République Tunisienne
Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique

Institut Supérieur des Sciences
Appliquées et de Technologie
de Sousse
Université de Sousse

DEPARTEMENT INFORMATIQUE

RAPPORT DE PROJET DE FIN D'ETUDES

En vue de l'obtention du:
Diplôme National d'Ingénieur Informatique
Option: Génie Logiciel-Architecture Logicielle

Reconstruction d'un modèle CAO cohérent à partir d'un nuage de points

Elaboré par :

Ben Makhlouf Aicha

Soutenu le 23/07/2016 devant le jury :

Président :	**Nom & Prénom Président**	**ISSAT de Sousse**
Examinateur :	**Nom & Prénom Examinateur**	**ISSAT de Sousse**
EncadrÁÎô :	**Louhichi Borhen**	**ISSAT de Sousse**
Encadrant Industriel :	**Subsol Gérard**	**LIRMM**

Année Universitaire : 201υ/201φ

Code Sujet: FI-

Dédicaces

***A** mon mari, l'homme adorable, pour son amour éternel et son encouragement*

***A** mon trésor d'amour maman chérie, pour ses sacrifices illimités*

***A** mon très cher papa, que dieu le protège de tout mal*

***A** mes deux chers frères qui m'aiment et que j'adore*

***A** ma deuxième famille Chaker : Mounir, Lilia et Mahdi, je vous aime trop*

***A** mes chers : Amira, Mohamed et Mariem**,** que dieu vous protège*

***A** mes chères : Marwa, Amal, Rym, Belkis, Souhir, Sirine*

***A** tous mes amis et ceux qui m'aiment*

*Que vous soyez tous fiers d'**Aicha***

Remerciement

À l'issue de la rédaction de cette recherche, j'avoue que mes travaux réalisés ne sont pas des travaux solitaires. En effet, je n'aurais jamais pu les réaliser sans le soutien des personnes dont l'intérêt manifesté à l'égard de ma recherche m'a permis de progresser mes premiers pas dans le domaine de recherche.

En premier lieu, j'exprime toute ma gratitude à mon encadreur Monsieur **Borhen LOUHICHI** qui n'a cessé de me diriger, conseiller et m'encourager. Je le remercie pour toute la confiance et le soutien qu'il m'a attribué, pour son temps précieux dont il m'a fait part et de m'avoir donné cette opportunité de travailler un tel sujet intéressant, qui est mon premier point de départ pour commencer mes recherches doctorales.

Je tiens à remercier aussi Monsieur **Gérard SUBSOL** (chargé de recherche CNRS) pour m'avoir permis d'effectuer ce projet de fin d'études au sein de l'équipe ICAR du Laboratoire d'Informatique, de Robotique et de Micro-électronique de Montpellier (LIRMM) et aussi pour l'intérêt qu'il m'a accordé tout au long de ce projet et pour m'avoir fait partager ses expertises dans le domaine.

Un grand merci à toute l'équipe ICAR, chaleureusement Mr **William PUECH**, Mr **Olivier STRAUSS** et aussi madame **Roseline Bénière** de l'entreprise C4W ainsi que tous mes amis doctorants et stagiaires rencontrés au fil de ces mois et également tous mes co-bureaux pour la bonne ambiance de travail !

Mes remerciements s'adressent aussi à tous les membres du jury qui m'ont honoré d'avoir accepté d'évaluer mon travail.

Enfin, pour ces cinq années de bonne ambiance, je remercie très chaleureusement mes enseignants à l'Institut Supérieur des Sciences Appliquées et de Technologie de Sousse pour leurs disponibilités et pour leurs conseils judicieux en mot d'encouragement, ils ont toujours été d'une aide précieuse et je les en suis très reconnaissante.

Table des matières

Table des figures

Liste des algorithmes

Introduction Générale

Le travail présenté dans ce rapport s'inscrit dans le cadre du Projet de Fin d'Etude (PFE) pour l'obtention d'un Diplôme National d'Ingénieur en Informatique spécialité Génie Logiciel– Architectures Logicielles à l'Institut Supérieur des Sciences Appliquées et de Technologie de Sousse (ISSATSo), Tunisie.

Dans le cadre de ce PFE, un stage est effectué au sein de l'équipe-projet ICAR (Image & Interaction) du Laboratoire d'Informatique, de Robotique et de Micro-électronique de Montpellier - LIRMM (CNRS/Université Montpellier), France. LIRMM est une unité mixte de recherche, dépendant conjointement de l'Université de Montpellier (UM) et du Centre National de la Recherche Scientifique (CNRS). Il est situé dans le Campus Saint-Priest de l'UM. Ses activités de recherche se positionnent pleinement au cœur des sciences et technologies de l'information, de la communication et des systèmes. Les activités de recherche du LIRMM concernent : la conception et la vérification de systèmes intégrés, mobiles, communicants, la modélisation de systèmes complexes à base d'agents, les études en algorithmique, bio-informatique, interactions homme-machine, robotique, etc. Le département informatique du LIRMM est organisé en quatorze équipes-projets. Il regroupe une centaine de chercheurs permanents et 70 doctorants. Parmi ces équipes, je cite l'équipe qui m'a accueillie : ICAR (Image & Interaction) qui développe *des activités de recherche associant l'interaction et le traitement des données visuelles telles que les images, les vidéos et les objets 3D*. Dans ce cadre nous nous sommes intéressés à la reconstruction du modèle géométrique tridimensionnel (3D) à partir d'un nuage de points.

En effet, de nos jours, la multiplication et l'accessibilité des scanners dans divers domaines comme le médical, la mécanique ou encore l'artistique ont permis la création de nombreux modèles numériques à partir d'un nuage de points relativement dense. L'information souhaitée par la suite est différente suivant le cas et le domaine d'utilisation. Par exemple, dans le domaine artistique l'objectif est de retrouver un modèle géométrique qui sert à reproduire l'image des objets. En mécanique, le nuage de points peut être un support pour inspecter la pièce fabriquée afin de vérifier la conformité avec la pièce nominale dans le but de valider certaines caractéristiques géométriques et dimensionnelles. Généralement, le modèle CAO – 3D représente le support numérique le plus fidèle au modèle réel qui est une information

primordiale le long de cycle de vie du produit (conception, calcul, fabrication, inspection, etc…). Aussi, le nuage de points peut servir à l'ingénierie inverse (Reverse Engineering) ou encore pour préparer un maillage pour des analyses par éléments finis.

Le but de ce projet de fin d'études est de développer un outil de reconstruction d'un modèle géométrique à partir d'un nuage de points. Ce nuage de points peut être le résultat d'un scan d'un objet 3D ou extrait d'un maillage 3D. L'objectif recherché est de concevoir un modèle numérique 3D continu et cohérent (modèle CAO) à partir d'un nuage de points. Pour atteindre ce but, il faut passer par plusieurs étapes.

Les étapes que nous sommes efforcés de suivre sont les suivantes :

- Retrouver les frontières de l'objet (les faces, les arêtes).
- Reconstruire les entités énumérées précédemment.
- Retrouver l'objet 3D.

Pour mener ces différentes étapes, nous sommes amenés à développer des algorithmes et méthodologies à savoir :

- Algorithmes de reconstruction des courbes (droite, cercle, Spline...)
- Algorithmes de reconstruction des primitives géométriques (plan, sphère, cylindre, cube, tore...).
- Méthodes de reconstruction des surfaces gauches à partir d'un nuage de points (Bézier, B-Spline, NURBS…).

L'objectif visé est de retrouver un modèle géométrique fidèle au modèle réel. Pour mener cela :

Le premier chapitre est consacré à une étude bibliographique du sujet proposé dans lequel on va présenter la conception assistée par ordinateur (CAO) et la reconstruction d'un objet 3D en citant quelques travaux élaborés dans ce domaine. Le deuxième chapitre sera dédié à détailler les différents types de surfaces (les surfaces classiques et les surfaces gauches). Le troisième chapitre sera consacré à présenter les algorithmes développés pour mener nos résultats. Dans la première partie de ce chapitre, l'algorithme général de la reconstruction d'un modèle CAO à partir d'un nuage de points 3D est présenté. Ensuite, une nouvelle approche pour reconstruire des primitives géométriques simples (un plan, un cylindre, une sphère, un cône, un tore…) en utilisant l'algorithme Levenberg Marquardt est proposée. Enfin, en se basant sur cet algorithme la méthode de la reconstruction d'une courbe et d'une surface B-Spline en optimisant les points de contrôle est bien détaillée. Le dernier chapitre montre la

réalisation de ce travail en énumérant les outils et les technologies utilisés et en présentant les résultats obtenus.

En conclusion, nous décrivons le bilan de ces travaux et nous indiquons les perspectives de développement des méthodes proposées.

Chapitre 1

Etat de l'art

Introduction

Ce chapitre est consacré à une revue de la littérature qui permettra de situer la présente recherche par rapport aux autres recherches effectuées sur le même sujet. En premier lieu, une revue générale sur la conception assistée par ordinateur et le maillage tridimensionnel est présentée. Ensuite, une revue sur la reconstruction 3D est exposée.

1.1. La Conception Assistée par Ordinateur

La conception assistée par ordinateur est l'ensemble de techniques et de méthodes de conception géométrique permettant de modéliser à l'aide d'un ordinateur des outils qui représentent des formes tridimensionnelles [1].

La conception assistée par ordinateur est un thème d'actualité. En effet, plusieurs travaux de recherches dans différentes spécialités (Ingénierie, médicine...) s'intéressent à la CAO en tant qu'un outil de modélisation virtuelle et de visualisation. En effet, le modèle CAO est le modèle le plus fidèle au modèle physique réel.

L'apparition des techniques de numérisation 3D tel que l'utilisation des scanners dans différents domaines est de plus en plus multipliée. La qualité des scans obtenus et les divers champs d'application laissent propager de nombreuses problématiques et sujets de recherche pour obtenir un modèle CAO cohérent.

Dans ce cadre, on s'intéresse à la reconstruction d'un modèle géométrique tridimensionnel à partir d'un nuage de point issu d'un scanner ou d'un maillage tridimensionnel. La reconstruction d'un objet 3D se fait soit à partir d'un maillage 3D soit à partir du nuage de points directement. Dans les sections suivantes, nous présentons chaque méthode de reconstruction et les travaux développés dans ce thème.

1.2. Maillage tridimensionnel

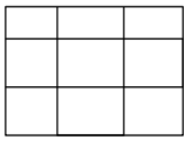
Un maillage 3D est un ensemble de sommets qui sont liés entre eux par des éléments qui sont des polygones comme des rectangles (Figure 1 (a)) ou des triangles (Figure 1 (b)).

(a) Maillage quadrangulaire (b) Maillage triangulaire

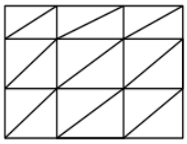
Figure 1 : Maillage quadrangulaire et maillage triangulaire

En fonction des valences décrivant le nombre d'arêtes par sommets, certaines catégories de maillages sont définies. Ainsi, si tous les sommets d'un maillage quadrangulaire, exceptés ceux du contour sont de valence 4, ces maillages sont dits réguliers (voir figure 2 (a)). De même,

si tous les sommets d'un maillage triangulaire, exceptés ceux du contour sont de valence 6, ces maillages sont également dits réguliers (voir figure 2 (b)).



(b) Maillage quadrangulaire régulier



(a) Maillage triangulaire régulier

Figure 2: Maillage quadrangulaire régulier et maillage triangulaire régulier

Ces maillages peuvent être construits en utilisant plusieurs méthodes. Ainsi, plusieurs types de maillage peuvent être identifiés en fonction de la manière dont ils ont été créés. Ces maillages peuvent être issus d'une acquisition numérique, d'une discrétisation, ou d'une déformation de maillage.

- Les maillages issus d'une acquisition numérique : Ce sont des maillages obtenus par différents procédés de numérisation. Dans certains cas, ce procédé de numérisation facilite la création des maillages. Dans d'autres cas, la numérisation ne permet pas faire des suppositions sur la structure du nuage de points. Il faut alors utiliser des méthodes de création de maillages triangulaires 3D à partir d'un nuage de points, comme par exemple celle de Hornung et Kobbelt dans [2].

- Les maillages issus de la discrétisation : Ce **s**ont des maillages qui sont obtenus à partir d'une représentation continue d'un objet. Une comparaison de plusieurs méthodes de discrétisation est proposée par Shawn et al. dans [3]. Nous pouvons distinguer principalement deux types d'objets continus : les objets mécaniques construits par l'assemblage de primitives géométriques simples et les objets de forme quelconque créés grâce à des surfaces paramétriques (surface de Bézier, surface B-Splines, NURBS...). La méthode classique pour discrétiser une surface paramétrique consiste à parcourir l'espace paramétrique $(\boldsymbol{u},\boldsymbol{v})$ en prenant successivement pour une valeur $\boldsymbol{u}$ des valeurs $\boldsymbol{v}$ (voir par exemple la méthode de Shikhare et al. [4]). Chaque "maille" du quadrillage ainsi obtenue est coupée en deux triangles.

- Les maillages déformés : On applique parfois des simulations et des déformations sur les maillages pour reproduire de manière numérique des phénomènes physiques, par exemple pour étudier le comportement d'un objet s'il rencontre un obstacle. Ces simulations sont généralement appliquées sur des maillages 3D à forte densité de sommets, ayant des caractéristiques géométriques spécifiques, et créent donc des maillages déformés ayant beaucoup de sommets et très peu bruités.

1.3 La reconstruction d'un objet à partir d'un maillage

Pour reconstruire le modèle CAO à partir de ces différents types de maillages, beaucoup de travaux ont été élaborés. Citons l'algorithme de Volpin [5] pour la reconstruction d'une surface NURBS à partir d'un maillage en passant par une simplification de ce dernier. La méthode de Volpin commence par la simplification du modèle maillé initial à travers la construction des régions restreintes en courbures conformément au modèle. Par la suite, il construit un maillage quadrilatéral du modèle et enfin il construit des surfaces lisses à ce maillage en utilisant une méthode appelée méthode d'énergie (Figure 3).

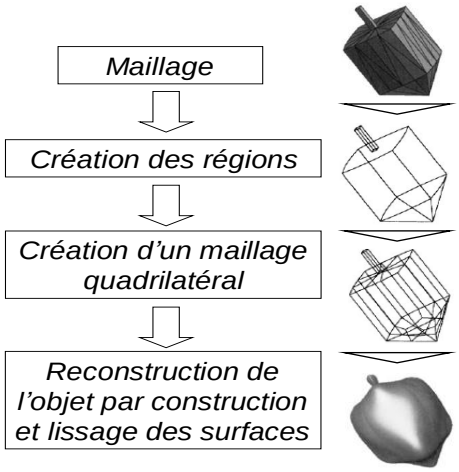


Figure 3: Simplification du maillage et construction de l'objet [5]

Pendant la première phase, simplification du modèle, Volpin se base sur des critères pour la création des régions restreintes en courbure qui sont l'angle minimal entre deux facettes de maillage pour qu'elles soient dans la même région et l'angle pour la création des arêtes entre les régions. Après le maillage quadratique, Volpin assure la continuité entre les surfaces reconstruites à partir des régions. Les régions restreintes sont de types planes, et la création de régions se fait selon trois critères :

- La distance entre une facette du maillage (maille) et un plan correspondant à une région.
- L'angle entre deux facettes adjacentes du maillage.
- L'angle entre un plan correspondant à une région construite et une facette.

Méthodes de raffinement du maillage 3D

Afin de bien évaluer les surfaces 3D porteuses de triangulation (maillage), des méthodes de raffinement sont appliquées sur le maillage. Ça permettra d'ajouter et d'améliorer l'information pour une meilleure reconstruction.

Pour raffiner ces maillages, Ren a développé [6] un algorithme de remaillage pour raffiner itérativement un maillage donné afin de se rapprocher de la forme réelle de l'objet maillé. Il s'agit d'insérer un nouveau nœud entre deux nœuds consécutifs de l'ancien maillage tout en respectant la nature et la forme de la surface. Cette subdivision consiste à diviser chaque triangle par quatre (Figure4).

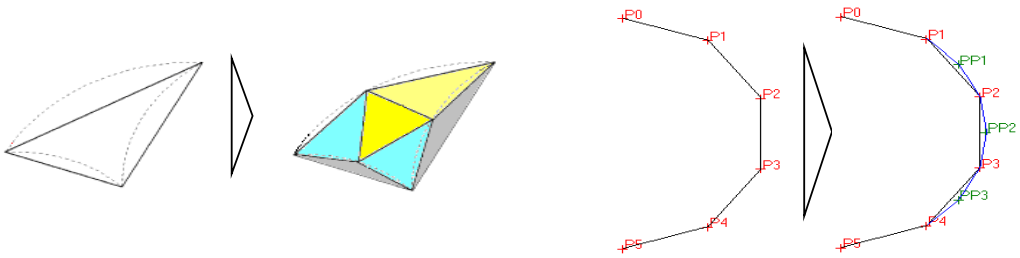


Figure 4: Raffinement successif suivant [6]

Dans le même contexte que Ren, des travaux se sont intéressés à l'approximation d'une surface par subdivision successive du maillage correspondant [7]. Un nouveau nœud est inséré entre deux nœuds consécutifs, afin que chaque arête soit divisée en deux et chaque triangle soit divisé en quatre triangles plus fins tout en respectant la forme de l'objet maillé. D'autres travaux sont basés sur des différents schémas de subdivisions connus (Schéma de Catmull-Clark, Schéma de Loop, Schéma de Butterfly…), pour raffiner un maillage successivement en vue d'obtenir une surface lisse [8] [9].

Pour l'évaluation d'une surface à partir d'un maillage, des travaux sont basés sur les triangles de Bézier. Dans ces travaux une surface triangulaire est calculée (triangle de Bézier) à partir de chaque triangle du maillage, de telle façon qu'elle soit continue avec les surfaces voisines (triangles voisins du maillage). Walton [10] a proposé un algorithme d'évaluation d'une surface de Bézier à partir d'un triangle dans un maillage surfacique. Cet algorithme a été ensuite amélioré par Owen [11]. L'algorithme consiste à calculer pour chacun des nœuds de la triangulation un vecteur. Ce vecteur sera ensuite utilisé comme la normale à la surface reconstruite. Par la suite, à partir des coordonnées des trois sommets du triangle et les trois vecteurs normaux en ces sommets, les autres points de contrôle sont calculés (Figure 5).

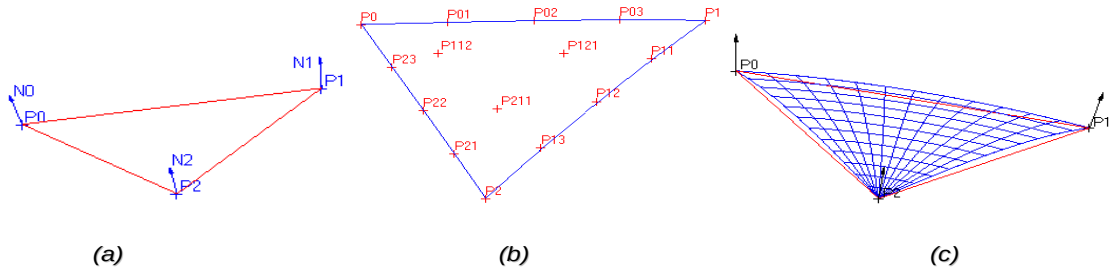


Figure 5: Evaluation d'une surface de Bézier

(a) Les normales associées à un triangle, (b) Les points de contrôle de la surface (Surface de Bézier) pour u = 0.5, v = 0.5 et w = 0.5, (c) Surface de Bézier porteuse du triangle.
Enfin, chaque point de cordonnées locales $(u, v\ et\ w)$ sur le triangle est évalué sur la surface porteuse de triangle en utilisant les points de contrôle calculés précédemment.

Dans [12], Bénière et al., ont proposé une nouvelle méthode pour reconstruire un modèle B-Rep[1] en se basant sur l'extraction des primitives en premier lieu, puis calculer leurs contours. En effet, un objet peut être construit en combinant des primitives géométriques comme des plans, des sphères, des cylindres, des cônes ou des surfaces paramétriques (par exemple Bézier, B-Splines, NURBS) et leurs contours (en utilisant les intersections entre les primitives). Dans la première étape, extraction des primitives géométriques, l'idée est d'abord de détecter le type de primitive géométrique qui correspond localement au maillage 3D et de calculer ensuite les paramètres qui donnent le meilleur ajustement. Pour la deuxième étape, ceci est un problème complexe. Il faut définir la relation entre toutes les primitives géométriques extraites, qui seront utilisées pour calculer les courbes d'intersection entre deux primitives géométriques. Ensuite, l'ensemble de ces courbes est combiné pour construire un contour continu d'une manière cohérente. Enfin, dans la troisième étape, les informations extraites ou reconstituées au cours des deux étapes précédentes doivent être combinées pour construire un modèle B-Rep cohérent.

Dans le même contexte, Auréhlien BEY et al. [13], ont traité le problème de la reconstruction de modèles CAO à partir de nuages de points acquis en environnement industriel, en s'appuyant sur des modèles 3D préexistants ainsi que sur des connaissances métier. Ils se sont intéressés aux modèles CAO décrits comme des assemblages de primitives géométriques simples telles que les plans, cônes, cylindres et tores. Chacune de ces primitives est un objet géométrique entièrement caractérisé par une position et une orientation, ainsi qu'un certain nombre de paramètres géométriques inhérents au type de la primitive. Ils ont traité plus particulièrement les parties cylindriques en proposant de formuler le problème de la

[1] B-Rep : une technique de modélisation 3D géométrique des solides par les surfaces

reconstruction comme la recherche de la configuration la plus probable vis-à-vis de multiples contraintes.

Louhichi et al. [14] ont proposé une méthode de reconstruction d'un modèle CAO à partir d'un maillage déformé. L'objectif est de retrouver un modèle CAO déformé. Donc comme information de départ, l'algorithme de reconstruction possède une CAO initiale, un maillage initial et un maillage déformé. Pour reconstruire les différentes faces déformées, une méthode basée sur l'estimation de déplacement pondérée est proposée. Cette méthode a pour objectif la résolution du problème des points non organisés d'une surface déformée. Le procédé d'estimation de déplacement pondérée met à jour la position des points de contrôle de la surface en treillis de surface (P°) en estimant leur déplacement (δ) en utilisant le déplacement induit par la moyenne pondérée de nœuds voisins du modèle de maillage lors de l'analyse par éléments finis. En notant les coordonnées des nœuds du modèle CAO maillé par M° et leurs coordonnées après l'analyse FE est l'ensemble $M^{\bullet}$, l'estimation de déplacement pondérée de chaque point de contrôle par (P_k°) associé à la surface est exprimé comme suit :

$$\delta_k = \frac{\sum_i^{n_k} \| P_k^{\circ} - M_{ki}^{\circ} \| \{ M_{ki}^{\bullet} - M_{ki}^{\circ} \}}{\sum_i^{n_k} \| P_k^{\circ} - M_{ki}^{\circ} \|} \; \forall k \mid P_k^{\circ} \in P^{\circ}$$

Aussi, $M_k^{\bullet}$ est l'ensemble de n_k nœuds voisins englobés dans un rayon R_k du point de contrôle P_k° vérifiant :

$$\left\| P_k^{\circ} - M_{ki}^{\circ} \right\| \leq R_k \; \forall i \mid M_{ki}^{\circ} \in M^{\circ}$$

Les nœuds $M_k^{\bullet}$ et les points de contrôle P_k° sont associés à la même surface. Après l'estimation du déplacement δ, les nouveaux points de contrôle ($P^{\bullet}$) de la surface (B-Spline, NURBS…) qui représentent le modèle CAO déformé sont calculés en utilisant cette relation :

$$P_k^{\bullet} = P_k^{\circ} + \delta_k$$

1.4 La reconstruction d'un objet à partir d'un nuage de points

Les surfaces de points ont beaucoup attiré l'attention récemment. En effet, les progrès des matériels depuis 10 ans permettent aujourd'hui de traiter des objets composés de plusieurs millions de triangles en temps réel. Mais dans le même temps, la définition des écrans n'a pas beaucoup évolué, et si l'on veut aller plus loin, et développer des solutions pour des objets de plusieurs centaines de millions, et même de milliards de points, on finit par générer des triangles plus petits qu'un pixel à l'écran. Dès lors, le paradigme du point s'est imposé comme une bonne

solution, réduisant l'espace mémoire des modèles. Le fait d'avoir des objets sous formes de nuages de points non organisés permet également de faire du streaming de points pour les applications en ligne, fournissant une visualisation progressive. Les points d'un tel nuage, appelés communément surfel (surface élément) dès lors qu'ils sont équipés d'un vecteur normal, peuvent également disposer de divers autres attributs : rayon définissant la portion de surface associée au surfel, couleur, coordonnées de textures, ou tout autre paramètre utile dans un domaine particulier, tels que la physique ou la médecine par exemple. En termes de rendu, de nombreux travaux ont été menés depuis 1998 sur la visualisation directe de telles surfaces. Ces techniques appelées communément « splatting » permettent d'associer une tâche à l'écran pour chaque pixel, ce qui comble les troue incombant à de telles surfaces. La qualité du rendu de telles surfaces va dépendre de l'algorithme choisi.

Malheureusement ces techniques de visualisation de points (rendu par points) se prêtent mal aux illuminations complexes, ne sont pas supportées par les matériels graphiques, gèrent mal les nuages de points non uniformes, ainsi que les détails à la surface tels que les arêtes vives. Le rendu par points n'est donc pas toujours une bonne solution.

Par contre, la modélisation par points présente de nombreux avantages : sélection aisée, travail en multi résolution, ou bien encore texturation des modèles. Ce dernier point, souvent délicat, a fait l'objet de plusieurs projets. La place des surfaces de points aujourd'hui est claire : un excellent paradigme de modélisation, qui unifie les autres, grâce notamment aux techniques de reconstruction de surfaces à partir de nuages de points, permettant de passer d'une représentation par points (utilisée pour modéliser l'objet) à une autre (par exemple pour utiliser l'objet en visualisation ou en simulation. Pour passer d'une représentation par points à une surface on est face à deux possibilités soit on passe par la reconstruction d'une triangulation qui peut être faite par divers méthodes comme celle du Ball-Pivoting et la reconstruction de Poisson. Ou bien on procède directement à la reconstruction de l'objet à partir de cet ensemble de points en utilisant plusieurs méthodes de reconstruction comme les surfaces NURBS et B-Splines qui sont expliquées dans le chapitre suivant.

La méthode du Ball-Pivoting (BPA) [15] consiste à calculer un maillage triangulaire interpolant un nuage de points donné. Généralement, les points sont des échantillons de la surface acquis avec de multiples analyses d'intervalle d'un objet. Le principe du BPA est très simple : Trois points forment un triangle à condition qu'une balle d'un rayon spécifié par l'utilisateur les touche sans contenir aucun autre point. A partir d'un triangle initial, la balle

pivote autour d'un bord tout en gardant le contact avec les extrémités du bord jusqu'à ce qu'elle touche un autre point, formant un autre triangle. Le processus se poursuit jusqu'à ce que tous les bords accessibles soient essayés, puis commence à partir d'un autre triangle, jusqu'à ce que tous les points soient pris en compte. Pour résumer, cette méthode sert à trouver un triangle du maillage en interpolant l'ensemble de points non organisés. Cet algorithme a été appliqué à des ensembles de données de millions de points représentant des scans réels des objets 3D complexes. La quantité relativement faible de mémoire requise de la méthode de BPA, son efficacité de temps, et la qualité des résultats obtenus se comparent favorablement avec les techniques existantes.

Aussi, la méthode de Poisson [16] est une des plus robustes méthodes de reconstruction de surface 3D qui utilise l'équation de Poisson pour interpoler un ensemble de points orientés. En calculant une fonction indicatrice χ définie comme 1 aux points situés à l'intérieur du modèle et 0 aux points extérieurs, on obtient la surface reconstruite par l'extraction d'une isosurface[2] appropriée. L'idée fondamentale de cette méthode, est qu'il existe une relation intégrale entre les points orientés échantillonnés à partir de la surface d'un modèle et de la fonction indicatrice du modèle. Plus précisément, le gradient de la fonction indicatrice est un champ de vecteur qui est égal à zéro presque partout (étant donné que la fonction indicatrice est constante presque partout) sauf en points proches de la surface, où elle est égale à la normale de la surface intérieure. Ainsi, les échantillons de points orientés peuvent être considérés comme des échantillons du gradient de la fonction indicatrice du modèle.

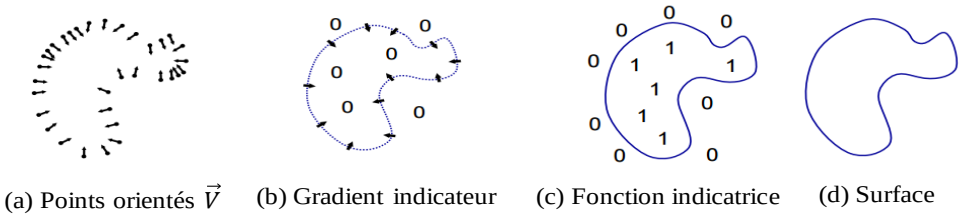
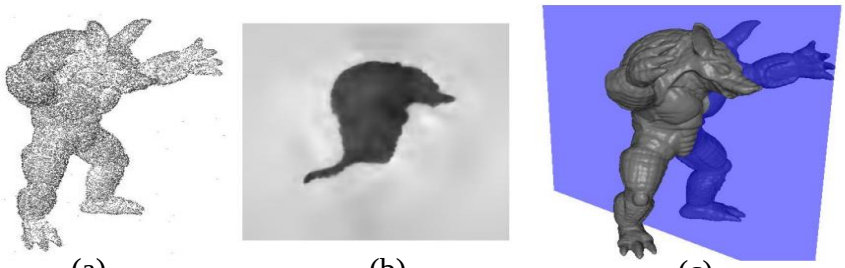


Figure 6: Illustration intuitive de la reconstruction de Poisson en 2D [16]

Le problème du calcul de la fonction indicatrice consiste à inverser l'opérateur gradient, à savoir trouver la fonction scalaire χ dont le gradient se rapproche d'un champ vectoriel $\vec{V}$

[2] Isosurface : En disposant d'une fonction qui donne à chaque point de l'espace une valeur ; puis en choisissant une de ces valeurs. Tous les points dont la valeur est *inférieure* au seuil que nous avons choisi sont *à l'intérieur* de la surface, et vice-versa.

défini par les points. Donc, on doit minimiser : $min_{\chi}\|\nabla\chi - \vec{V}\|$. En appliquant l'opérateur de divergence, ce problème se transforme à un problème de Poisson, donc il faut calculer la fonction scalaire χ dont le Laplacien (divergence du gradient) est égal à la divergence du champ de vecteurs $\vec{V}$: $\nabla\chi = \nabla.\nabla\chi = \nabla.\vec{V}$



(a) (b) (c)

Figure 7: Reconstruction de Poisson [16]

(a) Nuage de points de « Armadillo Man » (b) Visualisation de la fonction indicatrice (c) Reconstruction de Poisson

Dans le cadre de la reconstruction des surfaces B-Splines, la plupart des méthodes ont supposé que la surface est de type topologique simple. Par exemple, Dietz [17], Hoschek et Schneider [18], Rogers et Fog [19], et Sarkar et Menq [20] supposent que la surface est une région quadrilatérale déformée avec des limites rognées. Schmitt et al. [21] supposent que la surface est un cylindre déformé et utilisent un raffinement adaptatif pour obtenir la surface B-Spline.

Andersson et al. [22] , Fang et Gossard [23], et Milroy et al. [24] ont travaillé sur l'ajustement des surfaces B-Splines en obligeant l'utilisateur à définir manuellement le contour de patch (carreaux) soit par étiquetage « points limites » ou en dessinant des courbes limites sur une surface d'approximation. En outre, le paramétrage initial des points de données est essentiel dans le processus d'ajustement, comme a été démontré par Ma et Kruth [25], et ces systèmes nécessitent une intervention supplémentaire de l'utilisateur pour obtenir de bonnes distributions de paramètres initiaux.

Par contre, Eck et Hoppe [26] ont proposé une procédure de reconstruction automatique d'une surface B-Spline comme un produit tensoriel d'un ensemble de points 3D numérisés.

Dans [27] Hoppe et al. ont développé un algorithme qui prend un ensemble M de points non organisés $\{x_1, \dots, x_n\}$ pour avoir en sortie une surface valide qui approxime M. Ni la topologie, ni la présence de frontières et ni la géométrie de M sont supposés être connus à l'avance, tous sont inférées automatiquement à partir des données. Cet algorithme de

reconstruction de surface consiste en deux étapes. Dans la première partie, une fonction $f: D \rightarrow \mathbb{R}$ est définie tel que $D \subset \mathbb{R}^3$ est une région près de l'ensemble de données de telle sorte que f estime la distance géométrique signée à la surface inconnue M. L'ensemble zéro $Z(f)$ est l'estimation pour M. Dans la deuxième étape, un algorithme pour trouver le contour est utilisé pour approcher $Z(f)$ par une surface limitée.

Conclusion

Dans ce chapitre, on a commencé par une introduction à la CAO et aux techniques de maillage. Par la suite des différentes méthodes, de reconstruction d'un objet 3D à partir d'un nuage de points issu d'un maillage ou du scan, sont présentées.

Chapitre 2

Les surfaces CAO

Introduction

Dans ce deuxième chapitre, nous allons expliquer dans une première partie les caractéristiques des surfaces classiques (les primitives) en citant les principaux types. Ensuite, nous allons détailler les différentes caractéristiques des surfaces gauches à savoir : les surfaces de Bézier, B-Splines, NURBS...

2.1 Les surfaces classiques

Les surfaces classiques ou les primitives sont des formes géométriques de base, qui sont décrites par des formules mathématiques et qu'on peut modéliser plus parfaitement que d'autres objets complexes. Parmi ces primitives nous citons : le plan, le cylindre, le cône, la sphère, le tore…

Il existe principalement 5 types de surfaces classiques. Leurs équations sont données dans leurs repères locaux classiques à l'aide des paramètres ***u*** et ***v*** :

2.1.1 Plan

$S(u,v) = u*\vec{e}_1 + v*\vec{e}_2 + origine \quad avec \ \vec{e}_1 et \vec{e}_2 \ sont \ 2 \ vecteurs \ unités \ de \ plan$

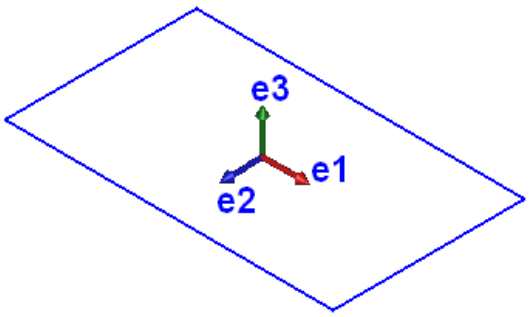


Figure 8: Surface plane

2.1.2 Cylindre

$S(u,v) = R*[\cos(u)*\vec{e}_1 + sin(u)*\vec{e}_2] + v*\vec{e}_3 + origine$ *avec R est le rayon du cylindre.*

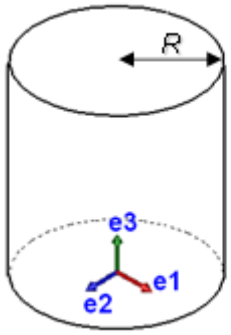


Figure 9: Surface cylindrique

2.1.3 Cône

Un cône d'angle α est défini comme suit :

$$S(u,v) = v * \tan(\alpha) * [\cos(u) * \vec{e}_1 + sin(u) * \vec{e}_2] + v * \vec{e}_3 + origine$$

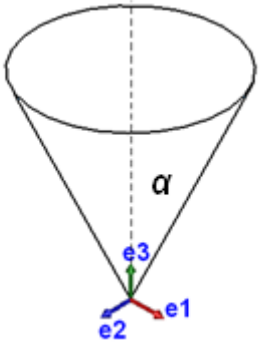


Figure 10: Surface conique

2.1.4 Tore

Un tore de grand rayon R1 et de petit rayon R2 est défini comme suit :

$$S(u,v) = (R_1 + R_2 * \cos(v)) * [\cos(u) * \vec{e}_1 + sin(u) * \vec{e}_2] + R_2 * \sin(v) * \vec{e}_3 + origine$$

Si $\boldsymbol{R_1} = 0$, on se retrouve avec l'équation d'une sphère.

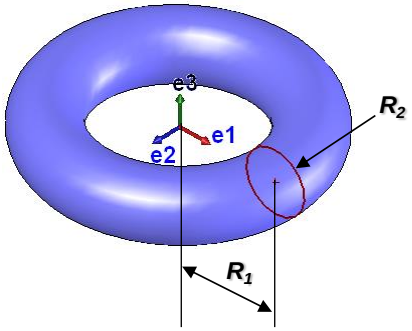


Figure 11: Surface torique

2.2 Les surfaces gauches

La création d'un modèle 3D nécessite parfois le choix d'entités géométriques (courbes et/ou surfaces) plus ou moins complexes. Pour décrire les formes libres (complexes), il s'agit d'adopter un modèle paramétrique de courbe ou de surface, défini à l'aide de fonctions polynomiales ou rationnelles par morceaux dans un système de coordonnées cartésiennes.
Dans la suite les principales surfaces complexes sont présentées :

2.2.1 Les surfaces Bilinéaires

On appelle surface Bilinéaire, une surface décrite par interpolation linéaire de 4 points (Fig. 12(a)). Ces quatre points définissent la matrice de contrôle de la surface [P], tel que :

$$[P] = \begin{bmatrix} P(0,0) \\ P(1,0) \\ P(0,1) \\ P(1,1) \end{bmatrix}$$

L'interpolation linéaire de ces quatre points s'écrit sous la forme suivante :

$$S(u,v) = (1-u)(1-v)P(0,0) + u(1-v)P(1,0) + (1-u)vP(0,1) + uvP(1,1)$$

Sous forme matricielle, elle s'écrit sous la forme suivante :

$$P(u,v) = [(1-u)(1-v) \quad u(1-v) \quad (1-u)v \quad uv] \begin{bmatrix} P(0,0) \\ P(1,0) \\ P(0,1) \\ P(1,1) \end{bmatrix}$$

$$Avec \quad 0 \le u \le 1 et\, 0 \le v \le 1$$

La surface bilinéaire est une surface qui est simple à gérer. L'inconvénient de cette surface c'est l'absence de la flexibilité : par exemple pour décrire une surface de révolution, il faut plusieurs carreaux bilinéaires (Fig. 12(b)).

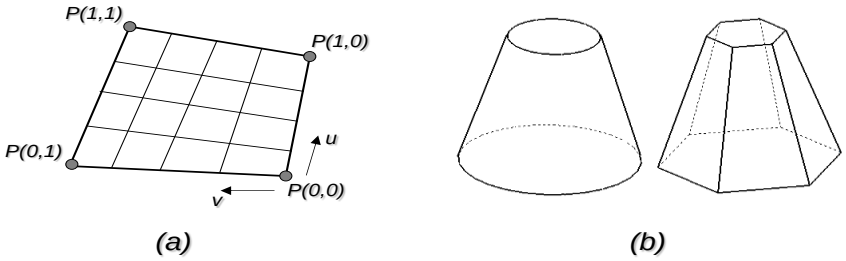


Figure 12: Modélisation d'un cône à l'aide de surfaces bilinéaires

(a) Surface Bilinéaire, (b) Cône reconstruit à l'aide de surfaces bilinéaires

2.2.2 Les surfaces de Coons

La surface de Coons est définie par un réseau de courbes croisées. Ceci a permis de définir des carreaux, qui sont définis complètement par interpolation surfacique à partir des frontières.

La Figure (Fig. 13) présente l'exemple de la construction d'une surface de COONS ***S(u, v)*** à partir de quatre courbes : ***P(u, 0)***, ***P(u, 1)***, ***P(0, v)*** et ***P(1, v)***.

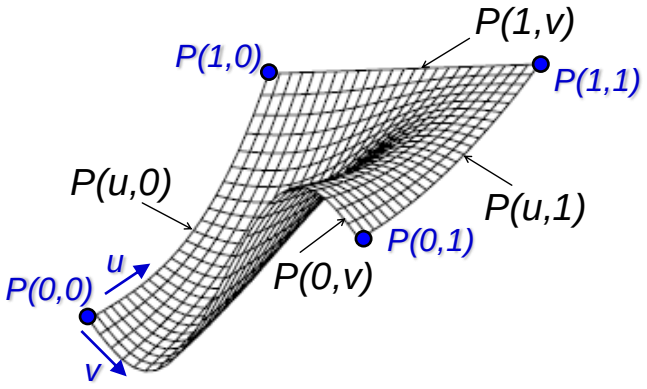


Figure 13: Surface de COONS

$$S(u,v) = (1-u \quad u \quad 1)\begin{pmatrix} -P(0,0) & -P(0,1) & P(0,v) \\ -P(1,0) & -P(1,1) & P(1,v) \\ P(u,0) & P(u,1) & P(0,0) \end{pmatrix}\begin{pmatrix} 1-v \\ v \\ 1 \end{pmatrix}$$

$0 \le u \le 1$

$0 \le v \le 1$

2.2.3 Courbes et surfaces polynomiales

Les polynômes de haut degré peuvent décrire des courbes complexes, mais ils demandent un grand nombre de paramètres dont la signification physique est parfois difficile à maîtriser. D'autre part, les instabilités numériques augmentent avec le degré des polynômes. De ce fait, les courbes et les surfaces cubiques (degré 3 maximum dans chaque direction), ont été reconnues comme étant un bon compromis de modélisation dans la plupart des applications. D'une manière générale, les courbes de degré maximum 2 sont appelées courbes quadratiques et les courbes de degré maximum 3 sont appelées courbes cubiques.

En 1963, Fergusson [28] a été le premier à introduire l'utilisation des courbes cubiques paramétriques dans le domaine de l'aéronautique (Boeing Co). Les segments de ces courbes paramétriques sont définis par des équations de la forme :

$$\vec{r} = \vec{r}(u) = \sum_{i=0}^{n} (u^i \vec{a}_i)$$

Où n est le degré de la courbe, le vecteur r décrit l'ensemble des points de la courbe lorsque u varie de 0 à 1. Les vecteurs $\vec{a}_i$ sont les paramètres de la courbe.

Les surfaces paramétriques sont exprimées comme suit :

$$\vec{r} = \vec{r}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m}(u^i v^j \vec{a}_{ij})$$

2.2.4 Courbes et surfaces polynomiales de Bernstein-Bézier

En 1970, Bézier a recombiné les termes de la paramétrisation cubique de Fergusson afin que la signification physique des coefficients vecteurs soit plus apparente. Les courbes de Bernstein-Bézier de degré ***n*** ont la forme suivante :

$$\vec{r} = \vec{r}(u) = \sum_{i=0}^{n} B_i^{\,n}(u)\vec{r}_i$$

Où n est le degré de la courbe et les $\vec{r}_i$ sont les vecteurs de positionnement des $(n+1)$ sommets ***(P0, P1, ..., Pn)*** d'un polygone appelé polygone caractéristique généralisé.

Les polynômes $B_i^{\,n}(u) = \dfrac{n!}{i!(n-i)!}u^i(1-u)^{n-i}$ sont appelés polynômes de Bernstein.

Exemple :

Si le nombre de points de contrôle égal à 4 donc le degré de l'approximation est égal à 3, la courbe est dite courbe de Bézier cubique.

$$P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3$$

Soit sous forme matricielle :

$$P(t) = \begin{pmatrix} t^3 & t^2 & t & 1 \end{pmatrix}\begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}\begin{pmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{pmatrix}$$

L'équation d'une surface de Bézier est de la forme suivante :

$$\vec{r} = \vec{r}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_i^{\,n}(u)B_j^{\,m}(v)\vec{r}_{ij}$$

Avantages des courbes et surfaces de Bézier :

- Chaque point de la courbe est une combinaison convexe de tous les points de contrôle.
- Pas de problème de non continuité aux raccords.
- Grande maniabilité des courbes.
- Elles ne sont pas modifiées par un changement d'axe.
- Les polynômes de Bernstein s'expriment de façon explicite.

Inconvénients de l'utilisation des courbes et surfaces de Bézier :

- Le déplacement d'un point de contrôle implique une modification complète de la courbe.
- Il est coûteux de rajouter un point.
- Le degré du polynôme augmente avec le nombre de points de contrôle. Ainsi, lorsque le nombre de points de contrôle est important, le calcul des fonctions de Bézier devient délicat.

2.2.5 Courbes et Surfaces B-Splines

➢ *Courbe B-Spline*

Une courbe B-Spline de degré p est définie comme suit :

$$C(u) = \sum_{i=0}^{n} N_{i,p}(u)\boldsymbol{P_i}$$

Tel que $\boldsymbol{P_i}$ sont les points de contrôle et $N_{i,p}(u)$ sont les fonctions de base de la courbe B-Spline de degré p définies dans le vecteur de nœuds.

Il y a trois étapes pour calculer un point sur la courbe B-Spline en fixant u :

- Trouver l'indice i tel que $u \in [u_i, u_{i+1}[$

- Calculer les fonctions de base $N_{i-p,p}(u), \dots, N_{i,p}(u)$

- Multiplier les valeurs des fonctions de base par les points de contrôle correspondants.

Pour calculer ce point, nous allons utiliser l'algorithme suivant :

Algorithme 1: Fonction Calcul-Point-Courbe-B-Spline

Données : n, p, u, U, P

Indice ← trouverIndice (n, p, u, U)

N ← Calcul-Base-B-Spline (Indice, p, u, U)

C ← 0

Pour i de 0 à p

C ← C + N[i]×P[Indice-p+i]

Fin pour

Fin

- ***Surface B-Spline***

Une surface B-Spline est obtenue à partir d'un réseau bidirectionnel de points de contrôle, deux vecteurs nodaux, et les produits des fonctions B-Splines unidimensionnelles [29].

$$S(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)P_{i,j}$$

Les vecteurs nodaux U et V ont respectivement $r+1$ et $s+1$ nœuds. Avec $r = n+p+1$ et $s = m+q+1$. Cette surface est de degré p suivant u et q suivant v.

Il y a 5 étapes pour définir un point dans une surface B-Spline en fixant les paramètres u et v :

- Trouver l'indice i, tel que $u \in [u_i, u_{i+1})$
- Calculer les fonctions de base $N_{i-p,p}(u), \dots, N_{i,p}(u)$
- Trouver l'indice j, tel que $v \in [v_j, v_{j+1})$
- Calculer la fonction de base $N_{j-q,q}(v), \dots, N_{j,q}(v)$
- Multiplier les valeurs des fonctions de base par les points de contrôle.

Donc, $S(u,v) = [N_{k,p}(u)]^T [P_{k,l}][N_{l,q}(v)]$ avec $i-p \le k \le i$ et $j-q \le l \le j$

$[N_{k,p}(u)]^T$: est un vecteur ligne de taille $p+1$.

$[N_{l,q}(v)]$: est un vecteur colonne de taille $q+1$.

$[P_{k,l}]$: est la matrice des points de contrôle de taille $(p+1) \times (q+1)$.

Les valeurs des nœuds pour une surface B-Spline doivent être entre 0 et 1. Donc, les vecteurs de nœuds vont prendre la forme suivante :

$$\underbrace{\mathbf{0}, \dots, \mathbf{0}}_{\text{p+1}} < \boldsymbol{u_{p+1}} < \cdots < \boldsymbol{u_n} < \underbrace{\mathbf{1}, \dots, \mathbf{1}}_{\text{p+1}}$$

L'écart entre deux nœuds consécutifs du sous-ensemble $u_{p+1}, \dots, u_n$ pour une surface B-Spline uniforme est identique et le vecteur est uniforme tel que $u_i = \frac{i-p-1}{(n+p+1)-2p-1}$ avec $i \in [p+1, n]$.

Exemple :

On prend le degré $p = 2$ et $q = 2$, la matrice des points de contrôle $P_{i,j}$ de taille 5×6

$$S(u,v) = \sum_{i=0}^{4}\sum_{j=0}^{5} N_{i,2}(u)N_{j,2}(v)P_{i,j}$$

Avec $U = (0,0,0, 2/5, 3/5, 1,1,1)$ et $V = (0,0,0, 1/5, 1/2, 4/5, 1,1,1)$

Pour calculer $S(1/5\,,3/5)$, on doit trouver l'indice de $u = 1/5$ dans U et l'indice de v dans V. En faisant appel à la fonction correspondante on trouve le premier indice est 2 puisque $1/5 \in [u_2, u_3)$ et le deuxième indice est égal à 4 puisque $3/5 \in [u_4, u_5)$.

Alors, $S\left(\frac{1}{5},\frac{3}{5}\right) = \left[N_{0,2}\left(\frac{1}{5}\right) \quad N_{1,2}\left(\frac{1}{5}\right) \quad N_{2,2}\left(\frac{1}{5}\right)\right] \times \begin{bmatrix} P_{0,2} & P_{0,3} & P_{0,4} \\ P_{1,2} & P_{1,3} & P_{1,4} \\ P_{2,2} & P_{2,3} & P_{2,4} \end{bmatrix} \begin{bmatrix} N_{2,2}\left(\frac{3}{5}\right) \\ N_{3,2}\left(\frac{3}{5}\right) \\ N_{4,2}\left(\frac{3}{5}\right) \end{bmatrix}$

Nous allons utiliser l'algorithme suivant pour calculer un point d'une surface B-Spline :

Algorithme 2: Fonction Calcul-Point-Surface-B-Spline

Données : n, p, U, m, q, V, P, u, v

IndiceU ← trouverIndice(n, p, u, U)

Nu ← Calcul-Base-B-Spline(indiceU, p, u, U)

IndiceV ← trouverIndice(m, q, v, V)

Nv ← Calcul-Base-B-Spline(indiceV, q, v, V)

Uind ← IndiceU - p

S ← 0

Pour l de 0 à q

    Aux ← 0

    Vind ← IndiceV – q + l

    Pour k de 0 à p

        Aux ← Aux + Nu[k]×P[Uind+k , Vind]

    Fin pour

    S ← S + Nv[l]×Aux

Fin pour

Fin

- *Fonctions de base des B-Splines*

On définit les fonctions de base de la façon suivante :

- Par un ensemble $\boldsymbol{U}$ de $\boldsymbol{r+1}$ réels $\boldsymbol{u_i}$ non décroissants : $\boldsymbol{u_0 \leq u_1 \leq \ldots \leq u_m}$.

$\boldsymbol{U}$: le vecteur nodal, et $\boldsymbol{u_i}$ sont les nœuds.

- Par un degré $\boldsymbol{p}$.

La i-ème fonction de base de B-Splines de degré p (ou d'ordre $p+1$) est définie par la formule doublement récursive de Cox-DeBoor :

$$N_i^0(u) = \begin{cases} 1 \; si \; u \in [u_i, u_{i+1}[\\ 0 \; sinon \end{cases}$$

$$N_i^p(u) = \frac{u - u_i}{u_{i+p} - u_i} N_i^{p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1}^{p-1}(u)$$

En utilisant cette relation de récurrence, le calcul de $p^{éme}$degré de fonctions de base génère une table triangulaire tronquée :

$$\begin{array}{cccc} N_{0,0} & & & \\ & N_{0,1} & & \\ N_{1,0} & & N_{0,2} & \\ & N_{1,1} & & N_{0,3} \\ N_{2,0} & & N_{1,2} & \\ & N_{2,1} & & N_{1,3} \\ N_{3,0} & & N_{2,2} & \\ & N_{3,1} & & \vdots \\ N_{4,0} & & \vdots & \\ & \vdots & & \\ \vdots & & & \end{array}$$

Figure 14: Table des B-Splines de base

Pour simplifier, on écrit $N_{i,p}$ au lieu de $N_{i,p}(u)$. Et puisque nous utilisons des intervalles de la forme $u \in [u_i, u_{i+1}[$, le problème dans l'évaluation des fonctions de base sont les cas particuliers lorsque $u = u_m$. On traite ce cas en retournant l'indice $n = m - p - 1$. Dans ce cas, $u \in (u_{m-p-1}, u_{m-p}]$.

Pour trouver l'indice i tel que $u \in [u_i, u_{i+1}[$, on fait appel à une fonction trouverIndice qui retourne un entier. L'algorithme de cette fonction est le suivant :

Algorithme 3: Fonction trouverIndice

Données : n, p, u, U

Si u = U[n + 1]

retourner (n)

Sinon

l ← p et h ← n+1

milieu ← (l+h) / 2

Tant que (u < U [milieu])

Si (u < U [milieu])

H ← milieu

Sinon

L ← milieu

Finsi

milieu ← (l+h) / 2

Fin Tant que

retourner (milieu)

Fin

Une foi nous avons l'indice de u, nous calculons les résultats des fonctions non nuls qui forment ce schéma triangulaire inverti :

$$\begin{array}{ccccc} & & & & N_{i-1,p} \\ & & N_{i-1,1} & & \\ N_{i,0} & & & \cdots & \vdots \\ & & N_{i,1} & & \\ & & & & N_{i,p} \end{array}$$

Figure 15 : Table triangulaire invertie des B-Splines de base

Exemple :

Soit $p = 2$, $U = \{ 0,0,0,1,2,3,4,4,5,5,5 \}$ et $u = 5/2$. Alors $i = 4$ puisque $u \in [u_4, u_5[$.
Dans ce cas, on calcule la table triangulaire suivante :

$$\begin{array}{ccc} & & N_{2,2}(5/2) \\ & N_{3,1}(5/2) & \\ N_{4,0}(5/2) & & N_{3,2}(5/2) \\ & N_{4,1}(5/2) & \\ & & N_{4,2}(5/2) \end{array}$$

Maintenant pour calculer les fonctions de base des B-Splines, nous allons utiliser l'algorithme suivant :

Algorithme 4: Fonction Calcul-Base-B-Spline

Données : i, p, u, U

N[0] ← 1

```
Pour j de 1 à p
    Gauche[j] ← u – U[i+1-j]
    Droit[j] ← U[i+j] – u
    Somme ← 0
    Pour r de 0 à j-1
        Aux ← N[r] / (Droit[r+1] +Gauche[j-r])
        N[r] ← Somme + Droit[r+1] * Aux
        Somme ← Gauche[j-r] * Aux
    Fin pour
Fin pour
Retourner (N)
Fin
```

- *Choix du vecteur nodal*

Le vecteur nodal est une séquence de valeur de paramètres qui détermine où et comment les points de contrôle vont affecter la forme. Le nombre de nœuds est égal au nombre de points de contrôle plus le degré de la forme plus 1.

À chaque fois que le paramètre entre dans un nouvel intervalle nodal, un nouveau point de contrôle devient actif tandis qu'un plus ancien devient inactif. Ceci garantit l'influence locale des points de contrôle. Pour $U = (u_0, u_1, \dots, u_m)$ le vecteur de nœuds va se présenter sous la forme suivante :

$$\boldsymbol{u_0 = \cdots u_p < u_{p+1} < \cdots < u_{m-p-1} < u_{m-p} = \cdots = u_m}$$

On repère $p + 1$ fois les valeurs extrêmes et les nœuds consécutifs peuvent avoir des valeurs identiques. Ceci est défini par la multiplicité d'un nœud et permet d'accentuer l'influence d'un point sur la forme.

2.2.6 Courbes et surfaces NURBS

Lorsque des vecteurs de nœuds non uniformes sont utilisés, ces B-Splines rationnelles sont connues sous le nom de NURBS (Non-Uniform Rational B-Spline) (Fig. 16). Les NURBS peuvent représenter de façon exacte toutes les formes polygonales, toutes les coniques, toutes les courbes polynomiales paramétriques par morceaux ainsi que toute courbe complexe. Ces courbes et surfaces d'une grande capacité de description sont utilisées par les modeleurs des logiciels de CAO.

En passants des surfaces de Bézier aux surfaces NURBS le contrôle local des points de contrôle augmente et le contrôle global sur la surface diminue.

NURBS est une des méthodes les plus utilisées pour la modélisation des objets 3D complexes car c'est une représentation standard des courbes et des surfaces [30]. En effet, NURBS est une solution aux problèmes des surfaces gauches. Cette méthode est la généralisation des courbes et des surfaces de Bézier et B-Spline.

Cette méthode interpole les points de contrôle reproduisant la forme de l'objet et permettant des changements dans cette forme [31]. L'un des avantages de cette méthode est sa souplesse pour la conception de formes complexes.

Elle est définie par ses variables paramétriques u et v comme suit :

$$S(u, v) = \frac{\sum_{i=0}^{m}\sum_{k=0}^{n} P_{i,k}.w_{i,k}.B_{i,s}(u).B_{k,l}(v)}{\sum_{i=0}^{m}\sum_{k=0}^{n} w_{i,k}.B_{i,s}(u).B_{k,l}(v)}$$

$B_{i,s}$ et $B_{k,l}$ sont les fonctions de base B-Splines de degré $s-1$ et $l-1$,

$P_{i,k}$: sont les points de contrôle et $w_{i,k}$ sont les poids qui définissent l'influence des points de contrôle à la courbe. Chaque point de contrôle a son propre poid.

Le nombre de nœuds est : $(m+s)+(n+l)$.

La surface change d'une façon prédictible selon le mouvement des points de contrôle, et les poids $w_{i,k}$ jouent des rôles importants dans le processus de reconstruction car ces facteurs déterminent combien un point de contrôle influence la forme de la surface localement.

Voilà pourquoi, les valeurs négatives ou les zéros dans les facteurs de pondération peuvent dégénérer la construction de la surface [32].

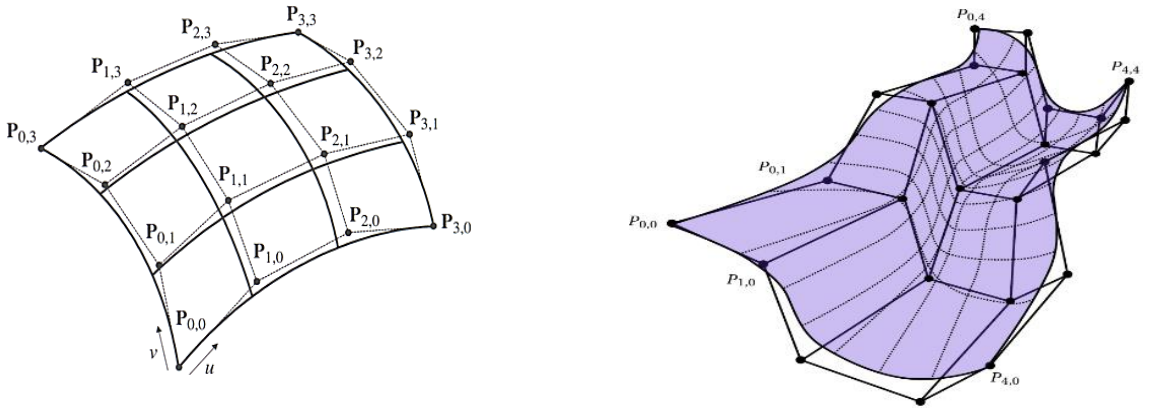


Figure 16: Surface NURBS

2.2.7 Synthèse sur les surfaces gauches

Les surfaces complexes (NURBS, B-Splines, Bézier) sont les plus utilisées pour modéliser les surfaces réels d'un objet. Cela nous a amené à faire une étude bibliographique sur les techniques de construction ou d'évaluation de ces surfaces à partir des informations issus d'un maillage (un nuage de points où une triangulation).

Les surfaces libres (Spline) sont pilotées par un réseau de points de contrôle. Ce réseau peut être raffiné en insérant des points de contrôle entre les points existants. Le maillage polygonal formé par le réseau de contrôle est déjà en quelque sorte une approximation de la surface Spline. Plus on affine ce réseau, plus il tend à se confondre avec la surface qu'il pilote (Fig. 17). Cela a permis de définir les surfaces de subdivision.

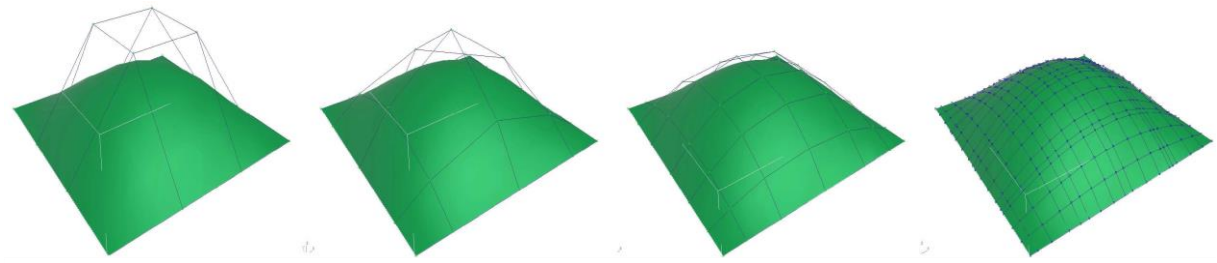


Figure 17: Raffinement successif d'un réseau de contrôle d'une surface Spline

Conclusion

Dans ce chapitre, nous avons présenté les différentes surfaces classiques ainsi que les surfaces gauches et cela afin de les bien maitriser pour les utiliser par la suite pour reconstruire l'objet géométrique 3D. En effet, il y a plusieurs algorithmes pour la reconstruction d'un objet 3D. Ils se diffèrent par la nature de l'objet à reconstruire. Certains algorithmes sont dédiés pour la reconstruction des géométries simples et d'autres pour la reconstruction des géométries complexes basées sur les surfaces gauches comme les surfaces B-Splines.

Chapitre 3

Algorithmes développés

Introduction

Nous présentons dans ce chapitre la solution proposée pour reconstruire un modèle CAO à partir d'un nuage de points 3D. Dans un premier lieu, l'algorithme général est présenté. Par la suite, les différentes techniques de reconstruction des entités géométriques et topologiques sont détaillées. Ces entités sont sous forme des primitives simples et des entités complexes (B-Splines...). La méthode de Levenberg Marquardt est principalement utilisée pour la reconstruction de la géométrie du modèle (courbes et surfaces).

3.1 Reconstruction d'un modèle géométrique à partir d'un nuage de points

3.1.1 Choix de modèle

Les modèles de CAO se basent dans la plupart de temps sur le modèle BREP (Boundary Representation [33]. Ce modèle est utilisé pour la reconstruction de modèle CAO en vue de garantir la compatibilité avec la plupart des logiciels de CAO. La modélisation BREP est basée sur les frontières (faces, contours, arêtes et sommets) (Fig. 18). Le modèle BREP est constitué d'informations topologiques et géométriques. Les informations topologiques existantes au sein d'un modèle BREP sont les connectivités, les liens et les orientations des différentes entités (faces, contours, arêtes et sommets). Les informations géométriques au sein du modèle BREP se résument dans les natures géométriques des surfaces et courbes porteuses des différentes faces et arêtes (face plane, courbe circulaire…).

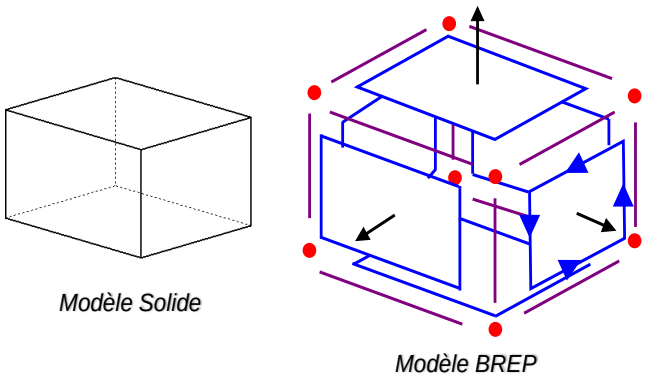


Figure 18 : Modèle BRE.

Les avantages de ce modèle sont :

- Ce modèle est facile en description, visualisation et transformation géométrique ;
- Il est unique (Pour un objet quelconque, il existe un seul modèle BREP) ;
- Il est disponible sur la plupart des systèmes CAO.

3.1.2 Algorithme général

D'une façon générale, l'algorithme de reconstruction suit la hiérarchie du modèle BREP (Fig. 19). Dans une première étape, on commence par l'identification de la topologie (l'identification des triangulations - ou nuage de points correspondants aux faces). Par la suite, les surfaces porteuses des faces sont déterminées à partir des informations précédentes. Dans une troisième étape, les contours sont ajoutés aux surfaces pour obtenir les faces afin d'aboutir au modèle BREP déformé.

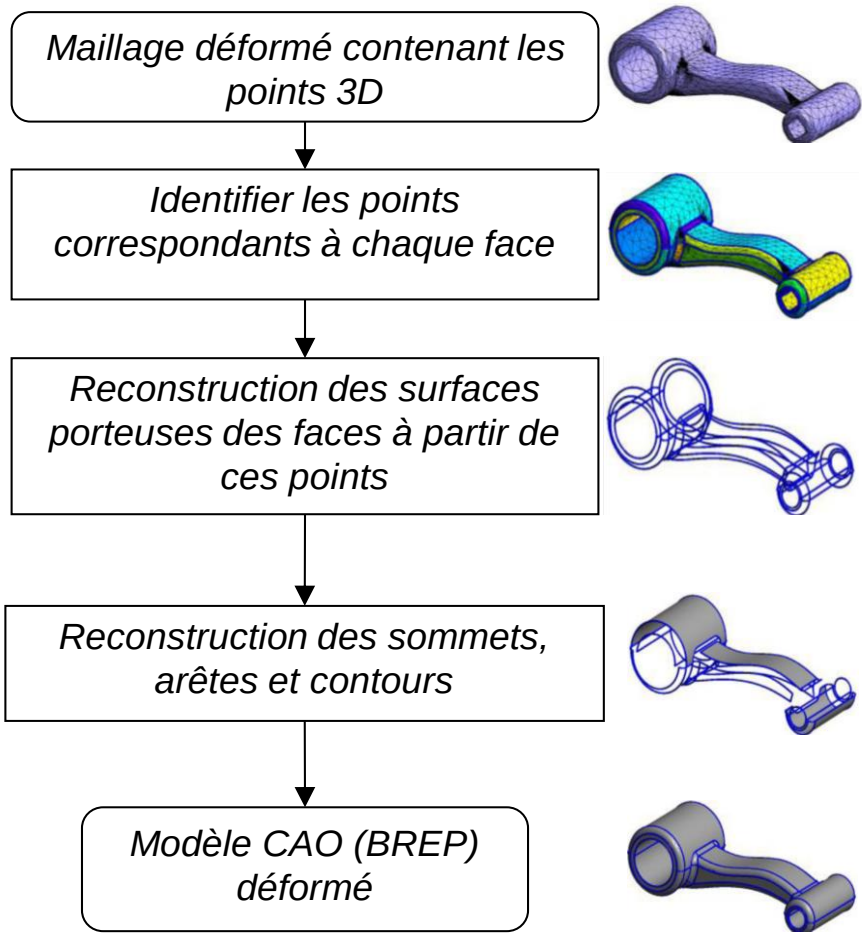


Figure 19: L'algorithme général de reconstruction [14]

Pour mener cet algorithme, une phase de génération des données est nécessaire. Donc, on doit disposer d'un fichier contenant les données suivantes :

- Le nombre de faces dans l'objet
- Le nombre d'arêtes dans chaque face
- Le nombre de points dans chaque arête

- Les points 3D de chaque arête ainsi que le déplacement de la déformation
- Le nombre de points dans chaque face
- Les points 3D de chaque face ainsi que le déplacement de la déformation

Les surfaces porteuses des faces ainsi les courbes porteuses des arêtes sont sous forme des entités classiques simples (cercle, droite, cylindre, tore...) ou de géométrie complexe (BSpline...). L'approche de reconstruction des entités géométriques est basée sur la méthode de **Levenberg Marquardt**. La littérature a montré que c'est une méthode très fiable et qui converge dès les premières itérations.

3.2 Reconstruction des primitives géométriques

Dans cette section, nous allons détailler l'algorithme de Levenberg Marquardt après nous allons définir les paramètres de chaque primitive géométrique.

3.2.1 Algorithme Levenberg Marquardt

Afin de reconstruire une primitive géométrique à partir d'un ensemble de points 3D, nous allons utiliser l'algorithme Levenberg Marquardt [34] qui minimise la fonction objectif qui est la somme des carrés de fonctions souvent non linéaires [35]. Or, l'application principale de cet algorithme est la régression au travers la méthode des moindres carrés.

L'idée principale de cet algorithme est de trouver le vecteur p, qui minimise la fonction objectif, $J(p) = \sum d_i^2(p)$. Avec d_i est la distance entre le point $\boldsymbol{x_i}$ et la géométrie définie par le vecteur de paramètres p. L'algorithme nécessite une estimation initiale ainsi que les premières dérivées de la fonction de distance.

C'est un algorithme itératif qui combine les avantages de l'algorithme du gradient et l'algorithme du Gauss-Newton. Il trouve une solution même s'il commence très loin du minimum. Lorsque la solution actuelle est loin de la bonne, l'algorithme se comporte comme l'algorithme du gradient : lent, mais garanti à converger. Lorsque la solution actuelle est proche de la solution correcte, elle devient une méthode de Gauss-Newton.

Une dérivation de l'algorithme [36] commence par une approximation de la fonction objectif $\boldsymbol{J}$ comme une fonction linéaire du vecteur $\boldsymbol{p}$

$$\boldsymbol{J}(\boldsymbol{p}) \approx \hat{}\boldsymbol{J}(\boldsymbol{p}) = \sum (\boldsymbol{d_i}(\boldsymbol{p_0}) + \boldsymbol{\nabla d_i}(\boldsymbol{p_0}).\boldsymbol{p})^2$$

$\boldsymbol{\nabla d_i}(\boldsymbol{p_0})$ est le gradient de $\boldsymbol{d_i}(\boldsymbol{p})$ évalué à une estimation initiale, $\boldsymbol{p_0}$. La dérivation examine alors comment réduire au minimum $\hat{}\boldsymbol{J}(\boldsymbol{p})$. On cherche $\boldsymbol{p_{nouveau}}$ tel que $\boldsymbol{J}(\boldsymbol{p_{nouveau}}) < \boldsymbol{J}(\boldsymbol{p_0})$.

Lorsqu'on trouve $\boldsymbol{p_{nouveau}}$, ce vecteur devient le nouveau $\boldsymbol{p_0}$ pour l'itération suivante de ce procédé. La solution $\boldsymbol{p}^*$ de chaque itération peut être exprimée comme suit :

$$\boldsymbol{p}^* = \boldsymbol{p}(\boldsymbol{\lambda}) = -\left(\boldsymbol{F_0^T F_0} + \boldsymbol{\lambda D^T D}\right)^{-1} \boldsymbol{F_0^T d}(\boldsymbol{p_0})$$

$\boldsymbol{F_0}$ est la matrice qui a $\boldsymbol{\nabla d_i}(\boldsymbol{p_0})$ à son i-ème ligne, $\boldsymbol{D}$ est la matrice de pondération appropriée, $\boldsymbol{d}(\boldsymbol{p_0})$ est le vecteur de résidus $\boldsymbol{d_i}(\boldsymbol{p_0})$, et $\boldsymbol{\lambda \geq 0}$ est une variable nommée paramètre de Levenberg Marquardt. Cet algorithme est présenté ci-dessous. La matrice $\boldsymbol{F_0^T F_0 + \lambda D^T D}$ est nommée $\boldsymbol{H}$ et le vecteur $\boldsymbol{F_0^T d}(\boldsymbol{p_0})$ est nommé $\boldsymbol{V}$. Cet algorithme comprend une modification proposée par Nash [29] qui consiste à utiliser une matrice de pondération définie de telle sorte que $\boldsymbol{D^T D}$ est la matrice d'identité plus la diagonale de $\boldsymbol{F_0^T F_0}$. L'utilisation de la matrice d'identité dans la définition de $\boldsymbol{D}$ force $\boldsymbol{H}$ pour être définie positive. Puisque $\boldsymbol{H}$ est positive, le système $\boldsymbol{Hx = -v}$ peut être résolu en utilisant la décomposition de Cholesky [37]. Pour appliquer cet algorithme, nous allons choisir les facteurs 10 et 0.04 pour incrémenter et décrémenter $\boldsymbol{\lambda}$ respectivement.

Algorithme 5: Algorithme de Levenberg Marquardt

Données : le vecteur initial $\boldsymbol{p_0}$

$\lambda \leftarrow 0{,}0001$

Répéter

Décrémenter λ

$U \leftarrow F_0^T F$

$V \leftarrow F_0^T \, d(p_0)$

$J_0 \leftarrow \sum d_i^2(P_0)$

Répéter

Incrémenter λ

$H \leftarrow U + \lambda(\mathrm{I} + \text{diagonale}(U_{11}, U_{22}, \ldots, U_{nn})$

Résoudre le système $H.X = -V$

$P_{nouveau} = P_0 + X$

$J_{nouveau} = \sum d_i^2(P_{nouveau})$

si J converge alors

Normaliser $P_{nouveau}$

$p_0 \leftarrow P_{nouveau}$

Retourner (p_0)

Finsi

Jusqu'à $(J_{nouveau} < J_0)$ **ou** itération limite est atteinte

Si $(J_{nouveau} < J_0)$ ***alors***

$p_0 \leftarrow P_{nouveau}$

Finsi

Jusqu'à une itération limite est atteinte

Fin

Pour la suite, on définit pour chaque primitive :

g_i : la distance entre le point $\boldsymbol{x_i}$ et le plan défini par le point $\boldsymbol{x}$ et la direction normale $\boldsymbol{a}$.

f_i : la distance entre le point $\boldsymbol{x_i}$ et la ligne définie par le point $\boldsymbol{x}$ et la direction normale $\boldsymbol{a}$.

$\boldsymbol{x}$: Un point de coordonnées (x, y, z)

3.2.2 Cas d'un plan

Les paramètres d'un plan sont :

$\boldsymbol{x}$: Un point dans le plan

$\boldsymbol{a}$: Le cosinus directeur de la normale au plan

L'équation de distance entre un point et la géométrie est :

$$d_i = d_i(\boldsymbol{x_i}) = d(\boldsymbol{x_i}, \boldsymbol{x}, a) = \boldsymbol{a}(\boldsymbol{x_i} - \boldsymbol{x}) = a(x_i - x) + b(y_i - y) + c(z_i - z)$$

La fonction objectif qu'on cherche à optimiser est :

$$J(\boldsymbol{x}, \boldsymbol{a}) = \sum {g_i}^2 = \sum [\boldsymbol{a}.(\boldsymbol{x_i} - \boldsymbol{x})]^2$$

En dérivant cette fonction objectif par rapport aux paramètres on obtient :

$$\frac{\partial J}{\partial x} = 2\sum g_i (g_i)_x$$

$$\frac{\partial J}{\partial y} = 2\sum g_i (g_i)_y$$

$$\frac{\partial J}{\partial z} = 2\sum g_i (g_i)_z$$

$$\frac{\partial J}{\partial A} = 2\sum g_i (g_i)_A$$

$$\frac{\partial J}{\partial B} = 2\sum g_i (g_i)_B$$

$$\frac{\partial J}{\partial C} = 2\sum g_i (g_i)_C$$

Les dérivées de la distance d_i par rapport aux paramètres du vecteur initial :

$$\frac{\partial d_i}{\partial x} = (g_i)_x$$
$$\frac{\partial d_i}{\partial y} = (g_i)_y$$
$$\frac{\partial d_i}{\partial z} = (g_i)_z$$
$$\frac{\partial d_i}{\partial A} = (g_i)_A$$
$$\frac{\partial d_i}{\partial B} = (g_i)_B$$
$$\frac{\partial d_i}{\partial C} = (g_i)_C$$

3.2.3 Cas d'une sphère

Les paramètres d'une sphère sont :

$\boldsymbol{x}$: Le centre de la sphère

r : Le rayon de la sphère

L'équation de distance est :

$$d_i = d_i(\boldsymbol{x_i}) = |\boldsymbol{x}_i - \boldsymbol{x}| - r$$

La fonction objectif qu'on cherche à optimiser est :

$$J(\boldsymbol{x}, \boldsymbol{a}) = \sum (|\boldsymbol{x}_i - \boldsymbol{x}| - r)^2$$

En dérivant cette fonction objectif par rapport aux paramètres on obtient :

$$\frac{\partial J}{\partial x} = -2 \sum d_i (x_i - x) / |\boldsymbol{x}_i - \boldsymbol{x}|$$
$$\frac{\partial J}{\partial y} = -2 \sum d_i (y_i - y) / |\boldsymbol{x}_i - \boldsymbol{x}|$$
$$\frac{\partial J}{\partial z} = -2 \sum d_i (z_i - z) / |\boldsymbol{x}_i - \boldsymbol{x}|$$
$$\frac{\partial J}{\partial r} = -2 \sum d_i$$

Les dérivées de la distance d_i par rapport aux paramètres du vecteur initial :

$$\frac{\partial d_i}{\partial x} = -(x_i - x) / |\boldsymbol{x}_i - \boldsymbol{x}|$$
$$\frac{\partial d_i}{\partial y} = -(y_i - y) / |\boldsymbol{x}_i - \boldsymbol{x}|$$

$$\frac{\partial d_i}{\partial z} = -(z_i - z)/|\boldsymbol{x}_i - \boldsymbol{x}|$$
$$\frac{\partial d_i}{\partial r} = -1$$

3.2.4 Cas d'un cylindre

Les paramètres d'un cylindre sont :

$\boldsymbol{x}$: Un point sur l'axe du cylindre

$\boldsymbol{A}$: direction de l'axe du cylindre

r : Le rayon du cylindre

L'équation de distance est :

$$d_i = d_i(\boldsymbol{x}_i) = f_i - r$$

Avec $f_i = \sqrt{|\boldsymbol{x}_i - \boldsymbol{x}|^2 - g_i^2}$

Et $g_i = \boldsymbol{a}(\boldsymbol{x_i} - \boldsymbol{x})$

La fonction objectif qu'on cherche à optimiser est :

$$J(\boldsymbol{x}, \boldsymbol{A}, r) = \sum (f_i - r)^2$$

En dérivant cette fonction objectif par rapport aux paramètres on obtient :

$$\frac{\partial J}{\partial x} = 2\sum (f_i - r)(f_i)_x$$
$$\frac{\partial J}{\partial y} = 2\sum (f_i - r)(f_i)_y$$
$$\frac{\partial J}{\partial z} = 2\sum (f_i - r)(f_i)_z$$
$$\frac{\partial J}{\partial A} = 2\sum (f_i - r)(f_i)_A$$
$$\frac{\partial J}{\partial B} = 2\sum (f_i - r)(f_i)_B$$
$$\frac{\partial J}{\partial C} = 2\sum (f_i - r)(f_i)_C$$
$$\frac{\partial J}{\partial r} = -2\sum (f_i - r)$$

Les dérivées de la distance d_i par rapport aux paramètres du vecteur initial :

$$\frac{\partial d_i}{\partial x} = (f_i)_x$$

$$\frac{\partial d_i}{\partial y} = (f_i)_y$$

$$\frac{\partial d_i}{\partial z} = (f_i)_z$$

$$\frac{\partial d_i}{\partial A} = (f_i)_A$$

$$\frac{\partial d_i}{\partial B} = (f_i)_B$$

$$\frac{\partial d_i}{\partial C} = (f_i)_C$$

$$\frac{\partial d_i}{\partial r} = -1$$

Pour calculer la direction de l'axe du cylindre $\boldsymbol{A} = (\mathrm{A}, \mathrm{B}, \mathrm{C})$, il faut calculer la matrice d'inertie suivante.

$$\begin{pmatrix} b^2 + c^2 & -a * b & -a * c \\ -a * b & a^2 + c^2 & -b * c \\ -a * c & -b * c & a^2 + b^2 \end{pmatrix}$$: matrice symétrique

Pour cela, nous décrivons l'algorithme qui calcule la matrice d'inertie :

Algorithme 6: Algorithme Matrice-Inertie

Données : nombre_points

Pour i de 1 à nombre_points

aa $\leftarrow$ aa + $(x_i - x_{centre})^2$

zz $\leftarrow zz + (z_i - z_{centre})^2$

yy $\leftarrow$ yy + $(y_i - y_{centre})^2$

ab $\leftarrow$ ab + $(x_i - x_{centre}) * (y_i - y_{centre})$

bc $\leftarrow$ bc + $(y_i - y_{centre}) * (z_i - z_{centre})$

ac $\leftarrow$ ac + $(x_i - x_{centre}) * (z_i - z_{centre})$

Fin pour

aa $\leftarrow$ aa $\times \frac{1}{nombre\_points}$

bb $\leftarrow$ bb $\times \frac{1}{nombre\_points}$

cc $\leftarrow$ cc $\times \frac{1}{nombre\_points}$

$$\text{ab} \leftarrow \text{ab} \times \frac{1}{nombre\_points}$$

$$\text{bc} \leftarrow \text{bc} \times \frac{1}{nombre\_points}$$

$$\text{ac} \leftarrow \text{ac} \times \frac{1}{nombre\_points}$$

Fin

Ensuite, nous cherchons les valeurs des vecteurs propres d'une matrice symétrique qui va retourner une matrice. Enfin, on prend la première colonne de cette matrice. Cette colonne sera un vecteur de taille 3 qui présente A $= \boldsymbol{A}_x$, B $= \boldsymbol{A}_y$, C $= \boldsymbol{A}_z$.

3.2.5 Cas d'un tore

Les paramètres d'un tore sont :

$\boldsymbol{x}$: Le centre du tore

$\boldsymbol{A}$: Direction de l'axe du tore

r : Le grand rayon

R : Le petit rayon

L'équation de distance est :

$$d_i = d_i(\boldsymbol{x_i}) = \sqrt{g_i^2 + (f_i - r)^2 - R}$$

$$\text{Avec} \quad f_i = \sqrt{|\boldsymbol{x}_i - \boldsymbol{x}|^2 - g_i^2}$$

$$\text{Et} \qquad g_i = \boldsymbol{a}(\boldsymbol{x_i} - \boldsymbol{x})$$

La fonction objectif qu'on cherche à optimiser est :

$$J(\boldsymbol{x}, \boldsymbol{A}, r, R) = \sum \left[\sqrt{g_i^2 + (f_i - r)^2} - R \right]^2$$

En dérivant cette fonction objectif par rapport aux paramètres on obtient :

$$\frac{\partial J}{\partial x} = 2 \sum \frac{d_i}{d_i + R} [g_i (g_i)_x + (f_i - r)(f_i)_x]$$

$$\frac{\partial J}{\partial y} = 2 \sum \frac{d_i}{d_i + R} [g_i (g_i)_y + (f_i - r)(f_i)_y]$$

$$\frac{\partial J}{\partial z} = 2 \sum \frac{d_i}{d_i + R} [g_i (g_i)_z + (f_i - r)(f_i)_z]$$

$$\frac{\partial J}{\partial A} = 2 \sum \frac{d_i}{d_i + R} [g_i (g_i)_A + (f_i - r)(f_i)_A]$$

$$\frac{\partial J}{\partial B} = 2\sum \frac{d_i}{d_i + R}[g_i(g_i)_B + (f_i - r)(f_i)_B]$$
$$\frac{\partial J}{\partial C} = 2\sum \frac{d_i}{d_i + R}[g_i(g_i)_C + (f_i - r)(f_i)_C]$$
$$\frac{\partial J}{\partial r} = 2\sum \frac{d_i}{d_i + R}[g_i(g_i)_r + (f_i - r)(f_i)_r]$$
$$\frac{\partial J}{\partial R} = -2\sum d_i$$

3.3 Optimisation des points de contrôle d'une courbe B-Spline

L'objectif dans cette partie est d'approximer une courbe B-Spline tout en optimisant les points de contrôle étant donné les points obtenus par un scan 3D.

3.3.1 Principe de la reconstruction d'une courbe B-Spline

Etant donnée un ensemble de points $\mathbf{q_l}$ $(\boldsymbol{l = 0, \dots, L + 1})$ avec les paramètres associés $\boldsymbol{u_l}$, nous voulons construire une courbe B-Spline de degré $\boldsymbol{n}$ (ordre $\boldsymbol{k = n + 1}$) avec $\boldsymbol{M + 1}$ points de contrôle $\mathbf{p_i}$.

$$\mathrm{S(u)} = \sum_{\mathrm{i=0}}^{\mathrm{M+1}} \mathrm{N_{i,k}}(u_l)\mathbf{p_i}$$

Cette courbe interpole le premier et le dernier points $\mathbf{q_0}$ et $\mathbf{q_{L+1}}$ et elle approxime les L points $\mathbf{q_l}$. Ce problème peut être résolu en utilisant la fonction de moindre carrée suivante :

$$J(p) = \sum_{l=1}^{L}\left(\sum_{i=1}^{M} N_{i,k}(u_l)\mathbf{p_i} - \mathbf{q_l}\right)^2$$

Pour résoudre ce problème de moindre carrée on peut utiliser plusieurs algorithmes d'optimisation comme l'algorithme de Gauss-Newton et l'algorithme de gradient. Nous avons choisi d'utiliser l'algorithme Levenberg Marquardt qui combine ces deux dernières méthodes.

Le but est de trouver M points de contrôle qui sont optimisés. Pour cela, on doit définir la différentielle de la fonction objectif par rapport à $\mathbf{p_j}$.

$$\frac{\partial J(P)}{\partial p_j} = 2\sum_{l=1}^{L}\left(\sum_{i=1}^{M} N_{i,k}(u_l)\mathbf{p_i} - \mathbf{q_l}\right)N_{j,k}(u_l) \quad (j = 1,2, \dots, M).$$

Et la dérivée de la distance par rapport à $\mathbf{p_j}$ est : $\frac{\partial d_i}{\partial p_j} = N_{j,k}(u_l)$.

3.3.2 Description de l'algorithme

On dispose d'un ensemble de points 3D en entrée. L'algorithme nécessite l'initialisation du degré de la courbe B-Spline ainsi l'initialisation d'un certain nombre de points de contrôle qu'on va optimiser jusqu'à trouver les bons points de contrôle qui approxime la courbe.
Les fonctions de base B-Spline sont calculées automatiquement grâce au degré de la courbe, le nombre de points de contrôle et les paramètres $\boldsymbol{u_l}$ associé à chaque point 3D.

L'algorithme Levenberg Marquardt est itératif, à chaque itération, on calcule les nouveaux points de contrôle jusqu'à minimiser la distance entre la courbe B-Spline définie par ces points de contrôle et les points 3D.

Algorithme 7: Algorithme Reconstruction d'une courbe B-Spline

Données : Points de contrôle **PX**, **PY**, **PZ**, le nombre de points **nbPoints**, Points échantillonnés **Q**, le degré **p**, le vecteur **UL** qui contient u_l correspondant à chaque q_l, le nombre de points de contrôle **n**

U ← Calcul-Vecteur-Nodal(n,p)

N ← Intialiser matrice (nbPoints , n , 0)

Pour i de 1 à nbPoints

    IndiceU ← trouverIndice(n, p, UL[i], U)

    Nu ← Calcul-Base-B-Spline(indiceU, p, UL[i], U)

    z ← 1

    Pour k de indiceU-p+1 à indiceU+1

        N[i,k] ← Nu[z]

            z← z + 1

    Fin pour

Fin pour

PointControleX ← Levenberg Marquardt (PX)

PointControleY ← Levenberg Marquardt (PY)

PointControleZ ← Levenberg Marquardt (PZ)

S ← Intialiser matrice (nbPoints , 3 , 0)

Pour i de 1 à nbPoints faire

    Aux1 ← 0, Aux2 ← 0, Aux3 ← 0

Pour j de 1 à n faire

Aux1 ← Aux1+N[i,j]* PointControleX [j]

Aux2 ← Aux2+N[i,j]* PointControleY [j]

Aux3 ← Aux3+N[i,j]* PointControleZ [j]

S [i,1] ← Aux1

S [i,2] ← Aux2

S [i,3] ← Aux3

Fin pour

Fin pour

Afficher la courbe passant par les points de la matrice S contenant les points 3D

Fin

Pour calculer le vecteur nodal à partir du degré et le nombre de points de contrôle, nous avons développé cette fonction :

Algorithme 8: Fonction Calcul-Vecteur-Nodal

Données : le nombre de points de contrôle $\boldsymbol{n}$, le degré $\boldsymbol{p}$

$\boldsymbol{Longueur} \leftarrow \boldsymbol{p+n+1}$

$U \leftarrow$ Intialiser vecteur $(Longueur\ , 0)$

Pour $\boldsymbol{i}$ de 1 à $\boldsymbol{p}$ faire

$\boldsymbol{U[i]} \leftarrow 0$

Fin pour

Pour $\boldsymbol{i}$ de $\boldsymbol{p+1}$ à $(\boldsymbol{Longueur-p})$ faire

$\boldsymbol{U[i]} \leftarrow (\boldsymbol{i-p-1})/(\boldsymbol{Longueur-(2\times p)-1})$

Fin pour

Pour $\boldsymbol{i}$ de $\boldsymbol{Longueur-p+1}$ à $\boldsymbol{Longueur}$ faire

$\boldsymbol{U[i]} \leftarrow 1$

Fin pour

Retourner $(\boldsymbol{U})$

Fin

3.4 Optimisation des points de contrôle d'une surface B-Spline

La surface B-Spline est donnée à titre unique avec le degré, les valeurs des vecteurs nodaux et des points de contrôle. Etant donné un ensemble de points 3D, l'objectif est d'approximer la surface B-Spline, tout en commençant par définir une surface de base formée

par des points de contrôle initiaux et ensuite appliquer l'algorithme d'optimisation Levenberg Marquardt pour estimer les points de contrôle de la surface à reconstruire.

La surface B-Spline est de degré $\boldsymbol{p}$ suivant $\boldsymbol{u}$ et $\boldsymbol{q}$ suivant $\boldsymbol{v}$. On note que $\boldsymbol{u}$ et $\boldsymbol{v}$ sont les paramètres de localisation qui localisent un point dans la surface [25]. Dans le contexte de la reconstruction d'une surface gauche le but est toujours de trouver l'ensemble de points de contrôle. L'approximation de la surface est considérée comme un problème de minimisation de distance entre un point du nuage de points et son correspondant dans la surface obtenue par les points de contrôle initiaux.

3.4.1 Fonctions des moindres carrés pour approximer une surface B-Spline

Le problème de la reconstruction de la surface est décrit par Weiss et al. dans [38] comme suit :

Nous avons un ensemble de points $\boldsymbol{S_{p_r}}$ associés à l'ensemble de points $\boldsymbol{p_r}$. C'est-à-dire pour chaque point $\boldsymbol{p_r}$ il existe un point dans la surface défini par un paramètre pair $(\boldsymbol{u_r}, \boldsymbol{v_r})$.
En utilisant la somme des moindres carrés nous approximons chaque point.
Aussi dans [39], Martin Aigner et Bert Jüttler, ont expliqué que lors de la reconstruction d'une surface à partir d'un ensemble de points, on minimise certaines distances entre les points et la surface. Ces distances peuvent être mesurées de différentes façons.

La distance géométrique à partir d'un point de donnée $\boldsymbol{p_r}$ à la surface est donnée par sa distance euclidienne (minimale) à la surface :

$$d_r = min\left\|\left(p_r - s_{u_r,v_r}\right)\right\|$$

La minimisation des carrés de la distance géométrique mène à un problème des moindres carrés décrit comme suit :

$$J(p) = \sum_{r=1}^{M} d_r^2$$

A cause de non linéarité, on doit utiliser une méthode itérative pour résoudre ce problème.

On commence par une surface de base, après on calcule - pour chaque point de donnée - les points les plus proches associés. En substituant ces points [40], on obtient un problème des moindres carrés pour les paramètres de la surface qui peuvent être résolus en utilisant des techniques d'optimisation. Aussi dans [41], Xiao-Diao et al. ont décrit le problème de trouver la distance minimale entre un point et une surface B-Spline. Dans [42], Yong Ming a expliqué

comment on peut approximer une surface B-Spline. Etant donné un ensemble de points $\boldsymbol{q_l}$ et les nœuds associés (u_l, v_l) avec $l = 0,1, \dots, L$. Nous voulons construire une surface B-Spline avec :

$$S(u,v) = \sum_{i=0}^{N}\sum_{j=0}^{M} N_{i,k_u}(u)N_{j,k_v}(v)\boldsymbol{P}_{i,j}$$

On peut reconstruire la surface passant par les points $\boldsymbol{q_l}$ en utilisant la méthode d'approximation des moindres carrés suivante :

$$J(\boldsymbol{P}) = \sum_{l=0}^{L}\left(\sum_{i=0}^{N}\sum_{j=0}^{M} N_{i,k_u}(u_l)N_{j,k_v}(v_l)\boldsymbol{P}_{i,j} - \boldsymbol{q}_l\right)^2 \rightarrow \min$$

$\boldsymbol{P}$: est une collection de $(N+1)\times(M+1)$ points de contrôle $\boldsymbol{P}_{i,j}$. Mathématiquement, c'est équivalent à résoudre les $(N+1)\times(M+1)$ équations linéaires :

$$\sum_{l=0}^{L}\left(\sum_{i=0}^{N}\sum_{j=0}^{M} N_{i,k_u}(u_l)N_{j,k_v}(v_l)\boldsymbol{P}_{i,j} - \boldsymbol{q}_l\right)N_{r,k_u}(u_l)N_{s,k_v}(v_l) = 0$$

$$avec\ (r = 0, \dots, N; s = 0, \dots, M).$$

Alternativement, nous pouvons écrire le système d'équations linéaires ci-dessus comme suit :

$$\sum_{l=0}^{L}\sum_{i=0}^{N}\sum_{j=0}^{M} N_{i,k_u}(u_l)N_{j,k_v}(v_l)\,N_{r,k_u}(u_l)N_{s,k_v}(v_l)\boldsymbol{P}_{i,j} = \sum_{l=0}^{L} N_{r,k_u}(u_l)N_{s,k_v}(v_l)\,\boldsymbol{q}_l$$

$avec\ (r = 0, \dots, N; s = 0, \dots, M).$

Comme on le voit, le système ci-dessus comporte quatre matrices de manipulation et de multiplication. Il est difficile de configurer la matrice finale sur la base de la représentation ci-dessus. Dans les parties suivantes, deux méthodes plus simples sont décrites pour résoudre le problème d'approximation, en fonction de la façon dont les points de $\boldsymbol{q}_l$ sont échantillonnés.

3.4.2 Approximation d'une surface basée sur des points sous forme de grille

Etant donné une grille de points $\boldsymbol{Q_{p,q}}$ et les nœuds associés (u_p, v_q) avec $p = 0,1, \dots, r$ et $q = 0,1, \dots, s$, nous voulons construire la surface B-Spline suivante :

$$S(u,v) = \sum_{i=0}^{N}\sum_{j=0}^{M} N_{i,k_u}(u)N_{j,k_v}(v)\boldsymbol{P}_{i,j}$$

Dans cette section, la méthode d'approximation se base seulement sur l'algorithme des moindres carrés pour approximer une courbe qui peut donc faire le meilleur usage des routines de courbe.

3.4.3 Approximation d'une surface basée sur des points aléatoires

Puisque $\boldsymbol{q}_l$ sont échantillonnés aléatoirement, la méthode précédente ne peut pas être appliquée. Alors, nous devons résoudre l'algorithme d'approximation des moindres carrés de l'ensemble de points. Pour éviter la manipulation et la multiplication de quatre matrices, on dérive une simple représentation.

Pour comprendre, nous commençons par une surface B-Spline biquadratique décrite comme suivant :

$$\begin{aligned} S(u,v) = {} & [p_{00}N_0(u) + p_{10}N_1(u) + p_{20}N_2(u)]N_0(v) \\ & +[p_{01}N_0(u) + p_{11}N_1(u) + p_{21}N_2(u)]N_1(v) \\ & +[p_{02}N_0(u) + p_{12}N_1(u) + p_{22}N_2(u)]N_2(v) \end{aligned}$$

Si nous conservons nos points de contrôle en réseau unidimensionnel comme $p_{00}, p_{10}, p_{20}, p_{01}, p_{11}, \dots, p_{22}$ avec un indice de $k = 0$ à $k = 8$ on peut écrire la surface comme :

$$S(u,v) = \sum_{k=0}^{8} p_k B_k(u,v)$$

Avec $B_{j\times 3+i} = N_i(u)N_j(v)$, il est clair que la surface B-Spline est représentée comme une courbe B-Spline.

Plus généralement, on peut étendre la surface B-Spline biquadratique à une surface B-Spline générique qui contient $(N+1)\times(M+1)$ points de contrôle. En notant $K = (N+1)\times(M+1) - 1$, on a :

$$S(u,v) = \sum_{k=0}^{K} p_k B_k(u,v)$$

Avec $B_{j\times(N+1)+i} = N_i(u)N_j(v)$. Par conséquent, le problème des moindres carrés devient :

$$J(\boldsymbol{P}) = \sum_{l=0}^{L}\left(\sum_{k=0}^{K} \boldsymbol{p}_k B_k(u_l, v_l) - \boldsymbol{q}_l\right)^2 \rightarrow \min$$

B : c'est une matrice de taille $(L+1) \times (K+1)$

$$\begin{pmatrix} B_0(u_0,v_0) & B_1(u_0,v_0) & \cdots & B_K(u_0,v_0) \\ B_0(u_1,v_1) & B_1(u_1,v_1) & \dots & B_K(u_1,v_1) \\ \vdots & \vdots & \vdots & \\ B_0(u_L,v_L) & B_1(u_L,v_L) & \dots & B_K(u_L,v_L) \end{pmatrix}$$

En utilisant le même principe que les courbes B-Splines, la dérivée de la fonction de distance en chaque point est $\frac{\partial d_i}{\partial p_j} = B_j(u_l, v_l)$ et la différentielle de la fonction objectif par rapport à $\mathbf{p_j}$ est la suivante :

$$\frac{\partial J(P)}{\partial p_j} = 2\sum_{l=0}^{L}\left(\sum_{i=0}^{K} \boldsymbol{p}_i B_i(u_l, v_l) - \boldsymbol{q}_l\right) B_j(u_l, v_l) \quad (j = 0,1,\dots,K).$$

Ci-dessous l'algorithme de la reconstruction d'une surface B-Spline à partir d'un ensemble de points :

Algorithme 9: Algorithme Reconstruction d'une surface B-Spline

Données : Les Vecteurs de points de contrôle **PX**, **PY**, **PZ**, nombre de points suivant u : **nbPointsU**, nombre de points suivant v : **nbPointsV**, le vecteur des points échantillonnés **Q**, le degré **p**, le degré **q,** les vecteurs **UL** et **VL** qui contiennent u_l et v_l correspondant à chaque q_l, le nombre de point de contrôle suivant u : **n,** le nombre de points de contrôle suivant v : **m**

U ← Calcul-Vecteur-Nodal (n,p)

V ← Calcul-Vecteur-Nodal (m,q)

N1 ← Intialiser matrice (nbPointsU , n , 0)

N2 ← Intialiser matrice (nbPointsV , m , 0)

B ← Intialiser matrice (nbPointsU × nbPointsV , n × m , 0)

Pour i de 1 à nbPointsU faire

    IndiceU ← trouverIndice(n, p, UL[i], U)

    Nu ← Calcul-Base-B-Spline(indiceU, p, UL[i], U)

    z ← 1

    Pour k de indiceU-p+1 à indiceU+1 faire

        N1[i,k] ← Nu[z]

            z ← z + 1

    Fin pour

Fin pour

Pour i de 1 à nbPointsV faire

```
        IndiceV ← trouverIndice(m, q, VL[i], V)
        Nv ← Calcul-Base-B-Spline(indiceV, q, VL[i], V)
        z ← 1
        Pour k de (indiceV-q+1) à (indiceV+1) faire
                N2[i,k] ← Nv[z]
                        z← z + 1
        Fin pour
Fin pour
Pour i de 1 à nbPointsU faire
        Pour j de 1 à nbPointsV faire
                Pour kq de 0 à m-1 faire
                        Pour kp de 0 à n-1 faire
                                B[k,(kq × m)+kp+1]=N1[i,kp+1] × N2 [j,kq+1]
                        Fin pour
                Fin pour
        Fin pour
Fin pour
PointControleX ← Levenberg Marquardt (PX)
PointControleY ← Levenberg Marquardt (PY)
PointControleZ ← Levenberg Marquardt (PZ)
S ← Intialiser matrice ((nbPointsU × nbPointsV) , 3 , 0)
Pour i de 1 à (nbPointsU × nbPointsV) faire
        Aux1 ← 0, Aux2 ← 0, Aux3 ← 0
        Pour j de 1 à (n×m) faire
                Aux1 ← Aux1+B[i,j]* PointControleX [j]
                Aux2 ← Aux2+B[i,j]* PointControleY [j]
                Aux3 ← Aux3+B[i,j]* PointControleZ [j]
                S [i,1] ← Aux1
                S [i,2] ← Aux2
                S [i,3] ← Aux3
        Fin pour
```

Fin pour

Afficher la surface passant par les points de la matrice S contenant les points 3D

Fin

Conclusion

Dans ce chapitre, l'algorithme général de la reconstruction d'un modèle CAO, à partir d'un nuage de points, est présenté. Par la suite les méthodes de reconstruction des entités géométriques (se basant sur l'algorithme de Levenberg Marquardt) sont détaillées. Ces entités géométriques sont sous forme des primitives simples ou des courbes et surfaces B-Splines. Dans le chapitre suivant, on va présenter les différents résultats issus de ces algorithmes.

Chapitre 4

Illustration

Introduction

Dans ce chapitre, les différentes étapes pour solutionner la problématique, sont exposées. Dans un premier lieu, le choix de l'outil Open Cascade, qui est utilisé avec la technologie C++, est discuté. Ensuite, nous montrons les différentes études réalisées ainsi les résultats retrouvés. Enfin, une mise en valeur de la performance de ces méthodes est présentée.

4.1 Implémentation informatique

4.1.1 l'outil Open Cascade

Open cascade est un logiciel libre pour la conception assistée par ordinateur et la modélisation tridimensionnelle. Il est composé d'une bibliothèque très riche d'objets utilisables en C++ .

L'architecture Open Cascade comprend les éléments suivants :

- OCAF (Open CASCADE Application Framework)

- Les classes de fondation

- Les données de modélisation

- Les algorithmes de modélisation

- Le maillage

- La visualisation

- L'échange de données

- Les outils de développement

- Les bibliothèques d'interfaces graphiques.

La plupart des fonctionnalités de Open Cascade est disponible sous la forme de bibliothèques C ++. Pour utiliser ces bibliothèques, on doit avoir un compilateur C++ installé dans la machine. VC++ 10 est utilisé pour des tests réguliers et pour construire un paquet binaire de sortie officielle de Open Cascade sur Windows. Visual C++ continent 10 projets C++ illustrant comment utiliser un module ou une fonction.

4.1.2 Visual Studio C++

Visual C++ est un environnement de développement intégré pour Windows, conçu par Microsoft pour les langages de programmation C et C++.

Différents outils peuvent être intégrés pour développer, compiler, déboguer un programme en C++ s'exécutant sur Windows. Open cascade est conçu pour enrichir les outils de C++ avec des classes, des méthodes et des fonctions de modélisation 3D. La combinaison de toutes ces ressources nous permet de créer des applications importantes.

4.1.3 La technologie C++

C++ est un langage de programmation compilé permettant la programmation sous de multiples paradigmes comme la programmation procédurale, la programmation orientée objet et la programmation générique.

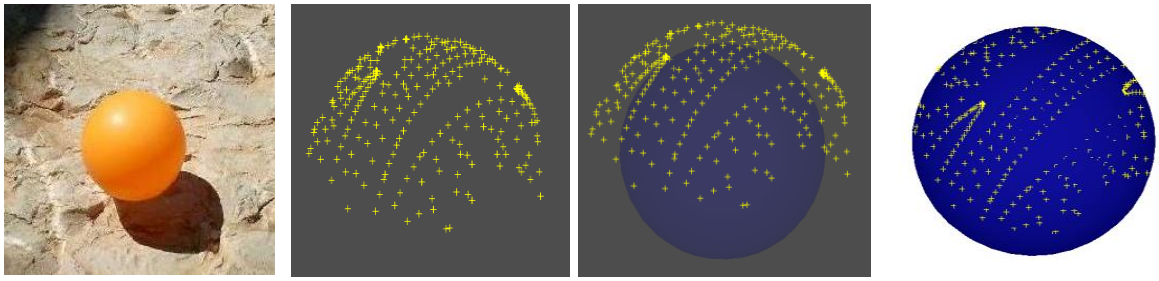
Le langage C++ est de plus en plus utilisé pour l'écriture des bibliothèques mathématiques, non pas pour son caractère orienté objet mais pour la généricité qu'il offre. La facilité de surcharge des opérateurs prédéfinis permet de traduire un algorithme en langage C++ en utilisant des notations assez intuitives, ce qui a le grand avantage de rendre le code clair et de repousser la frontière entre les types et fonctions définis par l'utilisateur et ceux prédéfinis par le langage.

4.2 Résultats

4.2.1 Résultats de reconstruction des primitives géométriques

Dans cette partie, nous allons présenter nos résultats de la reconstruction de quatre primitives géométriques en approximant le nuage de points obtenu par un scan 3D en utilisant l'algorithme d'optimisation Levenberg Marquardt (Chapitre 3).

- Reconstruction d'une sphère réelle de rayon *19.8 cm*



(a) (b) (c) (d)

Figure 20: Reconstruction d'une sphère réelle après 5 itérations

(a) sphère réelle de rayon 19.8; (b) nuage de points du scan 3D ; (c) approximation de la sphère ; (d) sphère reconstruite de rayon 19.8 après la 5ème itération

- Reconstruction d'un cylindre réel de rayon *34 cm*, hauteur *74 cm* et axe *(0.2,0.4,1)*

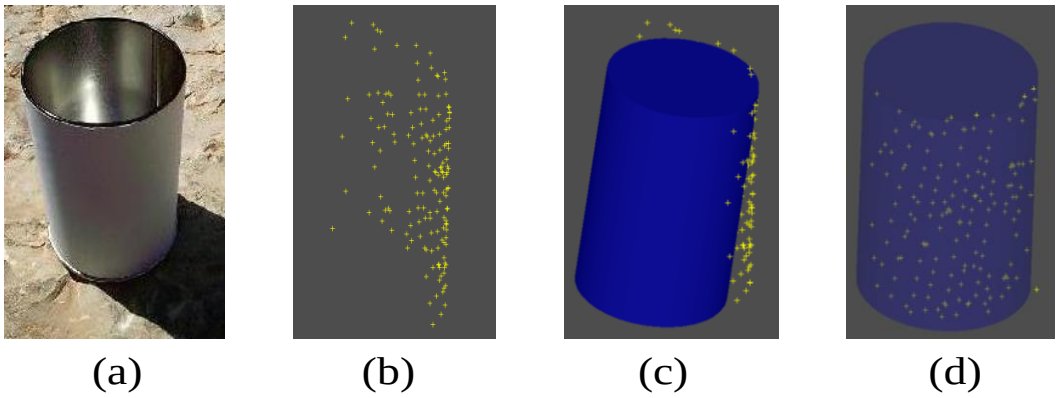


Figure 21: Reconstruction d'un cylindre réel après 2 itérations

(a) Cylindre réel de rayon 34, hauteur 74 et axe (0,0,1) ; (b) nuage de points du scan 3D ; (c) approximation du cylindre ;(d) cylindre reconstruit : rayon 34, hauteur 74, axe (0.2,0.4,1)

- Reconstruction d'un Tore de petit rayon *10 cm* et de grand rayon *30 cm*

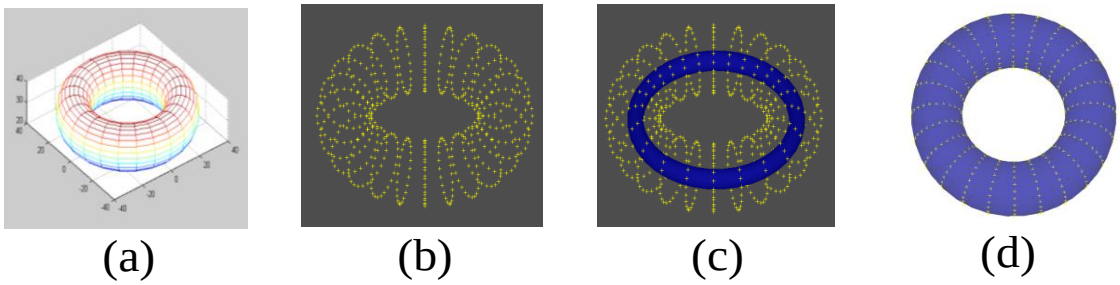


Figure 22: Reconstruction d'un tore après 5 itérations

(a) Tore d'un grand rayon 30 et petit rayon 10 ; (b) nuage de points du tore ; (c) approximation du tore ; (d) tore reconstruit de grand rayon 30 et petit rayon 10

- Reconstruction d'un plan réel à partir de son scan 3D d'axe *(-0.1,-0.2,1)*

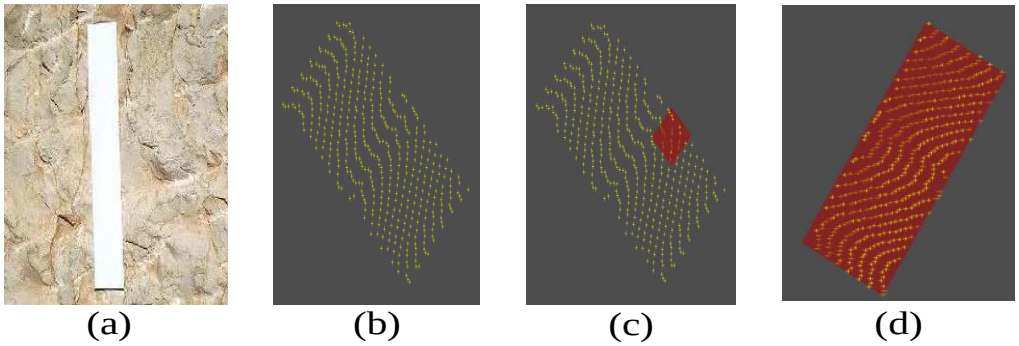


Figure 23: Reconstruction d'un plan réel après 2 itérations

(a) *Plan réel d'axe (-0.1,-0.2,1) ; (b) nuage de points du plan ; (c) approximation du plan ; (d) plan reconstruit à partir du nuage de points d'axe (-0.1,-0.2,1).*

4.2.2 Résultats de reconstruction de courbes et surfaces complexes

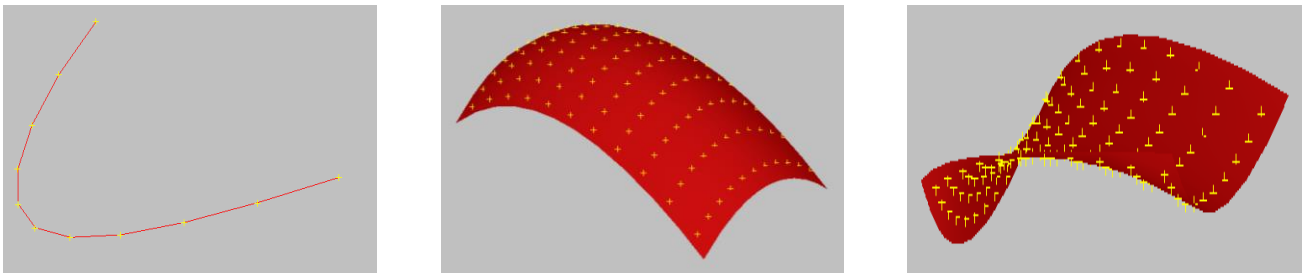


Figure 24: Optimisation des points de contrôle d'une courbe et deux surfaces B-Splines

(a) *Courbe B-Spline approximé de degré 3 ; (b) et (c) Surfaces B-Splines approximées de degré 2*

4.2.3 Résultats de quelques pièces mécaniques déformées reconstruites

L'objectif de ce travail est la reconstruction d'un modèle géométrique complet à partir d'un nuage de points. Dans notre cas, ce dernier est issue d'un maillage déformé. Dans ce qui suit, des résultats de reconstruction des modèles CAO sont présentés. Ces modèles sont composés à la base par des faces planes et des faces cylindriques. Après déformation, ces faces sont devenues complexes et modélisées par des surfaces de type B-Splines.

Ci-dessous (Fig. 25) le résultat de la reconstruction d'un objet mécanique simple formé de 8 faces présentant des faces planes et des faces cylindriques ainsi que le modèle déformé :

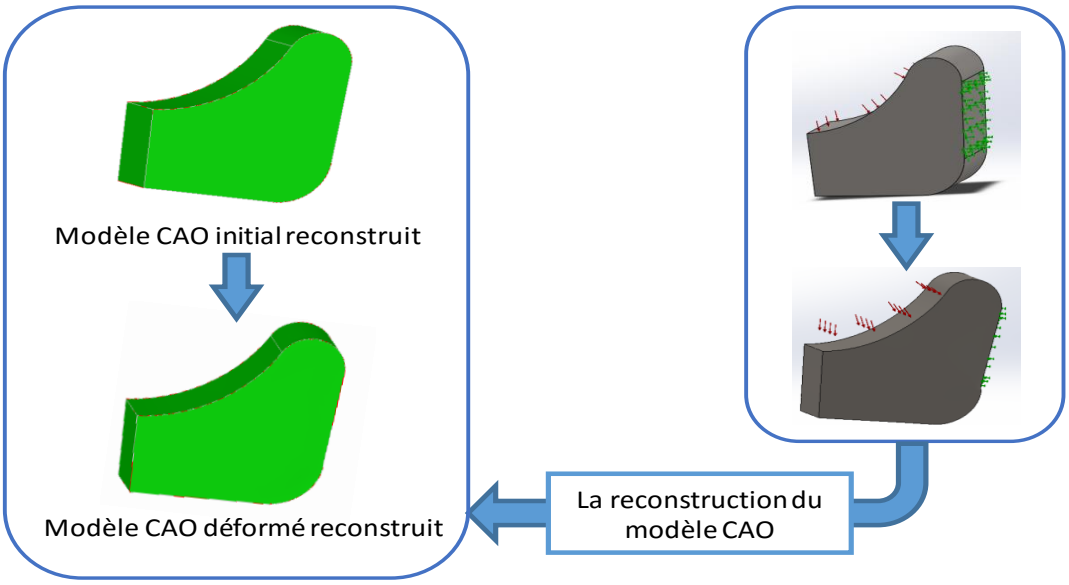


Figure 25: Reconstruction d'une première pièce mécanique déformée

La figure (Fig. 26) présente une pièce complexe contenant 14 faces dont 11 faces planes et une cylindrique. Ces surfaces deviennent des surfaces complexes (B-Splines) après la déformation :

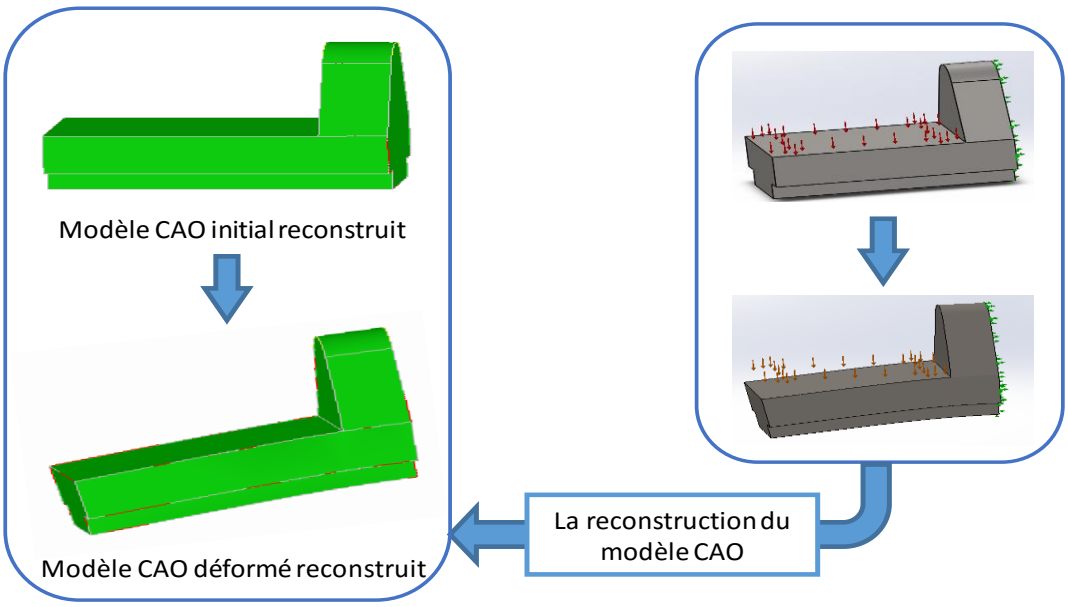


Figure 26: Reconstruction d'une deuxième pièce mécanique déformée

La figure (Fig. 27) présente une pièce mécanique contenant plus de surfaces cylindriques et des surfaces planes dont tous les surfaces se transforment en surfaces gauches (B-Splines dans notre cas) après déformation.

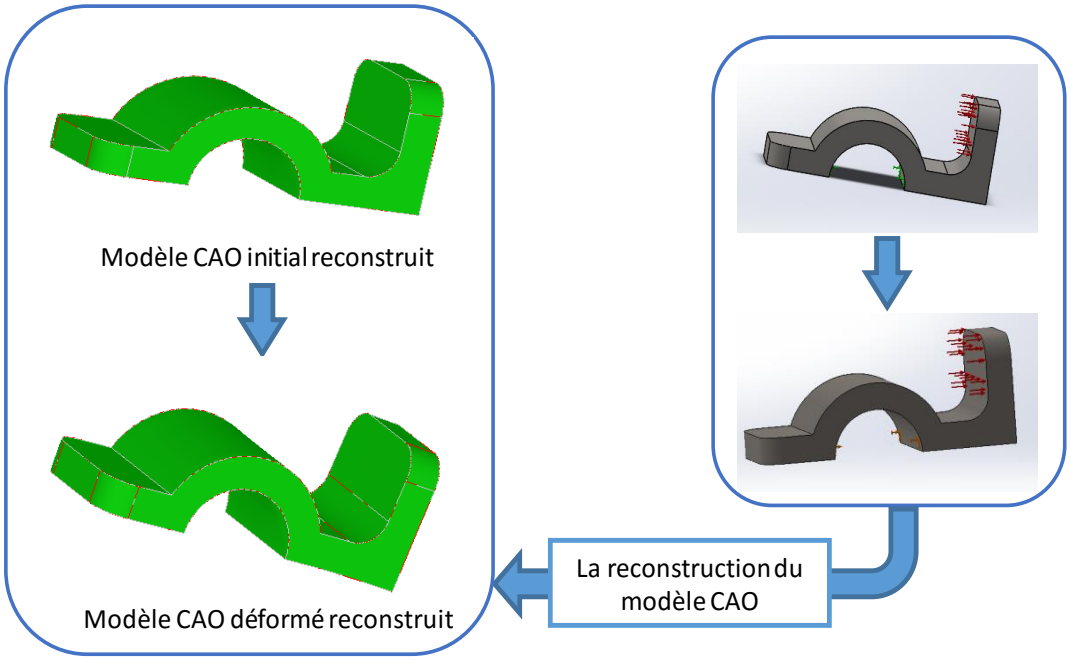


Figure 27: Reconstruction d'une troisième pièce mécanique déformée

La figure (Fig. 28) présente une pièce mécanique plus complexe contenant beaucoup de surfaces gauches.

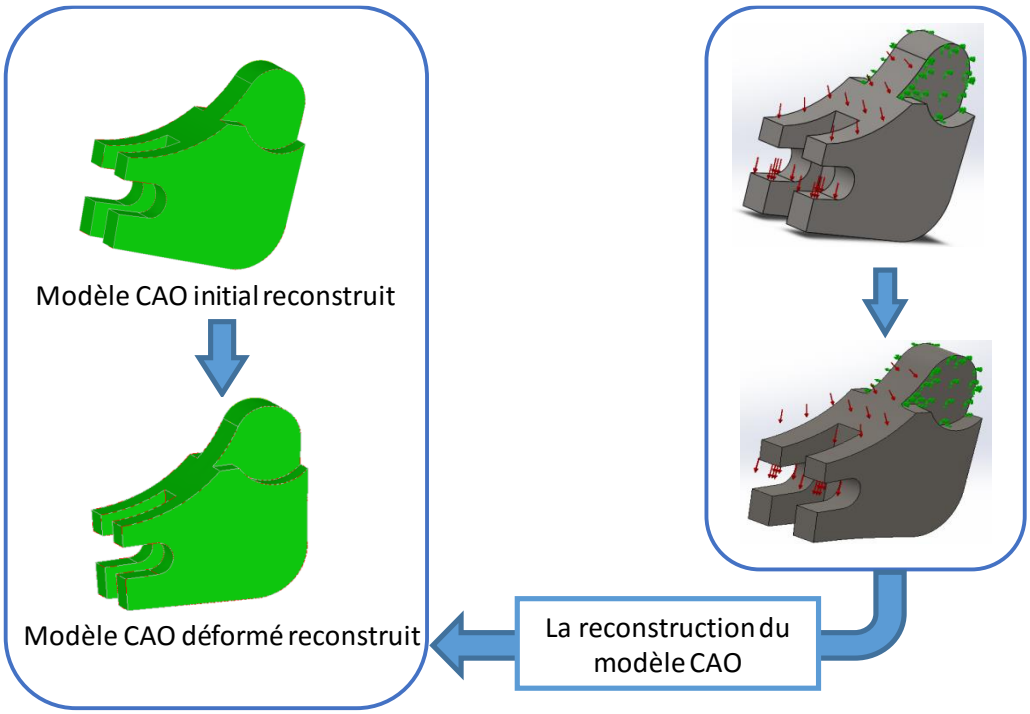


Figure 28: Reconstruction d'une quatrième pièce mécanique déformée

Conclusion

Dans ce dernier chapitre, nous avons donné une description de l'environnement logiciel de développement en présentant les technologies et les outils utilisés. Ensuite, nous avons présenté les résultats de reconstruction des surfaces : classiques (le cas de sphère, cylindre, tore et plan) ainsi complexes (B-Splines). Les résultats d'une courbe gauche sont aussi présentés. Bien évident, l'objectif final est de reconstruire un modèle au complet qui peut contenir différents entités géométriques et topologiques. Pour cette raison des tentatives de reconstruction d'un modèle CAO complet sont présentées dans le dernier paragraphe de ce chapitre.

Conclusion Générale

Dans ce rapport, la problématique de la reconstruction d'un modèle CAO cohérent à partir d'un nuage de points a été exposée. Cette problématique met en évidence la nécessité d'élaborer des méthodes pour reconstruire les différents types de surfaces et courbes nécessaires pour retrouver le modèle virtuel. En effet, un objet 3D est constitué d'un ensemble de courbes et surfaces. La complexité de la reconstruction se diffère suivant le type de l'entité. Concernant les surfaces, sont divisées en deux types : des surfaces classiques ou des primitives géométriques simples comme une sphère, un cylindre, un cube, un tore… et des surfaces complexes ou gauches qui sont définies par des équations polynômiales (surfaces de Bézier, surfaces B-Splines, NURBS…). Donc, nous avons proposé une méthode pour reconstruire les primitives géométriques à partir d'un ensemble de points en utilisant l'algorithme de Levenberg Marquardt qui minimise la distance entre ces points 3D et la géométrie définie par des paramètres optimisés à chaque itération. En fait, cet algorithme prend en entrée un vecteur qui contient des paramètres initiaux et après un certain nombre d'itérations, les paramètres finaux sont obtenus. Ces paramètres approximent les primitives géométriques ou les points de contrôle des courbes et des surfaces gauches comme nous l'avons montré pour les courbes et les surfaces B-Splines. En testant cet algorithme sur des cas réels des scans des primitives géométriques ensuite sur des courbes et surfaces B-Splines, nous avons montré sa robustesse et sa fiabilité.

Enfin, pour obtenir le modèle B-Rep final, on doit limier les surfaces reconstruites par les arêtes qui sont dans notre cas les courbes B-Splines. En combinant ces surfaces, nous obtenons le modèle CAO complet. Plusieurs recherches dans le domaine de la reconstruction tridimensionnelle, ont été présentées dans le premier chapitre afin de comparer nos travaux par rapport aux autres effectués dans ce domaine. Ensuite, dans le deuxième chapitre nous avons détaillé les différents types de surfaces qui constituent un objet 3D. Dans le troisième chapitre, nous avons présenté nos recherches réalisées et les algorithmes élaborés. Dans une première étape, l'algorithme général de la reconstruction d'un modèle CAO à partir d'un nuage de points 3D est présenté. Ensuite, nous avons expliqué l'algorithme de Levenberg Marquardt. Après, nous avons étudié quelques cas de primitives géométriques (sphère, plan, cylindre, tore) en mettant l'accent sur les paramètres qu'on doit passer à l'algorithme de Levenberg Marquardt pour chaque cas. Enfin, nous avons montré comment appliquer cet algorithme d'optimisation pour approximer les courbes et les surfaces B-Splines. Dans le dernier chapitre, nous avons justifié notre choix d'utiliser l'outil Open Cascade qui nécessite une connaissance du langage C++ et nous avons présenté ensuite les résultats obtenus.

Plusieurs perspectives peuvent être tirées de ce travail. En effet, nous visons à automatiser la reconstruction des courbes et surfaces B-Splines pour qu'on pourra optimiser, appart les points de contrôle, le degré et les vecteurs de nœuds. Aussi, nous allons travailler sur la reconstruction des surfaces lorsque les points numérisés ne sont pas organisés en trouvant les paramètres (u, v) de chaque point dans la surface. Sans oublier que dans la dernière partie du quatrième chapitre, nous avons présenté les résultats de la reconstruction de quelques objets mécaniques contenant des surfaces B-Splines déformées. Néanmoins, la reconstruction de ces derniers sera plus compliquée si les points obtenus sont non organisés, le contraire de ce que nous avons testé. Pour cela, plusieurs chercheurs ont expliqué ce problème en proposant des méthodes de reconstruction des objets 3D à partir de leurs maillages comme celle de Louhichi et al. [14]. Pour le cas de la reconstruction de cet objet déformé à partir d'un nuage de points directement, on pourra développer nos méthodes pour obtenir un objet reconstruit cohérent.

Enfin, le sujet étudié dans ce travail a un grand impact dans plusieurs domaines comme la mécanique, le médical, l'artistique et l'aéronautique… Pour cette raison, nous espérons que les contributions présentées ici seront précieuses pour les futures recherches dans le domaine.

References

[1] https://fr.wikipedia.org/wiki/Conception_assist%C3%A9e_par_ordinateur

[2] A. Hornung and L. Kobbelt, "Robust reconstruction of watertight 3d models from nonuniformly sampled point clouds without normal information," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006.

[3] S. Austin, R. Jerard and R. Drysdale, "Comparison of discretization algorithms for nurbs surfaces with application to numerically controlled maching," *Computer-Aided Design*, vol. 29, no. 1, p. 71–83, 1997.

[4] D. Shikhare, S. Gopalsamy, A. Reddy T.S., Patgawkar, S. Mahapatra, S.P. Mudur, K.P. Singh, I.Narayanswamy and L.Ravishankar, "Zeus: Surface modeling, surface grid generation,tetrahedral volume discretization," *Computers & Graphics*, vol. 23, no. 1, pp. 59-72, 1999.

[5] O. Volpin, A. Sheffer, M. Bercovier and L. Joskowicz, "Mesh simplification with smooth surface reconstruction," *Computer-Aided Design*, vol. 30, no. 11, pp. 875-882, 1998.

[6] B. Ren and I. Hagiwara, "Composite freeform surface reconstruction using recursive interpolating subdivision scheme," *Computers in Industry*, vol. 50, no. 3, pp. 265-275, 2003.

[7] X. Yang, "Surface interpolation of meshes by geometric subdivision," *Computer-Aided Design*, vol. 37, no. 5, pp. 497-508, 2005.

[8] Weiyin Ma and J P Kruth , "Parameterization of randomly measured points for least squares fitting of B-Spline curves and surfaces," *Computer Aided Design*, vol. 27, no. 9 , pp. 663-675, 1995.

[9] D. Rypl and Z. Bittnar, "Triangulation of 3D surfaces reconstructed by interpolating subdivision," *Computers & Structures*, vol. 82, no. 23-26, pp. 2093-2103, 2004.

[10] D. Walton and D. Meek, "A triangular G1 patch from boundary curves," *Computer-Aided Design*, vol. 28, no. 2, pp. 113-123, 1996.

[11] S. Owen and D. White, "International Journal for Numerical Methods in Engineering," *Mesh-based geometry*, vol. 58, no. 2, pp. 375-395, 2003.

[12] R. Bénière, G. Subsol, G. Gesquière, F. Le Breton and W. Puech, "A comprehensive process of reverse engineering from 3D meshes to CAD models," *Computer-Aided Design*, vol. 45, no. 11, pp. 1382-1393, 2013.

[13] A. Bey, R. Chaine, R. Marc, G. Thibault, "Reconstruction d'un nuage de points 3D étant donné un modèle CAO a priori," in *Reconnaissance des formes et intelligence artificielle*, Lyon, France, 2012.

[14] B. Louhichi, G. Abenhaim, A. Tahan, "CAD/CAE integration: updating the CAD model after a FEM analysis," *Int J Adv Manuf Technol*, vol. 76, no. 1-4, pp. 391-400, 2014.

[15] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE Trans. Visual. Comput. Graphics,* vol. 5, no. 4, p. 349–359, 1999.

[16] M. Kazhdan, M. Bolitho, H. Hoppe, "Poisson Surface Reconstruction," in *Symposium on Geometry Processing*, 2006.

[17] U. Dietz, "Erzeugung glatter Flachen aus MeBpunkten," Technical Report 1717, Department of Mathematics, University of Darmstadt, Germany, February 1995.

[18] J. Hoschek, F. Schneider and P. Wassum, "Optimal approximate conversion of spline surfaces," *Computer Aided Geometric Design,* vol. 6, no. 4, p. 293–306, 1989.

[19] D. Rogers and N. Fog , "Constrained B-spline curve and surface fitting," *Computer-Aided Design,* vol. 21, no. 10, p. 641–648, 1989.

[20] B. Sarkar and C. Menq, "Parameter optimization in approximating curves and surfaces to measurement data," *Computer Aided Geometric Design,* vol. 8, no. 4, p. 267–290, 1991.

[21] F. Schmitt, B. Barsky and W. Du, "An adaptive subdivision method for surface fitting from sampled data," *ACM SIGGRAPH Computer Graphics,* vol. 20, no. 4, p. 179–188, 1986.

[22] E. Andersson, R. Andersson, M. Boman, T. Elmroth, B. Dahlberg and B. Johansson, "Automatic construction of surfaces with prescribed shape," *Computer-Aided Design,* vol. 20, no. 6, p. 317–324, 1988.

[23] L. Fang , D. Gossard, "Reconstruction of smooth parametric surfaces from unorganized data points," in *Curves and Surfaces in Computer Vision and Graphics 3*, 1992.

[24] M. Milroy, C. Bradley, G. Vickers and D. Weir, "G1 continuity of B-spline surface patches in reverse engineering," *Computer-Aided Design,* vol. 27, no. 6, p. 471–478, 1995.

[25] W. Ma and J.P. Kruth, "Parametrization of randomly measured points for least squares fitting of B-spline curves and surfaces," *CAD,* vol. 27, p. 663–675, 663–675 1995.

[26] Eck, M. and Hoppe, H, "Automatic reconstruction of b-spline surfaces of arbitrary topological type," in *Computer Graphics (SIGGRAPH '96 Proceedings) (2nd edn.)*, vol. 30, 1996, pp. 325-334.

[27] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle, "Surface reconstruction from unorganized points," *ACM SIGGRAPH Computer Graphics,* vol. 26, no. 2, pp. 71--78, 1992.

[28] G. Fergusson, Journal of Geophysical Research, 68, 13, 1963.

[29] J. Nash, "Compact Numerical Methods for Computers," in *Linear Algebra and Function Minimisation*, Bristol, England, 1979.

[30] https://fr.wikipedia.org/wiki/NURBS

[31] L. Piegl, W. Tiller, "The NURBS Book," New York, 1997.

[32] D. Aquino, D. l Martins, J. Fonseca, and J.Mendes, "Three-Dimensional Surface Reconstruction Using NURBS," in *20th International Congress of Mechanical Engineering*,, Gramado-RS, Brazil, November 15-20, 2009.

[33] E. Mortenson, "Geometric Modeling," in *Wiley*, New York, 1985.

[34] C. Shakarji, "Least-squares fitting algorithms of the NIST algorithm testing system," *J. Res. Natl. Inst. Stand. Technol.*, vol. 103, no. 6, p. 633, 1998.

[35] https://fr.wikipedia.org/wiki/Algorithme_de_Levenberg-Marquardt

[36] J. More, "The Levenberg-Marquardt Algorithm: Implementation and Theory," in *Numerical Analysis: Proc. Biennial Conf on Numerical Analysis*, 1977.

[37] G. Golub and C. van Loan, "Matrix Computations," in *The Johns Hopkins University Press*, Baltimore, 1983.

[38] V. Weiss, L. Andor, G. Renner, T. Várady, "Advanced surface fitting techniques," *Computer Aided Geometric Design*, vol. 19, no. 1, pp. 19-42, January 2002.

[39] M. Aigner and B. Jüttler, "Gauss-Newton-type Techniques for Robustly Fitting Implicitly Defined Curves and Surfaces to Unorganized Data Points".

[40] J. Carr, R. Beatson, J. Cherrie, T. Mitchell, W. Fright, B. McCallum, "Reconstruction and Representation of 3D Objects with Radial Basis Functions," in *SIGGRAPH '01 Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 2001.

[41] X. Chen, G. Xu, J. Yong, G. Wang, J. Paul, "Computing the minimum distance between a point and a clamped B-spline surface," *Graphical Models*, vol. 71, no. 3, pp. 107-112, 2009.

[42] http://www.infogoaround.org/JBook/LeastSquaresApprx.pdf