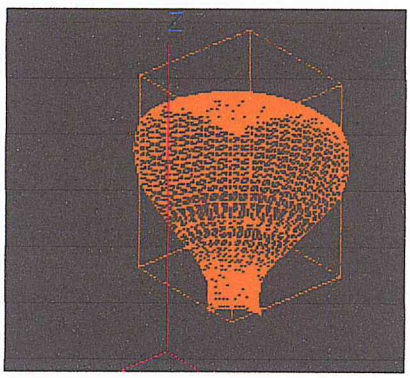


Figure III.6. Visualisation d'une cellule.

➢ ***Classification par « K-means » :*** elle permet d'affecter les triangles à des Clusters une fois que l'utilisateur fixe le nombre de Clusters et choisit les critères d'arrêt (Figure III.7).

Cette partie permet d'exécuter les tâches suivantes :

- Choisir une variante de la méthode « K-means » : K-means initial, Global K-means, Fast Global K-means et Incremental K-means.
- Créer des Clusters en fonction des paramètres introduits.
- Visualiser le temps d'exécution.
- Visualiser les Clusters et leurs enveloppes en filaire ou en rendu (Figure III.8.a).
- Visualiser un Cluster et son enveloppe en rendu ou en filaire (Figure III.8.b).

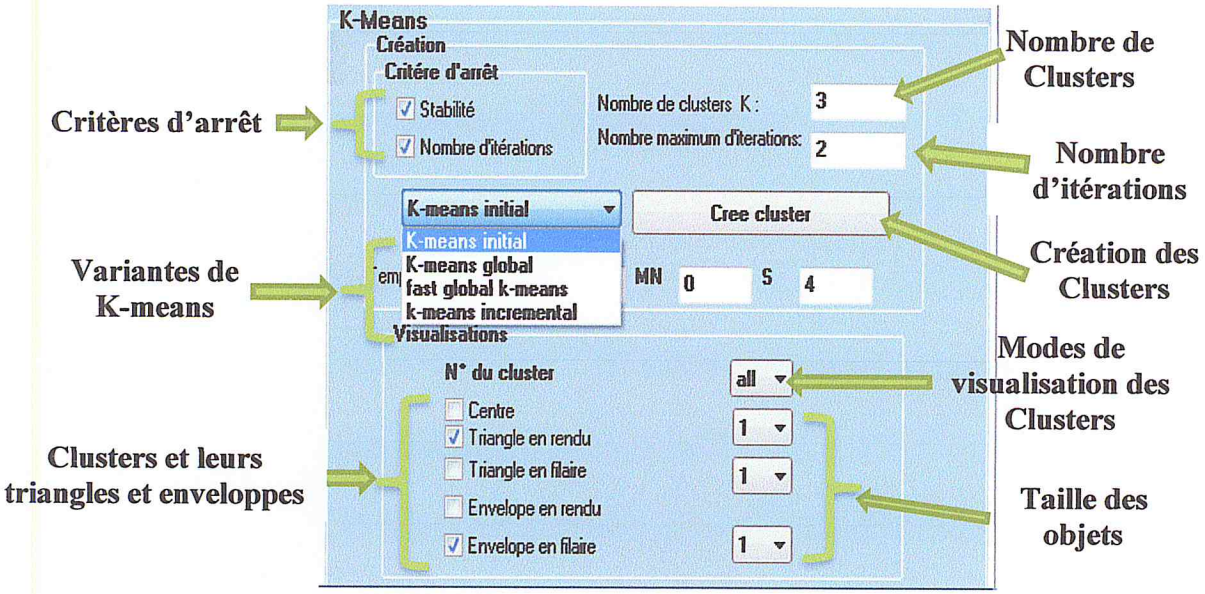
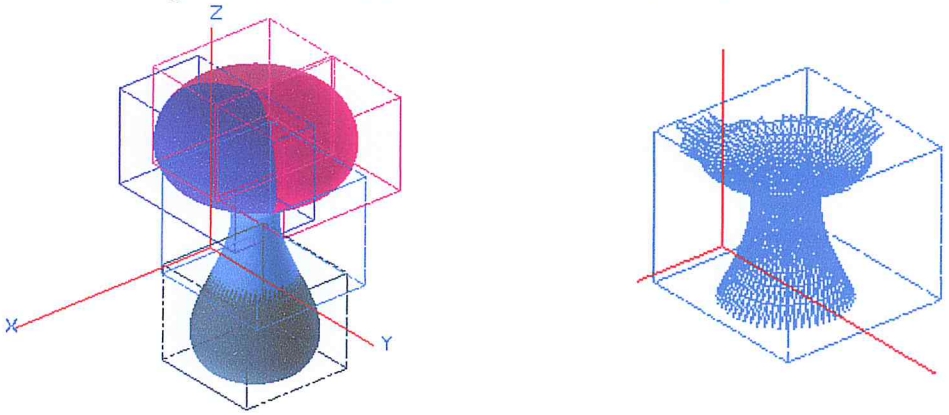


Figure III.7. Variantes de la méthode « K-means ».



a. Tous les Clusters en rendu. b. Paramètres d'un Cluster.

Figure III.8. Visualisation des Clusters.

2.1.3. Onglet « Grille » :

Cet onglet permet de créer les trois grilles sur les plans XY, XZ et YZ. Chaque grille est composée par des cellules caractérisées par leurs points limites, son centre et la droite orientée vers la direction perpendiculaire au plan (Figure III.9). Il permet de lancer les taches suivant :

- Créer les grilles selon les pas des cellules introduits par l'ingénieur.
- Visualiser les grilles pour chaque plan.
- Visualiser les centres des cellules pour chaque plan.
- Visualiser les droites des cellules pour chaque plan.

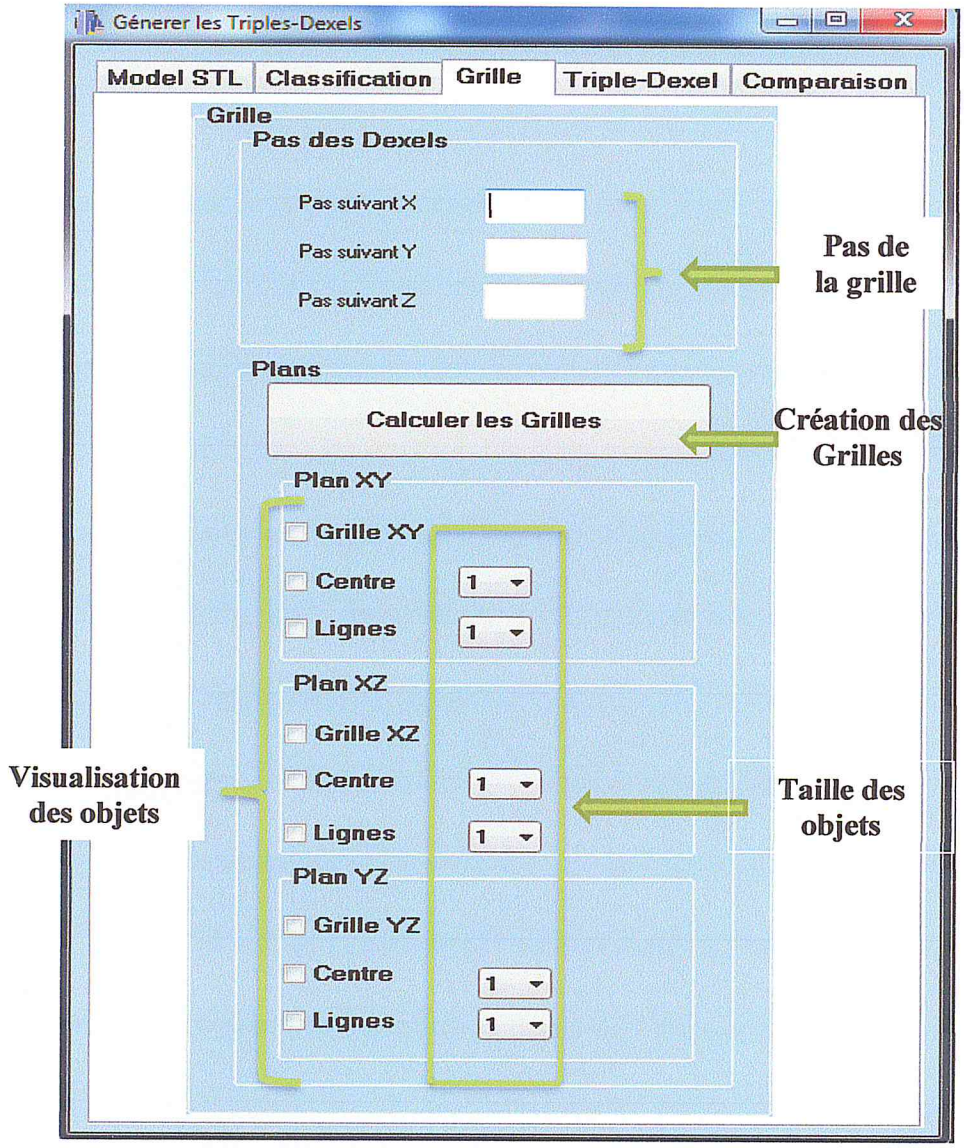
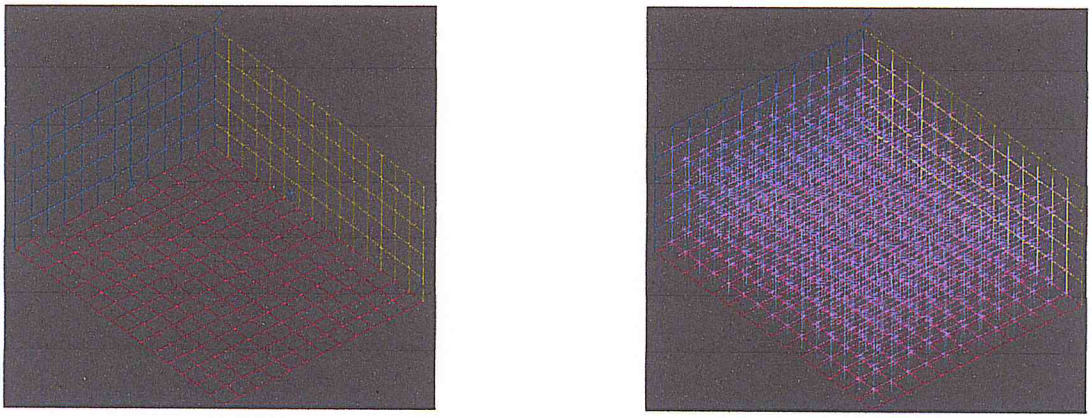


Figure III.9. Onglet « Grille ».



a. Grilles et centres des cellules. b. Droites des grilles.

Figure III.10. Visualisation des grilles.

2.1.4. Onglet « Triple-Dexel » :

Cet onglet permet de déterminer la modélisation volumique de la pièce en Triple-Dexels. Cette partie est divisée en deux sous-onglet :

- ***Onglet « Tous les Dexels » :*** elle permet de gérer les deux sous onglet-suivants :
  - **Onglet « Les trois plans » :** cette partie (Figure III.11) permet de gérer les actions principales suivantes :
    - Choisir le type de classification.
    - Choisir le mode de calcul (séquentiel ou parallèle).
    - Choisir une méthode d'appartenance parmi les méthodes proposées. Il est possible de gérer le degré de précision manuellement.
    - Créer les Triple-Dexels.

De plus, il est possible de réaliser les fonctions suivantes :

- Visualiser le volume de la pièce en Triple-Dexels.
- Visualiser les points d'intersection.
- Visualiser les segments et les Dexels.
- Sauvegarder le modèle Triple-Dexels de la pièce dans un fichier.

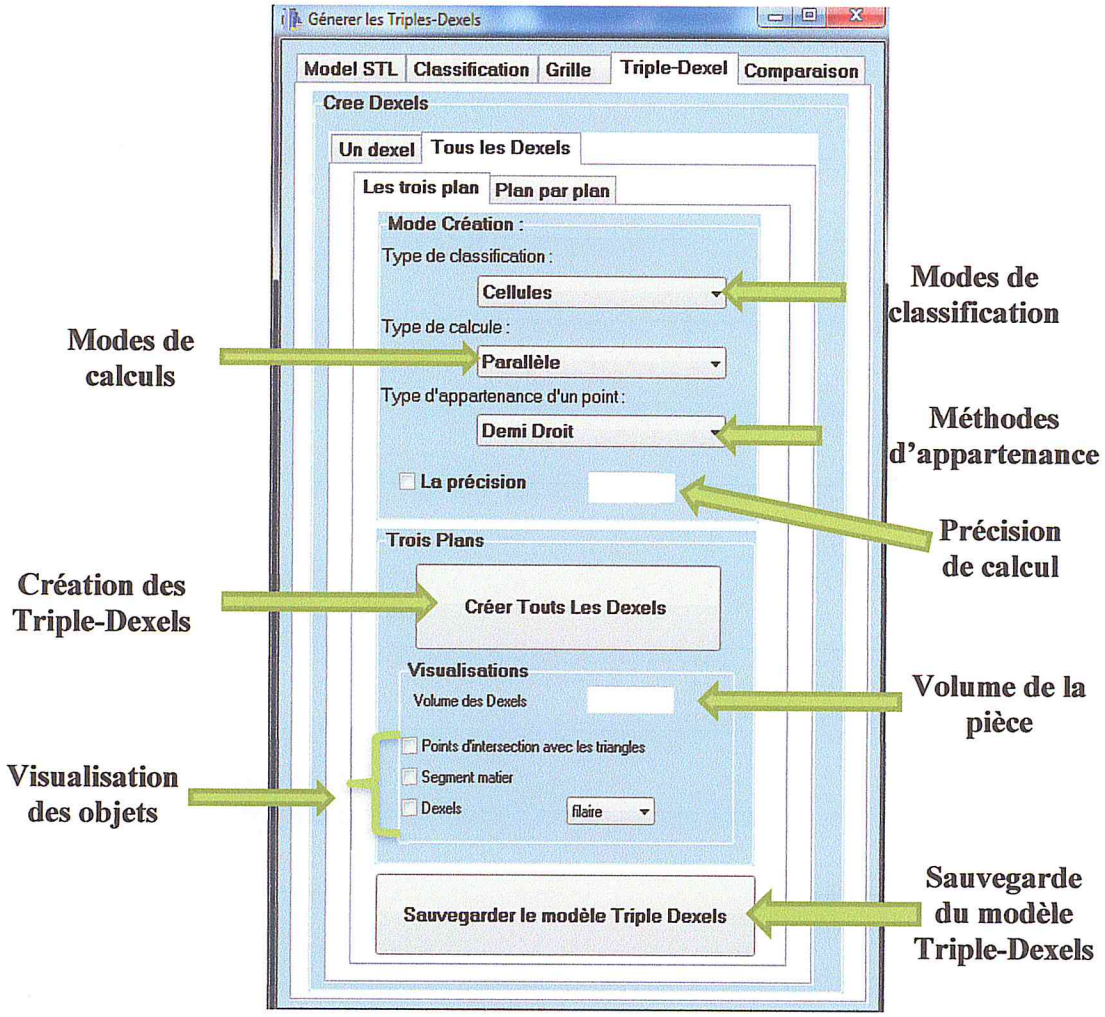


Figure III.11. Onglet « Triple-Dexel 1».

- **Onglet « Plan par plan » :** cet onglet permet de lancer les tâches suivantes pour plan par plan (Plan XY, Plan XZ et Plan YZ) (Figure III.12) :
  - Créer les Dexels pour chaque plan individuellement.
  - Visualiser le volume des Dexel pour chaque plan.
  - Visualiser les segments pour chaque plan.
  - Visualiser tous les Dexels.

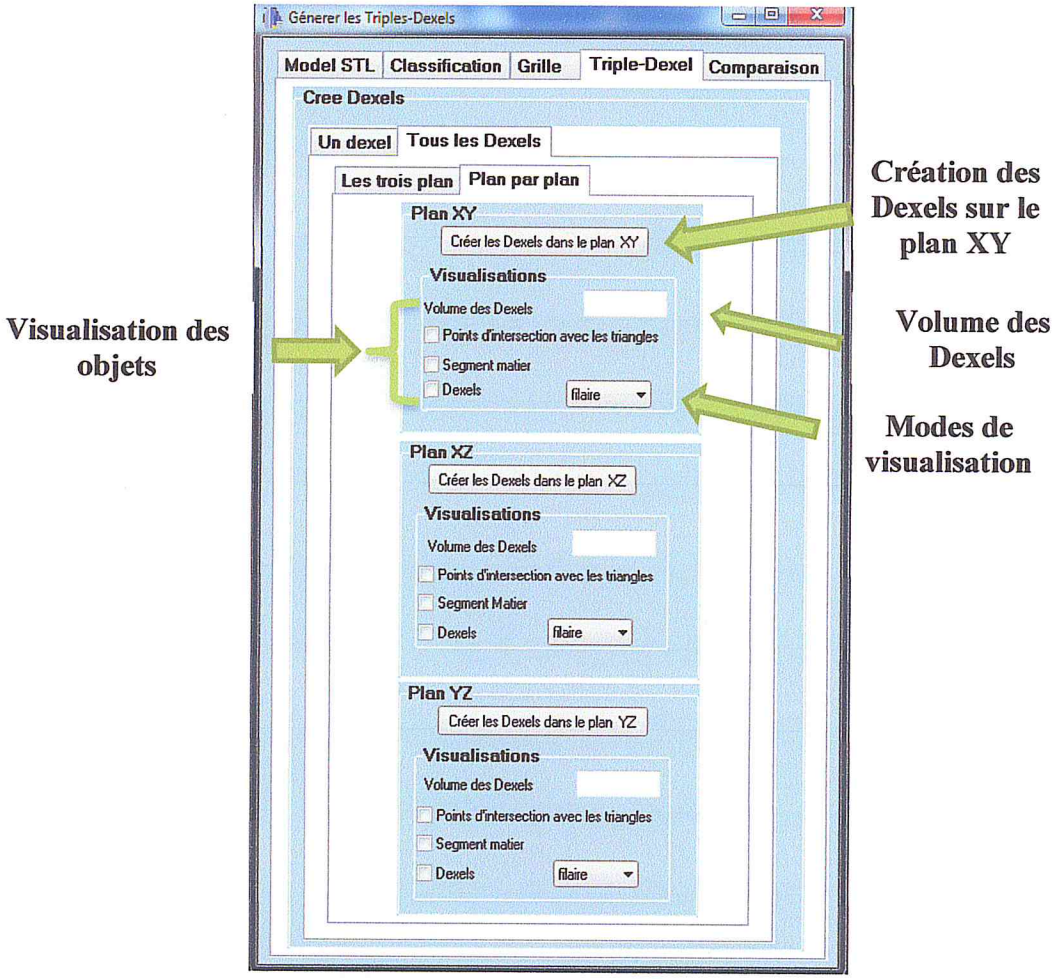


Figure III.12. Onglet « Triple-Dexel 2».

➢ ***Onglet « Un dexel » :*** cet onglet sert à visualiser droite par droite pour chaque plan (Figure III.13). Il permet de d'effectuer les tâches suivantes :

- Identifier la position de la droite.
- Visualiser les points début et fin de la droite.
- Visualiser les points d'intersection sur la droite.
- Visualiser les triangles qui sont traversés par la droite.
- Visualiser le nombre des points d'intersection.
- Visualiser les Dexels sur cette droite.

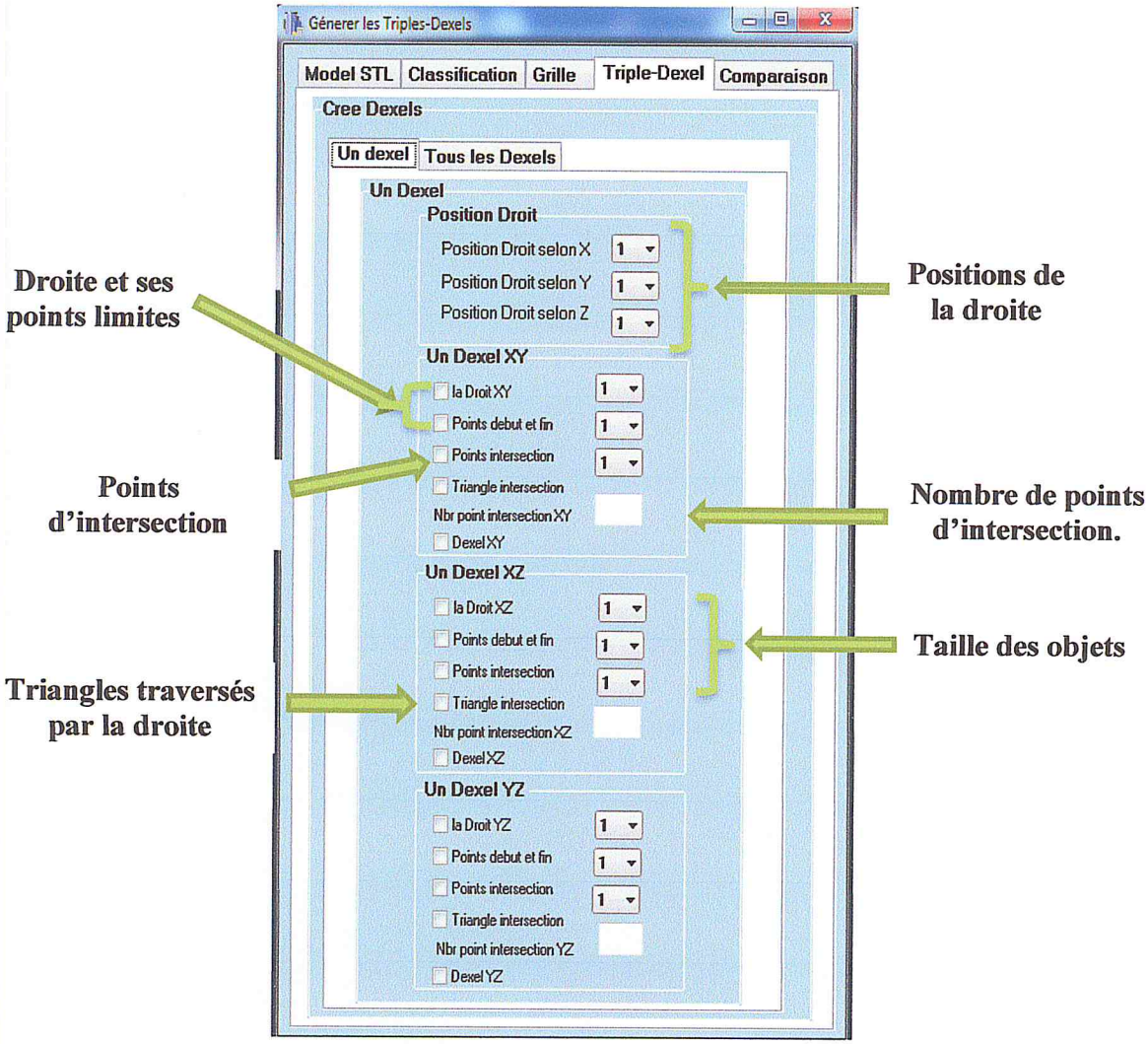


Figure III.13. Onglet « Triple-Dexel 3».

2.1.5. Onglet « Comparaison » :

Cet onglet sert à comparer les temps d'exécution et la création des Triples-Dexels entre l'approche séquentiel et parallèle. Il englobe deux sous-onglets

➢ ***Onglet « Paramètres » :*** l'utilisateur peut accéder directement à cet onglet après la lecture de fichier STL dans l'onglet « Model STL » (Figure III.14). Cet onglet permet de :

- Créer la classification.
- Identifier les méthodes à comparer.
- Identifier le mode de calcul (séquentiel ou parallèle).
- Choisir une méthode d'appartenance.
- Comparer les temps d'exécution.

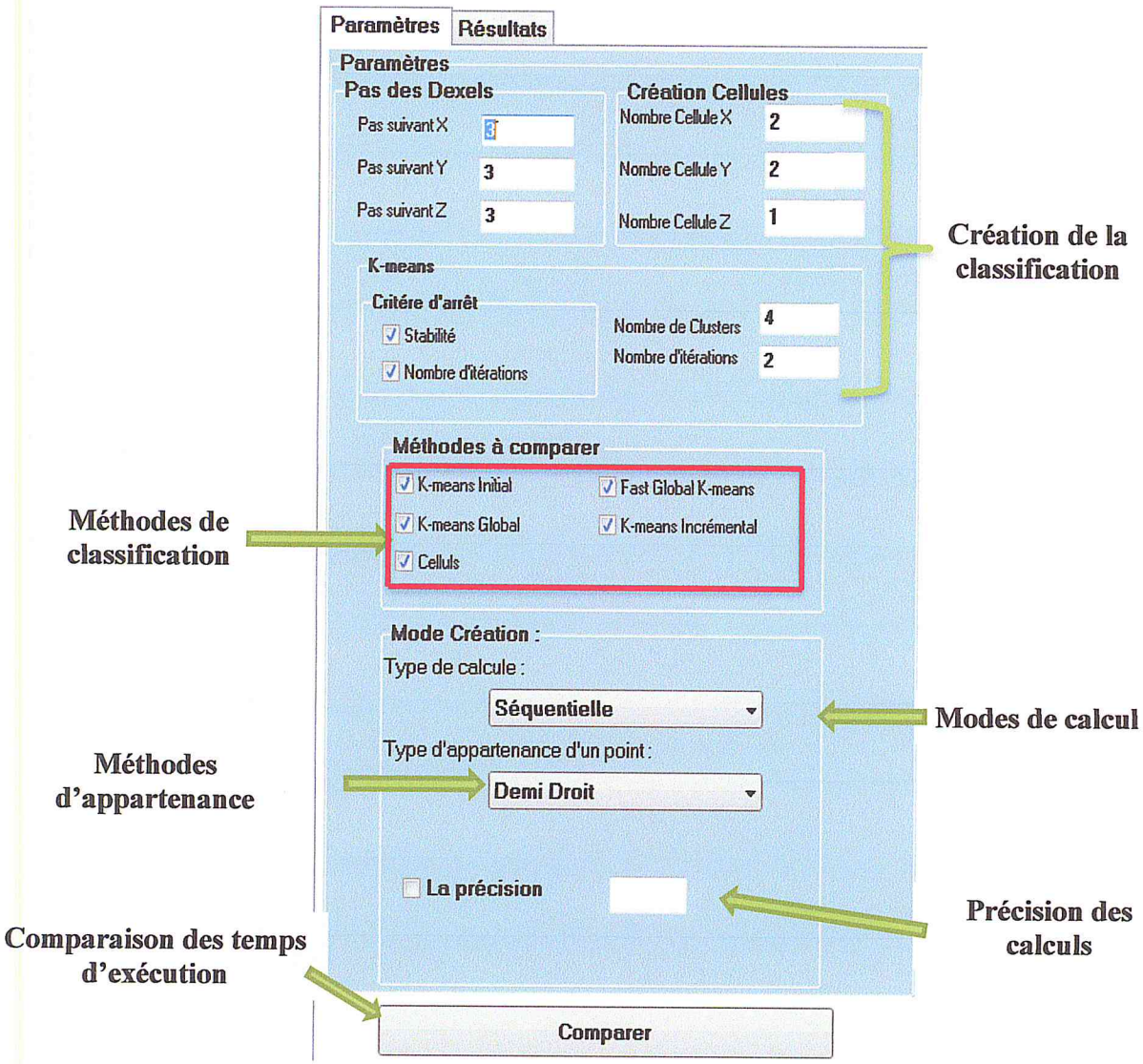


Figure III.14. Onglet « Comparaison / Paramètres ».

➢ ***Onglet « Résultat » :***

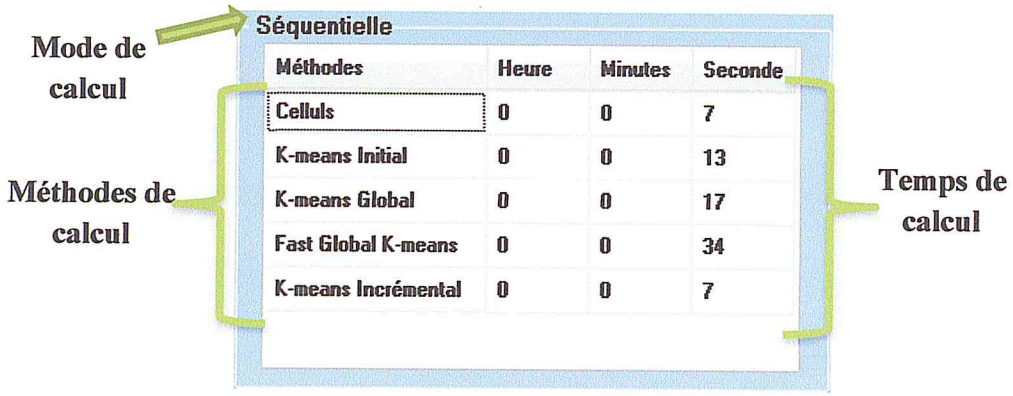


Figure III.15. Onglet « Comparaison / Résultat ».

3. Résultats :

Notre approche développée a été validée sur plusieurs exemples suie à la vérification des différents critères tels que :

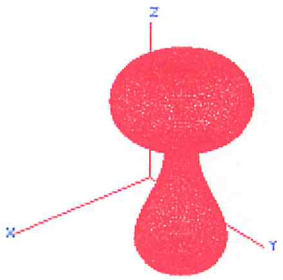
- Temps de calcul par rapport aux :
  - Méthodes de classification
  - Modes de calcul (parallèle ou séquentielle).
- Qualité des Dexels.
- Mémoire de la RAM.

3.1 Temps de calcul :

Le temps de calcul a été évalué par rapport aux critères suivants.

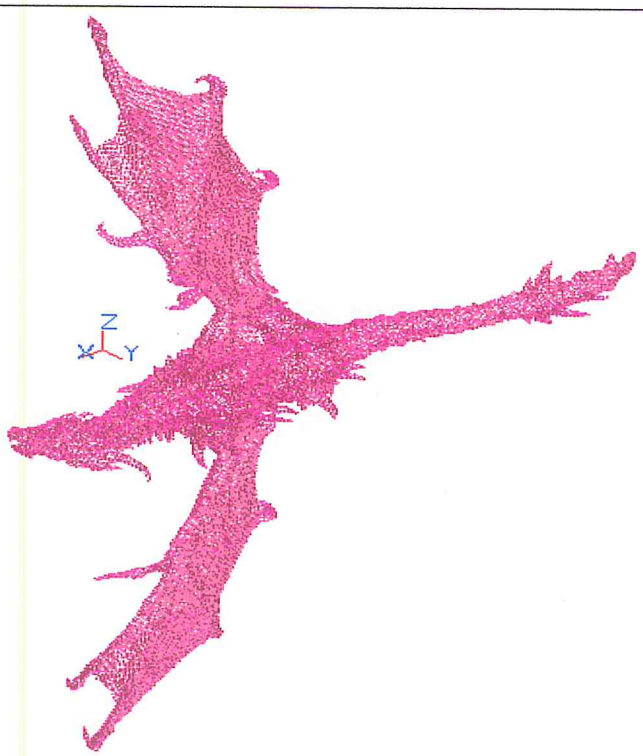
- ***Méthodes de classification :*** l'exemple utilisé pour la validation du temps de calcul selon le critère de classification est un poignet. Une comparaison entre les différentes méthodes de classification est illustrée sur le Tableau 1. Le temps de calcul est lié au paramètre du modèle lorsque la répartition des Clusters est homogène avec moins de chevauchement. Le processus de Clustering est plus efficace en termes de temps de calcul.

Tableau 1. Comparaison entre les méthodes de classification.

	Caractéristiques Modèle	Nombre de triangles 9076
	Cellules	Nbr selon X = 2
		Nbr selon Y = 2
		Nbr selon Z = 2
	Cluster	Nombre de cluster = 8
Méthode	**Temps**	**Gain du temps**
K_means intial	15 s	71.15 %
K_means Global	17 s	67.30 %
Fast Global k-means	16 s	69.23 %
K_means Incrémental	52 s	0 %
Cellule	26 s	50%

- ***Modes de calcul :*** l'exemple utilisé pour la validation du temps de calcul selon le mode de calcul est un dragon. Une comparaison entre les différents modes est donnée par le Tableau 2. Le temps de calcul est lié au paramètre du modèle. A chaque fois que nous diminuons les paramètres du pas de la grille, plus le nombre de droites augmente et par conséquent le temps de calcul. Dans cet exemple, nous avons utilisé la bibliothèque de programmation parallèle « PPL » pour comparer les performances des deux approches. Nous avons remarqué que la méthode parallèle est plus rapide que la méthode séquentielle. Plus le nombre de droites est grand, plus le gain de temps de l'approche parallèle par rapport à l'approche séquentielle est important. Les différents résultats sont donnés dans le Tableau 3.

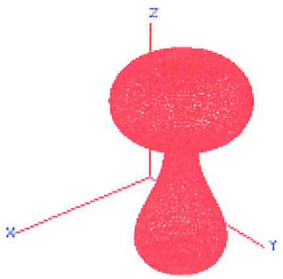
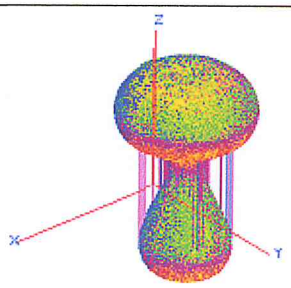
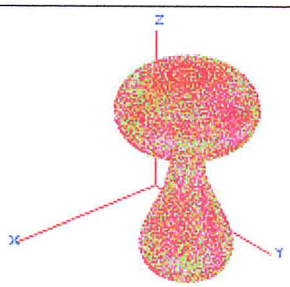
Tableau 2. Comparaison entre les modes de calcul parallèle et séquentiel.

<table>
<tr><td rowspan="10">  </td><td rowspan="4">Caractéristiques Modèle</td><td>Longueur = 1333.25 mm</td></tr>
<tr><td>Largeur = 450.129 mm</td></tr>
<tr><td>Hauteur = 1793.906 mm</td></tr>
<tr><td>Triangle= 16074</td></tr>
<tr><td rowspan="4">Cellule</td><td>Nbr selon X = 10</td></tr>
<tr><td>Nbr selon Y = 10</td></tr>
<tr><td>Nbr selon Z = 10</td></tr>
<tr><td>Nbr = 1000 cellules</td></tr>
<tr><td rowspan="4">Grille</td><td>Pas X = 5 mm</td></tr>
<tr><td>Pas Y = 5 mm</td></tr>
<tr><td>Pas Z = 5 mm</td></tr>
<tr><td>Nbr de droit = 151 Mille</td></tr>
<tr><td colspan="2">Mode de calcul</td><td>Temps</td><td>Gain en temps</td></tr>
<tr><td colspan="2">Parallèle</td><td>1053 s</td><td>33.16 %</td></tr>
<tr><td colspan="2">Séquentiel</td><td>1574 s</td><td>0 %</td></tr>
</table>

3.2 Qualité des Dexels :

Pour prouver que la qualité des Dexels créés par notre programme est bonne, nous avons comparé notre programme avec un ancien travail qui négligeait la concavité et la convexité du modèle. Ce dernier créait des Dexels en plus, chose qui dégrade la qualité du modèle. D'un autre côté, notre programme traite ce problème et crée des modèles de qualité. Dans le Tableau 3 nous avons pris l'exemple d'un poignet et nous avons comparé le rendu de notre programme avec l'ancien.

Tableau 3. Comparaison entre la qualité des Triple-Dexels.

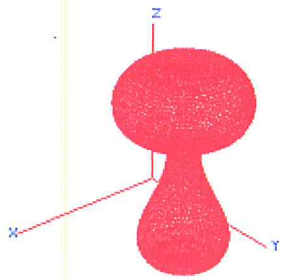
<table>
<tr><td rowspan="6">  </td><td>Caractéristiques Modèle</td><td>Nombre de triangle = 9076</td></tr>
<tr><td rowspan="4">Cellules</td><td>Nbr selon X = 2</td></tr>
<tr><td>Nbr selon X = 2</td></tr>
<tr><td>Nbr selon X = 2</td></tr>
<tr><td>Nbr = 8 cellules</td></tr>
<tr><td rowspan="3">Grille</td><td>Pas X = 0.1 mm</td></tr>
<tr><td>Pas Y = 0.1 mm</td></tr>
<tr><td>Pas Z = 0.1 mm</td></tr>
<tr><td>Dexels</td><td> 
Ancien travail</td><td> 
Notre travail</td></tr>
</table>

3.3 Mémoire de la RAM :

Nous reprenons l'exemple du poignet pour faire une comparaison entre les deux modes de calcul par rapport au coût en mémoire. A chaque fois que nous diminuons les paramètres du pas de la grille, plus le nombre de droites augmente et plus la consommation de l'espace mémoire (RAM) devient plus importante. Dans cet exemple, nous avons utilisé la bibliothèque de programmation parallèle « PPL » pour comparer les performances de la méthode parallèle avec la méthode séquentielle. Nous avons remarqué que la méthode parallèle consomme moins d'espace mémoire que la méthode séquentielle. Plus le pas devient

petit, plus le gain en espace mémoire est optimal. Nous ne pouvons pas donner les résultats de cette comparaison si les paramètres du pas de la grille sont trop petits car les deux méthodes pourront causer une saturation de la mémoire. Pour résoudre ce problème, nous avons sacrifié un peu la qualité des Dexels en augmentant le pas. De cette manière nous pouvons mieux voir les gains en espace mémoire apportés par la méthode parallèle par rapport à la méthode séquentielle. Les différents résultats sont donnés dans le Tableau 4.

Tableau 4. Comparaison entre la mémoire de la RAM.

<table>
<tr><td rowspan="5">  </td><td rowspan="4">Caractéristiques Modèle</td><td>Longueur = 40.084 mm</td></tr>
<tr><td>Largeur = 40.044 mm</td></tr>
<tr><td>Hauteur = 62.114 mm</td></tr>
<tr><td>Triangle= 9076</td></tr>
<tr><td rowspan="4">Cellule</td><td>Nbr selon X = 50</td></tr>
<tr><td>Nbr selon Y = 50</td></tr>
<tr><td>Nbr selon Z = 50</td></tr>
<tr><td>Nbr = 125000 cellules</td></tr>
</table>

Grille	Mode de calcul	Mémoire RAM	Gain en mémoire
Pas X = 0.09 mm Pas Y = 0.09 mm Pas Z = 0.09 mm Nbre de droites = 800 000	Parallèle	1, 010,148 K	?? %
	séquentiel	mémoire saturée	?? %
Pas X = 0.2 mm Pas Y = 0.2 mm Pas Z = 0.2 mm Nbre de droites = 164 554	Parallèle	370,020 K	7,5 %
	séquentiel	400,000 K	0 %

Conclusion :

Dans ce chapitre, nous avons présenté toutes les formes que nous avons créées et intégrées dans l'application logicielle de CFAO, permettant la représentation volumique des pièces quel que soit la complexité géométrique des formes d'une manière précise rapide et avec un gain en mémoire.

CONCLUSION GENERALE

CONCLUSION GENERALE

Durant le présent projet de fin d'étude, nous avons procédé à proposer et à implémenter une approche optimale permettant la modélisation volumique des pièces de formes complexes, définies par leurs modèles STL, en utilisant la technique des « Triple-Dexels » afin de l'employer dans la simulation d'enlèvement de matière lors du processus d'usinage des surfaces complexes sur des fraiseuses multiaxes.

Ce travail représente une contribution importante dans le projet mené par l'équipe CFAO du CDTA intitulé « Production Automatisée des Pièces Complexes sur des Fraiseuses Multiaxes».

L'élaboration de ce travail nous a permis, de mettre en pratique l'ensemble des connaissances acquises tout au long de notre cursus et de maitriser certains domaines scientifiques étudiés à l'USDB, et d'autre part, de préparer notre intégration dans le monde professionnel.

Les principaux résultats de notre travail sont :

- ✓ Evitement les limites de l'approche développée antérieurement et proposer d'autres méthodes.
- ✓ Génération du modèle en « Dexels » suivant les trois axes X, Y et Z en proposant une méthode générique.
- ✓ Résolution du problème des formes de fortes concavités et convexités.
- ✓ Mise en œuvre d'une approche pour le calcul parallèle du modèle en Triple-Dexels. Cette dernière a permis un gain de temps considérable comparé au mode séquentiel.

En perspective de notre travail, nous recommandons de traiter les points suivants :

- ✓ Résoudre le problème d'appartenance qui représente une contrainte majeure dans la génération du modèle Triple-Dexels. Ce problème est lié aux erreurs d'arrondis.
- ✓ Déterminer les pas optimums suivant chaque axe permettant de générer un modèle Triple-Dexel avec une précision donnée.
- ✓ Exploitation de la carte graphique par l'utilisation du GPU pour accélérer davantage les calculs parallèles.

REFERENCES BIBLIOGRAPHIQUES

BIBLIOGRAPHIE :

[1]. "FABRICATION MECANIQUE". **DEPEYRE, Philippe.** s.l. : Faculté des Sciences et Technologies, 2005.

[2] d. C. Tournier, «Contribution à la conception des formes complexes : La surface d'usinage en fraisage 5 axes isocrête,» Laboratoire Universitaire de Recherche en Production Automatisée, 12 décembre 2001.

[3] K. Bouhadja, M. Bey, K. Sebti, H. Moulay, «Modélisation Volumique des Pièces de Formes Complexes par Triple-Dexels,» Centre de Développement des Technologies Avancées (CDTA), Alger, Algérie, 2016.

[4] K. Bouhadja, «Fabrication des surfaces de forme gauche Rapport de recherche,» 2013.

[5] P. Dubois, A. Aoussat, R. Duchamp, P. Dubois, A. Aoussat, and R. Duchamp, "Prototypage rapide, généralités To cite this version : HAL Id : hal-01064800 Science Arts & Métiers (SAM)," pp. 9–10 2014.

[6] Z. C. L. K. a. N. B. M. Bey, «Automatisation de l'Opération de Tréflage.Rapport de recherche,» 2011.

[7] K. Bouhadja, «Classification of simulation methods in machinnig on multi-axis machines,» 2014.

[8] X. X. a. Y. L. Y. Zhang, «Numerical control machining simulation: a comprehensive survey ». International Journal of Computer Integrated Manufacturing, 24,» 2011.

[9] H. K.Lilia, «Conception et Dévlopement d'une Application de simulation d'Enlèvement de Matière Lors de la finition des Surfaces Complexes sur Fraiseuses 05-axes».

[10] Q. C.Zezhong, «A Partical Approach Generationg Steepest Ascent ToolPaths for Three-Axis Finiche Milling of Compound NURBS Surfaces,» 2007.

[11] S.Abainia, «Usinage Techniques de simulation d'Usinages,» 2009.

[12] S. C. ASSOULINE, «SIMULATION NUMERIQUE DE L'USINAGE A L'ECHELLE MACROSCOPIQUE: PRISE EN COPMTE D'UNE PIECE DEFORMABLE,» PARIS, décembre 2005.

[13] W. Z. Y.-S. L. Yongfu Ren, «Tri-Dexel Volumetric Modeling for Haptic Sculpting and Virtual Prototyping of Complex Surfaces,» Proceedings of NAMRI/SME, Vol. 41, State University Raleigh, North Carolina, USA, 2013.

[14] S. W. L. &. A. Nestler, «Virtual workpiece: workpiece representation for material removal process,» International Journal of Advanced Manufacturing Technology, January 2011.

[15] M. B. K.Bouhadja, Survey On Simulation Methods In Multi-Axis Machining, Transactions on Engineering Technologies: World Congress on Engineering: Springer, 2014.

[16] M. L. W. Zhang, «Surface reconstruction using dexel data from three sets of orthogonal rays,» Faculty Research & Creative Works, 2009.

[17] K. M. Laurent Tapie, «Analyses de difficultés d'usinage pour les pièces de,» Nantes, France, 2008.

[18] [En ligne]. Available: http://lurpa.ens-paris-saclay.fr/version-francaise/recherche/equipe-geo3d/modelisation-et-simulation-realiste-du-processus-fao-cn-machine-outil-344683.kjsp?RH=1265970149354.

[19] L. P. G. e. Environnement, «TRAITEMENT NUMÉRIQUE DES IMAGES Classifications non supervisées,» Université Toulouse le Mirail, Toulouse II, 2004.

[20] Z. e. L.Zaoui, "Proposition d'une solution au problème d'initialisation cas du K-means," Université des sciences et de la technologie d'Oran MB, Université Mohamed Boudiaf USTO -BP 1505 El Mnaouer -ORAN - Algérie.

[21] V. M. &. V. J. Likas A., "The global k-means clustering algorithm," Pattern Recognition, 2003.

[22] "Point in triangle test," [Online]. Available: http://blackpawn.com/texts/pointinpoly/.

[23] D.Pastre, "Résolution des systémes linéaires Méthode de Gauss," Université René Descartes UFR de mathématiques et informatique, 2003/2004.

[24] J.-P. Marco et L. Lazzarini (dir.), Mathématiques L1 : Cours complet avec fiches de révision, 1000 tests et exercices corrigés, Pearson, p. 479, 2013.

[25] W. Zhang, "«Virtual Prototyping with Surface Reconstruction and Freeform Geometric Modeling Using Level-set Method ».," Missouri University of Science and Technology, 2008.

[26] "Analyse et Conception des Systèmes d'Information – Méthodes Objet Le langage de modélisation objet UML," pp. 1–44, 1998.

[27] P. Deitel, "*C++ How to progra*"m, 20 Hall, 8eme éd, P 1104, 2011.

[28] The Industry's Foundation for High Performance Graphics. *opengl.* [En ligne] https://www.opengl.org/.

[29] https://en.cppreference.com/w/cpp/compiler_support (dernier accès 03/07/2018)

[30]Embarcadero.http://docwiki.embarcadero.com/RADStudio/Tokyo/fr/Utilisation_de_la_biblioth%C3%A8que_de_programmation_parall%C3%A8le. (Dernier accès 03/07/2018)