UNIVERSITE SAAD DAHLEB DE BLIDA

Faculté des Sciences

Département d'Informatique

MEMOIRE DE FIN D'ETUDES

Pour l'obtention

Du Diplôme de Master en Informatique

Option : systèmes d'informatiques et réseaux

THEME

Vers la Parallélisation de la Génération du Trajet d'Outils pour la Finition des Pièces Complexes sur des Fraiseuses 05-Axes

Réalisé par :

M^elle^. CHIKHAOUI Romaissa

M^elle^. SALEM Saliha

Soutenu devant :

Mr. BEY Mohamed	CDTA	Encadreur
Mr. BENDIFALLAH Hassène	CDTA	Encadreur
Mme. ZAHRA Fatma Zohra	USDB	Promotrice
Mme.lahiani Nesrine	USDB	Président
Mme. Daoud Hayat	USDB	Examinateur

2022/2023

Remerciement

*Nous remercions notre créateur **ALLAH** le tout-puissant, de nous avoir donné les forces, la santé, la volonté et le courage afin d'entamer et de terminer ce modeste travail.*

*Nous tenons à exprimer toute notre reconnaissance à notre encadreur Monsieur **Bey Mohamed** pour son inestimable aide et soutient, ses orientations et conseils précieux, durant toute cette période d'encadrement.*

*Nous tenons à remercier aussi toute l'équipe du **CDTA**.*

Nous remercions nos très chers parents qui ont toujours été là pour nous.

Nous remercions nos familles. Nous adressons nos sincères remerciements à toutes les personnes telle que dans l'impossibilité de citer tous les noms qui par leurs paroles, conseils et leurs critiques ont guidé nos réflexions et ont accepté de nous rencontrer et de répondre à nos questions durant notre recherche. Nos vifs remerciements vont aux membres de jury pour avoir accepté de juger notre travail. Nous voudrons exprimer notre reconnaissance envers nos chères amies et nos collègues qui nous ont apporté leurs soutiens morals et intellectuels tout au long de nos démarches. À tous ces intervenants, nous présentons nos remerciements, notre respect et gratitude.

MERCI A VOUS TOUS

Dédicace

Dédicace en témoignage d'amour et d'affection, je dédie ce travail avec une grande fierté à mes parents qui ont été d'un dévouement exemplaire et d'un réconfort inestimable. À mes frères Aymen et Oussama, mes sœurs Fadwa et Oumaima et toute ma famille en reconnaissance de leurs encouragements. Une spéciale dédicace à mes amies pour leurs sympathies, leurs amitiés et leurs solidarités envers moi.

Romaissa

Dédicace

Je dédie ce modeste travail :

A mes très chers parents pour leurs sacrifices et

la confiance qu'ils m'accordent

Que Dieu me les garde

A mon petit frère et ma sœur

A tous les membres de ma famille

A tous mes amis qui me soutiennent

La liste est longue !

Saliha

Résumé

Le recours à l'usinage sur des machines numériques à 05-axes s'impose pour fabriquer des pièces complexes géométriquement. En ce mode d'usinage, pour chaque point de contact outil-surface, une infinité d'orientations de l'axe de l'outil sont possibles en raison des deux degrés de rotation supplémentaires. La complexité des calculs nécessaires à la génération de la trajectoire de l'outil d'usinage tout en évitant les interférences et les collisions engendre un temps d'exécution très important. En effet, cette procédure est considérée comme un problème de calcul intensif.

Par conséquent, l'objectif de ce travail est la proposition d'une approche permettant la parallélisation du calcul des outils optimums (cylindriques, hémisphériques et toriques) et du trajet d'outils lors de la finition des pièces complexes, définies par leurs modèles STL, en utilisant la stratégie « Z-Constant », sur des fraiseuses numériques 05-axes. Il s'agit de concevoir, de développer et d'intégrer à la plateforme logicielle de l'équipe « CFAO » du CDTA un module logiciel graphique et interactif permettant de réaliser les tâches exigées.

Les tests ont montré l'efficacité de l'approche proposée pour parallélisation des calculs sur des GPUs. En effet, le temps d'exécution a été réduit d'une façon considérable.

Mots Clés : Parallélisation, Combinaison d'Outils Hémisphérique, Finition d'usinage, Interférence, Fraiseuse 05-axes, Générations des trajectoires.

Abstract

The use of machining on 05-axis digital machines is essential to manufacture geometrically complex parts. In this machining mode, for each tool-surface contact point, an infinite number of tool axis orientations are possible due to the two additional degrees of rotation. The complexity of the calculations required to generate the trajectory of the machining tool while avoiding interference and collisions generates a very long execution time. Indeed, this procedure is considered a computationally intensive problem.

Therefore, the objective of this work is the proposal of an approach allowing the parallelization of the calculation of the optimum tools (cylindrical, hemispherical and toric) and the path of tools during the finishing of complex parts, defined by their STL models. , using the “Z-Constant” strategy, on 05-axis CNC milling machines. This involves designing, developing and integrating into the software platform of the “CAD/CAM” team of the CDTA a graphical and interactive software module allowing the required tasks to be carried out.

The tests showed the effectiveness of the proposed approach for parallelizing calculations on GPUs. Indeed, the execution time has been reduced considerably.

Keywords: Parallelization, Combination of Hemispherical Tools, Machining Finishing, Interference, 05-axis milling machine, Path Generations.

ملخص

يعد استخدام الآلات الرقمية ذات المحور 05 أمرًا ضروريًا لتصنيع الأجزاء المعقدة هندسيًا. في وضع المعالجة هذا، لكل نقطة تلامس مع سطح الأداة، من الممكن عدد لا حصر له من اتجاهات محور الأداة بسبب درجتين إضافيتين للدوران. إن تعقيد العمليات الحسابية المطلوبة لإنشاء مسار أداة المعالجة مع تجنب التداخل والاصطدامات يولد وقت تنفيذ طويل جدًا. في الواقع، يعتبر هذا الإجراء مشكلة حسابية مكثفة. لذلك، فإن الهدف من هذا العمل هو اقتراح نهج يسمح بموازنة حساب الأدوات المثلى (الأسطوانية، نصف الكروية) ومسار الأدوات أثناء الانتهاء من الأجزاء المعقدة، المحددة بواسطة نماذج STL الخاصة بهم. باستخدام استراتيجية "Z-Constant" على آلات طحن CNC ذات 5 محاور. يتضمن ذلك تصميم وتطوير ودمج منصة برمجية لفريق "CAD / CAM" التابع لـ CDTA، وحدة برمجية رسومية وتفاعلية تسمح بتنفيذ المهام المطلوبة. أظهرت الاختبارات فعالية النهج المقترح لموازنة الحسابات على وحدات معالجة الرسومات. في الواقع، تم تقليل وقت التنفيذ بشكل كبير.

الكلمات المفتاحية: الموازاة، مجموعة الأدوات نصف الكروية، التشطيب الآلي، التداخل، آلة الطحن ذات المحور 05، توليد المسار

Table des matières

CHAPITRE02 : étude conceptuelle, démarche de résolution et solutions proposes

CHAPITRE03 : implémentation informatique, tests et validations

Liste des Figures

CHAPITRE01 État de l'art

CHAPITRE02: etude conceptuelle, démarche de resolution et solutions proposes

CHAPITRE03 : implémentation informatique, tests et validations

Liste des tableaux

CHAPITRE 01 : État de l'art

CHAPITRE 03 : implémentation informatique, tests et validations

INTRODUCTION GENERALE

Préambule

Le processus de réalisation des pièces de formes complexes par usinage est un processus important dans les domaines de l'aéronautique, de l'automobile, des moules et des matrices. Avec la croissance de la complexité géométrique et technologique des formes complexes, le recours à l'usinage sur des machines numériques multiaxes de 03-axes à 05-axes s'impose.

L'usinage 05-axes est maintenant répandu dans les grands groupes industriels. Le plus souvent, ces formes complexes ne sont pas exprimables par des formules mathématiques connues, ce qui donne le nom des surfaces libres « surfaces gauches ». C'est la raison pour laquelle, des modèles géométriques ont été développés pour les représenter, ce qui rend la conception de ces pièces plus ou moins simples.

Avec l'intégration de l'outil informatique, un nombre important de logiciels de conception « Conception Assistée par Ordinateur » « CAO » sont développés et commercialisés pour aider les concepteurs dans leurs tâches de conception. Une fois la pièce est conçue, il est possible de prendre son modèle en utilisant différents formats d'échange de données standards (STL, STEP, IGES, etc.). Dans certains cas où la géométrie de la pièce est complexe que sa conception dans un logiciel de « CAO » devienne couteuse ou impossible, le processus du Reverse Engineering est utilisé. Dans ce processus, une maquette (prototype) est réalisée manuellement et elle est scannée par un dispositif spécifique pour récupérer un nuage de points très dense. Ce dernier est traité pour le convertir en un modèle triangulé (modèle STL).

Pour la partie production de ces pièces, les machines-outils qui produisent ces pièces ont profité de l'outil informatique. Une commande numérique a été intégrée à ces machines pour automatiser le processus d'usinage. Parmi ces machines, les fraiseuses à 03-axes, à 04-axes et à 05-axes qui sont utilisées pour la fabrication de ces pièces. Leurs formes finales sont obtenues en trois opérations : ébauchage, demi-finition et finition. Pendant l'usinage, l'outil suit une trajectoire complexe dans l'espace 3D et de ce fait des problèmes d'interférence, entre la partie active de l'outil et la surface à usiner notamment au niveau des interférences globales représentant une collision entre l'outil et son environnement.(machine, outils, pièce, …etc.), Pour cela, il est impératif d'éliminer ces problèmes avant de lancer l'usinage réel.

Dans la pratique industrielle, un compromis est toujours recherché entre la qualité des pièces usinées et les temps d'usinage pour réduire les coûts. Afin d'optimiser les temps d'usinage et d'améliorer la qualité des pièces, plusieurs approches développées, considèrent soit les conditions d'usinage soit les stratégies d'usinage soit la sélection des outils optimums. Dans cet ordre d'idées et dans le cadre du programme de recherche piloté par l'équipe « CFAO » « Conception et Fabrication Assistées par Ordinateur » du « CDTA » « Centre de Développement des Technologies Avancés », il nous a été demandé de proposer une approche permettant la parallélisation du calcul des outils optimums hémisphériques et l'évitement des collisions lors de la finition des pièces complexes.

Problématique

Les pièces mécaniques de formes complexes sont utilisées dans diverses industries automobile, aérospatial, ...etc. Elles sont conçues pour répondre à des contraintes fonctionnelles et/ou de style. Elles sont caractérisées par une topologie complexe composée d'un ensemble de surfaces concaves, convexes ou à double courbure. Avec la croissance de la complexité géométrique et technologique des formes complexes, le recours à l'usinage sur des machines numériques à 05-axes s'impose. Pour ce mode d'usinage, chaque point de contact outil-surface, une infinité d'orientations de l'axe de l'outil sont possibles en raison des deux degrés de rotation supplémentaires. Ces orientations augmentent les conditions d'accès de l'outil aux différentes zones de la pièce mais d'un autre côté, les risques de collisions et d'interférences deviennent plus importants. Parmi toutes les orientations possibles, il est indispensable de choisir celles évitant ces problèmes et optimisant l'usinage.

Objectif du travail

Ce travail s'insère dans le cadre de développement d'une plateforme logicielle graphique et interactive sous Windows pour la production des pièces 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 finition des pièces complexes, définies par leurs modèles STL, en utilisant des outils hémisphériques sur des fraiseuses numériques à 05-axes, en considérant la stratégie d'usinage « Z-Constant ». L'objectif de ce projet est la proposition d'une approche permettant la parallélisation du calcul des outils hémisphériques optimums et leurs orientations permettant d'éviter les problèmes d'interférences et de collisions, leurs combinaisons ainsi que la génération du trajet d'outils. Il s'agit de concevoir, de développer et

d’intégrer à la plateforme logicielle de l’équipe « CFAO » un module logiciel graphique et interactif permettant de réaliser les tâches exigées. Ce travail permettra de réduire le cycle de développement de nouveaux produits, d’augmenter la productivité et de réduire les temps d’usinage et par conséquent les coûts. De plus, la topologie des surfaces exige la combinaison de différentes formes d’outils dans le but d’améliorer la qualité des surfaces usinées. Dans le but de minimiser les temps d’analyse et de calcul, il est très judicieux de penser à paralléliser les calculs séquentiels en utilisant les performances des cartes graphiques « GPU ».

Structuration du mémoire

Le présent mémoire est composé des chapitres suivants :

- est consacré à l’étude du processus d’usinage des pièces complexes et l’étude du calcul parallèle.
- Le deuxième chapitre est réservé à la présentation des solutions proposées.
- Le troisième chapitre détaille l’implémentation informatique et présente les tests réalisés pour valider l’approche proposée.

Ce mémoire se termine par une conclusion générale et des perspectives.

CHAPITRE 01 :

État de l'art

1. Introduction

La surface d'une pièce est sa peau qui la limite dans l'espace et la sépare du milieu environnant. Elle assure plusieurs fonctions telles que l'aptitude aux frottements et la résistance aux contraintes mécaniques. Les surfaces les plus répandues dans le domaine de l'industrie manufacturière sont les surfaces complexes appelées « surfaces gauches ». Elles sont utilisées dans l'industrie automobile, les coques de navires, les pièces en aérospatial, etc. Elles sont caractérisées par une topologie complexe. Le processus de réalisation de ces surfaces a subi de grandes évolutions ces dernières décennies par l'introduction de la « CFAO » « Conception et Fabrication Assistées par Ordinateur » et de la commande numérique dans les machines de production. Ce processus passe par des étapes successives et complémentaires : conception, simulation, production et contrôle de la pièce réalisée.

La suite de ce chapitre est organisée en trois parties. Dans la première partie, une présentation du processus de production des pièces mécaniques. La deuxième partie est consacrée à l'étude des méthodes de représentation et de conception des surfaces gauches. La dernière partie est réservée au processus de production des surfaces gauches en 05-axes.

2. Processus de production de pièces mécaniques

Le processus de production des pièces mécaniques passe par les étapes suivantes :

- ***Besoin :*** la nécessité est la mère de l'invention.
- ***Naissance de l'idée :*** c'est l'analyse des besoins.
- ***Conception « CAO » :*** c'est l'étude, la recherche et l'élaboration du cahier des charges. Le recours à des outils de conception tels que la « CAO » (Conception Assistée par Ordinateur) pour modéliser un objet en 3D[1]**.**
- ***Fabrication « FAO » :*** la Fabrication Assistée par Ordinateur « FAO » permet de générer le programme de pilotage d'une machine-outil à commande numérique « MOCN ».

Dans le cas où les modules de « CAO » et de « FAO » sont dans le même environnement, les informations circulent directement entre les deux. Dans le cas contraire, la circulation des données requiert des formats d'échange de données tels que STEP, IGES, STL, etc [1].

- ***Choix des outils :*** il comprend les formes, les dimensions et le matériau.
- ***Usinage :*** il définit les techniques de fabrication de pièces mécaniques. Le processus d'usinage d'une pièce passe par trois opérations (ébauchage, demi-finition et finition).

3. Représentation des surfaces

3.1. Définition des surfaces complexes

Appelées aussi surfaces gauches. Elles sont caractérisées par une topologie complexe. Les modèles des surfaces sont classés en deux grandes catégories.

3.1.1. Modèles non paramétriques

Ces surfaces peuvent être représentées sous deux différentes formes[2] :

- ***Forme explicite :*** elle est définie par l'équation suivante (Figure 1) :

$$Z = F(X, Y) \tag{1.1}$$

- ***Forme implicite :*** elle est définie par une fonction de trois variables (Figure 2)[2]:

$$F(X, Y, Z) = 0 \tag{1.2}$$

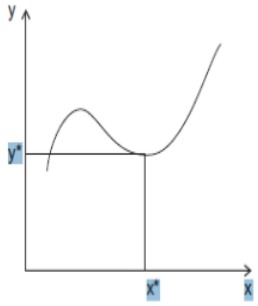


Figure 1 : Forme explicite d'une surface.

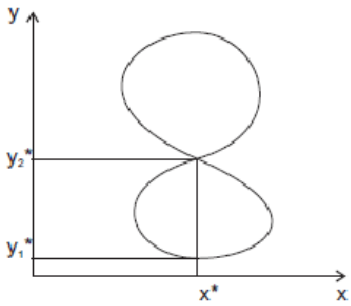


Figure 2 : Forme implicite d'une surface.

3.1.2. Modèles paramétriques

Ces modèles permettent de créer une représentation continue des surfaces d'un objet à l'aide de formulations polynômiales ou rationnelles définies dans un espace bi-paramétrique**[2].**

3.1.2.1. Définition des surfaces paramétriques

Une surface paramétrique est définie par un ensemble de trois fonctions, une pour chaque coordonnée. Ces fonctions dépendent de deux paramètres *u* et *v* appartenant à l'intervalle [0,1]. Une surface paramétrique est donnée par la forme suivante :

$$F(u, v) = (x(u, v), y(u, v), z(u, v)) \tag{1.3}$$

Pour chaque (*u*, *v*) lui correspond un point F (*u*, *v*) sur la surface paramétrique (Figure 3). Plusieurs surfaces paramétriques sont joints pour former une surface plus complexe **[2].**

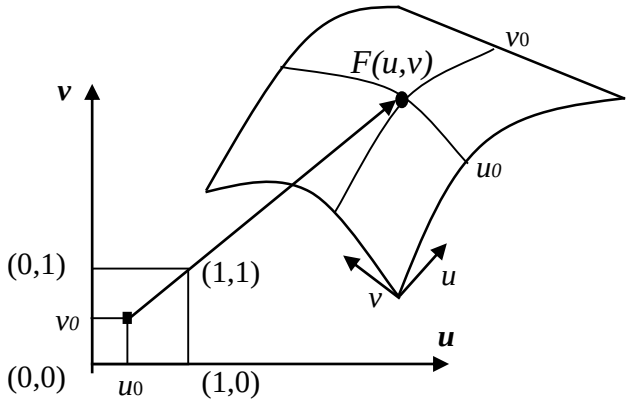


Figure 3 : Surface paramétrique**[2]**.

3.1.2.2. Propriétés géométriques des surfaces paramétriques

- ***Vecteurs tangents :*** les vecteurs tangents à la surface paramétrique au point **F** (*u*, *v*) sont (Figure 4) :
  - Le vecteur tangent dans la direction *u* **:**

$$\overrightarrow{T_u} = \frac{\partial F}{\partial u} = \left(\frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u}\right) \quad (1.4)$$

  - Le vecteur tangent dans la direction *v* :

$$\overrightarrow{T_v} = \frac{\partial F}{\partial v} = \left(\frac{\partial x}{\partial v}, \frac{\partial y}{\partial v}, \frac{\partial z}{\partial v}\right) \quad (1.5)$$

Ces deux vecteurs définissent le plan tangent à la surface au point **F**(*u*, *v*) (Figure 4).

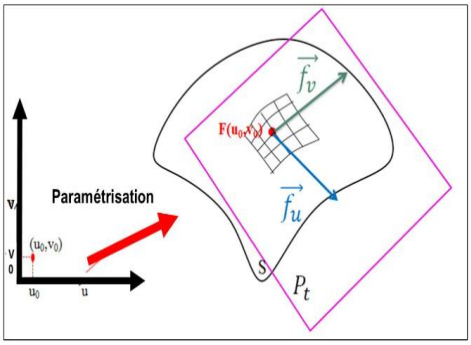


Figure 4 : Plan tangent à une surface paramétrique en un point.

- ***Vecteur normal :*** le vecteur normal unitaire **n**(*u*, *v*) à la surface au point **F**(*u*, *v*) est le produit vectoriel des deux vecteurs tangents (Figure 5). Il est donné par :

$$\vec{n} = \frac{\overrightarrow{T_u} * \overrightarrow{T_v}}{\left\| \overrightarrow{T_u} * \overrightarrow{T_v} \right\|} \quad (1.6)$$

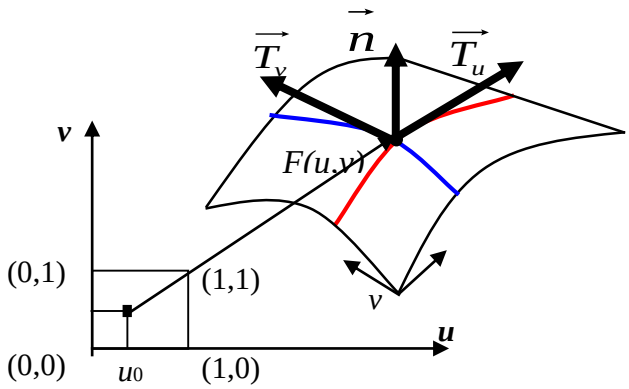


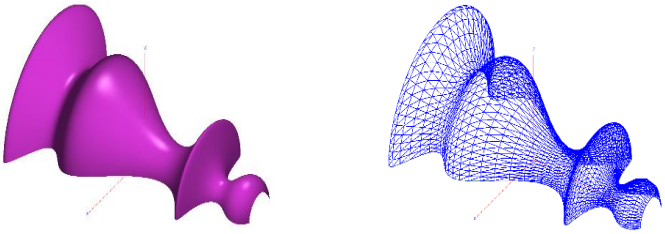
Figure 5 : Vecteurs tangents et vecteur normal d'une surface paramétrique.

4. Formats d'échange de données

Dans un système où la « CAO » et la « FAO » sont intégrées dans un seul environnement, l'information passe directement. Dans le cas où la « FAO » et la « CAO » sont dans deux environnements différents, il est nécessaire d'utiliser des formats d'échange de données « translateurs » pour assurer le passage de l'information entre la « FAO » et la « CAO ». Pour cela, plusieurs formats d'échange de données sont créés tels que IGES, STL, STEP, …etc**[1]**.

4.1. Format STL

Le format STL a été développé pour exporter les modèles « CAO » vers les machines de prototypage rapide. Le format STL ne décrit que les surfaces d'un objet en 3D (Figure2) **[3]** par la décomposition des surfaces en triangles chacun d'eux est décrit par son vecteur normal et par ses trois sommets (Figure 7)[3].



a. Surface de base. **b**. Modèle STL.

Figure 6 : Exemple d'un model STL**.**

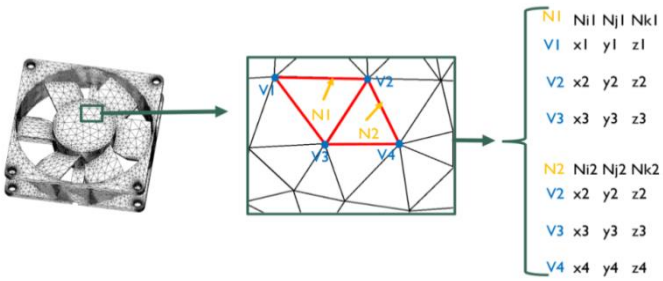


Figure 7 : Format STL.

4.2. Structures d'un fichier STL

Il existe deux Formats de fichier STL, un format ASCII et un format binaire. Le format est spécifié comme un fichier ASCII qui est plus descriptif et lisible mais gros de taille et un fichier binaire, moins de taille mais non lisible et trop condensé. Le Format ASCII commence par une ligne de description précédée par le mot réservé « Solid ». Par la suite, chaque triangle est défini par les composantes de sa normale et par les coordonnées de ses trois sommets. La fin du fichier est indiquée par « endsolid » (Figure 8**).**

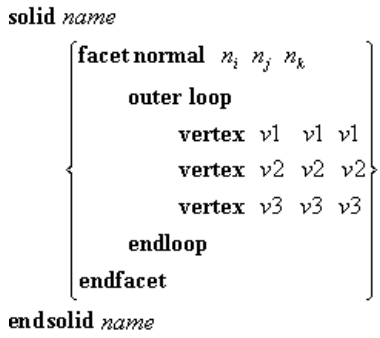


Figure 8 : Syntaxe d'un fichier STL ASCII.

4.3. Caractéristiques du format STL

Les caractéristiques du format STL sont présentées dans le tableau suivant **:**

Tableau 1. Caractéristiques du format STL.

Format	Géométrie	Volume	Texture couleur	Echelle	Représentation	Vecteur normal
Simple ; des triangles	3D	Fermé	Aucune	Aucune	Pas d'équation, approximation	Dirigé vers l'extérieur

❖ **Avantages du format STL**

- Le format STL est simple et il est très répondu.
- Il est facile à trancher pour calculer les intersections des triangles avec des plans.

❖ **Inconvénients du format STL :** pour représenter fidèlement des formes très complexes, il est indispensable d'augmenter le nombre de triangles, ce qui rend le fichier très volumineux.

5. Usinages des surfaces gauches

La conception et la fabrication sont deux fonctions essentielles du processus de réalisation d'un produit. La tendance actuelle consiste à utiliser des processus commandés numériquement

appelés les systèmes « CFAO ». Ce processus doit permettre de garantir la fidélité entre la pièce produite et les spécifications fonctionnelles. La production d'une pièce passe par (Figure 9) :

- Conception de la pièce : la pièce est conçue sur ordinateur grâce aux outils de « CAO ».
- Génération des trajectoires d'usinage : ces trajectoires indiquent les chemins suivis par les outils pour usiner la pièce.
- Usinage de la pièce : il est réalisé en traduisant les trajectoires d'usinage en un programme « G-Code » à exécuter par la commande numérique de la machine-outil[1]**.**

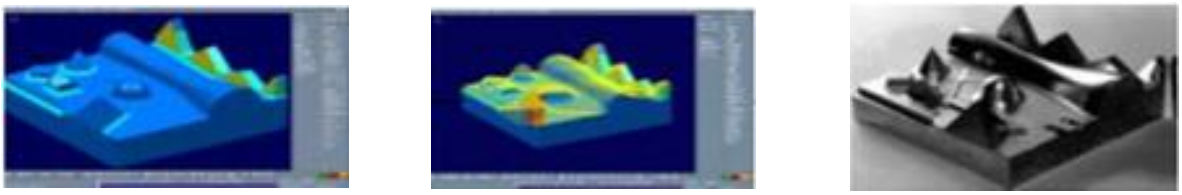


Figure 9 : Processus d'usinage.

5.1. Usinage des pièces complexes

L'usinage consiste à enlever de la matière pour donner à la pièce brute la forme et les dimensions exigées dans les dessins techniques à l'aide d'une machine-outil. Lors de l'usinage, l'enlèvement de matière est réalisé par la combinaison 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). Les procédés d'usinage les plus fréquents sont le tournage et le fraisage. Dans ce travail, le fraisage est considéré dans lequel l'outil est animé d'un mouvement de rotation (mouvement de coupe) et la pièce est animée d'un mouvement de translation (mouvement d'avance). Les surfaces gauches sont usinées sur des fraiseuses à 03-axes, à 04-axes et à 05-axes en fonction de leurs complexités géométriques**.**

5.2. Machine-outil à commande numérique

Une machine-outil est destinée à exécuter un usinage, ou autre tâche répétitive, avec une précision et une puissance adaptées. C'est un moyen de production destiné à maintenir un outil fixe, mobile, ou tournant, et à lui imprimer un mouvement pour usiner ou déformer une pièce ou un ensemble fixé sur une table fixe ou mobile. Une machine-outil à commande numérique « MOCN » est une machine-outil qui fonctionne d'une manière automatique. Elle se compose d'une partie opérative et d'une partie commande. Les déplacements des outils sont décrits dans un programme [4]**.**

5.2.1. Orientations et axes normalisés

C'est la direction suivant laquelle le mouvement est commandé numériquement en continu en position et en vitesse (Figure 10).

- Axe Z : il est parallèle à l'axe de la broche principale.
- Axe X : est associé au mouvement qui définit le plus grand déplacement dans le plan perpendiculaire à l'axe Z.
- Axe Y : il forme avec les axes X et Z un trièdre de sens direct.
- Axes A, B et C : rotations autours des axes X, Y et Z respectivement.

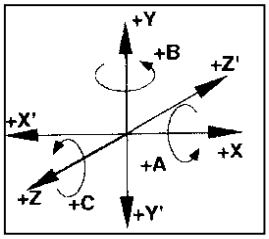
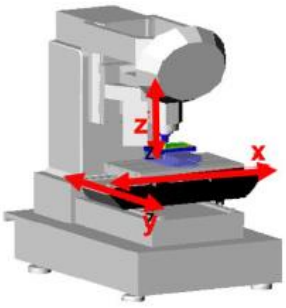


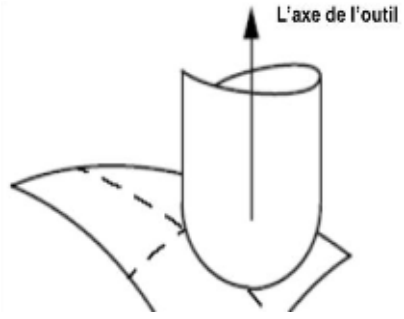
Figure 10 : Axes d'une machine-outil.

5.2.2. Usinage en 03-axes

Dans ce mode, trois translations simultanées suivant les axes X, Y et Z de la machine sont possibles (Figure 11.a). Lors de la finition en 03-axes, pour chaque point de contact entre l'outil et la surface, il existe une unique position de l'outil tangente à la surface (Figure 11.b). Cette technique est plus appropriée pour des pièces sans parties en contre-dépouille[4]**.**



a. Fraiseuse 03-axes.



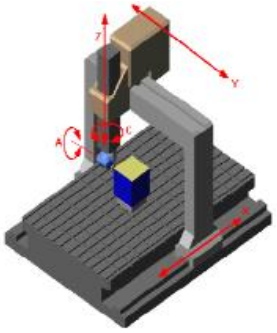
b. Orientation de l'outil en 03-axes.

Figure 11 : Usinage en 03-axes.

5.2.3. Usinage en 05-axes :

L'usinage en 05-axes fait appel à des machines-outils à cinq (05) degrés de liberté correspondant aux trois (03) axes linéaires de translation, auxquels s'ajoutent deux axes rotationnels (Figure 12.a). Une telle cinématique permet d'approcher la pièce dans toutes les

directions et de la traiter sur cinq côtés en une seule opération (Figure 12.b). Cette technique est adaptée aux pièces complexes avec des parties en contre-dépouille [3]**.**



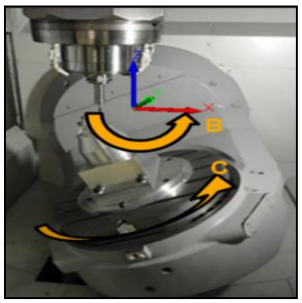
a. Fraiseuse 05-axes.

b. Orientation de l'outil en 05-axes.

Figure 12 : Usinage en 05-axes.

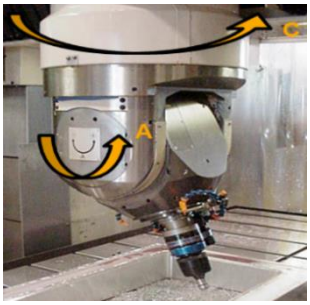
Les configurations possibles des fraiseuses 05-axes sont les suivantes [4]:

- Deux orientations sur la table (Figure 13 a).
- Orientation sur la table et orientation sur la tête (Figure 13. b).
- Deux orientations sur la tête (Figure 13. c).



a. Table seule.

b. Table et tête.



c. Tête seule.

Figure 13 : Répartition des axes de rotation.

5.3. Phases d'usinage

Pour transformer une pièce brute de son état initial à son état final (pièce finie), il faut passer par trois phases (Figure 14) :

- ✓ ***Ébauchage :*** pour l'enlèvement d'un maximum de matière en un minimum de temps.
- ✓ ***Demi-finition*** *:* pour avoir une surépaisseur d'usinage uniforme.
- ✓ ***Finition*** *:* pour obtenir la forme finale de la pièce

a. Ebauchage.

b. Demi-finition.

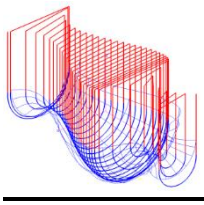
c. Finition.

Figure 14 : Phases d'usinage.

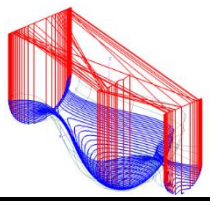
5.4. Stratégies d'usinage en finition

Une stratégie d'usinage est une méthodologie utilisée pour générer une série d'opérations pour réaliser une forme donnée. Elle permet d'associer un ensemble d'opérations comprenant la définition des outils, des conditions de coupe et des trajectoires d'usinage. Comme il n'y a pas une forme d'outil permettant de générer des surfaces gauches directement, ces dernières sont usinées par balayage de plusieurs outils selon une direction privilégiée (mode de balayage). Le choix d'un mode est basé sur un critère d'optimisation du temps d'usinage ou de la qualité des surfaces. Les stratégies les plus utilisées pour la finition des surfaces gauches sont **:**

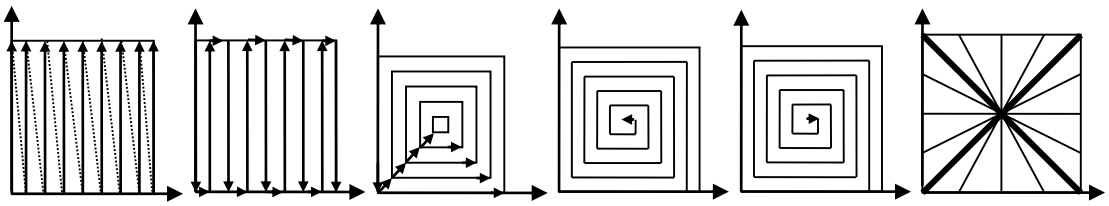
- Plans parallèles : le trajet d'usinage est obtenu par la construction des courbes d'intersection de la surface avec des plans verticaux parallèles (Figure 15.a).
- Z-Constant : le trajet d'usinage est obtenu par l'intersection de la surface avec des plans horizontaux avec différentes valeurs de Z (Figure 15.b).
- Iso paramétriques : le trajet d'usinage est généré dans le plan paramétrique de la surface et ensuite transformé dans l'espace 3D (Figure 15.c**)**.



a. Plans parallèles.



b. Z-Constant.



c. Iso paramétriques.

Figure 15 : Stratégies d'usinage en finition des surfaces gauches.

5.5. Outils de finition

Trois formes d'outils sont utilisées dans l'usinage des pièces de formes complexes [**3**] **:**

❖ ***Fraises cylindriques :*** elles conviennent aux travaux d'ébauche et de demi-finition (Figure 16.a). Leur avantage majeur réside dans leurs parties actives, le cylindre situé sur un rayon éloigné de l'axe de l'outil où la vitesse est non nulle.

❖ ***Fraises hémisphériques :*** ces fraises peuvent pivoter autour du point centre outil (Figure 16.b). Leurs positionnements font apparaître un angle d'inclinaison de la broche pour éloigner le point de contact de l'axe d'outil pour obtenir une vitesse de coupe plus élevée**[5].**

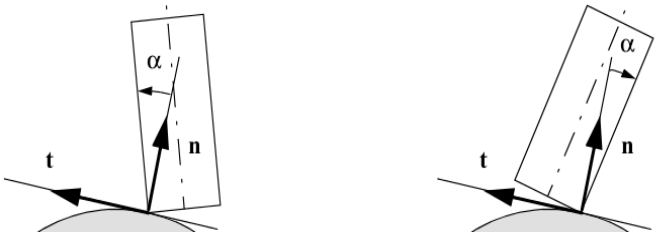
❖ ***Fraises toriques :*** elles ont une partie active, le tore, situé sur un rayon éloigné de l'axe de l'outil. La conséquence directe est que la vitesse de coupe est non nulle (Figure 16.c**)[5].**

a. Cylindrique. b. Hémisphérique. c. Torique.

Figure 16 : Formes de fraises.

5.6. Positionnement des différentes formes d'outils

Le positionnement d'un outil sur une surface gauche est décomposé en deux étapes : la définition du point de contact outil/pièce et ensuite l'orientation de l'axe de l'outil. L'axe de l'outil n'est pas forcément aligné avec la normale à la surface au point considéré. Il est possible d'introduire un angle α positif entre l'axe de l'outil et la normale dans le plan (n, t) qui inclinera l'outil dans le sens du mouvement d'avance (Figure 17). Il peut aussi introduire un angle α négatif c'est à dire dans le sens inverse du mouvement d'avance**[5].**

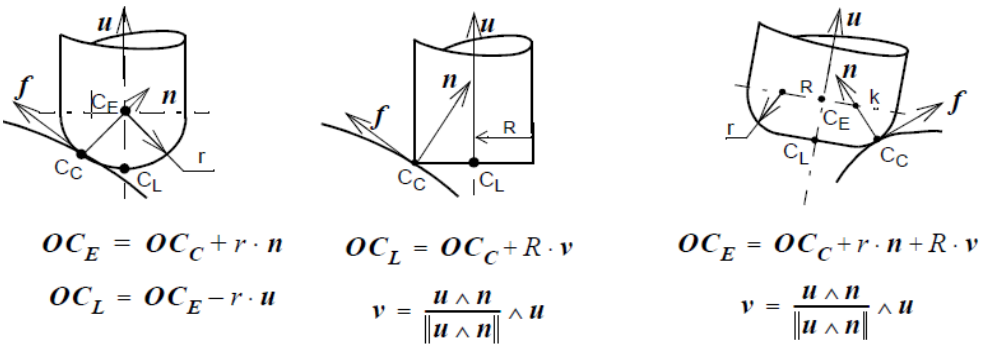


a. Détalonnage positif : α>0 b. Détalonnage négatif : α<0

Figure 17 : Types de détalonnage.

Le positionnement de l'outil sur la surface consiste à déterminer un couple (P, u), où P représente le point piloté, caractéristique de la géométrie de l'outil, et u le vecteur directeur de

l'axe de l'outil. Le point piloté de l'outil est le point CL (Cutter Location), point extrémité de l'outil. Ce couple est défini localement par rapport à la surface à usiner au niveau du point de contact outil-pièce C_C (Figure 18). Le repère local (C_C, f, n, t) est tel que f représente le vecteur tangent à la courbe suivie, n le vecteur normal à la surface, et t le vecteur issu du produit vectoriel de f par n. La (Figure 18) positionne les points caractéristiques de l'outil suivant sa géométrie. C_E représente le centre de l'outil, R le rayon principal de l'outil et r le rayon du tore[4].



a. Hémisphérique. b. Cylindrique. c. Torique.

Figure 18 : Points caractéristiques suivant la géométrie de l'outil.

Une fois la position de l'extrémité de l'outil déterminée, il reste à définir l'orientation de son axe. Elle est donnée dans le repère local (C_C, f, n, t) par deux angles de rotation (Figure 19). Deux modes sont utilisés pour l'inclinaison de l'axe de l'outil :

❖ ***Premier mode :*** l'orientation de l'axe de l'outil est obtenue par une rotation de θt autour de t suivie d'une rotation de θn autour de n**[3].**

❖ ***Deuxième mode :*** l'orientation de l'axe de l'outil est obtenue par une rotation de α dans le plan contenant la direction d'avance f suivie d'une rotation de β dans le plan perpendiculaire à la direction d'avance f**[3].**

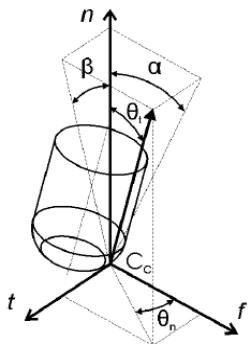
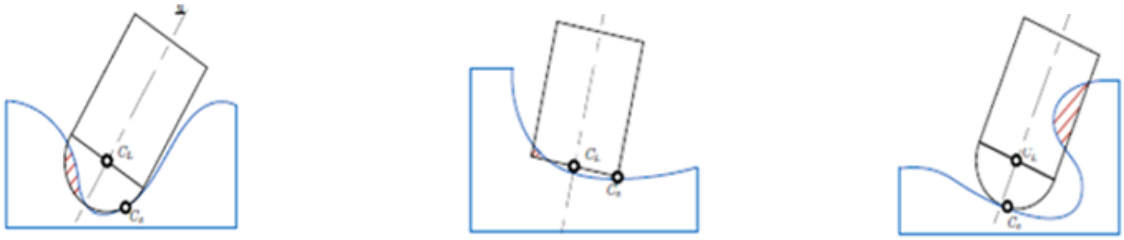


Figure 19 : Angles d'orientation de l'axe outil.

5.7. Problèmes d'interférences

L'usinage en 05-axes est puissant et peut résoudre de nombreux problèmes rencontrés en usinage 03-axes. Malgré cet avantage, les deux rotations supplémentaires augmentent considérablement les risques d'interférences et de collisions. Ces problèmes peuvent apparaître au niveau des positionnements d'outils calculés, lors de l'interpolation de la trajectoire ou lors des mouvements hors matière. L'orientation efficace de l'axe de l'outil pour un point de contact outil pièce donné, représente le problème majeur le plus rencontré en usinage 05-axes. Lors du positionnement de l'outil, trois (03) types de problèmes peuvent être rencontrés.

- ***Interférences locales :*** elles traduisent un enlèvement de matière excessif par la partie active de l'outil sur la surface à usiner (Figure 20.a)**[3].**
- ***Interférences vers l'arrière :*** elles sont des pénétrations intempestives de l'arrière de l'outil dans la surface à usiner (Figure 20.b)**[3].**
- ***Collisions (interférences globales) :*** elles représentent des collisions entre l'ensemble (corps d'outil, porte outil, broche) et l'ensemble (pièce, porte pièce, etc.) (Figure 20.c)**[3].**



a. Interférence locale. b. Interférence vers l'arrière. c. Collision.

Figure 20 : Types de problèmes d'usinage.

5.8. Evitement des interférences et des collisions

Pour un point de contact C_C, les paramètres d'orientation de l'outil choisis ne garantissent pas un usinage sans interférences et sans collisions. La gestion des interférences est un problème difficile à résoudre en usinage 05-axes. Parmi les méthodes de détection des interférences existantes, il y a celles qui sont basées sur les caractéristiques locales de la surface à usiner et celles qui optent pour une représentation approchée par des points ou des modèles polyédriques. L'objectif consiste alors à positionner l'outil hors interférence afin d'obtenir un enlèvement de matière maximum. Pour éviter les interférences et les collisions, deux solutions existent :

- Changement d'outil ou bien l'orientation de l'outil.

- Pour les collisions, il suffit juste de réorienter l'outil, quel que soit sa géométrie.

La maîtrise des interférences et des collisions lors de la génération de trajectoires d'outils est une condition nécessaire pour garantir la conformité de la pièce en termes de géométrie et de contraintes imposées dans le cahier des charges[6]**.**

5.9. Génération du trajet d'outil en 05-axes

La génération d'un trajet d'outils sans interférences et sans collisions pour l'usinage des pièces complexes sur des fraiseuses en 05-axes passe principalement par les étapes suivantes :

1. Calcul des points de contact C_C entre la surface à usiner et l'outil.
2. Calcul de l'outil optimum pour chaque point de contact évitant les interférences.
3. Sélection du nombre d'outils à utiliser.
4. Affectation des outils aux points de contact C_C.
5. Détermination de l'orientation de l'outil évitant les collisions.
6. Simulation virtuelle de l'usinage.
7. Simulation virtuelle des mouvements de la machine d'usinage.
8. Génération du trajet d'outil.

Ce processus requiert un temps de calcul très important **[3]** et il est considéré comme un problème de calcul intensif. Par conséquent il est indispensable de réduire le temps de calcul afin de minimiser les coûts, le cycle de développement de nouveaux produits et augmenter la productivité.

6. Définition de calcul intensif

Le calcul intensif est un domaine de l'informatique qui consiste à utiliser des systèmes de traitement haute performance pour résoudre des problèmes complexes nécessitant une quantité massive de calculs. Les systèmes de calcul intensif peuvent inclure des ordinateurs centraux de grande capacité, des clusters de calcul, des systèmes de grille et des cartes graphiques (GPU).). Les applications de calcul intensif incluent la simulation numérique, le traitement de données massives et la visualisation scientifique. Les algorithmes de calcul intensif sont souvent basés sur des techniques de traitement parallèle et distribué pour augmenter la vitesse et l'efficacité des calculs**[7]**.

A. Calcul intensif distribué : Le calcul intensif distribué est une méthode d'exécution de tâches de calcul intensif qui utilise plusieurs ordinateurs pour traiter plusieurs tâches

simultanément. Cela peut se faire en utilisant un nœud de traitement unique qui distribue les tâches à plusieurs ordinateurs ou en utilisant un réseau d'ordinateurs pour traiter les tâches de manière simultanée. Le calcul intensif distribué est souvent utilisé pour effectuer des tâches complexes qui nécessitent un grand nombre de ressources de calcul**[7].**

B. **Calcul intensif parallèle :** Le calcul intensif parallèle est une méthode d'exécution de tâches de calcul intensif qui utilise plusieurs processeurs pour traiter une seule tâche de manière simultanée. Cela peut se faire en utilisant des CPU multi-cœurs ou des GPU pour accélérer les tâches de calcul intensif. Le calcul intensif parallèle est souvent utilisé pour accélérer les applications qui exécutent des algorithmes mathématiques complexes**[7].**

Les GPUs sont particulièrement adaptés aux calculs massivement parallèles, qui peuvent être effectués plus rapidement qu'avec un processeur central (CPU) traditionnel donc dans notre projet nous utilisons le parallélisme

6.1. Définition des GPU

GPU est l'acronyme de Graphics Processing Unit (unité de traitement graphique en français). Communément appelée processeur graphique, c'est une puce informatique qui permet d'optimiser l'affichage et le rendu des images 2D et 3D ainsi que des vidéos. Il effectue des calculs mathématiques très rapides qui sont entièrement dédiés au traitement de données graphiques. Le processeur graphique (GPU) n'est pas à confondre avec la carte graphique ou carte vidéo puisqu'il s'agit de deux composants distincts. En effet, le GPU est un élément constitutif de la carte graphique d'un appareil informatique. Il est associé à la mémoire et à d'autres composants qui permettent à la carte vidéo de fonctionner. Le GPU représente donc pour une carte graphique ce qu'est le CPU pour la carte mère**[8].**

6.2. La différence entre une CPU et un GPU

La principale différence entre un CPU (unité centrale de traitement) et un GPU (unité de traitement graphique) réside dans leur architecture et leur fonctionnement.

Le CPU est conçu pour effectuer des tâches générales de traitement de données. Il est optimisé pour effectuer des opérations séquentielles, où chaque opération doit être effectuée avant de passer à la suivante. Les processeurs modernes ont souvent plusieurs cœurs, ce qui leur permet de traiter plusieurs tâches en parallèle.

Le GPU, quant à lui, est conçu pour accélérer le traitement des opérations graphiques et des calculs parallèles. Il est optimisé pour effectuer de nombreuses opérations en parallèle sur de grandes quantités de données, ce qui le rend particulièrement utile pour des applications telles que l'infographie, la modélisation 3D, la reconnaissance d'images et l'apprentissage en profondeur.

En termes simples, le CPU est idéal pour les tâches qui nécessitent une puissance de traitement générale, tandis que le GPU est plus efficace pour les tâches hautement parallèles impliquant de grandes quantités de données

Tableau 2 : la différence entre une CPU et un GPU**[9]**.

CPu	**Gpu**
unité centrale de traitement	unité de traitement graphique
quelques cœurs	Beaucoup de cœurs
Faible latence	Haut débit
Bon pour le traitement en série	Bon pour les traitements en parallèle
Peut effectuer une poignée d'opérations à la fois	Peut effectuer des milliers d'opérations à la fois

6.3. Types des GPUs

Les GPU disponibles sur le marché actuellement peuvent être classés en deux familles : les GPU intégrés et ceux dédiés (discrète GPU). La différence entre ces deux familles de processeurs graphiques se situe au niveau de leur architecture et de leur mode d'utilisation.

6.3.1. Les GPU intégrés

GPU intégré : Les GPU intégrés, également appelés IGP (Integrated Graphics Processor), sont intégrés directement sur la carte mère d'un ordinateur ou dans le processeur lui-même. Ils partagent les ressources système (telles que la mémoire) avec le processeur principal de l'ordinateur. Les GPU intégrés offrent des performances graphiques de base et sont couramment utilisés dans les ordinateurs portables et les ordinateurs de bureau grand public. Ils sont adaptés aux tâches quotidiennes, à la navigation web, au multimédia et à certains jeux moins exigeants**[10].**

6.3.2. Les GPU dédiés

Les GPU dédiés, également appelés cartes graphiques dédiées, sont des cartes d'extension distinctes qui se branchent sur la carte mère de l'ordinateur. Contrairement aux GPU intégrés, les GPU dédiés disposent de leur propre mémoire vidéo dédiée et de ressources système indépendantes. Ils offrent des performances graphiques plus élevées, une meilleure qualité d'image et sont capables de gérer des charges de travail plus lourdes, telles que les jeux vidéo modernes, la modélisation 3D avancée, l'édition vidéo professionnelle et les applications de calcul intensif**[10].**

Les GPU dédiés sont généralement plus puissants et plus performants que les GPU intégrés en raison de leurs caractéristiques spécifiques et de leur architecture dédiée au traitement graphique. Ils sont également plus flexibles en termes d'extensions et de mises à niveau, car il est possible de remplacer ou d'ajouter une carte graphique dédiée pour améliorer les performances graphiques d'un système.

6.4. GPGPUS

Les GPU Sont des processeurs qui peuvent fonctionner en parallèle en exécutant un seul noyau sur plusieurs enregistrements d'un flux à la fois. Un flux est simplement un ensemble d'enregistrements qui nécessitent un calcul similaire

6.4.1. Programmation des GPUs

Les avantages de la programmabilité GPU sont si irrésistibles que de nombreux programmeurs ont commencé à réécrire leurs applications à l'aide de GPGPU. De cette façon, ils peuvent utiliser la capacité de calcul élevée du GPU. Il existe plusieurs SDK et API disponibles pour la programmation des GPU pour un calcul à usage général qui est fondamentalement autre qu'à des fins graphiques, par exemple NVIDIA CUDA, ATI Stream SDK, Open CL, HMPP et accélérateur PGI. La sélection de la bonne approche pour accélérer un programme dépend d'un certain nombre de facteurs, notamment le langage actuellement utilisé, la portabilité, les fonctionnalités logicielles prises en charge et d'autres considérations en fonction du projet. À l'heure actuelle, CUDA est la méthode prédominante pour l'accélération GPGPU, bien qu'il n'est pris en charge que par les GPU NVIDIA. À plus long terme, Open CL promet de devenir la norme indépendante du fournisseur pour la programmation d'architectures multi cœurs hétérogènes. Comme il est largement basé sur CUDA, il y aura, espérons-le, relativement peu de difficultés pour passer de l'API propriétaire de CUDA à Open CL**[11].**

7. Conclusion

Dans ce chapitre, nous avons présenté d'une manière générale les différentes méthodes de modélisation des surfaces gauches en mettant en avant le modèle discret plus particulièrement le modèle « STL » support de notre étude. Par la suite, nous avons explicité le processus d'usinage des surfaces gauches sur des fraiseuses numériques à 05-axes et les différentes stratégies d'usinages. Et la présentation des machines-outils à commande numérique et les différents outils d'usinages. Ensuite, nous avons présenté le calcul intensif et les solutions de parallélisation sur des GPUs.

.

CHAPITRE 02 :

ETUDE CONCEPTUELLE, DEMARCHE DE RESOLUTION ET SOLUTIONS PROPOSEES

1. Introduction

Après avoir présenté dans le chapitre précédent un état de l'art et des généralités indispensables pour la compréhension de notre travail et pour donner une vue générale de notre application. Le présent chapitre sera consacré à cerner les étapes entreprises pour réaliser ce travail. Ces étapes sont, en premier lieu, présentées dans l'architecture générale de l'application.

2. Architecture générale de l'application logicielle

Les principales étapes de l'application logicielle à développer, dans un ordre chronologique, sont décrites par l'organigramme général donné par la Figure 21.

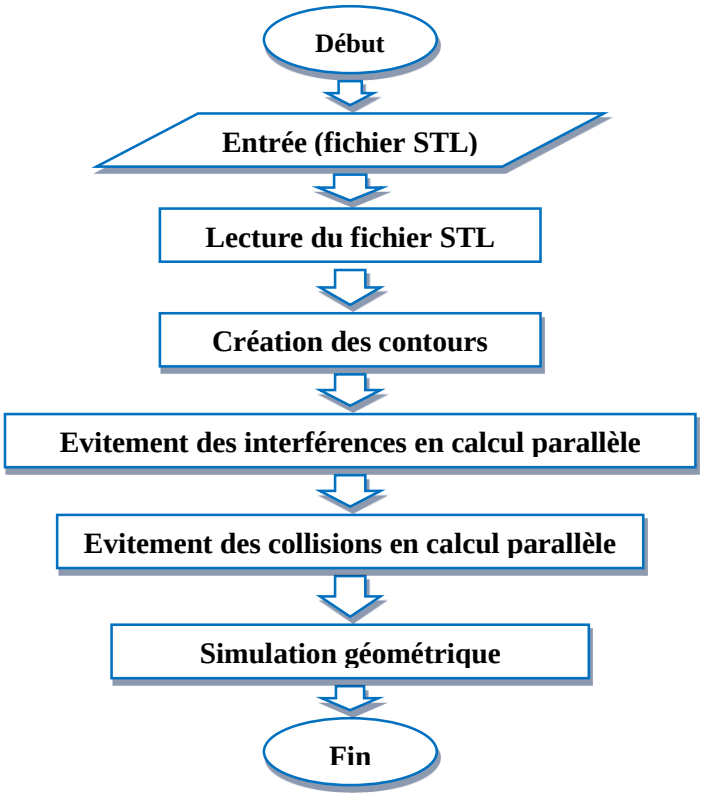


Figure 21 : Organigramme général.

Chaque grande étape comporte plusieurs sous-tâches :

1. Lecture de la forme de la pièce à usiner à partir d'un fichier STL :
    - Récupération des paramètres de la forme brute.
    - Création du vecteur des sommets.
    - Création du vecteur des triangles.
    - Estimation du vecteur normale en chaque sommet.

2. Création des contours :
    - Création du vecteur de plans.
    - Création de la liste des bandes.
    - Calcul des points de contact C_C entre les plans et les triangles.
    - Estimation de la normale aux points d'intersection.
    - Création de la liste des segments d'intersection dans chaque plan.
    - Chaînage des segments et création des contours.
3. Gestion des interférences et des collisions entre l'outil et la pièce pour chaque contour :
    - Génération des points centre-outil (points de passage C_E) pour chaque contour.
    - Positionnement de l'outil en chaque point de passage.
    - Détection des interférences de l'outil en chaque point de passage.
    - Correction des interférences avec le calcul parallèle.
    - Détection des collisions en chaque point de passage de l'outil.
    - Correction des collisions avec le calcul parallèle.
4. Simulation géométrique des mouvements de l'outil.

3. Solutions proposées

3.1. Lecture de la forme de la pièce à usiner à partir d'un fichier Stl

La lecture d'un fichier STL est la première étape dans notre application logicielle. Elle consiste à récupérer les informations et les paramètres définissant la forme de la pièce à usiner. Les informations récupérées sont un vecteur des sommets sans redondance et un vecteur des triangles. L'algorithme de lecture d'un fichier STL est déjà implémenté, alors notre objectif est la récupération de la liste des sommets et de la liste des triangles et les mettre dans nos propres structures de données. A cet effet, nous avons choisi des tableaux de pointeurs vers les sommets et vers les triangles pour économiser l'espace mémoire et pour profiter de l'accès direct et rapide aux données. Après la récupération de ces données, plusieurs autres informations sont déterminées.

3.2. Création des contours

La création des contours est une fonctionnalité complexe. Cette étape est composée de plusieurs sous-tâches séquentielles .Un contour est composé par des segments d'intersection des triangles avec un seul plan où chaque deux points voisins doivent avoir un point d'intersection commun (Figure 22). Chaque extrémité d'un segment représente un point de contact C_C entre la surface à usiner et l'outil.

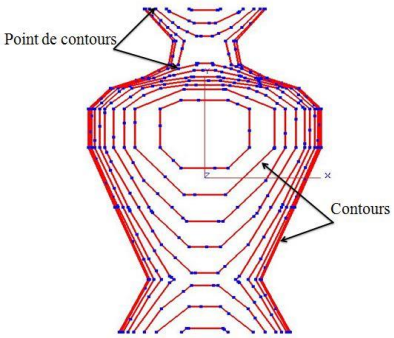


Figure 22: Contours d'usinage.

3.3. Gestion des interférences et des collisions avec le calcul parallèle

3.3.1. Enrichissement du modèle STL par l'insertion de points de manière aléatoire

Lorsque le modèle STL est grossier, il induit des erreurs lors de la détection des interférences et des collisions. Ceci est dû au fait que la densité des points n'est pas importante et l'outil peut pénétrer dans la surface sans être détecté. Afin de remédier à ce problème, la densité des points est augmentée. L'approche utilisée consiste à insérer (ajouter) de nouveaux points dans les triangles en se basant sur les coordonnées barycentriques. Le nombre de points à insérer pour chaque triangle est déterminé en fonction de l'aire du triangle et de la densité (nombre de points par millimètre carré) spécifiée par l'utilisateur.

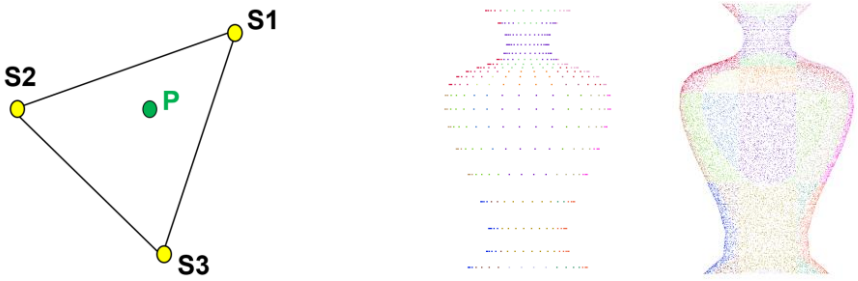
Pour chaque triangle de sommets S_1, S_2, et S_3, les coordonnées d'un point P quelconque du plan du triangle sont données par (Figure 3.a) :

$$\overrightarrow{OP} = \alpha * \overrightarrow{OS_1} + \beta * \overrightarrow{OS_2} + \gamma * \overrightarrow{OS_3}$$

Avec les deux conditions suivantes :

$$0 \leq \alpha, \beta, \gamma \leq 1$$
$$\alpha + \beta + \gamma = 1$$

La Figure 23 montre un modèle STL avant et après enrichissement.



a. Insertion d'un point. b. Modèle avant et après enrichissement.

Figure 23 : Enrichissement du modèle STL.

3.3.2. Création des cellules

Dans le but d'optimiser et d'accélérer la détection des interférences et des collisions, les points insérés sont affectés à des blocs parallélépipédiques de mêmes dimensions. Chaque bloc est appelé « Cellule ». La création des cellules (matrice tridimensionnelle) consiste à subdiviser le brut en des blocs parallélépipédiques de mêmes dimensions (Figure 24). Cette subdivision nécessite la spécification du nombre de cellules N_x, N_y et N_z suivant les trois axes X, Y et Z. A partir de ces données, les pas et les limites de chaque cellule sont calculés.

- Calcul des pas suivant les axes X, Y et Z :

$$Dx = \text{longueur du brut}/ N_x$$

$$Dy = \text{largeur du brut}/ N_y$$

$$Dz = \text{hauteur du brut}/ N_z$$

- Calcul des limites de chaque cellule : les calculs sont résumés dans le Tableau 1.

Tableau 3 :Limites d'une cellule.

	Extrémités minimales	**Extrémités maximales**
Pour i de 0 à N_x	$X_{min} = Brut\_X_{min} + i* Dx$	$X_{max} = X_{min} + Dx$
Pour j de 0 à N_y	$Y_{min} = Brut\_Y_{min} + j* Dy$	$Y_{max} = Y_{min} + Dy$
Pour k de 0 à N_z	$Z_{min} = Brut\_Z_{min} + k* Dz$	$Z_{max} = Z_{min} + Dz$

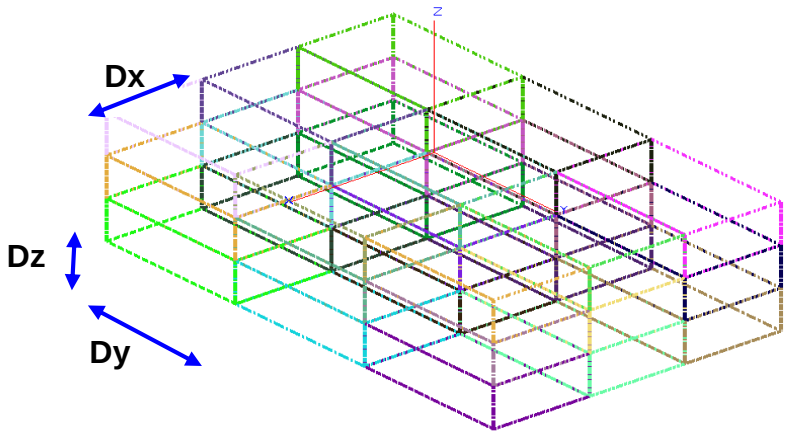


Figure 24 : Création des cellules

3.3.3. Affectation des points insérés aux cellules

Après la création de la matrice tridimensionnelle de cellules, chaque point inséré doit être affecté à la cellule correspondante. Cette affectation est réalisée par comparaison entre les coordonnées de chaque point inséré et les limites maximale et minimale de chaque cellule.

Pour un point S de coordonnées X_S, Y_S et Z_S et une cellule avec ses limites X_{min}, X_{max}, Y_{min}, Y_{max}, Z_{min} et Z_{max}, le point S appartint à la cellule si les conditions suivantes sont vérifiées :

$$\begin{cases} X_{min} < X_s < X_{max} \\ Y_{min} < Y_s < Y_{max} \\ Z_{min} < Z_s < Z_{max} \end{cases}$$

Après la nouvelle représentation des points et pour minimiser le nombre de points à tester dans la détection des interférences et des collisions, pour l'interférence, seules les cellules se chevauchant avec l'enveloppe de la partie active sont considérées et pour la collision seules les cellules se chevauchant avec l'enveloppe du corps de l'outil sont considérées. La Figure 25 montre l'affectation des points insérés aux cellules créées.

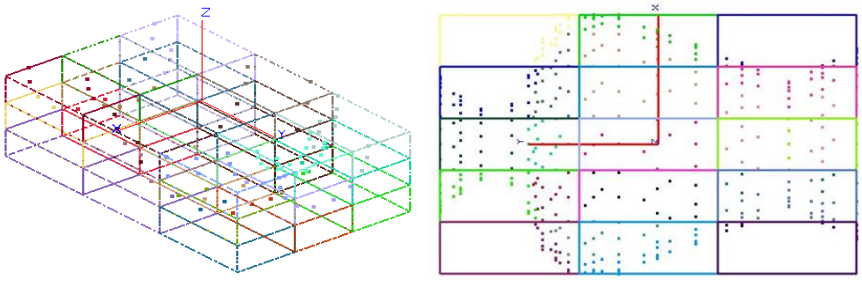


Figure 25 : Affectation des points aux cellules.

3.3.4. Détection et évitement des interférences pour un point de contact donné :

Pour traiter les interférences et les collisions, nous devons connaître la base de données des

outils hémisphériques disponibles. Chaque outil est défini par son rayon et sa longueur.

3.3.4.1. Création de la liste des points de passage de l'outil :

Pour positionner l'outil, nous avons besoin de connaître les coordonnées du centre de la sphère (partie active de l'outil) pour déduire les coordonnées de ses extrémités et de son enveloppe. Les coordonnés du centre sont calculées à partir du point d'intersection (point de contact outil-surface et de la normale en ce point).

❖ **Calcul des coordonnées du point centre outil :** les coordonnées du centre outil sont calculées comme suit. Soit un outil de rayon R (rayon de sa sphère) et soit P le point d'intersection (point de contact C_C) de coordonnées Xp ,Yp et Zp. Au point P, les composantes du vecteur normal unitaire au point P sont Ux, Uy et Uz (Figure 26). Les coordonnées Xc, Yc et Zc du point C centre d'outil sont données par :

$$Xc = Xp + R \bullet Ux$$

$$Yc = Yp + R \bullet Uy$$

$$Zc = Zp + R \bullet Uz$$

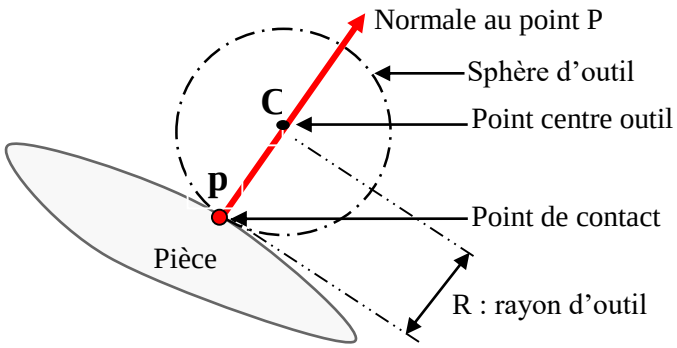


Figure 26 : Positionnement d'un outil hémisphérique.

❖ **Positionnement** de l'outil : après le calcul des centres outil, nous pouvons maintenant positionner l'outil pour usiner le segment en question. Initialement, nous positionnons l'outil verticalement au plan, c'est-à-dire son axe est parallèle à l'axe Z.

❖ **Détection des interférences :** l'outil est en interférence avec la pièce si l'intersection du volume de sa partie active (sphère) avec le volume de la pièce est un ensemble non vide. Pour vérifier l'intersection des volumes, il suffit de vérifier l'appartenance des points insérés à la sphère de l'outil. La vérification de l'intersection de la sphère avec tous les points insérés est très couteuse en temps de calcul. Pour le minimiser, il faut réduire le nombre de points à vérifier par l'identification et l'utilisation des cellules en chevauchement avec la partie active de l'outil.

❖ **Recherche des cellules au voisinage de la sphère :** pour résoudre ce problème, nous allons positionner virtuellement l'enveloppe de la sphère dans la matrice des cellules des points insérés et par la suite nous allons déterminer les cellules qui se chevauchent avec l'enveloppe de la sphère . Donc, nous allons identifier les indices (i, j, k) de chaque cellule déterminée.

❖ **Calcul de l'enveloppe de la sphère (cube) :** les coordonnées des points extremums (coin gauche bas et coin droit haut) sont calculées comme suit :

Extrémité minimale (coin gauche bas) :

$$Xmin = Xc - R$$

$$Ymin = Yc - R$$

$$Zmin = Zc - R$$

Extrémité maximale (coin droit haut) :

$$Xmax = Xc + R$$

$$Ymax = Yc + R$$

$$Zmax = Zc + R$$

Avec R est le rayon de la sphère, (Xc, Yc, Zc) sont les coordonnées du point centre outil.

Pour déterminer les cellules dans une matrice tridimensionnelle, nous devons connaître suivant chaque direction à quelle position (indice) nous commençons la sélection et à quelle position (indice) nous devons terminer la sélection. Par conséquence, six indices doivent être déterminés (Figure 27).

- Imin et Imax pour la direction suivant l'axe X.
- Jmin et Jmax pour la direction suivant l'axe Y.
- Kmin et Kmax pour la direction suivant l'axe Z.

Après le calcul des deux extrémités de l'enveloppe de la sphère, nous pouvons facilement extraire les valeurs des indices i, j et k comme suit :Imin=Xmin/Dx

$$Imax = Xmax/Dx$$

$$Jmin = Ymin/Dy$$

$$Jmax = Ymax/Dy$$

$$Kmin = Zmin/Dz$$

$$Kmax = Zmax/Dz$$

A la fin de cette étape, nous avons déterminé les cellules concernées par la vérification des

interférences. Il nous reste la récupération de la liste des points insérés de ces cellules et de vérifier l'appartenance de ces points à la sphère.

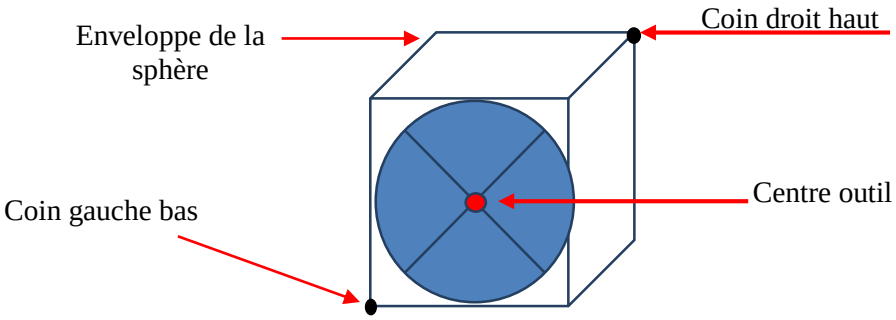


figure 27 : Enveloppe de la sphère de l'outil.

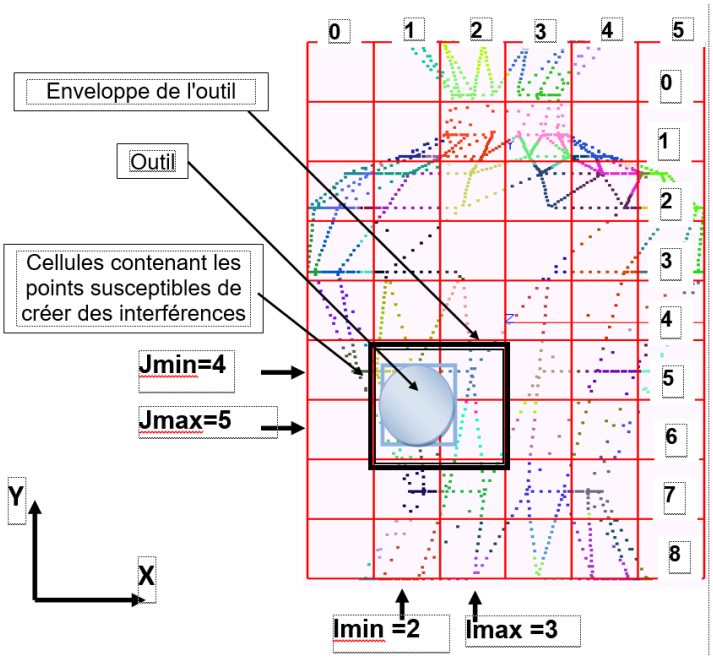


Figure 28 : Cellules candidates à l'interférence.

❖ **Vérification si un point inséré est en interférence avec la sphère :** pour résoudre ce problème, nous utilisons l'équation de la sphère. Soit C (Xc, Yc, Zc) un point qui représente le centre de la sphère et R son rayon. Un point P(Xp, Yp, Zp) appartient à la surface de la sphère si l'équation suivante est vérifiée (Figure 29) :

$$(Xp - Xc)^2 + (Yp - Yc)^2 + (Yp - Yc)^2 = R^2$$

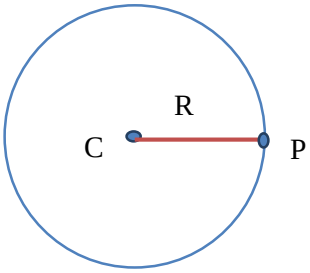


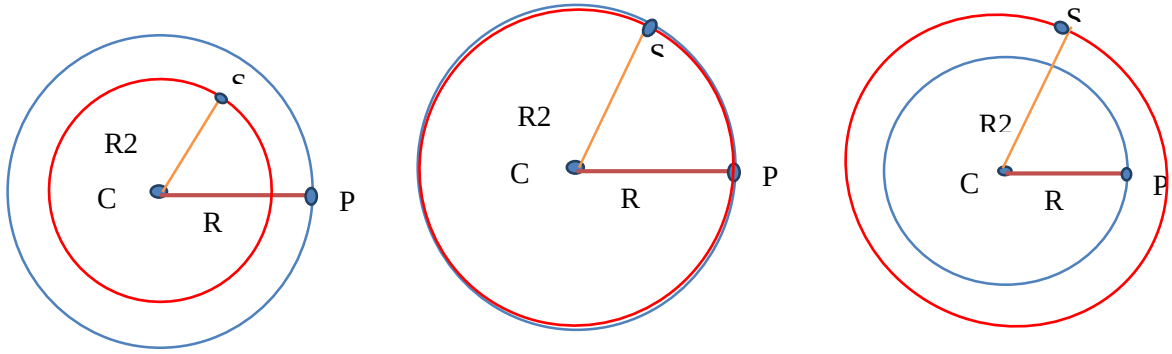
Figure 29 : Point sur la surface de la sphère.

Soit un point S(Xs, Ys, Zs) appartenant à une deuxième sphère de rayon R2. Alors, les coordonnées du point S vérifient l'équation suivante :

$$(Xs - Xc)^2 + (Ys - Yc)^2 + (Ys - Yc)^2 = (R2)^2$$

Donc :

- Si R2 < R, alors la deuxième sphère se trouve à l'intérieur de la première sphère, ce qui implique l'existence d'interférence (Figure 30.a).
- Si R2 = R, alors les deux sphères sont identiques, ce qui implique absence d'interférence (Figure 30.b).
- Si R2 > R, alors la première sphère se trouve à l'intérieur de la deuxième sphère. Dans ce cas, il n'y a pas d'interférence (Figure 30.c).



a. A l'intérieur de la sphère. b. Sur la sphère. c. A l'extérieur de la sphère.

Figure 30 : Position d'un point par rapport à la surface de la sphère d'outil.

3.3.5. Outil hémisphérique optimum pour un point de contact donné

Lors de l'usinage, il est indispensable de choisir le plus grand outil pour avoir des vitesses d'usinage plus importantes et par conséquent des temps d'usinage très faibles. Mais ce choix peut engendrer des problèmes d'interférences. Donc, l'outil hémisphérique optimum en un point de contact est l'outil ayant le plus grand rayon et qui ne crée pas des problèmes d'interférences. Il est déterminé par les étapes suivantes :

1. Trier la base de données des outils suivant les rayons dans un ordre décroissant.
2. Sélectionner le plus grand outil.
3. Calculer le point centre outil et l'enveloppe de la partie active de l'outil (sphère).
4. Identifier les cellules qui se chevauchent avec l'enveloppe de la partie active.

5. Vérifier l'existence ou l'absence des interférences avec les points des cellules identifiées. Deux cas sont à considérer :
    - Si absence d'interférences, alors c'est l'outil hémisphérique optimum.
    - Si existence d'interférences, alors sélectionner l'outil suivant dans la base de données et aller à l'étape 3.

Ce processus est répété pour tous les points de contact de tous les contours d'usinage. Il est clair que le calcul de l'outil optimum en un point de contact est indépendant de tous les autres points de contact. Comme l'usinage en finition des pièces complexes sur des fraiseuses numériques à 05-axes exige par la spécification d'une profondeur de passe très petite de l'ordre du centième de millimètre (0.01 mm). Cette profondeur avec le très grand nombre de triangles représentant la pièce, le calcul des contours d'usinage génère un important nombre de contours et par conséquent un nombre de points très important qui peut atteindre des millions. Pour réduire le temps de calcul des outils optimums, nous avons pensé au calcul parallèle par l'utilisation de la carte graphique.

3.3.6. Affectation de l'outil optimum à un contour

L'outil optimum d'un contour est celui qui élimine les problèmes d'interférences en chaque point de contact de ce dernier. Plusieurs stratégies peuvent être proposées pour l'affectation des outils optimums d'un contour donné. Dans le contexte de ce projet, un seul outil est assigné à chaque contour de la manière suivante :

1. Déterminer l'outil ayant le plus petit rayon parmi tous les outils optimums de tous les points de contact du contour.
2. Affecter cet outil à tous les points du contour.
3. Positionner l'outil déterminé en chaque point de contact et calculer les coordonnées des points centre outil et extrémité outil.

3.3.7. Détection et évitement des collisions pour un point de contact donné

Après le positionnement de l'outil et la correction des interférences, nous devons vérifier l'existence des problèmes de collisions entre l'outil et la pièce. Le principe est le même qu'avec la détection des interférences, c'est-à-dire le chevauchement entre l'enveloppe du cylindre de l'outil (partie non active) et le volume de la pièce. Dans ce cas précis, nous devons chercher une méthode pour calculer l'intersection.

❖ **Détection de la collision :** maintenant l'outil est déjà positionné. Comme nous avons vu précédemment dans la détection des interférences, nous devons déterminer les cellules de la matrice tridimensionnelle susceptibles de provoquer des collisions de la même manière, c'est-à-dire nous calculons l'enveloppe du cylindre, plus exactement les deux extrémités maximale et minimale et par la suite nous déterminons les indices i, j et k des cellules.

Il reste donc la vérification de l'appartenance des points des cellules déterminées à l'enveloppe du cylindre. Dans ce cas, le test est différent de celui de l'interférence.

❖ **Vérification si un point appartient au cylindre :** la vérification de l'appartenance d'un point à un cylindre d'orientation quelconque est composée de deux étapes :

- ***Première étape :*** calculer la distance D entre le point S et la droite représentant l'axe du cylindre. Par la suite, la comparer avec le rayon R du cylindre. Deux cas sont considérés (Figure 31) :
  - Si D < R, alors il y a possibilité de collision.
  - Si D > R, alors le point S est en dehors du cylindre et donc pas de collision.

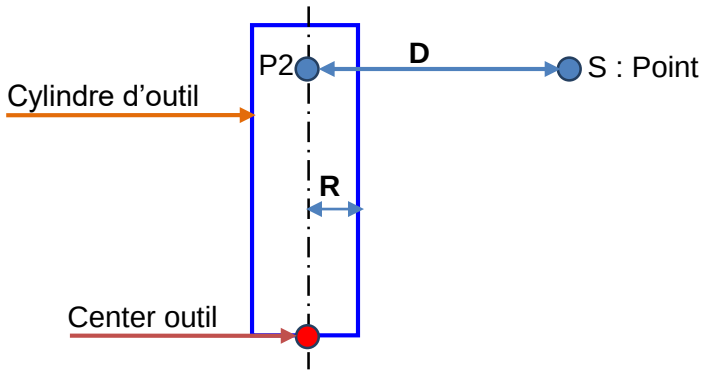


Figure 31 : Position d'un point par rapport à l'axe du cylindre.

- ***Deuxième étape :*** si le point S est candidat à une collision, pour terminer le test de vérification des collisions, nous projetons ce sommet sur la droite portant l'axe du cylindre et par la suite nous calculons la distance D1 entre l'extrémité supérieure et le point résultant de la projection de S et la distance D2 entre l'extrémité inferieure et le point (Figure 32). Si « L » est la longueur du cylindre, deux cas sont à considérer :
  - Si D1 + D2 = L, alors le point S appartient au volume du cylindre et donc il crée une collision avec l'outil.

- Si D1 + D2 > L, alors le point S est en dehors de l'enveloppe du cylindre et donc il ne crée pas de collision.

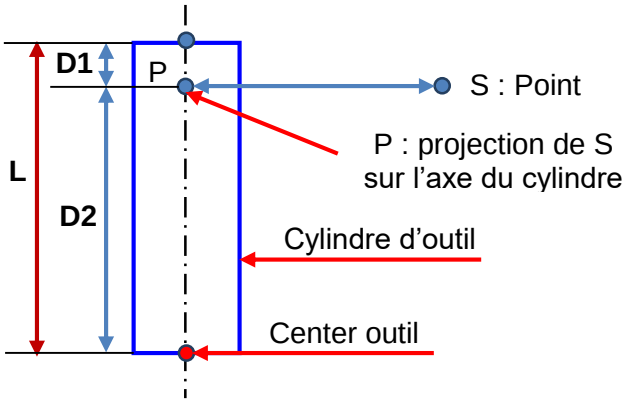


Figure 32 : Point en collision avec le cylindre de l'outil.

La détection des collisions est effectuée en chaque position d'outil (point centre outil). Après cette étape, les angles d'orientation de l'outil pour toutes les positions sont égaux à zéro (pas d'orientations). Les positions d'outil qui provoquent une collision doivent être changées en lançant une procédure de correction des collisions.

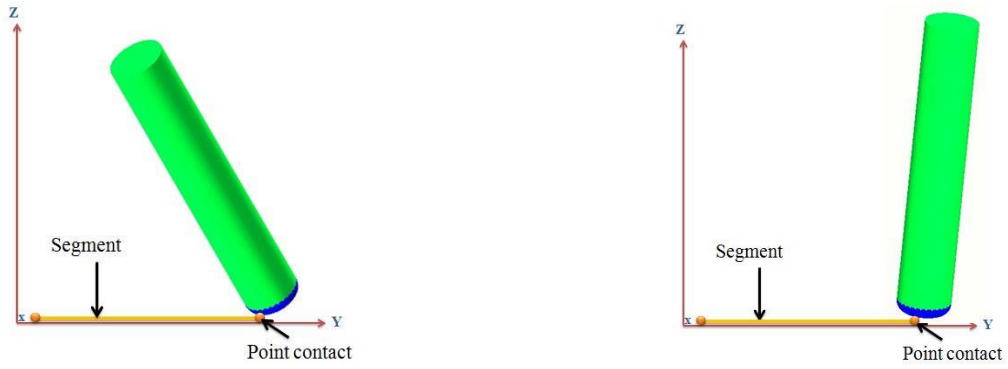
❖ **Correction des collisions :** avant de lancer l'opération d'usinage réel, il est indispensable de corriger les collisions en chaque point centre d'outil. Si la correction est impossible, alors nous devons éliminer le segment correspondant au point centre outil de l'opération d'usinage. La collision est due à une mauvaise orientation de l'axe de d'outil. Pour éviter ce problème, nous pouvons incliner l'outil avec un angle déterminé afin d'avoir une nouvelle orientation qui ne provoque pas une collision. Cet angle doit être sauvegardé pour chaque opération de correction.

L'évitement de la collision consiste à proposer une stratégie d'orientation de l'axe de l'outil permettant de dégager l'outil de sa position de chevauchement avec la surface. Dans ce travail, deux modes d'orientation de l'axe de l'outil sont proposées (Figure 33) :

- Inclinaison de l'outil vers le segment dans le sens de la vitesse d'avance.
- Rotation de l'outil autour du segment formé par le point de contact actuel et le point de contact suivant.

Pour les deux modes, pour chaque point de contact, l'outil initialement est positionné verticalement et en cas de collision, l'outil est orienté « autour du »/« vers le » segment jusqu'à la détermination d'un angle qui oriente l'outil hors collision. Ces deux modes sont lancés séquentiellement sans aucune priorité au préalable. Néanmoins, pour un expert en fonction de

la géométrie de la surface à usiner il peut décider lequel des deux modes doit être lancé en premier.



a. Inclinaison vers le segment. b. Rotation autour du segment.

Figure 33 : Modes d'orientation de l'outil.

❖ **Orientation de l'outil :** l'orientation de l'outil a pour objectif de calculer les angles d'inclinaison permettant de positionner l'outil sans collision. L'orientation de l'outil peut être réduite à l'orientation de l'extrémité supérieure de l'outil. Donc, l'orientation ne concerne pas le point centre outil qui constitue l'extrémité inferieure de l'outil (Figure 34). Après la rotation de ce point avec un angle alpha et calcul de ses coordonnées, nous pouvons positionner l'outil dans cette nouvelle position et par la suite application du test de vérification de la collision. S'il n'y a pas de collision, nous sauvegardons l'angle alpha et nous passons à un nouveau point centre outil. Si la collision existe toujours, nous incrémentons l'angle alpha avec un pas et nous procédons à l'orientation et au test de collision ainsi de suite jusqu'à l'obtention d'un angle adéquat tout en respectant les angles d'orientation maximales de l'outil définis par le constructeur. Si la collision n'est pas éliminée, nous devons éviter l'usinage du segment correspondant. Dans notre approche, la correction est faite en trois étapes séquentielles en fonction de la situation rencontrée :

- ***Première étape :*** l'orientation de l'outil est faite uniquement dans le sens d'usinage (vers le segment) avec un angle alpha incrémenté par un pas « pas alpha ». Si nous parcourons tout l'intervalle de variation de « alpha » sans résoudre le problème, alors nous passons à la deuxième étape.
- ***Deuxième étape :*** nous repositionnons l'outil dans la position initiale (position verticale) au point centre outil et nous procédons à orienter l'outil uniquement autour du segment (vers l'extérieur de la pièce) avec un angle « beta » incrémenté chaque fois

avec un pas « pas beta ». Si nous parcourons tout l'intervalle de variation de beta sans résoudre le problème, alors nous passons à la troisième étape.

- ***Troisième étape :*** cette fois, les deux orientations (vers le segment et autour du segment) sont combinées. Donc, une orientation est faite vers le segment avec un pas « pas alpha » et pour chaque nouvelle orientation d'angle « alpha » calculée, nous orientons l'axe de l'outil autour du segment en considérant tout l'intervalle de variation de beta. Cette procédure est répétée jusqu'à l'obtention des angles « alpha » et « beta » évitant les collisions. Si les collisions persistent toujours, le segment considéré n'est pas usiné et il est retiré du trajet d'outil.

D'après ce qui précède, nous constatons que l'orientation de l'extrémité supérieure (point dans l'espace) du cylindre d'outil est effectuée autour d'un axe quelconque passant par le point centre outil et ne passant pas par l'origine.

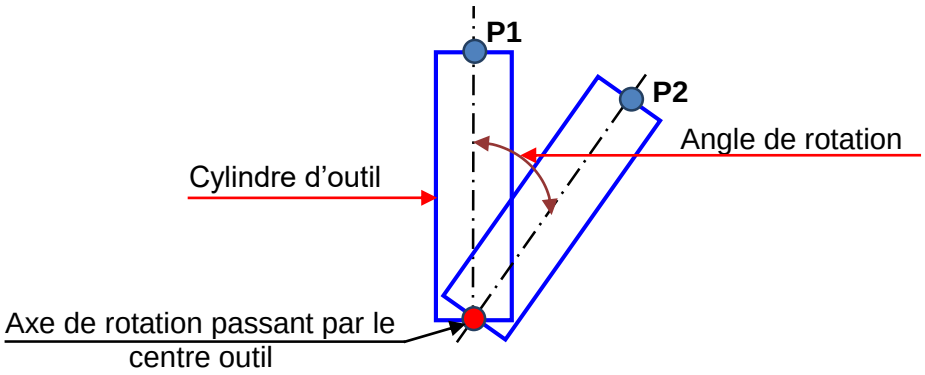


Figure 34 : Rotation de l'outil autour d'un axe.

Comment orienter un point autour d'un axe quelconque : nous avons vu que l'axe d'orientation ne passe pas forcément par l'origine. Ce problème peut être résolu par le changement de base (changement de repère). Le nouveau repère aura comme origine le point centre outil et ses axes sont parallèles aux axes du repère global. Donc, nous effectuons l'orientation de l'extrémité supérieure de l'outil dans le nouveau système d'axes après calcul de ses nouvelles coordonnées (Figure 35).

L'orientation dans le sens d'usinage se fait autour d'un axe $\vec{U}$ passant par le point centre outil. Cet axe est perpendiculaire à l'axe Z et au segment d'usinage considéré dont les deux points centres outil sont C1 et C2 générés à partir du point du début du segment et du point de la fin du segment. Cet axe est le résultat du produit vectoriel des deux vecteurs unitaires de l'axe Z (0, 0, 1) et du vecteur $\overrightarrow{C1C2}$. Par contre, lors de l'orientation autour du segment

d'usinage, l'axe $\vec{U}$ est le vecteur unitaire du segment d'usinage reliant les points centre outil C1 et C2.

Si nous orientons le point P1(X1, Y1, Z1) dans le nouveau repère avec un angle α, alors les coordonnées de son image P2(X2, Y2, Z2) sont calculées par le produit matriciel de la matrice de transformation avec le vecteur $\begin{pmatrix} X1 \\ Y1 \\ Z1 \end{pmatrix}$comme suit :

Soit U(u_x,u, u_z) le vecteur unitaire de l'axe de rotation, la matrice de transformation des coordonnées est donnée par :

$$\begin{bmatrix} \mathrm{ux}^2 + (1 - ux^2) * c & \mathrm{ux} * \mathrm{uy} * (1 - \mathrm{c}) - \mathrm{uz} * \mathrm{s} & \mathrm{ux} * \mathrm{uz}(1 - \mathrm{c}) + \mathrm{uy} * \mathrm{s} \\ \mathrm{ux} * \mathrm{uy} * (1 - \mathrm{c}) - \mathrm{uy} * \mathrm{s} & \mathrm{uy}^2 + (1 - \mathrm{uy}^2) * \mathrm{c} & \mathrm{uy} * \mathrm{uz}(1 - \mathrm{c}) - \mathrm{ux} * \mathrm{s} \\ \mathrm{ux} * \mathrm{uz} * (1 - \mathrm{c}) - \mathrm{uy} * \mathrm{s} & \mathrm{uy} * \mathrm{uz} * (1 - \mathrm{c}) + \mathrm{ux} * \mathrm{s} & uz^2 + (1 - uz^2) * c \end{bmatrix}$$

Où :

c=cos(α) et s=sin (α)

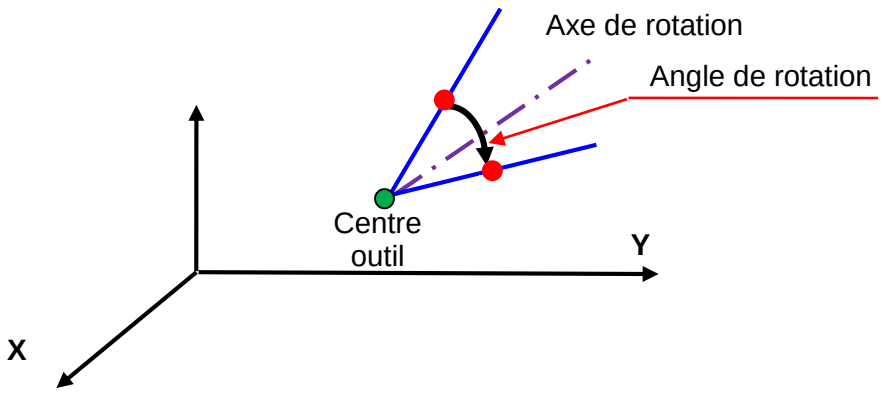


Figure 35 : Rotation d'un point P1 autour d'un axe quelconque.

3.3.8. Génération du trajet d'outils

Une fois que les problèmes d'interférences et de collisions sont détectés et évites, il est possible maintenant de passer la génération du trajet d'outils. Comme dans ce travail, à chaque contour est associé un outil hémisphérique optimum, l'usinage de la pièce complexe requiert l'utilisation de plusieurs d'outils et donc d'une fraiseuse numérique avec un magasin d'outils pour effectuer automatiquement les changements d'outils. Pour cela, deux modes sont proposés lors de la génération du trajet d'outils final :

- ***Premier mode :*** usinage des contours séquentiellement sans tenir compte de l'outil utilisé et du temps de changement d'outils. C'est le mode sans optimisation.
- ***Deuxième mode :*** usinage des contours ayant le même outil séquentiellement. C'est le mode avec optimisation. Chaque outil est utilisé une seule fois ce qui permet de réduire les temps d'usinage puisque les temps de changement d'outils sont réduits au maximum.

3.4. Optimisation du temps par le calcul parallèle

Au début, nous avons pensé à la création et à l'utilisation des bibliothèques dynamiques « dll » pour pouvoir appeler directement les fonctions de la plateforme logicielle développée par l'équipe « CFAO ». Plusieurs solutions et tentatives ont été faites dans ce sens, mais nous n'avons pas pu les intégrer entre deux environnements différents. Ce choix a été abandonné puisque nous avons trouvé beaucoup de difficultés. Pour remédier à ces difficultés, nous sommes passées à l'utilisation des « Sockets » pour permettre le transfert des données entre les deux environnements de développements à savoir Builder C++ et Microsoft Visual C++. L'architecture générale de l'approche proposée est représentée par la Figure 16.

Ordinateurs distincts du réseau, mais les Sockets peuvent également être utilisées pour communiquer localement (interprocessus) sur un même ordinateur. Les Sockets sont bidirectionnelles, ce qui signifie que l'un ou l'autre côté de la connexion est capable d'envoyer et de recevoir des données. Parfois, une application qui initie la communication est appelée « Client » et l'autre application est appelée « Serveur ».

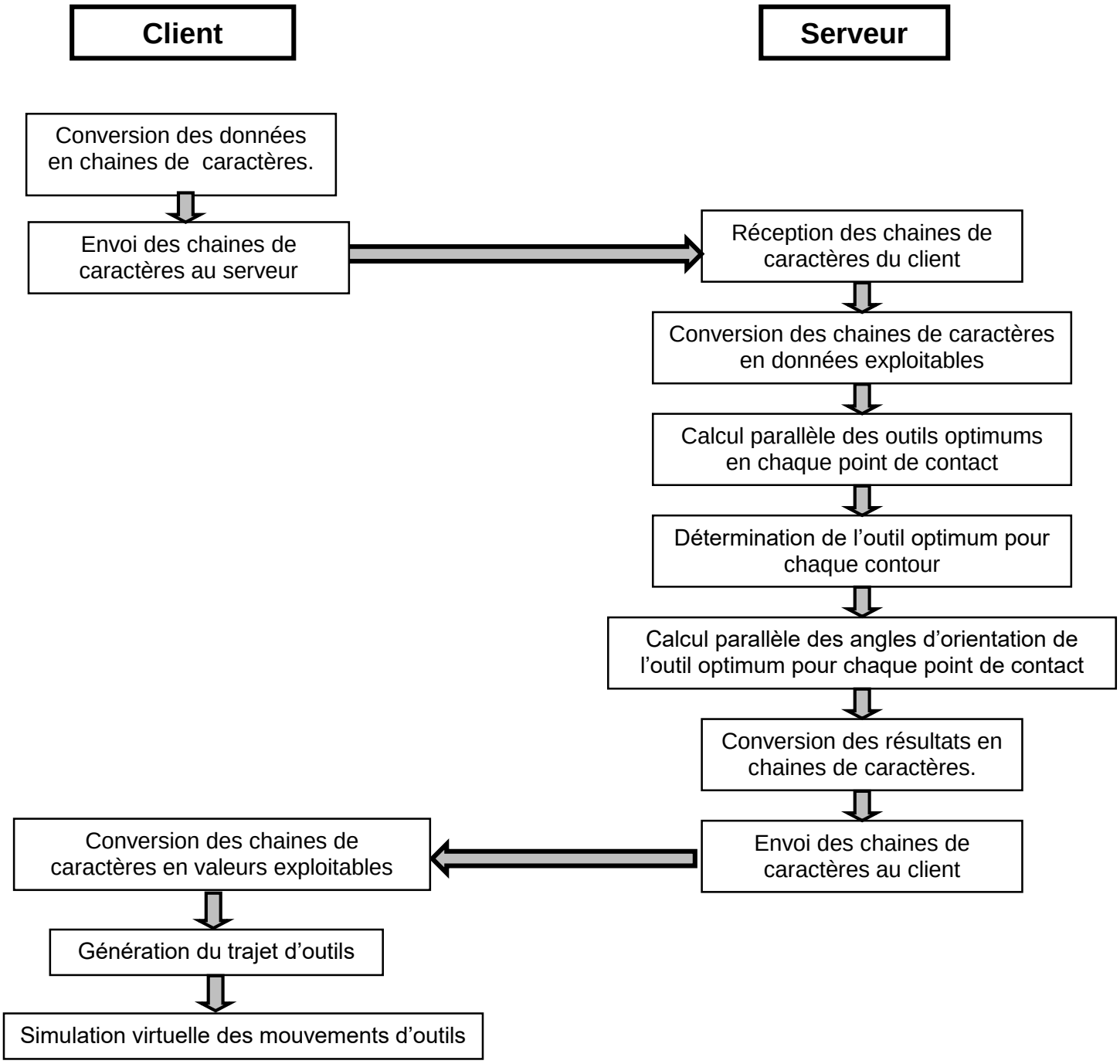


Figure 36 : Architecture générale de l'approche proposée.

3.4.1. Création des sockets

Une socket représente une seule connexion entre exactement deux logiciels. Plus de deux logiciels peuvent communiquer avec des systèmes « Client/serveur » ou distribués à l'aide de plusieurs Sockets. Les logiciels basés sur des Sockets s'exécutent généralement sur deux ordinateurs distincts du réseau, mais les Sockets peuvent également être utilisés pour

communiquer localement (interprocessus) sur un même ordinateur. Les Sockets sont bidirectionnelles, ce qui signifie que l'un ou l'autre côté de la connexion est capable d'envoyer et de recevoir des données. Parfois, une application qui initie la communication est appelée « Client » et l'autre application est appelée « Serveur ».

3.4.2. Etablissement de la liaison

Les sockets permettent d'établir une connexion TCP/IP entre deux (02) programmes (Figure 37). Il n'y a pas de restrictions sur la localisation des programmes et les deux programmes peuvent très bien installés sur la même machine (local host=127.0.0.1).

Une connexion TCP/IP est identifiée par les quatre (04) données suivantes :

- Adresse IP du programme 1.
- Port du programme 1 (en général, on laisse le système choisir un port).
- Adresse IP du programme 2.
- Port du programme 2.

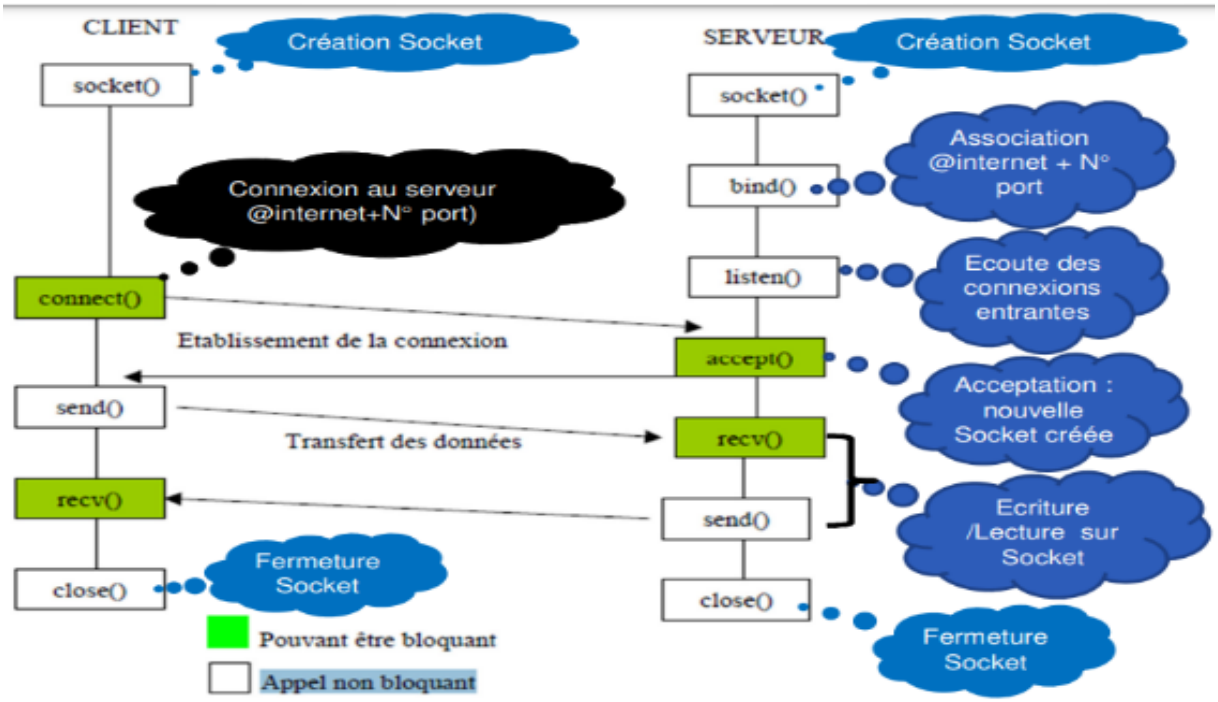


Figure 37 : Schéma d'une communication avec une Socket.

Dans notre projet, nous avons besoin de transférer les données du Builder C++ au Microsoft Visual Studio C++ où le « Client » est le Builder et le « Serveur » est le Visual (Figure 38).

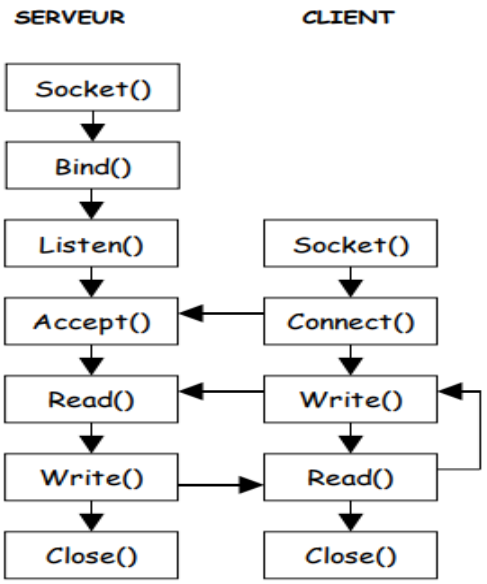


Figure 38 : Schéma d'une communication.

3.4.3. Envoi des données au « Serveur »

Comme le travail entrepris est une continuité d'un autre travail où les calculs relatifs aux évitements des interférences et des collisions étaient lancés séquentiellement, les données indispensables aux différents calculs sont préétablies et doivent être envoyées au « Serveur » pour effectuer tous les calculs parallèles. Les données à envoyer sont : les paramètres des outils, les paramètres des cellules, les paramètres des contours d'usinage et les paramètres des angles d'orientation de l'outil. Pour l'envoi de ces paramètres, les « Sockets » sont utilisées.

❖ **Paramètres d'outils :** l'outil est défini selon des critères géométriques et technologiques. Le premier indicateur caractérise la forme et les dimensions de l'outil en relation avec la forme finale à obtenir. Les caractéristiques géométriques définissant l'outil peuvent être en fraisage par exemple le diamètre, la longueur utile coupante. Pour l'outil hémisphérique, il faut envoyer la longueur et le rayon d'outil ainsi que le nombre total des outils de la base de données des outils (Figure 39).

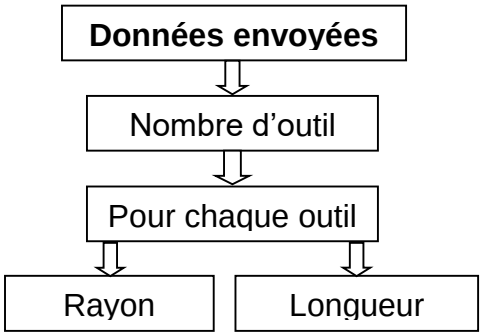


Figure 39 : Paramètres d'outils à envoyer.

❖ **Paramètres des cellules :** les paramètres des cellules à envoyer sont le nombre de cellules suivant les trois axes X, Y et Z. Par la suite, pour chaque cellule, il faut envoyer ses extrémités maximales et minimales ainsi que les coordonnées de ses points (Figure 40).

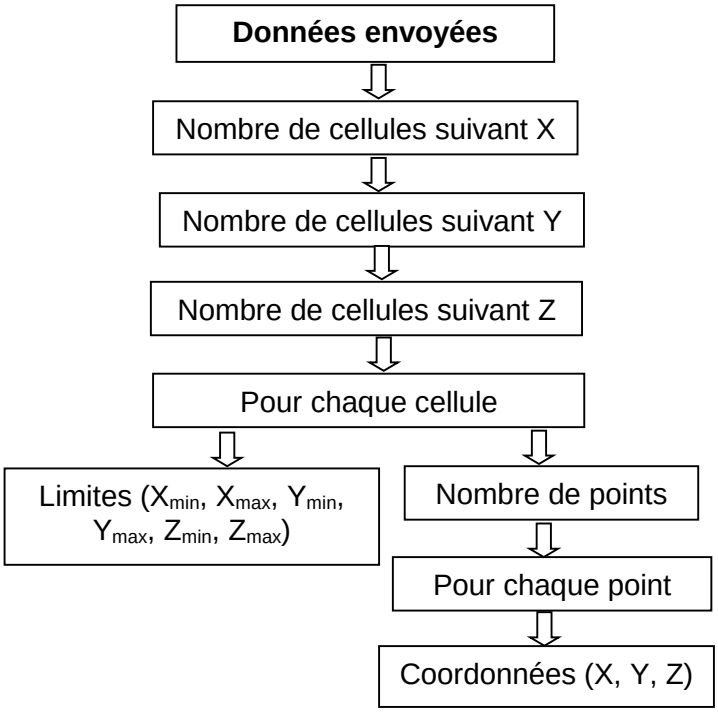


Figure 40 : Paramètres des cellules à envoyer.

❖ **Paramètres des contours :** le premier paramètre à envoyer est le nombre de contours. Par la suite, pour chaque contour il faut spécifier le nombre de points et pour chaque point transférer ses paramètres (coordonnées X, Y et Z ainsi que les composantes du vecteur normal unitaire Nx, Ny, Nz (Figure 41).

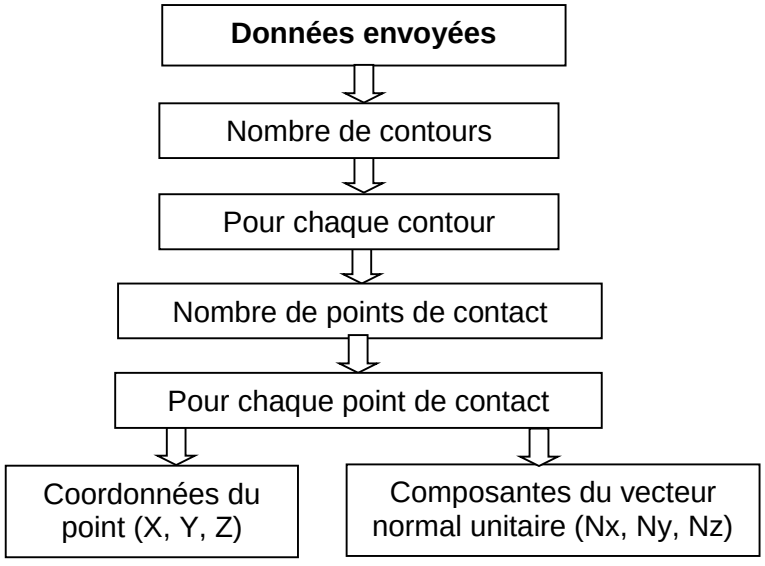


Figure 41 : Paramètres des contours à envoyer.

❖ **Paramètres des angles d'orientation alpha et beta :** chaque fraiseuse numérique possède des paramètres spécifiques déterminés par le constructeur. Ces paramètres concernent les limites inférieures et supérieures des deux angles d'orientation de l'outil autour de ses axes rotatifs. Ces angles sont à spécifier et à envoyer au « Serveur » (Figure 42). De plus, il faut envoyer les incréments de chaque angle lors de la phase d'évitement des collisions.

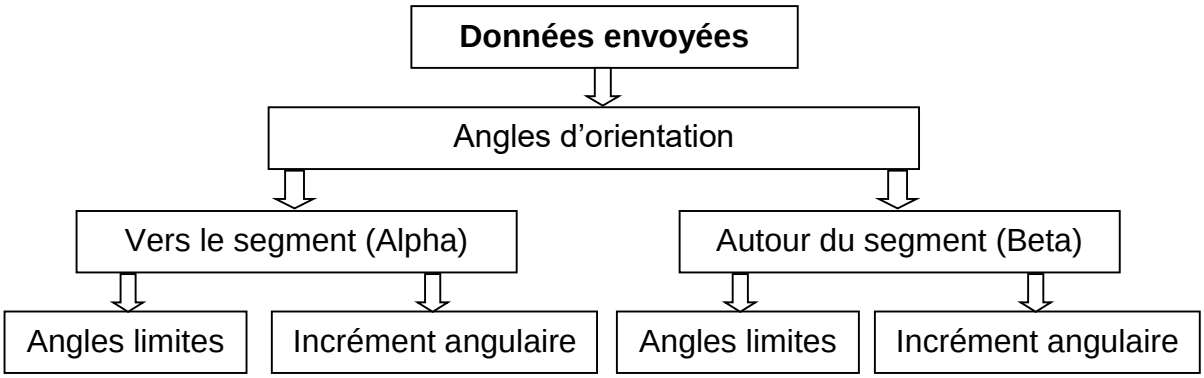


Figure 42 : Paramètres des angles d'orientation des outils à envoyer.

❖ **Réception des données :** après l'envoi des données à partir du Builder C++, le Visual Studio C++ reçoit ces données en utilisant les « Sockets » TCP. Nous devons que les données envoyées sont de natures hétérogènes avec des valeurs entières, réelles et booléennes. Donc, pour uniformiser les traitements, les données sont envoyées et reçues sous formes de chaines de caractères. Après leur réception, ces données sont converties selon le type des données.

3.4.4. Calcul parallèle avec « CUDA »

Dans le précédent projet, l'évitement des interférences et des collisions est mené par un calcul séquentiel en considérant un seul point de contact à la fois. Sachant bien la finition des pièces complexes exige la génération d'un trajet d'outils composé de centaines de milliers de points voire des millions de points en fonction des précisions demandées, de l'étendu de la pièce et de sa complexité géométrique. Le traitement séquentiel de cet important nombre de points augmente considérablement les temps de calcul et par conséquent les coûts et le cycle de développement de nouveaux produits. Pour réduire considérablement, dans ce projet, nous avons utilisé le calcul parallèle en programmant la carte graphique « GPU » avec « CUDA ».

« CUDA » est utilisé afin de développer des logiciels pour les processeurs graphiques et pour développer une variété d'applications à usage général pour les « GPU » qui sont de nature hautement parallèle et fonctionnent sur des centaines de cœurs de processeurs du « GPU ». Dans ce projet, nous avons implémenté les approches d'évitement des interférences et des collisions en utilisant le calcul parallèle avec « CUDA ».

❖ **Evitement des interférences :** lors du calcul parallèle, le calcul de l'outil optimum en chaque point de contact est lancé simultanément pour un grand nombre de points de contact. Cette manière permet de réduire considérablement les temps de calcul. Dans le cas de présence d'interférence, l'outil est remplacé par celui qui le succède dans la base de données des outils. Notons que les outils sont classés dans la base de données dans un ordre décroissant de leurs rayons. Si tous les outils de la base de données sont consommés avant l'évitement des interférences, le point est retiré du trajet. Si pour un contour, il existe un point où les interférences ne sont évitées, alors ce contour n'est pas usiné. Dans ce cas de Figure, la base de données des outils doit être enrichie par des outils plus petits pour pouvoir assurer l'usinage de toute la pièce. A la fin des calculs, un seul outil optimum, défini par son rayon et sa longueur, est affecté pour tous les points du même contour.

❖ **Evitement des collisions :** lors du calcul parallèle, le calcul de l'orientation de l'outil optimum en chaque point de contact est lancé simultanément pour un grand nombre de points de contact. Cette manière permet de réduire considérablement les temps de calcul. L'évitement de la collision consiste à proposer une stratégie d'orientation de l'outil permettant de dégager l'outil de sa position de chevauchement avec la surface. Dans notre travail nous avons proposé une approche qui combine une inclinaison de l'outil vers le segment dans le sens de l'avance (alpha) et une rotation autour du même segment (beta). Si pour un contour, il existe un point où les collisions ne sont évitées, alors ce contour n'est pas usiné. Dans ce cas de Figure, la base de données des outils doit être enrichie par des outils plus petits pour pouvoir assurer l'usinage de toute la pièce.

A la fin des calculs, pour chaque point de contact sont calculées les coordonnées des points centre outil « C_E » et point extrémité outil « C_L » ainsi que le vecteur axe d'outil « U ».

3.4.5. Réception des résultats

Après la fin des calculs d'évitement des interférences et des collisions par le calcul parallèle au niveau du « Serveur », celui-ci convertit les résultats en chaines de caractères et les envois au « Client ». Ce dernier, convertit ces chaines de caractères en données exploitables dans la génération du trajet d'outils. Dans la phase de réception des données, pour chaque point de contact « C_C »les informations reçues sont les suivantes (Figure 43) :

- Existence ou absence des interférences.
- Existence ou absence des collisions.
- Rayon « R » et longueur d'outil « L ».

- Coordonnées du point centre outil « C_E ».
- Coordonnées du point extrémité outil « C_L ».
- Composantes du vecteur unitaire axe d'outil « U ».

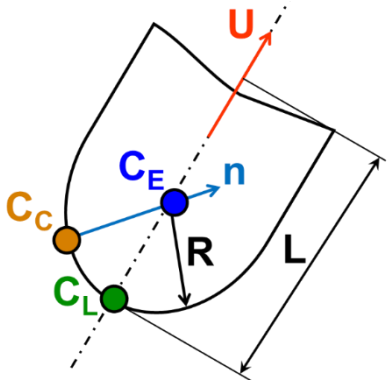


Figure 43 : Paramètres reçus pour chaque point de contact.

3.4.6. Génération du trajet d'outils et simulation virtuelle des mouvements :

Une fois que toutes les données sont reçues, le trajet d'outils est généré suivant deux modes :

- ***Sans optimisation :*** usinage séquentiel de tous les contours.
- ***Avec optimisation :*** réduction du nombre de changements d'outils par groupement et usinage des contours ayant le même outil.

Pour les deux modes, une simulation virtuelle des mouvements des outils doit être lancée pour vérifier visuellement le trajet des différents outils avant de passer à la génération du programme d'usinage et à l'usinage réel.

4. Conclusion

Dans ce chapitre, nous avons présenté l'architecture générale de notre application logicielle et les problèmes rencontrés avec les différentes solutions que nous avons proposées pour pouvoir générer un trajet d'outil sain (sans interférences et sans collisions) en utilisant « CUDA » lors de la finition des surfaces complexes définies par des modèles STL sur des fraiseuses numériques 05-axes en utilisant des outils hémisphériques.

Le test et la validation de notre application sur des surfaces réelles seront considérés dans le chapitre suivant.

CHAPITRE 03 :

IMPLEMENTATION INFORMATIQUE, TESTS ET VALIDATIONS

1. Introduction

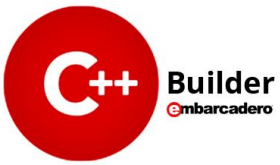
Après avoir présenté l'approche proposée dans le chapitre précédent, l'objectif de ce chapitre est l'illustration des différentes tâches que notre système peut effectuer. Nous commençons par définir les outils de développement informatique ainsi qu'une vue générale détaillée en utilisant des captures d'écrans. Par la suite, une validation est réalisée par un ensemble de tests qui ont été préparés pour chaque fonctionnalité de l'application.

2. Présentation des langages utilisés

Comme notre système est appelé à être intégré dans l'environnement de production des surfaces complexes développé par l'équipe « CFAO » du « CDTA ». Les outils de développement utilisés lors de la mise en œuvre de notre application (C++) sont les mêmes que ceux utilisés par l'équipe « CFAO » afin de se conformer à la tendance qui veut que la majorité des systèmes de « CFAO » soient développés en utilisant le C++.

2.1. Présentation du langage C++

Le langage C++, inventé par Bjarne Stroustrup vers 1983, est une évolution orientée objets du langage C de Brian Kernighan et Denis Ritchie. Il s'est enrichi, au cours de la décennie 1980, parallèlement à la stabilisation et la normalisation de C. Ce langage repose sur les mêmes mécanismes d'écriture et de génération. Il apporte notamment la gestion des exceptions, la gestion des références, la surcharge des opérateurs et les Templates, …etc. Enfin, une rétrocompatibilité a été gardée où les programmes en C sont compilés sans difficulté avec un compilateur C++. Comme tout langage, C++ dispose d'une bibliothèque standard, c'est-à-dire de fonctions et de classes prédéfinies. Elles comportent notamment de nombreux patrons de classes et de fonctions permettant de mettre en œuvre les structures de données les plus importantes (vecteurs dynamiques, listes chaînées, chaînes, …etc.) et les algorithmes les plus usuels. Parmi les environnements de développement AnjutaDevStudio, C++ Builder, Code :: Blocks (open-source), Dev-C++, Eclipse (open-source), Microsoft Visual C++, …etc.

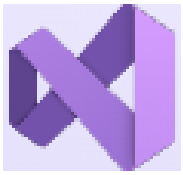


2.2. Presentation embarcadero C++ Builder 10 Seattle

C++ Builder est un logiciel de logiciel de développement rapide d'applications « RAD » conçu par Borland qui reprend les mêmes concepts, la même interface et la même bibliothèque que Delphi en utilisant le langage C++. Il permet de créer rapidement des applications Win32

et Win64 ainsi qu'une interface graphique avec son éditeur de ressources. Il est compatible avec la version de norme ISO C++ de 2011. **Embarcadero C ++ Builder 10 Seattle** est le moyen le plus rapide de créer et de mettre à jour des applications riches en données, hyper connectées et visuellement engageantes pour Windows 10, Mac, Mobile, IoT et plus encore en utilisant le standard C ++.

2.3. Présentation du Visual Studio



Visual Studio est un ensemble complet d'outils de développement permettant de générer des applications Web ASP.NET, des services Web XML, des applications bureautiques et des applications mobiles. Visual Basic, Visual C++, Visual C# et Visual J# utilisent tous le même environnement de développement intégré « IDE » (Integrated Développement Environnement), qui leur permet de partager des outils et facilite la création de solutions faisant appel à plusieurs langages. Ces langages permettent de mieux tirer parti des fonctionnalités du Framework.NET, qui fournit un accès à des technologies clés simplifiant le développement d'applications Web ASP et de Services Web XML grâce à Visual Web Developer.

Visual Studio est un IDE puissant qui prend en charge plusieurs langages de programmation. De plus, il intègre des outils de base de données et de Cloud Computing, notamment Azure, SQL et SQLite. Visual Studio - souvent abrégé en VS - est un IDE utilisé pour éditer, compiler, tester et exécuter du code de programme. Visual Studio 2022 est le premier 64 bits de VS. Microsoft a publié la version 17.0 le 8 novembre 2021. La version 17.1.1 a été publiée le 23 février 2022. Ils existent trois éditions de Visual Studio :

- Communauté VS : il s'agit d'une édition gratuite, d'un IDE complet pour les étudiants, les développeurs open source et individuels.
- VS Professionnel : il s'agit d'outils de développement, de services et d'avantages d'abonnement pour les petites équipes.
- VS Entreprise : il s'agit d'une solution de bout en bout pour répondre aux besoins exigeants de qualité et d'évolutivité des équipes de toutes tailles.

2.4. Présentation de CUDA

CUDA est une plateforme de calcul parallèle et un modèle de programmation inventés par NVIDIA. Il permet des augmentations spectaculaires des performances de calcul en exploitant la puissance

de l'unité de traitement graphique « GPU ». CUDA a été développé avec plusieurs objectifs de conception :

- Fournir un petit ensemble d'extensions aux langages de programmation standard, comme C, qui permettent une implémentation simple d'algorithmes parallèles. Avec CUDA C/C++, les programmeurs peuvent se concentrer sur la tâche de parallélisation des algorithmes plutôt que de passer du temps sur leur implémentation.

- Prend en charge le calcul hétérogène où les applications utilisent à la fois le « CPU » et le « GPU ». Les parties série des applications sont exécutées sur le « CPU » et les parties parallèles sont déchargées sur le « GPU ». En tant que tel, « CUDA » peut être appliqué progressivement aux applications existantes. Le « CPU » et le « GPU » sont traités comme des périphériques distincts qui ont leurs propres espaces mémoire. Cette configuration permet également des calculs simultanés sur le « CPU » et le « GPU » sans contention pour les ressources mémoires. Pour installer « CUDA », il faut tout d'abords vérifier que le système dispos d'un « GPU » compatible avec « CUDA ».

2.4.1. Architecture cuda

CUDA Architecture se compose de trois parties de base, qui aident le programmeur à utiliser efficacement toute la capacité de calcul de la carte graphique du système. CUDA Architecture divise le périphérique en grilles, blocs et threads dans une structure hiérarchique[**11]:**

- **La grille :** Une grille est un groupe de threads exécutant tous le même noyau. Ces threads ne sont pas synchronisés. Chaque appel à CUDA à partir du processeur est effectué via une grille. Démarrage d'une grille sur le processeur est une opération synchrone, mais plusieurs grilles peuvent s'exécuter à la fois. Sur les systèmes multi-GPU, les grilles ne peuvent pas être partagées entre les GPU car ils utilisent plusieurs grilles pour une efficacité maximale.
- **Le bloc** : Les grilles sont composées de blocs. Chaque bloc est une unité logique contenant un nombre de threads de coordination, une certaine quantité de mémoire partagée. Tout comme les grilles ne sont pas partagées entre GPU, les blocs ne sont pas partagés entre multiprocesseurs.
- **Thread** : Les blocs sont composés de fils. Les threads sont exécutés sur les cœurs individuels du multiprocesseur, mais contrairement aux grilles et aux blocs, ils ne sont pas limités à un seul cœur. Habituellement, il peut y avoir 1024 threads par bloc.

2.4.2. Le principe de fonctionnement de CUDA

La plate-forme CUDA est conçue pour fonctionner avec des langages de programmation tels que C, C ++ et Fortran. Ceci rend plus facile, pour les intéressés de la programmation parallèle, l'accès aux ressources GPU, contrairement aux API antérieures. Un programme CUDA démarre son exécution dans le CP U et utilise la mémoire centrale RAM, puis, arrivé à la partie parallèle, le programme passe par plusieurs étapes. Les étapes utilisées par CUDA pour exécuter la partie parallèle sur le GPU sont comme suit:

- **Étape 01 : Copie vers GPU**

Après avoir défini et préparé les données à charger dans la mémoire GPU, CUDA copie ces données, ainsi que les paramètres de la fonction à exécuter kernel, de la mémoire centrale (RAM), vers la mémoire globale du GP U. Si les textures et les constantes sont définies, elles sont alors chargées de la mémoire centrale (RAM) vers leurs mémoires respectives sur le gpu

- **Étape 02 : chargement des instructions**

Instruire les unités de calcul du GPU des instructions de la fonction kernel à exécuter qui sont préalablement chargées dans la mémoire GPU. Par ailleurs, le nombre de threads et de blocs à utiliser (choisis par le programmeur) sont communiqués au GP U.

- **Étape 03 : Exécution sur GPU**

Après avoir reçu les instructions et les données, le GP U répartit les threads en blocs et définit les warps puis exécute en parallèle les instructions dans chaque cœur du GP U.

- **Étape 04 : Récupération des résultats**

Au terme de l'exécution de tous les threads, on procède à la copie des résultats du GP U vers la mémoire centrale RAM. Le CP U prend la relève et termine l'exécution du programme.

3. Présentation de l'application développée

L'approche développée est composée de deux parties complémentaires :

- ***Première partie :*** c'est le « Client » qui est développée sous « Embarcadero » en utilisant Builder C++. C'est une application graphique où la fenêtre principale est composée de quatre (04) onglets (Figure 46).
-

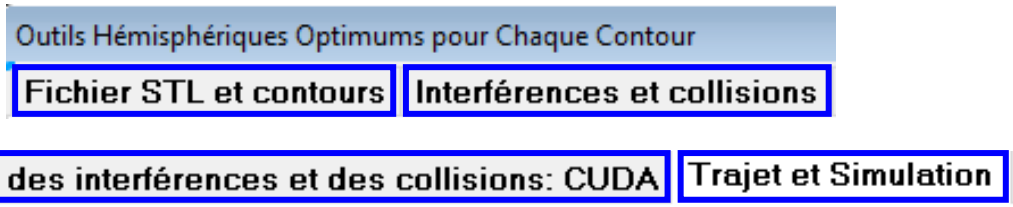


Figure 44 : Onglets de l'application « Client ».

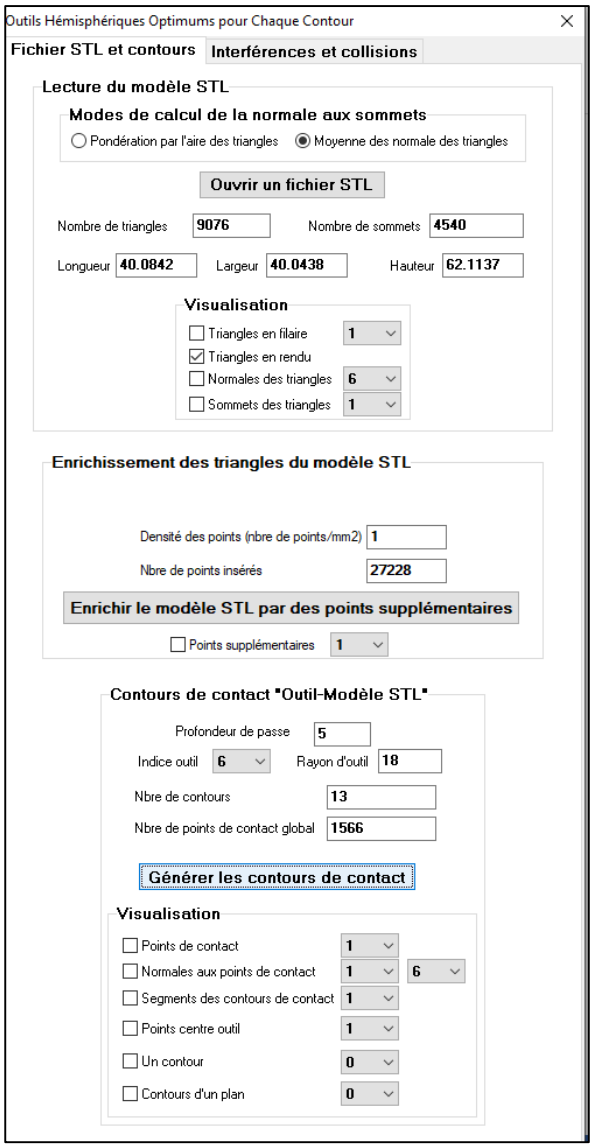
Les onglets de l'application « Client » sont (Figure 44) :

- ***Onglet « Fichier STL et contours » :*** son rôle consiste en la lecture du modèle STL de la pièce et le calcul des contours de contact.

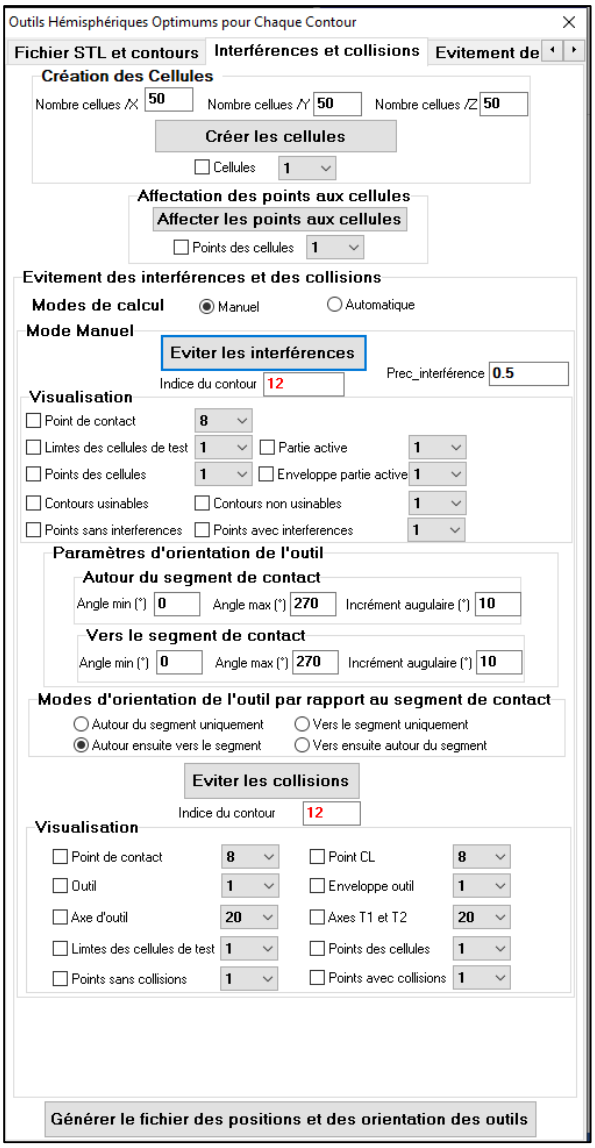
- ***Onglet « Interférences et collisions » :*** cet onglet est réservé au calcul des outils optimums et leurs orientations lors du calcul séquentiel.

- ***Onglet « Evitement des interférences et des collisions : CUDA » :*** cet onglet est réservé au calcul des outils optimums et leurs orientations lors du calcul parallèle.

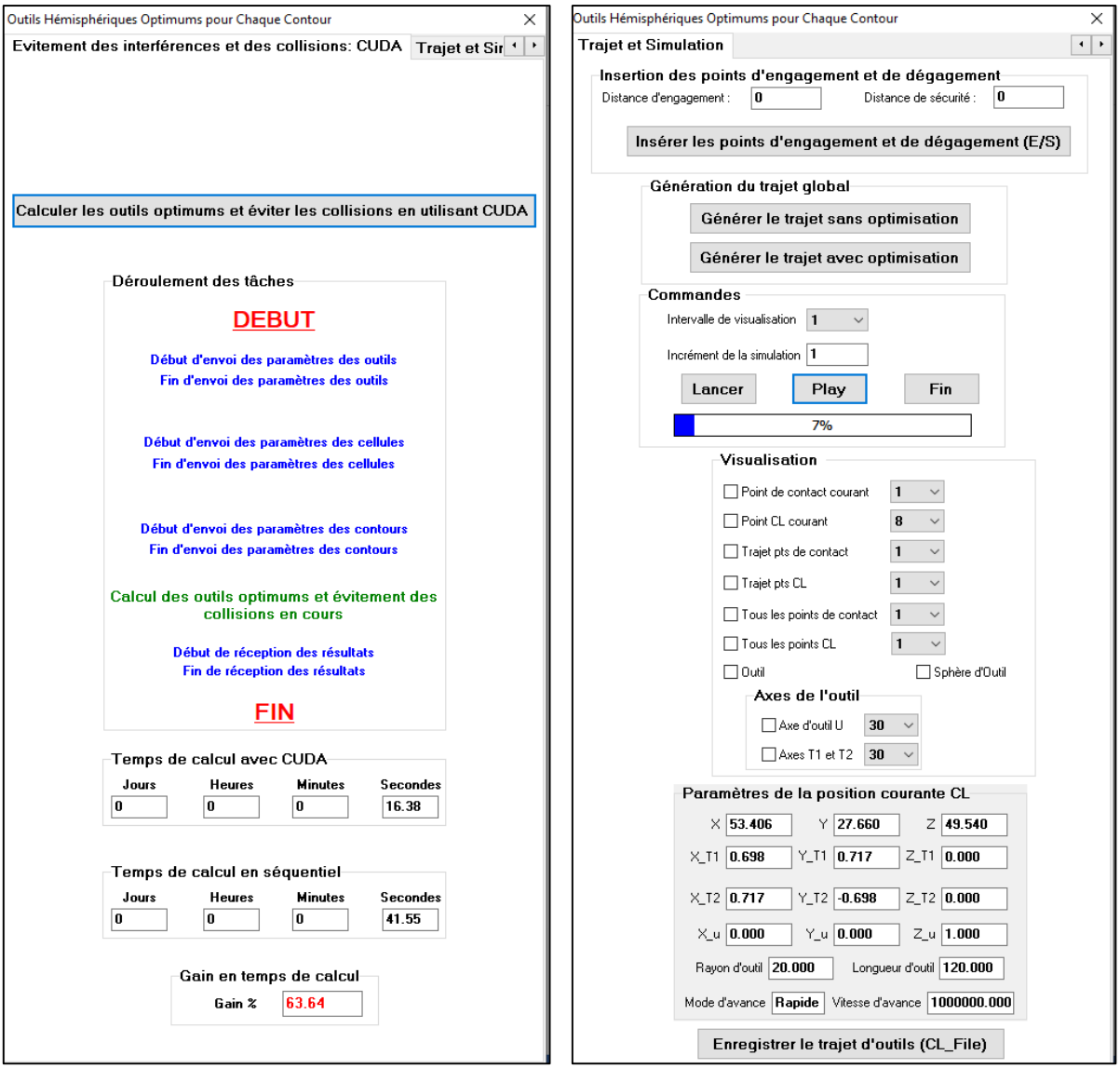
- ***Onglet « Trajet et simulation » :*** il est utilisé pour générer le trajet d'outils et pour la simulation virtuelle des mouvements des outils par rapport aux surfaces à usiner.



a. Fichier STL et contours.



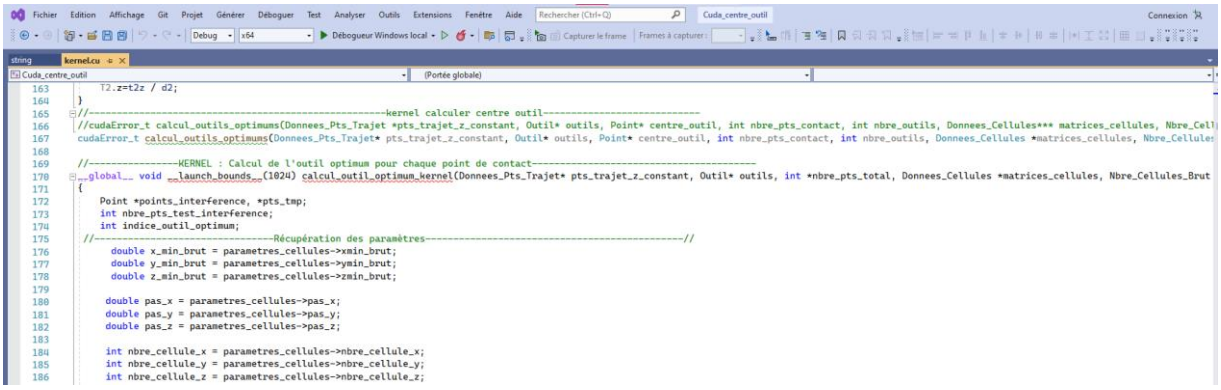
b. Interférences et collisions.



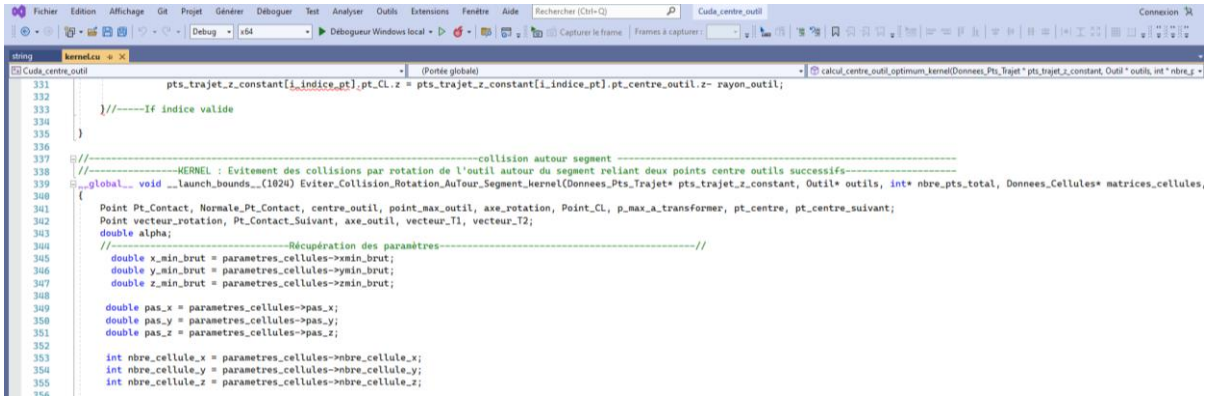
c. Interférences et collisions : CUDA. d. Trajet et simulation.

Figure 45: Onglets de l'application « Client ».

- ***Deuxième partie :*** c'est le « Serveur » qui est développée sous Visual Studio. C'est une application console pour les calculs parallèles. Les principales fonctions développées pour le calcul parallèle sont données par la Figure 46.

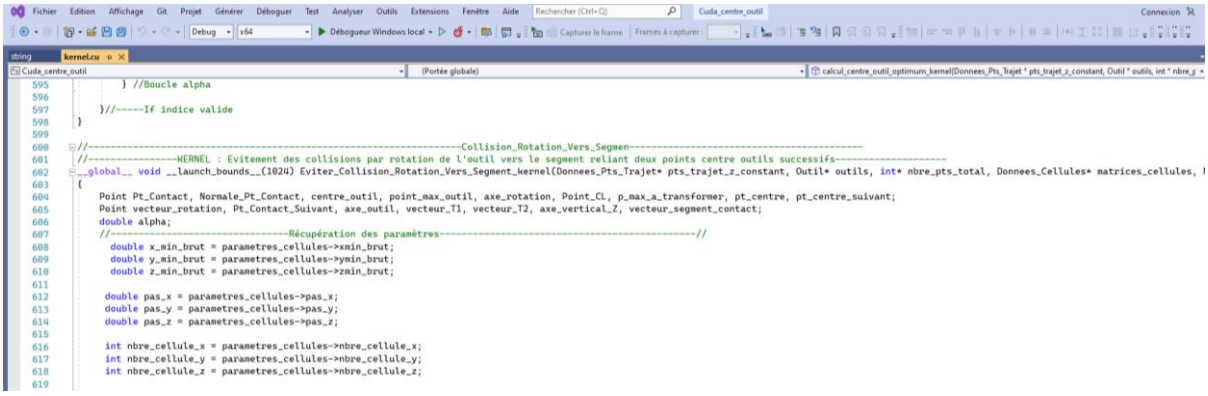


a. Calcul des outils optimums par « CUDA ».



```
                pts_trajet_z_constant[i_indice_pt].pt_CL.z = pts_trajet_z_constant[i_indice_pt].pt_centre_outil.z- rayon_outil;

    }//-----If indice valide

}

//-------------------------------------------------------------------collision autour segment ---------------------------------------------------
//----------------KERNEL : Evitement des collisions par rotation de l'outil autour du segment reliant deux points centre outils successifs-------------------
__global__ void __launch_bounds__(1024) Eviter_Collision_Rotation_AuTour_Segment_kernel(Donnees_Pts_Trajet* pts_trajet_z_constant, Outil* outils, int* nbre_pts_total, Donnees_Cellules* matrices_cellules,
{
    Point Pt_Contact, Normale_Pt_Contact, centre_outil, point_max_outil, axe_rotation, Point_CL, p_max_a_transformer, pt_centre, pt_centre_suivant;
    Point vecteur_rotation, Pt_Contact_Suivant, axe_outil, vecteur_T1, vecteur_T2;
    double alpha;
    //-----------------------------------Récupération des paramètres-----------------------------------------------//
      double x_min_brut = parametres_cellules->xmin_brut;
      double y_min_brut = parametres_cellules->ymin_brut;
      double z_min_brut = parametres_cellules->zmin_brut;

     double pas_x = parametres_cellules->pas_x;
     double pas_y = parametres_cellules->pas_y;
     double pas_z = parametres_cellules->pas_z;

     int nbre_cellule_x = parametres_cellules->nbre_cellule_x;
     int nbre_cellule_y = parametres_cellules->nbre_cellule_y;
     int nbre_cellule_z = parametres_cellules->nbre_cellule_z;
```

b. Evitement des collisions par rotation de l'outil autour du segment par « CUDA ».



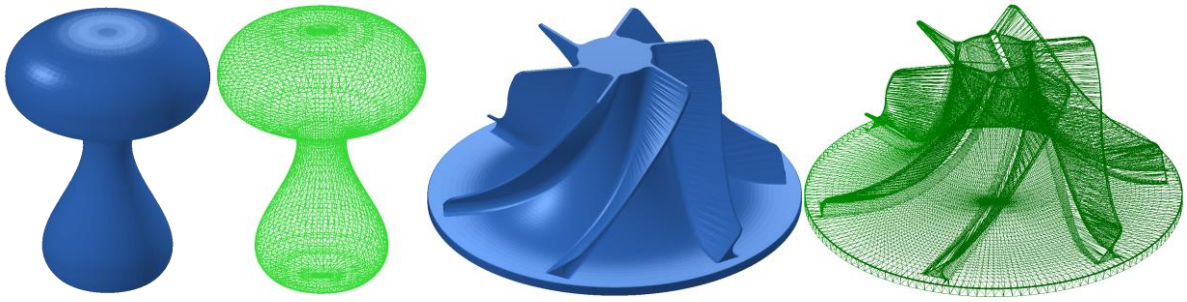
```
        } //Boucle alpha

    }//-----If indice valide
}

//-------------------------------------------------------------------Collision_Rotation_Vers_Segmen-----------------------------------------------
//----------------KERNEL : Evitement des collisions par rotation de l'outil vers le segment reliant deux points centre outils successifs-------------------
__global__ void __launch_bounds__(1024) Eviter_Collision_Rotation_Vers_Segment_kernel(Donnees_Pts_Trajet* pts_trajet_z_constant, Outil* outils, int* nbre_pts_total, Donnees_Cellules* matrices_cellules,
{
    Point Pt_Contact, Normale_Pt_Contact, centre_outil, point_max_outil, axe_rotation, Point_CL, p_max_a_transformer, pt_centre, pt_centre_suivant;
    Point vecteur_rotation, Pt_Contact_Suivant, axe_outil, vecteur_T1, vecteur_T2, axe_vertical_Z, vecteur_segment_contact;
    double alpha;
    //-----------------------------------Récupération des paramètres-----------------------------------------------//
      double x_min_brut = parametres_cellules->xmin_brut;
      double y_min_brut = parametres_cellules->ymin_brut;
      double z_min_brut = parametres_cellules->zmin_brut;

     double pas_x = parametres_cellules->pas_x;
     double pas_y = parametres_cellules->pas_y;
     double pas_z = parametres_cellules->pas_z;

     int nbre_cellule_x = parametres_cellules->nbre_cellule_x;
     int nbre_cellule_y = parametres_cellules->nbre_cellule_y;
     int nbre_cellule_z = parametres_cellules->nbre_cellule_z;
```

c. Evitement des collisions par rotation de l'outil vers le segment par « CUDA ».

Figure 46 : Fonctions de l'application « Serveur ».

4. Tests et validations

Les tests et les validations sont effectués sur deux pièces (Figure 47). Ces pièces composées de surfaces de géométries très complexes sont conçues dans un logiciel spécialisé de « CAO ». La première pièce est une pièce de révolution utilisée pour montrer toutes les fonctionnalités du module développé. La deuxième pièce est une pièce réelle appelée roue à ailettes utilisées dans les turbocompresseurs. Le modèle continu de chaque pièce est approximé par un ensemble de triangles sauvegardés dans un fichier STL qui sera utilisé comme une donnée d'entrée pour la plateforme logicielle.



a. Première pièce de test. b. Deuxième pièce de test.

Figure 47 : Modèles « CAO » et « STL » des deux pièces de test

Le Tableaux 2 donne les principales informations des deux pièces.

Tableau 4 : Paramètres des deux modèles de pièces.

Modèles de pièce	Nombre de triangles	Nombre de sommets	Longueur (mm)	Largeur (mm)	Epaisseur (mm)
Modèle 1	9076	4540	39.93	39.89	63.11
Modèle 2	65248	32626	399.99	399.99	180

L'interface du module logiciel précédemment développé avec nos améliorations contient une seule fenêtre composée de quatre pages « onglets » représentant la séquence des actions à mener pour aboutir au résultat final. Ces pages sont :

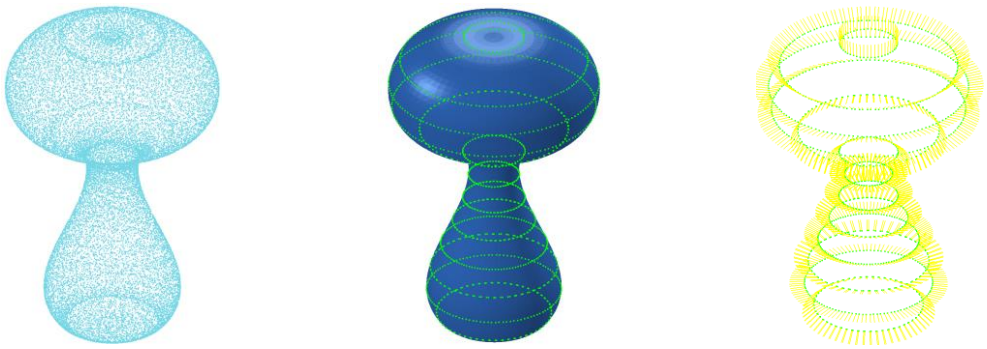
- Fichier STL et contours
- Evitement des interférences et des collisions par un calcul séquentiel.
- Evitement des interférences et des collisions par un calcul parallèle « CUDA ».
- Simulation virtuelle des mouvements des outils.

4.1. Fichier STL et contours

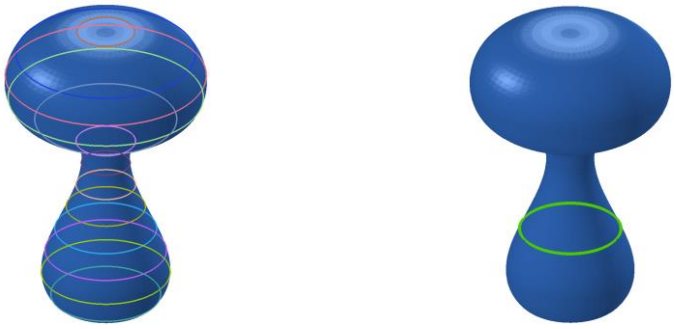
La première page est « Fichier STL et contours ». Dans cette page, l'utilisateur doit cliquer sur le bouton « Ouvrir fichier STL » pour sélectionner une pièce à analyser. Après cette étape, l'utilisateur doit spécifier la densité des points à insérer pour chaque triangle et clique sur le bouton « Enrichir le modèle STL par des points supplémentaires ». A la fin, il doit spécifier une profondeur de passer et cliquer sur le bouton « Générer les contours de contact ». Il est possible de visualiser les informations suivantes :

- Points supplémentaires.
- Points de contact.
- Normales aux points de contact.
- Segments des contours de contact.
- Un seul contour.
- Les contours d'un plan de coupe donné.

Pour le premier modèle, la densité des points et la profondeur de passe sont fixées égales à 4points/mm^2 et 5mm. Avec ces données, le nombre de points insérés est égal à 29132 et le nombre de contours est égal à 13. La Figure 48 montre les différents résultats pour ce modèle.



a. Points insérés. b. Points de contact. c. Normales aux points de contact.



d. Contours de contact. e. Contour d'un plan de coupe.

Figure 48: Résultats de la première page pour le premier modèle.

La Figure 49 montre les contours pour le deuxième modèle de test relatifs à une profondeur de passe de 5mm.

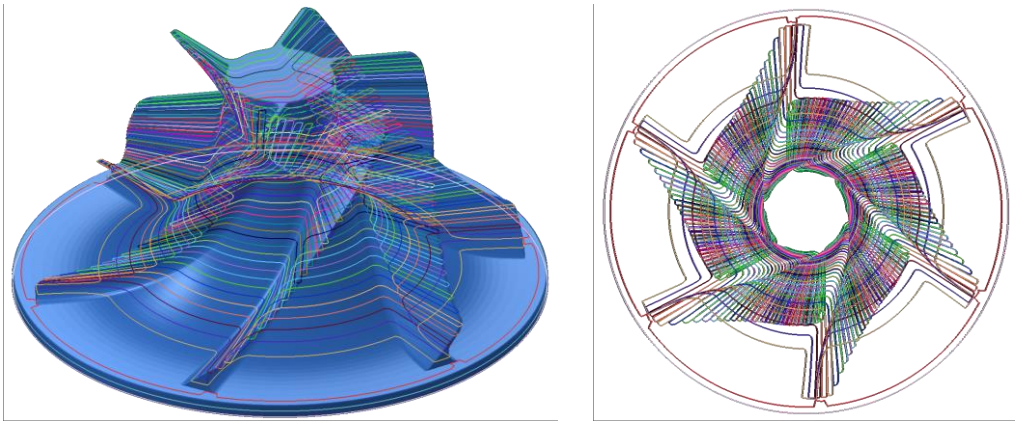


Figure 49 : Résultats des contours pour le deuxième modèle.

4.2. Evitement des interférences et des collisions par un calcul séquentiel

La deuxième page est « Interférences et collisions». Dans cette page, les problèmes d'interférences et de collisions sont éliminés par le choix des outils hémisphériques optimums suivi de l'orientation de ces outils en chaque point de contact en passant par plusieurs étapes. La première étape requiert la spécification du nombre de cellules suivant les trois axes X, Y et Z suivie d'un clic sur le bouton « Créer cellules » pour créer une matrice tridimensionnelle des cellules. Dans la deuxième étape, les points supplémentaires sont affectés aux cellules suite au clic sur le bouton « Affecter les points aux cellules ». Il est possible de visualiser les cellules et ses points. Comme exemple, les nombres de cellules sont pris égaux à 5, 5 et 10 (Figure 50). Plus les nombres des cellules sont grands, plus les temps des tests sont réduits.

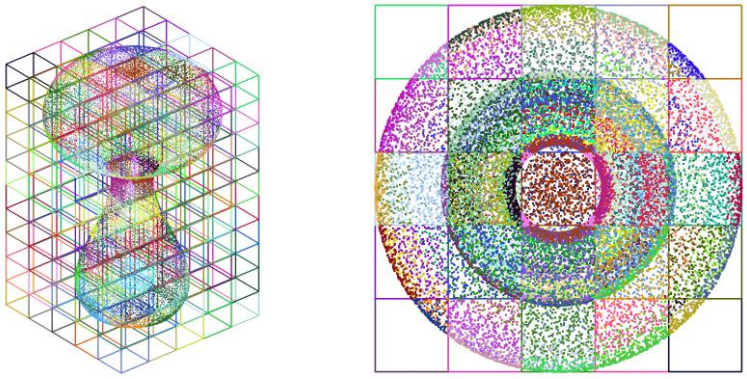


Figure 50 : Création des cellules et affectation des points supplémentaires.

L'évitement des interférences, la détermination des outils optimums hémisphériques pour chaque point de contact et l'identification de l'outil optimum pour chaque contour sont traités dans la troisième étape. Lors de cette étape, il est possible de visualiser :

- Point de contact courant.
- Limites des cellules en chevauchement avec la partie active de l'outil.
- Points des cellules en chevauchement.
- Contours usinables et contours non usinables.
- Points sans interférences et points avec interférences.
- Partie active de l'outil et son enveloppe.

La Figure 51 montre quelques informations visualisées pour différents points de contact.

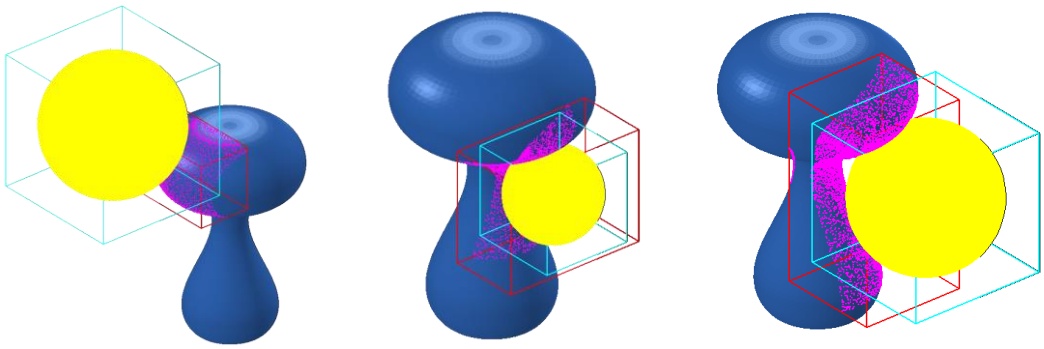


Figure 51 : Evitement des interférences en différents points de contact.

Après la phase des calculs, il ressort que tous les contours sont usinables (Figure 52).

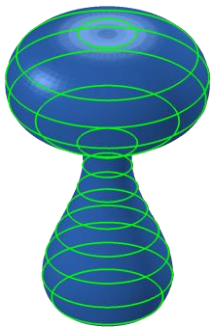


Figure 52 : Contours usinables.

L'évitement des collisions est considéré dans la quatrième étape. Lors de cette étape, l'utilisateur doit spécifier les angles limites et l'incrément angulaire pour chaque mode d'orientation de l'outil. Il est possible de visualiser les informations suivantes :

- Point de contact courant.
- Outil et axes d'outil.
- Limites des cellules en chevauchement avec la partie cylindrique de l'outil.

- Points des cellules en chevauchement.
- Point extrémité d'outil « C_L ».
- Enveloppe d'outil.
- Points sans collisions et points avec collisions.

La Figure 53 montre quelques informations visualisées pour différents points de contact.

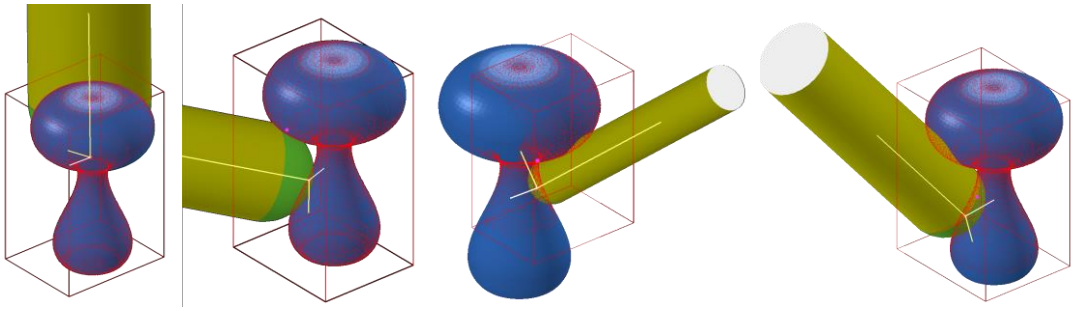
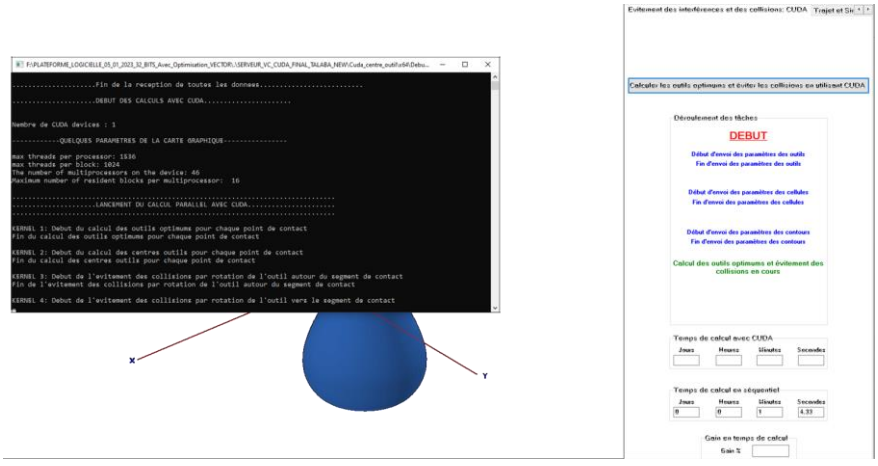


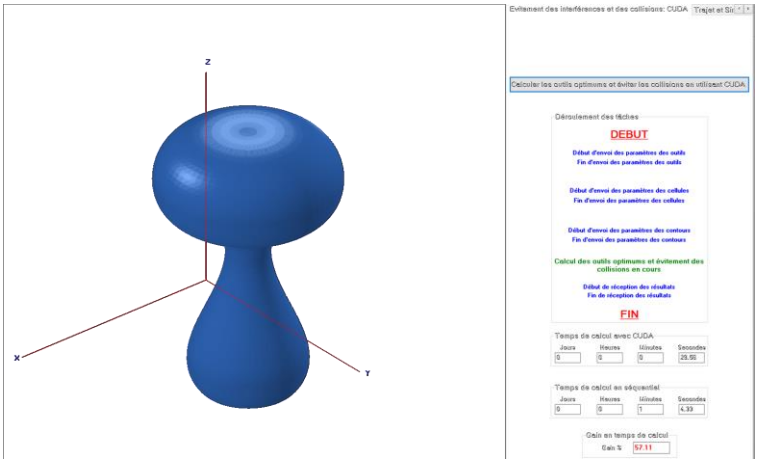
Figure 53 : Evitement des collisions en différents points de contact en séquentiel.

4.3. Evitement des interférences et des collisions en utilisant « CUDA »

La troisième page est « Evitement des interférences et des collisions : CUDA ». Dans cette page, le calcul parallèle avec « CUDA » est utilisé pour éviter les interférences et les collisions. En cliquant sur le bouton « calculer les outils optimum et éviter les collisions en utilisant CUDA », tous les paramètres nécessaires au calcul (cellules, outils, contours, angles d'orientation) sont envoyés au « Serveur » pour le calcul parallèle. Après leur réception, les calculs sont lancés en parallèle au niveau du « Serveur ». Une fois les calculs finalisés, les résultats sont envoyés du « Serveur » au « Client » pour générer le trajet d'outils et le simuler (Figure 54). Les temps de calcul en séquentiel, en parallèle et le gain de temps sont affichés.



a. Envoi des données du « Client » au « Serveur ».



b. Envoi des résultats du « Serveur » au « Client ».

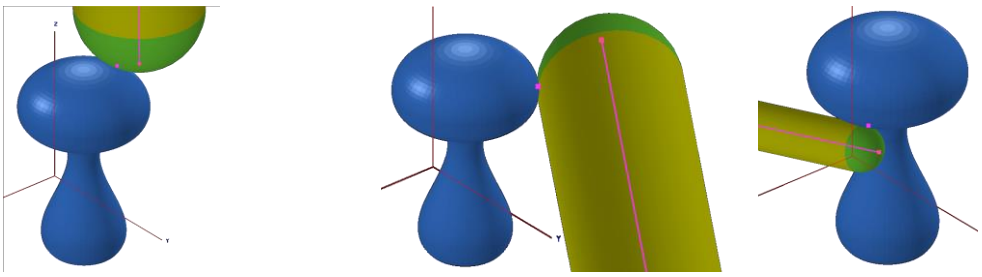
Figure 54 : Evitement des interférences et des collisions en parallèle.

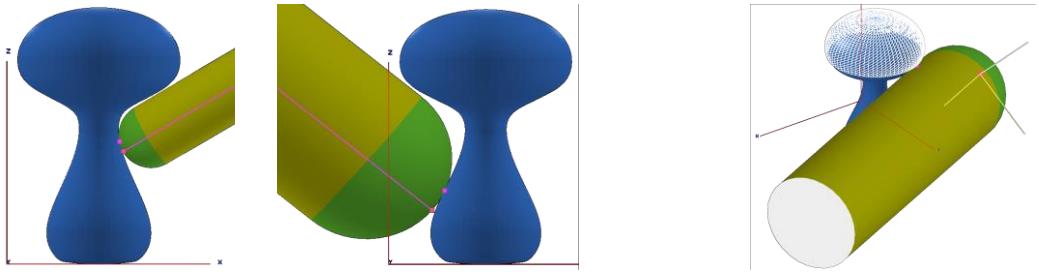
4.4. Simulation virtuelle des mouvements des outils

La quatrième page est « Trajet et simulation » où l'utilisateur doit spécifier les distances d'engagement (de dégagement) et de sécurité. Par un clic sur le bouton « Insérer les points d'engagement et de dégagement (E/S) », deux points au début et deux points à la fin de chaque contour sont insérés. L'utilisateur peut générer un trajet sans optimisation ou un trajet avec optimisation. Une fois le trajet généré, il est possible de visualiser :

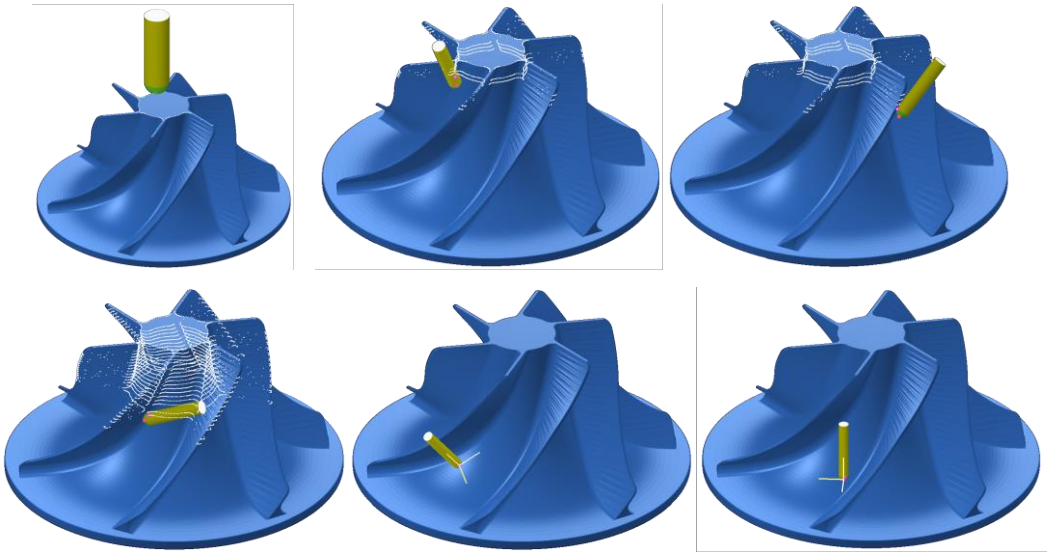
- Point de contact courant « C_C » et point extrémité d'outil « C_L ».
- Trajet des points de contact et trajet des points extrémité d'outil.
- Paramètres de l'outil et son mode d'avance ainsi que ses axes.
- Coordonnées du point de contact, composantes des vecteurs d'outils.

La Figure 55 montre les outils optimums pour différents points de contact pour les deux modèles considérés. Ces Figures montrent que les dimensions ainsi que les orientations des outils changent en fonction de la position du point de contact par rapport à la surface.





a. Premier modèle.



b. Deuxième modèle.

Figure 55 : Positions et orientations des outils lors de la simulation des mouvements.

5. Comparaison entre l'approche séquentielle et l'approche parallèle

Pour mettre en évidence l'efficacité et le gain considérable en termes de temps de calcul, les deux modèles sont testés pour différentes profondeurs de passe. Les calculs sont menés sur une station de travail avec les caractéristiques suivantes :

- Processeur : Intel(R) Core(TM) i9-10900K CPU @ 3.70GHz 3.70 GHz
- RAM : 32.0 Go
- Carte graphique : NVIDIA GeForce RTX 3070
  - Nombre de cœurs NVIDIA CUDA : 5888
  - Fréquence Boost (GHz) : 1.73
  - Fréquence de base (GHz) : 1.50
  - Mémoire standard : 8 Go GDDR6
  - Résolution numérique maximale : 7680x4320

Pour l'évitement des collisions, les paramètres des angles d'orientation pour les deux modèles de pièces sont donnés dans le Tableau 5 et le Tableau 6.

Tableau 5 : Paramètres d'orientation de l'outil pour le premier modèle.

Orientations	Angle minimum	Angle maximum	Incrément angulaire
Rotation autour du segment	0	90	10
Rotation vers le segment	0	270	10

Tableau 6 : Paramètres d'orientation de l'outil pour le deuxième modèle.

Orientations	Angle minimum	Angle maximum	Incrément angulaire
Rotation autour du segment	0	270	10
Rotation vers le segment	0	270	10

Avant de lancer les calculs pour les deux modèles, les nombres de cellules sont fixés à 50 suivant les axes X, Y et Z tandis que la densité des points est prise égale à 1point/mm^2.

Les résultats sont donnés dans le Tableau 5 et le Tableau 6. Il ressort des résultats obtenus que plus la profondeur de passe est petite plus le gain est important. Pour les deux modèles, pour des profondeurs de passe proche de la réalité (usinage en finition), les gains de temps sont plus grands que 98%. Ces résultats montrent que l'utilisation des processeurs graphiques « GPU » permet une réduction très importante des temps de calcul, ce qui permet de réduire les coûts, le cycle de développement de nouveaux produits et de permettre aux ingénieurs de tester un nombre plus grands de configurations et de cas possibles.

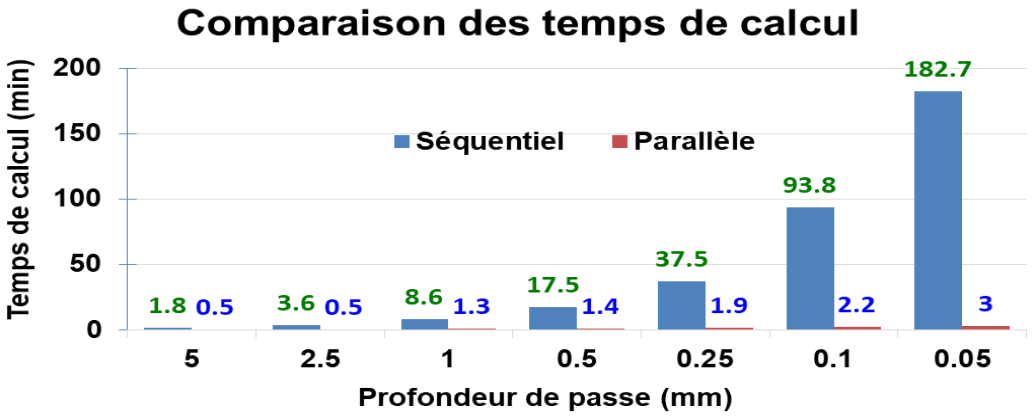
Tableau 7 : Résultats pour le premier modèle de test.

Profondeur de passe (mm)	Nombre de contours	Nombre de points de contact	Temps de calcul en séquentiel	Temps de calcul en parallèle	Gain de temps
5	12	1464	1min 46.28s	0min 45.06s	58.89
2.5	25	3205	3min 38.52s	0min 50.38s	77.71
1	63	7877	8min 33.13s	1min 19.97s	84.93
0.5	126	15864	17min 27.64s	1min 26.25s	92.04
0.25	252	31640	37min 29.44s	1min 58.69s	94.90
0.1	631	78963	1h 33min 48.33s	2min 9.27s	97.78
0.05	1262	157938	3h 2min 39.53s	3min 0.01s	98.41

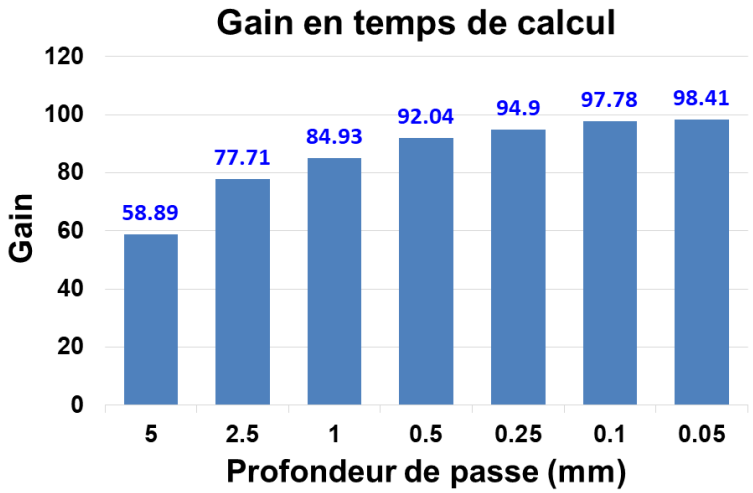
Tableau 8 : Résultats pour le deuxième modèle de test.

Profondeur de passe (mm)	Nombre de contours	Nombre de points de contact	Temps de calcul en séquentiel	Temps de calcul en parallèle	Gain de temps
10	19	13813	1min 36.59s	0min 57.31s	45.84
5	37	28064	3min 12.86s	1min 1.44s	70.95
2.5	73	56015	6min 18.30s	1min 7.20s	83.83
1	181	140241	16min 7.84s	1min 47.06s	89.93
0.5	361	280436	32min 6.39s	2min 51.63s	91.90
0.25	721	560496	1h 05 min 27.66s	5min 22.05s	92.52

La Figure 56 montre les diagrammes de comparaison des temps de calcul entre le calcul séquentiel et le calcul parallèle avec les gains pour chaque profondeur de passe.



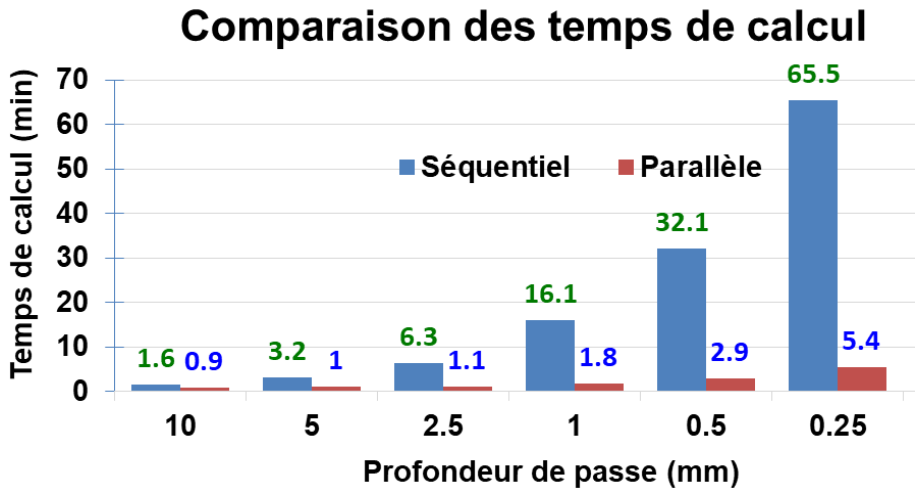
a. Temps de calcul pour les deux approches.



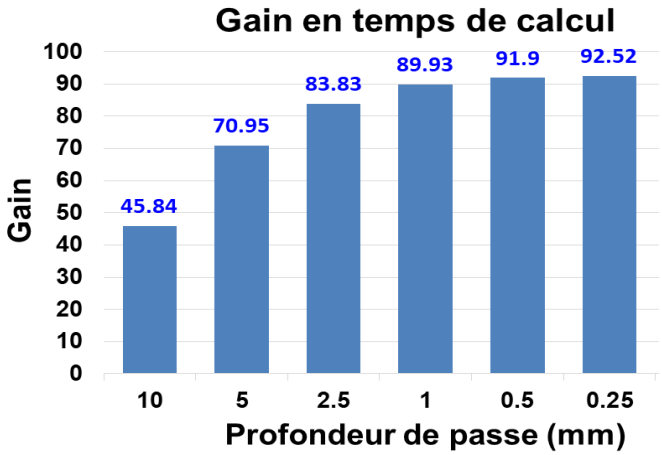
b. Gain de temps.

Figure 56 : Comparaison des résultats pour le premier modèle.

La Figure 57 montre les diagrammes de comparaison des temps de calcul entre le calcul séquentiel et le calcul parallèle avec les gains pour chaque profondeur de passe.



a. Temps de calcul pour les deux approches.



b. Gain de temps.

Figure 57 : Comparaison des résultats pour le deuxième modèle.

Les résultats obtenus montrent l'importance du calcul parallèle dans le processus de développement de nouveaux produits en raison de la réduction très considérable des temps de calcul par rapport au calcul séquentiel.

6. Conclusion

Dans ce chapitre, nous avons présenté notre application développée et les résultats de test obtenus. Au début, nous avons présenté les environnements de développement, le langage de programmation utilisé. Ensuite, nous avons présenté ses différentes interfaces et fonctionnalités tout en enrichissant cette partie par des tests de validations sur deux exemples depuis la lecture du fichier STL jusqu'à la simulation du trajet d'usinage en 05-axes.

CONCLUSION GENERALE

Nous avons présenté dans ce mémoire notre projet qui consiste à proposer, concevoir et implémenter un module logiciel permettant l'optimisation de la génération du trajet d'outil par parallélisation des calculs des évitements des interférences pour déterminer les outils optimums hémisphériques ainsi que leurs orientations pour éviter les collisions lors de la finition des pièces complexes, définies par des modèles STL, par la stratégie d'usinage Z-Constant sur des fraiseuses numériques à 05-axes. Ce module logiciel permet de générer une trajectoire d'outils sans interférences et sans collisions.

Lors de la réalisation de ce projet, une étude bibliographique a été faite sur les modèles de représentation des surfaces complexes ainsi que sur les machines-outils à commande numérique et les différentes stratégies d'usinage utilisées dans la phase de fabrication. Cette étude a été complétée par une étude conceptuelle de l'application et une implémentation informatique des solutions proposées aux différentes questions posées durant la réalisation de cette approche sous Windows en utilisant le langage de programmation « C++ », les environnements de développement « Builder C++ » et « Microsoft Visual Studio C++ ». A la fin, des tests de validation ont été menés sur deux pièces, l'une est d'une forme géométrique très complexe.

Le principal résultat obtenu est l'intégration à la plateforme logicielle de production des pièces de formes complexes développée par l'équipe « CFAO », d'un module logiciel graphique interactif sous Windows permettant de réaliser les tâches suivantes :

- ✓ Récupération des paramètres du modèle STL de la pièce.
- ✓ Récupération des contours d'usinage.
- ✓ Enrichissement du modèle STL par l'insertion aléatoire de nouveaux points.
- ✓ Création des cellules et affectation des points insérés aux cellules.
- ✓ Création de la « Socket » et établissement de la liaison entre le « Client » (Builder C++) et le « Serveur » (Microsoft Visual Studio C++).
- ✓ Paramétrage et envoi des données.
- ✓ Evitement des interférences aux points de contact et détermination des outils hémisphériques optimums par un calcul parallèle.

- ✓ Détermination de l'outil hémisphérique optimum pour chaque contour.
- ✓ Evitement des collisions interférences aux points de contact et détermination de l'orientation de l'outil hémisphérique par un calcul parallèle.
- ✓ Paramétrage et réception des résultats.
- ✓ Génération du trajet d'outils.
- ✓ Simulation virtuelle des mouvements d'outils.
- ✓ Calcul du gain de temps entre le calcul séquentiel et le calcul parallèle.

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

- ➢ Combinaison de plusieurs outils hémisphériques pour l'usinage de chaque contour.
- ➢ Combinaison de plusieurs formes d'outils pour l'usinage de la pièce.
- ➢ Intégration du calcul parallèle lors de la détermination des outils optimums cylindriques et toriques.
- ➢ Proposition d'une nouvelle approche pour l'orientation de l'outil pour éviter les collisions.
- ➢ Intégrer le calcul parallèle dans la simulation virtuelle de l'usinage.
- ➢ Génération du programme d'usinage « G-Code » pour n'importe quelle cinématique des fraiseuses numériques 05-axes.
- ➢ Considération de l'environnement de l'usinage (porte outil, montage d'usinage, …etc.) lors du test des collisions.
- ➢ Finition des surfaces gauches avec le profil de la fraise (fraisage en roulant).
- ➢ Simulation virtuelle de la cinématique des fraiseuses numériques 05-axes.

REFERENCES BIBLIOGRAPHIQUES

[1] "Finition des Pièces Complexes par la Stratégie « Plans Parallèles » sur des Fraiseuses Numériques à 03-Axes." https://di.univ-blida.dz/xmlui/handle/123456789/4135 (accessed Jun. 03, 2023).

[2] M. Bey, Z. Tchantchane, L. Kheidri, and N. Benhenda, "Automatisation de l ' Opération de Tréflage des Surfaces Gauches à Partir des Modèles STL," pp. 7–8, 2011.

[3] A. Lakrib. and M. Boutagga., "Conception et réalisation d'une application pour finition des surfaces complexes sur des fraiseuses à 05-axes.," Jun. 2015, Accessed: may. 23, 2023. [Online]. Available: https://di.univ-blida.dz/jspui/handle/123456789/10887

[4] M. Bey, "Optimisation de l ' Opération de Finition des Surfaces Gauches par la Combinaison des Formes et des Dimensions d ' Outils à Partir des Modèles STL," no. December, 2017.

[5] M. Bey *et al.*, "Finition des Surfaces Gauches par Combinaison de Différentes Formes et Dimensions d ' Outils," no. April, pp. 10–11, 2012.

[6] "Combinaison des Outils Hémisphériques, Cylindriques et Toriques pour la Finition des Surfaces Complexes sur des Fraiseuses à 05-Axes," 2019, Accessed: Jun. 10, 2023. [Online]. Available: https://di.univ-blida.dz/jspui/handle/123456789/2204

[7] "Différence entre le calcul parallèle et distribué »wiki utile Comparez la différence entre des termes similaires - La Technologie - 2023." https://fr.strephonsays.com/parallel-and-vs-distributed-computing-2413 (accessed may. 03, 2023).

[8] "Définition | GPU - Graphical Processing Unit - Processeur graphique - Microprocesseur graphique | Futura Tech." https://www.futura-sciences.com/tech/definitions/informatique-gpu-5739/ (accessed Jun. 10, 2023).

[9] "CPU VS GPU : les différences - malekal.com." https://www.malekal.com/cpu-vs-gpu-les-differences/ (accessed Jun. 10, 2023).

[10] "La vraie différence entre CPU et GPU | Jedha Bootcamp." https://www.jedha.co/blog/vraie-difference-cpu-gpu (accessed Jun. 10, 2023).

[11] J. Ghorpade, "GPGPU Processing in CUDA Architecture," *Adv. Comput. An Int. J.*, vol. 3, no. 1, pp. 105–120, 2012, doi: 10.5121/acij.2012.3109.