UNIVERSITE DE HAUTE-ALSACE
École Doctorale "Mathématiques, Sciences de l'Information et de l'Ingénieur" (ED 269)

Modélisation géométrique à partir de croquis

par

Nicolas Chérin

présentée en vue d'obtenir le grade de Docteur,
spécialité informatique

Remerciements

Je tiens à remercier mon directeur de thèse Mahmoud Melkemi pour m'avoir permis d'effectuer ce travail de thèse au sein du LMIA de l'Université de Haute Alsace. Je voudrais ensuite remercier Frédéric Cordier pour l'aide ainsi que les conseils qu'il m'a apportée tout au long de ces années.

J'exprime toute ma gratitude aux chercheurs qui ont bien voulu prendre le rôle de rapporteur de mon mémoire : Anne Verroust-Blondeti et Laurent Grisoni, ainsi qu'aux chercheurs qui ont rejoint le jury : Youssef Chahir et Gilles Gesquière.

Et enfin je voudrais remercier les membres de ma famille ainsi que mes amis pour leurs soutiens.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Présentation du sujet et motivations

Dans ce mémoire de thèse, je présente les différents travaux de recherche que j'ai effectués au sein du LMIA [1] de l'Université de Haute Alsace. Dans ce premier chapitre, je décrirai en quoi consiste la modélisation géométrique ainsi que la modélisation par croquis. Je présenterai ensuite les intérêts qu'il pourrait y avoir à utiliser la technique de la modélisation par croquis.

1.1.1 Modélisation géométrique

La technique de la modélisation géométrique consiste à créer une représentation mathématique de modèles tridimensionnels. C'est une technique très importante de l'informatique graphique. Elle est utilisée par de nombreuses industries comme la modélisation des personnages pour les jeux vidéo, les effets spéciaux pour le cinéma, le dessin industriel, les simulateurs pour les opérations chirurgicales, etc.

L'exercice de la modélisation nécessite l'utilisation de logiciel de modélisation comme Maya, 3DSMax, Blender, etc. (voir figure 1.1).

Ces logiciels sont très complexes et demandent un savoir-faire important. Par exemple, la modélisation des personnages pour les jeux vidéo peut demander plusieurs jours de travail. En effet la modélisation de personnage consiste en général à créer un premier modèle possédant un faible nombre de polygones (par exemple avec 3DSMax) puis de l'affiner (modèle haute résolution pouvant aller jusqu'a plusieurs millions de polygones)

1. Laboratoire Mathématiques Informatiques et Application

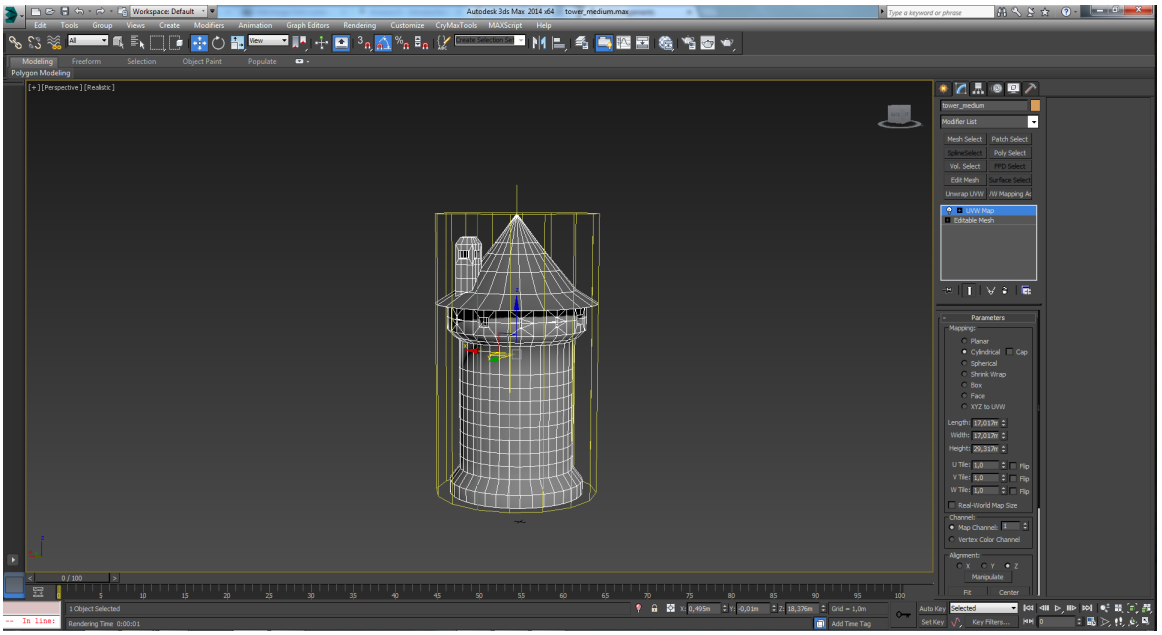

FIGURE 1.1 : Modélisation d'un objet à l'aide du logiciel 3DSMax

avec un logiciel de sculpture 3D comme *ZBrush*. L'étape finale étant d'appliquer des textures sur le modèle 3D.

La principale raison de la difficulté du travail de modélisation est due au fait que les artistes disposent d'outils qui fonctionnent en deux dimensions (souris, tablettes, etc.) pour créer des modèles en trois dimensions.

1.1.2 Modélisation par croquis

1.1.2.1 Qu'est-ce que la modélisation par croquis ?

La modélisation à l'aide de croquis a pour but de construire une forme tridimensionnelle à partir d'un dessin en deux dimensions. L'utilisateur dessine la silhouette de l'objet à reconstruire sur le plan de *dessin* ($z = 0$). Cette silhouette est la projection orthogonale des courbes formant la silhouette 3D sur le plan ($z = 0$) (voir figure 1.2).

Dessiner un croquis est assez simple, en effet la plupart des personnes sont capables de dessiner en très peu de temps des croquis de formes tridimensionnelles. C'est une façon naturelle de communiquer des idées rapidement. En effet, lorsque nous regardons un croquis nous sommes capables d'interpréter et de construire une représentation mentale de la forme dessinée. Pour être capable de créer des algorithmes qui permettent à l'ordinateur de créer une forme tridimensionnelle à partir d'un dessin, il peut être intéressant de comprendre comment le cerveau humain les interprète (voir section suivante). Dans

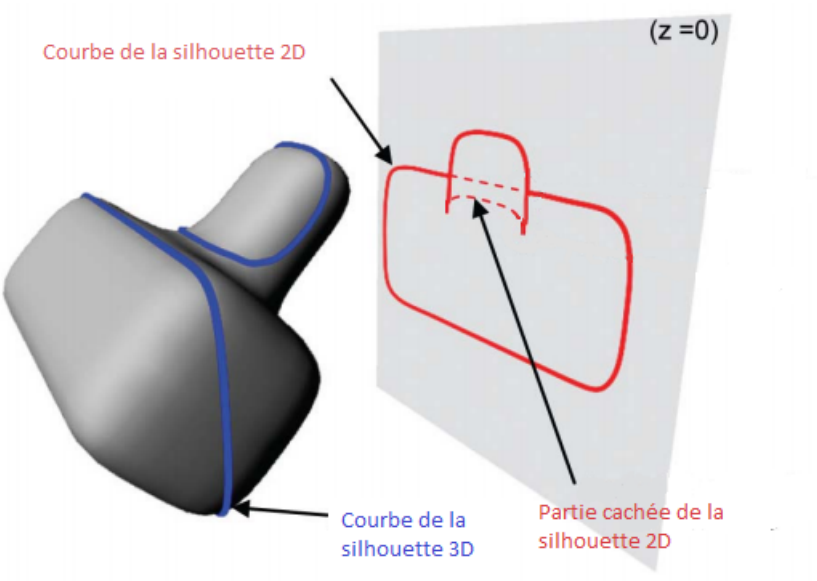

FIGURE 1.2 : Silhouette d'une forme 3D et sa projection orthogonale [1]

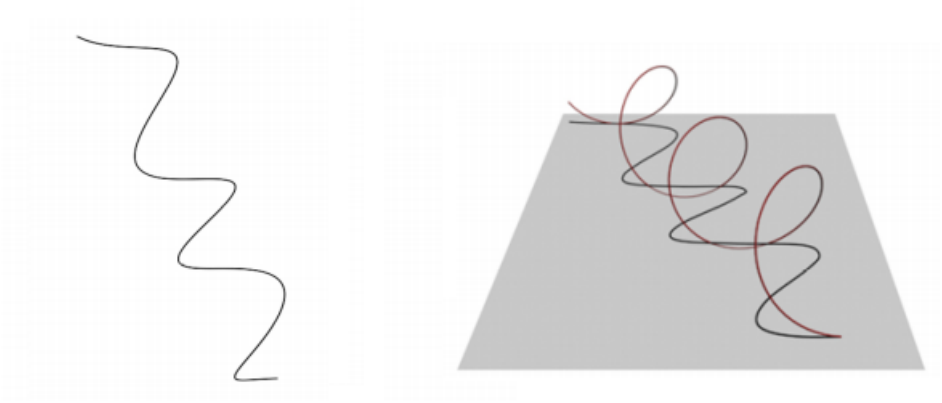

FIGURE 1.3 : À gauche le dessin en deux dimensions donné à l'algorithme par l'utilisateur et à droite la courbe tridimensionnelle générée par mon algorithme [2]

ce mémoire je m'intéresserai dans un premier temps à la construction de courbes tridimensionnelles à partir de croquis. Par exemple, la figure 1.3 montre la reconstruction d'une courbe 3D à l'aide d'hélices. Dans un second temps, j'aborderai le problème de la reconstruction de surfaces à partir de croquis. La figure 1.4 est un exemple d'une reconstruction d'une forme 3D (à droite) à partir du croquis (à gauche).

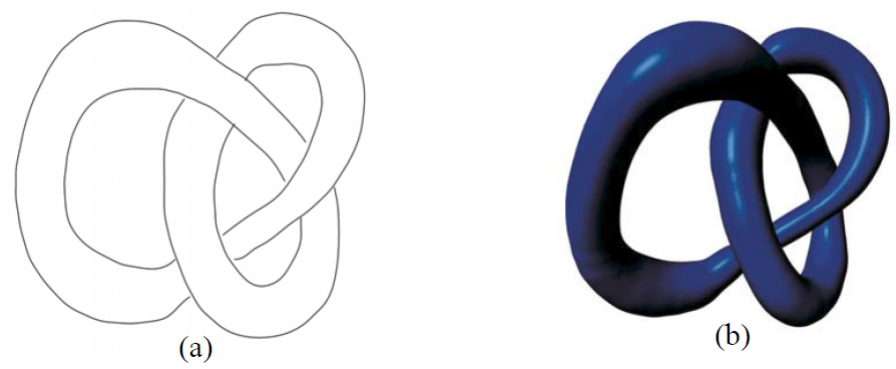

FIGURE 1.4 : Croquis (a) et sa reconstruction (b) (image extraite de [3])

1.1.2.2 Comment interprétons-nous un dessin ?

La difficulté majeure de la modélisation par croquis réside dans le fait qu'il y a une infinité de solutions possibles pour la reconstruction d'une forme tridimensionnelle à partir d'un croquis . En effet le dessin du cube (figure 1.5) peut être interprété d'une infinité de manières. Pourquoi interprète-t-on la figure 1.5 plus volontiers comme un cube que comme un des autres choix possibles ?

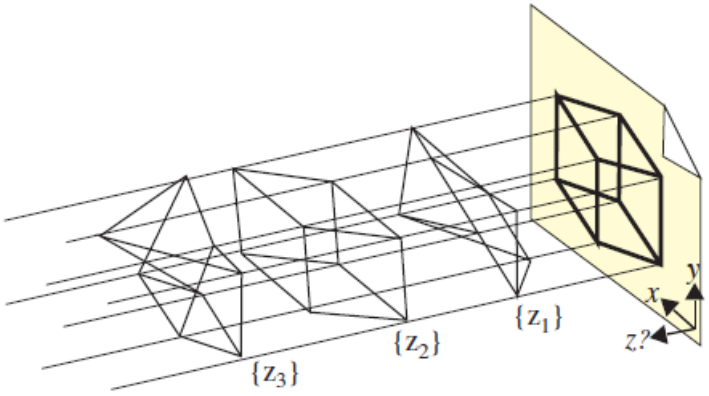

FIGURE 1.5 : Il existe de nombreuses interprétations possibles du dessin 2D (reproduit de [4])

Lorsque l'on regarde l'interprétation z_2, des changements du point de vue ne modifient pas la nature de l'objet, cela reste un cube. Alors que pour les autres solutions , un petit changement du point de vue suffit à révéler que ces interprétations ne sont pas des cubes. Hoffmann [5] appelle ces interprétations *vues accidentelles*, ces *vues accidentelles* sont dites *non stables*. Pour autant, la majorité des points de vue sont stables sous des changements mineurs . Notre système visuel favorise les interprétations des objets *stables*, les objets les plus simples.

Comprendre comment une personne perçoit les formes nous permet de comprendre aussi comment elle dessine. En effet lorsqu'on dessine un objet, il est peu probable de le représenter d'un point de vue accidentel. Cette règle associée avec notre mémoire nous permet de comprendre des images que nous n'avons jamais vues avant. Par exemple si on nous montre une image ou seulement une silhouette d'un autobus nous sommes capables de visualiser cet objet et de déduire approximativement ses caractéristiques (géométrie, échelle, etc.).

Pour diminuer la difficulté de la reconstruction de formes 3D à partir de croquis il est possible de poser des contraintes sur le modèle à reconstruire, par exemple [6] propose un modèle qui reconstruit une forme 3D en utilisant quatre contraintes différentes : *la symétrie*, la *planarité*, la *compacité* et la *surface minimale*. Comme nous le verrons par

la suite une autre approche est de se restreindre les types de formes que l'on reconstruit, par exemple créer une méthode spécialisée dans la création d'arbres à partir de dessins.

1.1.2.3 Différentes techniques de modélisation par croquis

Afin de résoudre ce problème, les chercheurs ont développé de nombreuses méthodes que l'on peut grouper sous différentes catégories :

- *Modélisation de formes constituées d'arêtes* : La plupart de ces méthodes traitent des croquis constitués de segments de droites et courbes 3D planaires. l'objectif est de calculer une forme en fil de fer.
- *Modélisation de formes spécifiques* : Étant donné qu'il est très difficile de proposer une méthode capable de reconstruire n'importe quels types de formes, certaines personnes ont développé des méthodes spécialisées dans la reconstruction d'un certain type d'objets (par exemple les cheveux, les vêtements, les arbres, etc.)
- *Modélisation à l'aide de base de données* : Il est possible d'utiliser une base de donnée contenant des modèles (*templates*) de formes.
- *Modélisation itérative* : D'autres méthodes proposent à l'utilisateur de construire une forme 3D en plusieurs étapes, l'utilisateur dispose d'outils qui lui permettent d'ajouter, de modifier ou de supprimer des éléments de la forme à reconstruire.
- *Modélisation à l'aide de plusieurs vues* : La modélisation à partir d'un seul croquis étant difficile à résoudre, certains chercheurs ont travaillé sur la reconstruction à partir de deux ou plusieurs croquis.

1.1.2.4 Intérêts de la modélisation par croquis

Les applications liées à la modélisation par croquis seraient nombreuses :

- Dans le domaine de l'infographie, la modélisation géométrique pour les jeux vidéo, le dessin industriel, les effets spéciaux ... etc. serait plus rapide et moins coûteuse
- la modélisation à l'aide de croquis pourrait être utilisée par tout le monde. Nous avons tous, un jour ou l'autre, fait un croquis pour expliquer le chemin à prendre pour aller à un lieu précis, un croquis pour l'agencement d'une cuisine et d'une salle de séjour, ou croquis pour expliquer le fonctionnement d'une machine, etc.

1.2 Plan du mémoire

Le mémoire est organisé de la façon suivante : le chapitre 2 présente l'état de l'art. Je décris ensuite dans les chapitres 3 et 4 deux algorithmes permettant de reconstruire une

courbe tridimensionnelle à partir d'un croquis. Le chapitre 5 porte sur la reconstruction de surfaces sous forme de bas-reliefs à partir de croquis.

Chapitre 2

État de l'art

Dans ce chapitre, nous passons en revue différents types de méthodes qui existent pour résoudre le problème de la modélisation par croquis.

2.1 Perception du dessin

Le système visuel humain est complexe pourtant il fonctionne sans effort de notre part. En effet nous sommes capables de construire instantanément une représentation mentale en 3D de la plupart des images que nous regardons. Comme je le mentionnais dans l'introduction, il existe une infinité d'interprétations pour une image donnée.

Reprenons l'image de notre cube en deux dimensions (figure 2.1), pourquoi interprète-t-on plutôt cette image comme un cube que comme un des nombreux autres choix possibles ? Pour comprendre cela, il faut introduire les notions de règles visuelles. Hoffmann dans [5] dresse un certain nombre de règles qui sont utilisées par notre système visuel pour interpréter un dessin.

La première règle importante est la règle de *vue générique*.

2.1.1 Vue générique

Une vue générique est une vue d'un objet qui est dit stable, c'est-à-dire que changer légèrement le point de vue , ne change pas l'interprétation que l'on a de l'objet. Par exemple la vue du cube de la figure 2.2 n'est pas générique, il suffit de changer un peu le point de vue (voir figure 2.3) pour que les six segments de droites ne se joignent plus, c'est une vue accidentelle.

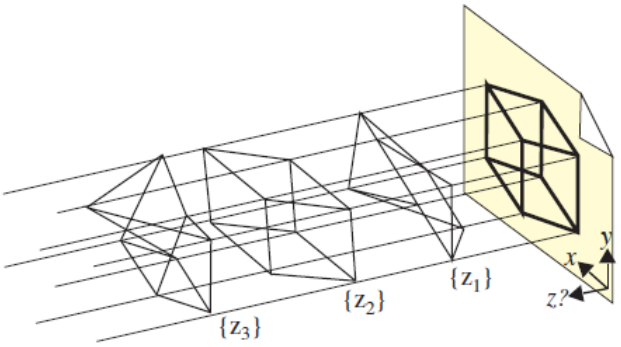

FIGURE 2.1 : Il existe de nombreuses interprétations possibles du dessin 2D (reproduit de [4])

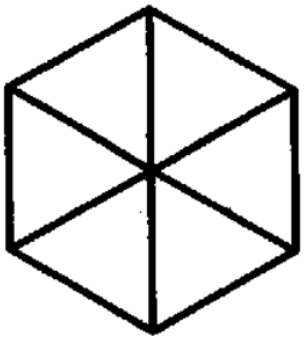

FIGURE 2.2 : Cube de Kopfermann [4]

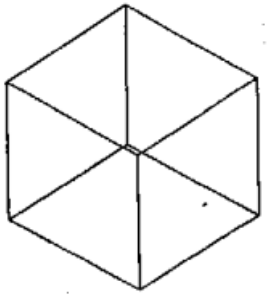

FIGURE 2.3 : Vue générique

2.1.2 Règles visuelles

Basées sur ce ce principe de vue générique/ accidentelle, [5] dresse un certain nombre de règles qui permettent de comprendre comment on interprète un dessin. Par exemple les deux premières règles visuelles sont :

— Règle 1 : Toujours interpréter une ligne droite dans une image comme une ligne droite en 3D.

— Règle 2 : Si les extrémités de deux lignes se touchent dans une image, alors les interpréter de façon à ce qu'elles coïncident aussi en 3D.

La première règle peut nous aider à comprendre pourquoi il est difficile d'interpréter la figure 2.2 comme un cube, en effet selon celle-ci les lignes dans la figure 2.2 doivent être interprétées comme des lignes droites ce qui empêche d'interpréter la figure comme un cube.

2.2 Acquisition et filtrage du dessin

2.2.1 Acquisition

La première étape dans la plupart des méthodes de modélisation à partir de croquis est l'acquisition du dessin. Utiliser la souris pour dessiner est le moyen le plus simple à mettre en oeuvre, mais il ne donne pas les mêmes sensations que le dessin avec un crayon sur du papier. En effet, dessiner avec un crayon et du papier est un moyen de communication qui est assez riche. Un artiste peut transmettre des informations non seulement avec la forme générale du dessin, mais aussi en variant la pression sur le crayon et le style de traits. La texture du papier est aussi importante, car elle permet de fournir des sensations à l'artiste.

Des travaux ont été effectués afin de transférer ces aspects dans le domaine digital. En effet, beaucoup de palettes graphiques sont par exemple sensibles à la pression et ne fournissent donc pas uniquement la position de stylet. Les dispositifs Haptic [7] fournissent des réactions à l'utilisateur à travers le dispositif utilisé pour dessiner. Par exemple des vibrations basses fréquences afin de simuler la friction entre le crayon et le papier.

Une autre solution pour acquérir le dessin est de scanner celui-ci. Cela peu fonctionner pour certain domaine, par exemple scanner des dessins d'architecture. Mais cette approche est très difficile et manque de solutions robustes. En général nous préférons des systèmes interactifs qui fournissent plus d'informations à l'application et des réponses à l'utilisateur.

L'étape suivante est le filtrage du dessin pour réduire les éventuelles imprécisions de l'utilisateur et les erreurs liées au matériel. Je vais présenter quelques méthodes de filtrage fréquemment utilisées.

2.2.2 Filtrage

2.2.2.1 Ajustement

Un dessin peut contenir un nombre important de points sans grandes significations. Ajuster le dessin à une autre représentation peut avoir l'avantage de le simplifier et de permettre de le comparer à d'autres dessins. Par exemple il est possible de simplifier les traits du dessin par des courbes. [8] et [9] utilisent des courbes de Bezier , [10] et [11] utilisent des B-splines. L'utilisation de courbes à l'avantage de fournir peu d'erreurs d'approximation au détriment du temps de calcul.

Beaucoup de dessins contiennent des sections linéaires et des sections courbées, des méthodes ont été développées afin d'approximer le dessin par des segments des droites et par des courbes douces : [12] [9] [13] [14] . Par exemple la méthode développée par [12] utilise les données de courbure et de vitesse afin d'approximer au mieux le dessin.

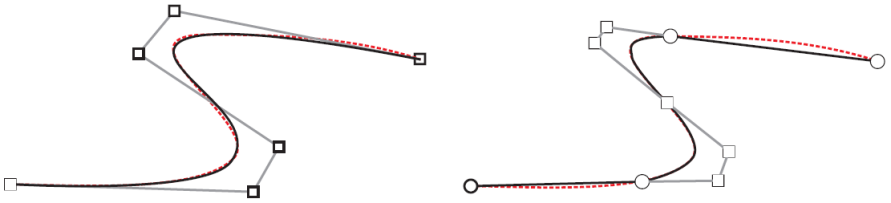

FIGURE 2.4 : Cette figure montre l'ajustement à des courbes. À gauche B-splines, à droite courbes de Bezier (figure extraite de [15])

2.2.2.2 Correction du dessin et traitements de dessin hachuré

L'ajustement à une autre représentation est utile pour les applications où la précision est importante. En revanche pour les applications qui font peu de suppositions sur les intentions de l'utilisateur, ajuster le dessin à une autre représentation peut détruire des parties importantes du dessin. Dans ce cas l'utilisateur doit pouvoir dessiner exactement ce qu'il veut et avoir la possibilité de corriger lui même ses erreurs.

Cette technique permet à l'utilisateur de dessiner de nouveaux traits par-dessus le dessin existant, le système utilise alors ces nouveaux traits pour corriger le dessin (voir figure 2.5).

Les méthodes suivantes [12] [16] [17] utilisent cette méthode avant d'interpréter le dessin. Il existe un autre type *d'oversketching* utilisé par les artistes ou le dessin est constitué

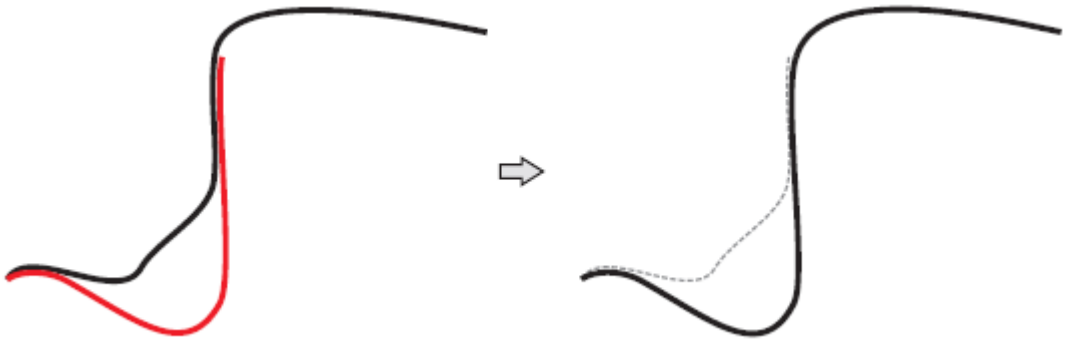

FIGURE 2.5 : Le trait en rouge est la correction du dessin par l'utilisateur (figure extraite de [15])

de traits courts qui se chevauchent et forme un seul objet (voir figure 2.6). Il existe des systèmes [18] [19] [20] [21] qui autorisent ce type de dessin et qui automatiquement mélangent ces traits pour en former un seul.

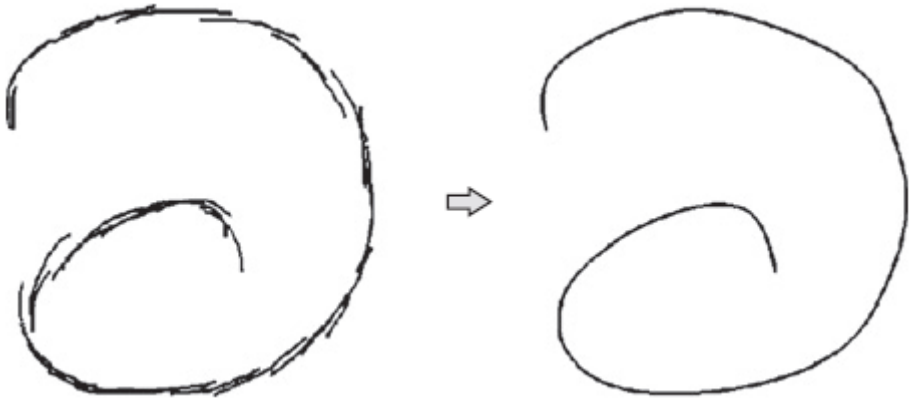

FIGURE 2.6 : Les traits qui forment la figure de gauche sont mélangés en un seul pour former celle de droite (figure extraite de [15])

2.3 Modélisation de formes constituées d'arêtes

Dans les années 70, David Huffman et Maxwell Clowes (entre autres) ont commencé à apprendre à un ordinateur à reconstruire des scènes 3D à partir d'images de polyèdres (voir figure 2.7 [1]).

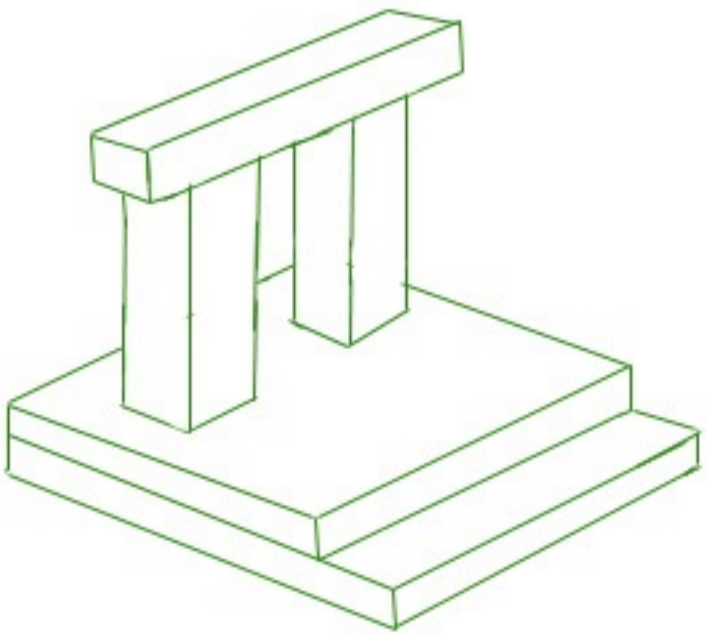

FIGURE 2.7 : Exemple d'image constituée de polyèdres

Pour analyser ces images Huffman dans [22] propose un ensemble de règles basées sur l'étiquetage des lignes et des sommets composant le croquis. Ces règles permettent à l'ordinateur de construire la structure de la forme dessinée sur le croquis. En particulier, elles permettent de définir les arêtes et les sommets qui sont convexes et concaves. Les règles de base sont les suivantes :

- Un "+" représente une arête convexe dont les deux faces correspondantes sont visibles
- Un "-" représente une arête concave dont les deux faces correspondantes sont visibles
- Une flèche est utilisée pour représenter une arête dont une des deux faces correspondantes est cachée. La face visible se trouve à droite de la flèche

La figure 2.8 montre un exemple d'étiquetage de lignes (*line labeling* en anglais).

1. http ://cs.stanford.edu/people/eroberts/courses/soco/projects/1997-98/computer-vision/linelabeling.html

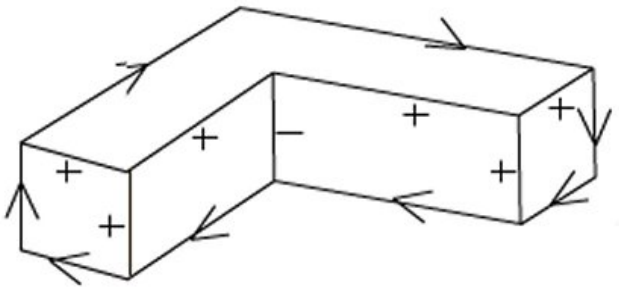

FIGURE 2.8 : Exemple d'étiquetage

Ce travail a été ensuite repris et développé par [23]. L'application principale des premiers travaux apparus est la conception assistée par ordinateur (CAO). Dans la plupart des cas, l'utilisateur fournit un croquis constitué essentiellement de segments de droite. L'objectif étant de calculer une forme en fils de fer dont la projection orthogonale ou en perspective correspond au croquis.

Pour la plupart des méthodes apparues après les années 90, la reconstruction est faite à l'aide d'une optimisation avec fonction coût dont il faut calculer le minimum.

Par exemple dans le travail de [24], la fonction coût est calculée de façon à optimiser un certain nombre de critères comme l'orthogonalité et la planarité des faces, le parallélisme des lignes, etc. La figure 2.9 montre les résultats obtenus pour différents croquis.

Une grande partie des travaux [25], [4], [26], qui suivent sont basés sur le même principe. La principale différence est la formulation de la fonction coût utilisée pour la reconstruction. Par exemple, dans l'article de [26] les variables de la fonction objectif sont les paramètres des plans qui passent à travers les faces du modèle à reconstruire. Ces méthodes sont bien adaptées pour les surfaces composées d'un faible nombre de faces polygonales comme les polyèdres par exemple

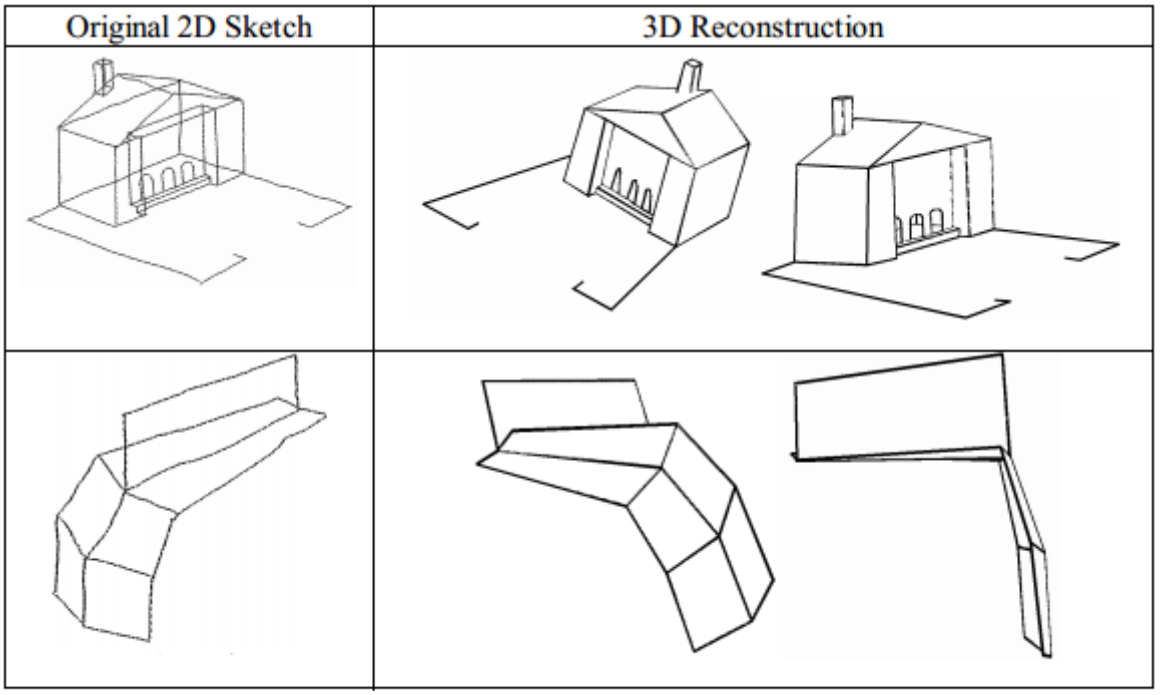

FIGURE 2.9 : Résultats de la méthode[24]

2.4 Modélisation de formes spécifiques

Constatant qu'il est très difficile de proposer une méthode capable de modéliser n'importe quelle forme, plusieurs chercheurs ont travaillé sur des méthodes pour la modélisation de formes spécifiques. L'idée principale de cette classe de méthodes est d'utiliser un certain nombre d'hypothèses sur la forme à reconstruire. Ces hypothèses sont souvent très fortes ; elles limitent les formes qui peuvent être reconstruites, mais elles permettent de simplifier grandement le problème.

2.4.1 Modélisation de plantes et arbres

En informatique graphique la modélisation d'objets naturels tels que les arbres représente un défi important au vu de leurs complexités, la forme des arbres étant très particulière il est naturel que certains chercheurs développent des méthodes spécifiques pour ce type de formes.

Dans le travail d'Okabe [27] le système reconstruit un arbre en 3D à partir d'un dessin en faisant la supposition que la distance entre les branches est la plus grande possible. La figure suivante (2.10) montre trois exemples de reconstructions.

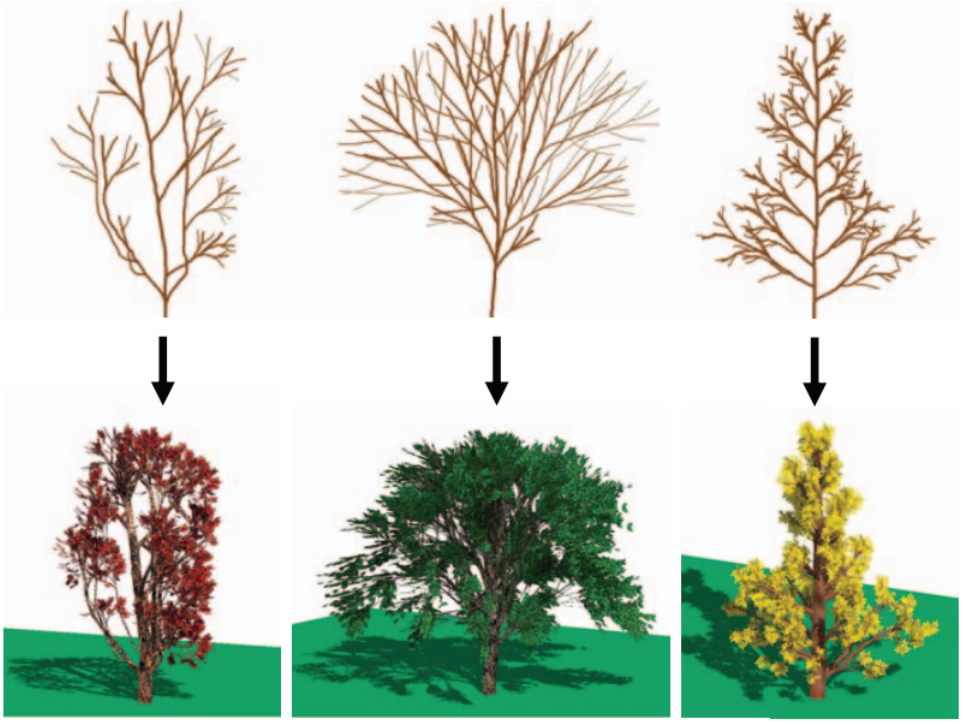

FIGURE 2.10 : Exemples de reconstructions [27]

Les auteurs de [28] ont développé un système permettant à l'utilisateur de créer des arbres à partir de dessins de branches et en utilisant une base de données contenant des modèles d'arbres. Cela rejoint comme on le verra ultérieurement les méthodes de modélisation à partir de bases de données. Dans cette méthode la base de données contient un certain nombre d'arbres modèles et leurs paramètres. La première étape consiste à reconstruire les branches dessinées par l'utilisateur à l'aide de l'arbre modèle le plus proche présent dans la base de données. Étant donné que l'arbre dessiné est souvent épars, l'étape suivante consiste à complexifier l'arbre (étape de propagation). La dernière étape consiste à construire les feuilles de l'arbre.

FIGURE 2.11 : Reconstruction d'un arbre à partir d'un dessin, sur cette figure (extraite de [28]) le dessin se trouve à gauche, puis se trouve le résultat de la reconstruction des branches suivi de l'étape de propagation. Les deux dernières images montrent le modèle 3D final sous deux vues différentes

[29] ont présenté un système qui permet de construire le modèle d'un arbre à partir des croquis de la silhouette de l'arbre à différentes échelles (voir figure 2.12).

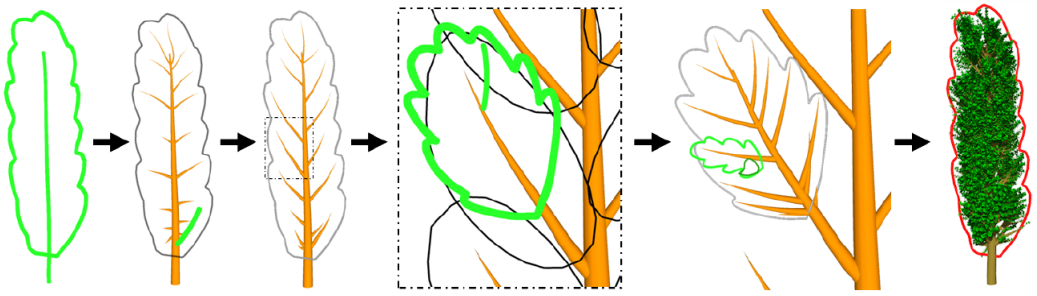

FIGURE 2.12 : Creation d'un arbre en dessinant (traits verts sur la figure) successivement les silhouettes du feuillage des arbres à différentes échelles (de l'arbre complet à une feuille). [29]

Dans le travail de [30], la méthode de modélisation des arbres est basée sur un modèle procédural (figure 2.13). Les auteurs ont développé un système procédural permettant la génération d'arbres complexes. La génération procédurale de l'arbre repose sur un système de compétition entre les branches pour l'espace et l'accès à la lumière. Ce système de compétition est mis en jeu pendant tout le processus de croissance de l'arbre.

L'utilisateur peut intervenir sur ce processus, en effet il peut contrôler la direction de croissance de l'arbre, changer les paramètres de croissance et éditer les formes générées.

FIGURE 2.13 : Méthode procédurale pour générer des arbres

2.4.2 Modélisation de cheveux

De la même façon qu'il existe des méthodes dédiées à la modélisation de plantes et d'arbres, certains chercheurs ont aussi développé des méthodes permettant la modélisation de cheveux à partir de croquis. Dans ce cas-ci, les cheveux sont modélisés à l'aide de courbes tridimensionnelles. Par exemple dans l'article de [31], ils sont modélisés par des cylindres généralisés.

Dans celui de [32], ils sont représentés par des hélices circulaires (figure 2.14), l'utilisateur dessine des brins de cheveux sur une vue du côté de la tête du personnage puis à partir de ces dessins leur méthode génère automatiquement les cheveux en trois dimensions. Contrairement à la méthode que nous avons développée dans [2] l'orientation des hélices reste toujours la même.

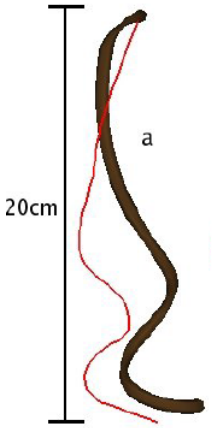

FIGURE 2.14 : Modélisation de brin de cheveux à l'aide d'hélices circulaire

Dans l'article de [33], la forme des cheveux est calculée à l'aide d'un champ vectoriel. Dans cette méthode l'utilisateur manipule des courbes 3D à l'aide de traits, ces courbes servent de contraintes pour la construction des cheveux. La figure 2.15 montre un exemple de style de cheveux construit à l'aide de cette méthode.

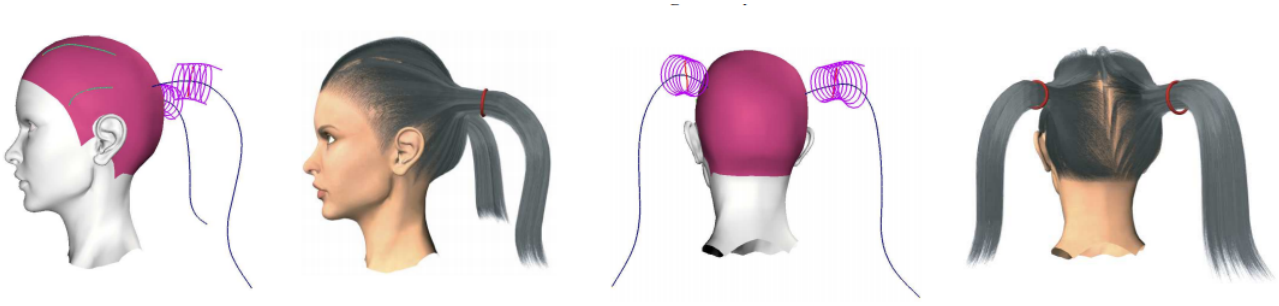

FIGURE 2.15 : Exemple de coupe de cheveux crée avec le système de [33]

2.4.3 Modélisation de vêtements

Il existe de nombreux algorithmes qui permettent de modéliser des vêtements pour des personnages virtuels à partir de croquis. Ces méthodes spécialisées pour les vêtements ont un certain nombre de points communs. La forme à reconstruire est une surface polygonale. Cette surface est construite à partir de la forme du corps sur lequel vient se poser le vêtement. La plupart de ces méthodes proposent de construire le vêtement de façon itérative, c'est à dire, en plusieurs étapes en utilisant différents widgets qui permettent de dessiner les bords de l'habit, les coutures, les plis, etc.

Selon [34], l'habillage de personnage virtuel pose trois problèmes :

- Le design du vêtement
- Placer le vêtement sur le personnage
- Faire en sorte que le vêtement semble correct d'un point de vue physique

[35] ont été parmi les premiers à introduire des méthodes qui permettent la modélisation de vêtement pour des personnages virtuels à partir de croquis. Dans ce système l'utilisateur dessine directement des lignes sur un mannequin virtuel en 3D (voir figure 2.16). Le dessin est utilisé pour un créer un vêtement qui est conforme avec la forme du mannequin. Ce système produit de bons résultats pour des dessins de vêtement simple, mais perd en réalisme pour des vêtements plus complexes.

FIGURE 2.16 : Exemple de résultat obtenu avec la méthode de [35]

Les auteurs de [34] utilisent la même technique que précédemment, mais augmentent le nombre de types de lignes qui peuvent être tracées. En effet ils introduisent des notations

spéciales pour les ourlets et les plis .L'utilisateur trace d'abord le contour de l'avant et de l'arrière du vêtement, à partir de cela le système fait des déductions sur la forme générale du vêtement.

Afin d'augmenter le réalisme, des vêtements [36] ont développé une méthode qui ajoute des plis générés de façon procédurale. Récemment [37] ont développé une méthode qui a pour but de créer des vêtements à la physique plus réaliste que les méthodes précédentes. La figure 4.7 est une comparaison entre le résultat produit par la première méthode [35] et cette méthode. On peut remarquer que le vêtement produit par [37] est plus réaliste que celui de la première méthode.

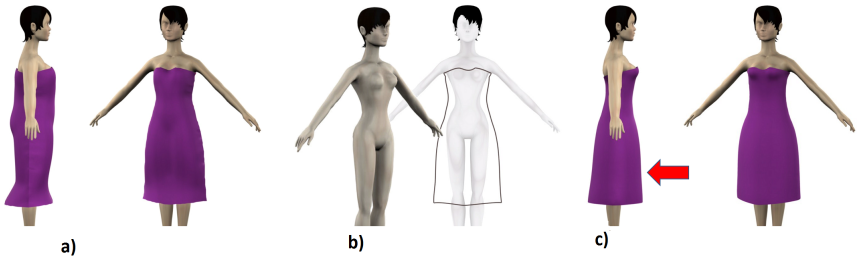

FIGURE 2.17 : Comparaison entre [35] (à gauche) et [37] (à droite)

2.4.4 Modélisation à partir d'une base de données de formes

L'idée principale de cette approche est de modéliser la forme à reconstruire en assemblant des formes existantes venant d'une base de données. Cette base de données peut être composée de modèles paramétriques dont nous pouvons changer les dimensions et les formes en modifiant les paramètres [38] [39] . L'autre solution est une base de données qui contient des formes segmentées, c'est à dire découpées en morceaux [40]. Ces morceaux peuvent être ensuite assemblés pour construire de nouvelles formes. Le principal désavantage de ces méthodes est qu'elles ne peuvent construire que des formes correspondantes à celles qui sont dans la base de données.

FIGURE 2.18 : Les traits (en rouge) sont utilisés pour retrouver des morceaux dans la base de données et les intégrer dans le modèle 3D [40]

2.4.5 Modélisation d'autres types de formes

De multiples autres méthodes ont été proposées pour la reconstruction de formes spécifiques à partir de croquis : le corps humain [41], les bâtiments [42], les nuages [43], le système vasculaire [44], etc. Comme je l'ai dit précédemment, leur principal défaut est que ce sont des méthodes spécialisées pour la reconstruction de formes spécifiques.

2.5 Modélisation itérative

L'idée de cette classe de méthodes est de construire la forme 3D de façon itérative, c'est à dire, en plusieurs étapes. L'utilisateur/utilisatrice dispose généralement de plusieurs outils qui lui permettent de changer le point de vue du croquis, d'ajouter, de supprimer ou de modifier des éléments de la forme a reconstruire a l'aide de widgets ou a l'aide d'une interface gestuelle. D'une certaine façon, ces méthodes sont à mi-chemin entre les méthodes qui n'utilisent qu'un croquis en entrée et les logiciels commerciaux de modélisation géométriques comme Maya ou 3DS max. Le principal avantage de ces méthodes est qu'elles permettent de construire des formes beaucoup plus compliquées par rapport à celles n'utilisant qu'un croquis. En particulier, elles permettent d'obtenir des formes avec une plus grande précision puisque l'utilisateur/utilisatrice peut utiliser le zoom sur

la partie du croquis qui l'intéresse. L'inconvénient est qu'elles nécessitent de la part de l'utilisateur/utilisatrice d'apprendre à manipuler les widgets. C'est pour cette classe de méthodes que se trouve le plus grand nombre de travaux de recherche.

Certaines de ces méthodes utilisent une surface implicite pour représenter la surface à reconstruire [45] [46] [47] [48]. D'autres utilisent un maillage [49] [50] [20] [51]. Par exemple, la figure 2.19 présente des résultats obtenus en utilisant la méthode de [50].

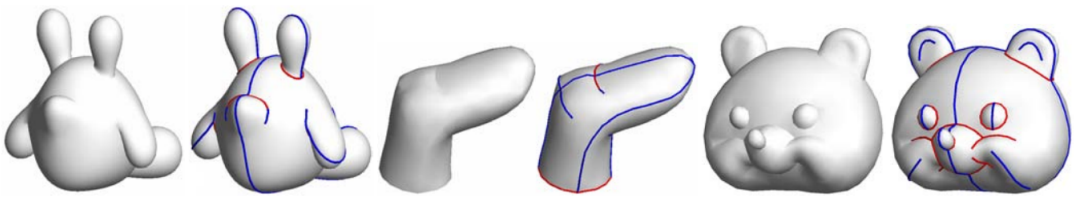

FIGURE 2.19 : Différents résultats obtenus en utilisant FiberMesh [50], l'utilisateur définit les courbes de contrôles (bleu : courbes douces, rouge : courbes aiguës) et le système construit des surfaces interpolant ces courbes

D'autres méthodes ont aussi été développées pour la modélisation de courbes tridimensionnelles [52] [53].

2.6 Modélisation à partir de plusieurs vues

Comme la modélisation à partir d'un seul croquis est difficile à résoudre, plusieurs personnes ont travaillé sur la reconstruction à partir de deux ou plusieurs croquis ; ces croquis correspondent généralement à des vues orthogonales (vue de face, vue de côté .etc.). Contrairement aux méthodes de modélisation itérative, l'utilisateur/utilisatrice ne dispose pas de moyen pour corriger ou modifier une forme reconstruite. Les seules données utilisées sont les croquis. Par exemple L'article [54] présente une méthode pour créer une forme 3D en utilisant la géométrie de construction de solides (CSG en anglais : "Constructive Solid Geometry"). La *CSG* est une technique de modélisation géométrique qui représente un objet solide comme une combinaison d'objets de base (cube, cylindre, sphère, etc.). La figure 2.21 montre un exemple d'objet construit à l'aide d'opérations booléennes successives (union, intersection ou différence) .

L'article [55] présente une étude de la reconstruction utilisant la géométrie épipolaire. Le principal problème de cette classe de méthodes est qu'elle n'est pas pratique pour l'utilisateur/utilisatrice. En particulier, il est difficile pour une personne de dessiner avec précision deux vues d'une même forme. Le problème devient particulièrement compliqué lorsque la forme à reconstruire est composée de multiples éléments.

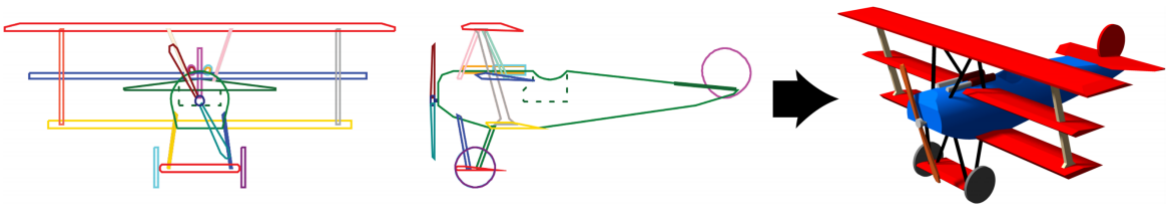

FIGURE 2.20 : Construction d'un modèle 3D à l'aide des deux silhouettes (à gauche) [54]

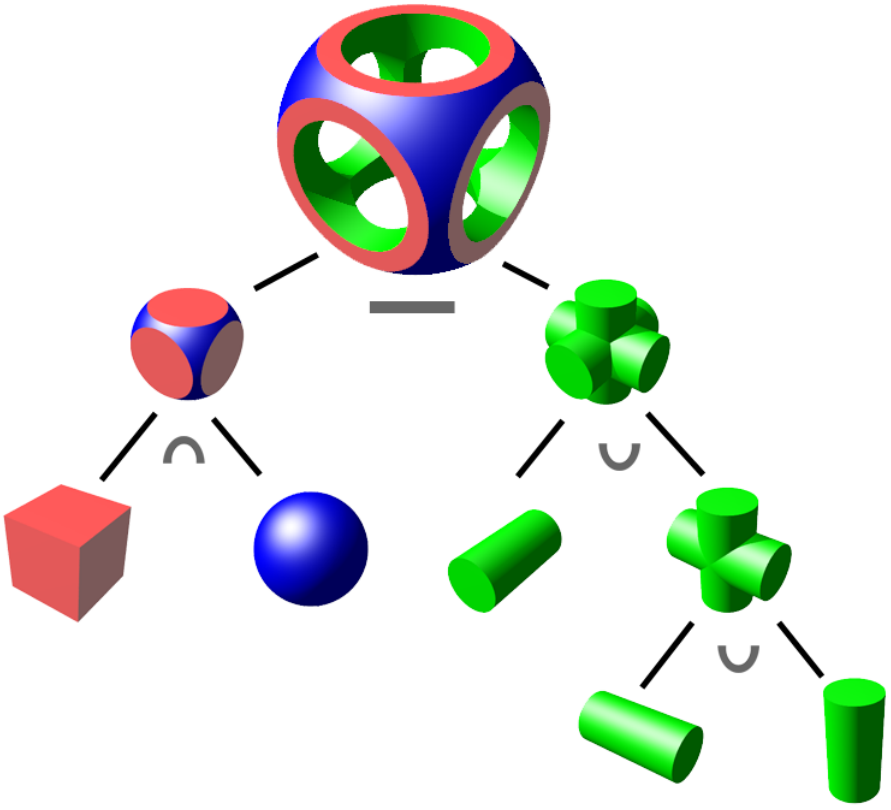

FIGURE 2.21 : Un objet construit à l'aide de la géométrie de construction de solide peut être représenté par un arbre binaire. Les feuilles représentent les objets de base et les noeuds les opérations. (extrait de http://en.wikipedia.org/wiki/Constructive_solid_geometry)

2.7 Modélisation d'une surface à l'aide d'un croquis représentant sa silhouette

Les données d'entrée sont un croquis représentant la silhouette de la surface à reconstruire. Ce croquis est généralement composé de courbes polygonales. L'objectif est de reconstruire une surface telle que la projection orthogonale de la silhouette de la surface corresponde au croquis. Contrairement aux méthodes décrites dans les sections 2.5 et 2.6, ces méthodes sont fidèles aux principes du croquis. L'utilisateur n'a pas à manipuler des widgets pour créer la forme tridimensionnelle.

Le problème de la reconstruction à partir d'un croquis est particulièrement difficile. Contrairement aux autres approches, très peu de travaux ont été proposés. [56] ont publié une méthode (voir figure 2.22) qui permet de reconstruire des formes simples à partir du croquis de la silhouette. Dans une première étape, la méthode analyse le croquis pour calculer les parties cachées de la silhouette. Cette méthode d'analyse du croquis présente des défauts qui limitent la complexité des formes qui peuvent être reconstruites. Dans la deuxième étape, la forme 3D est calculée à l'aide d'un maillage dont les positions des sommets sont calculées à l'aide d'une méthode d'optimisation.

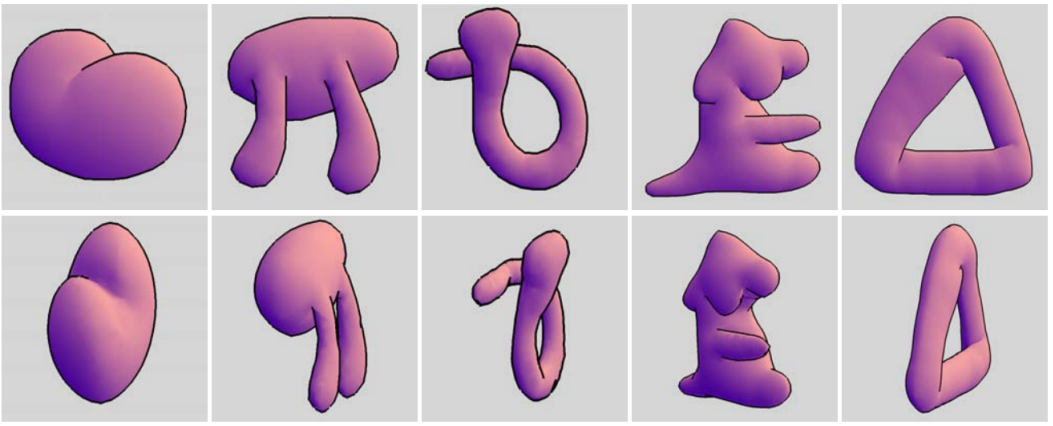

FIGURE 2.22 : Résultats obtenus à l'aide de la méthode de [56], la ligne du haut montre les formes du point de vue du dessin et la ligne du bas montre les formes sous un angle différent

2.8 Objectifs

Mon objectif est le développement de méthodes de reconstruction qui suivent les principes suivants :

— La méthode doit pouvoir être utilisée pour modéliser une grande variété de formes ou de courbes. L'objectif n'est pas de développer une méthode pour la modélisation de formes spécifiques (vêtements, arbres, etc.).
— La méthode doit utiliser un croquis dessiné à main levée, c'est à dire, le dessin est composé de courbes et non pas de segments rectilignes. Cette façon de dessiner les croquis est la plus naturelle.

Dans le chapitre suivant, je présente une méthode permettant de générer des courbes constituées de morceaux d'hélices à partir de croquis.

Chapitre 3

Modélisation de courbes constituées de morceaux d'hélices à partir de croquis

Dans ce chapitre nous présentons une méthode pour reconstruire des courbes tridimensionelles à partir de croquis. Cette méthode génère à partir d'un dessin une courbe constituée de morceaux d'hélices. Les données en entrée sont une courbe polygonale C_I définie dans le plan (x,y). L'objectif est de calculer une courbe 3D composée d'hélices circulaires et dont la projection orthogonale passe au plus près des sommets de la courbe donnée en entrée. Les hélices sont des courbes tridimensionnelles dont la courbure et la torsion sont constantes. Ces formes sont très courantes dans la nature et les objets créés par les hommes.

3.1 Les hélices

Une hélice est une courbe tridimensionnelle dont la courbure et la torsion sont constantes. L'équation de l'hélice qui passe par l'origine du système de coordonnées et dont l'axe principal est parallèle à l'axe y est :

$$\begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} r(cos(t) - 1) \\ pt \\ rsin(t) \end{pmatrix} \tag{3.1}$$

Avec $r \geq 0$ et $p \geq 0$, r et p étant respectivement le rayon et le pas de l'hélice.

3.2 Modélisation de morceaux d'hélices à partir de croquis

3.2.1 Aperçu de la méthode

L'élément principal de notre approche est la méthode pour calculer l'hélice telle que sa projection orthogonale soit égale à une partie ou l'intégralité de la courbe polygonale 2D C_i. Dans la section 3.2.2, nous décrivons quelques propriétés de l'hélice projetée. En utilisant ces propriétés, nous présentons une méthode pour calculer l'ajustement d'une hélice projetée à une courbe d'entrée C_I.

Dans la plupart des cas, l'hélice reconstruite ne correspond pas à l'intégralité de la courbe d'entrée C_I, en effet il est rare que l'utilisateur dessine une courbe qui est une projection exacte d'une hélice. Ainsi, l'ajustement est effectué de manière récursive avec plusieurs hélices. Après avoir ajustée la première hélice H_1 à C_I, nous identifions les parties de C_I pour lesquelles les deux courbes H_1 et C_I ne correspondent plus. Nous calculons ensuite l'ajustement avec l'hélice suivante H_2. Ce processus est répété récursivement jusqu'à ce que l'intégralité de la courbe d'entrée soit mise en correspondance avec une hélice. Le résultat est une courbe constituée de morceaux d'hélices circulaires qui passe au plus près des sommets de la courbe C_I. Ce processus est expliqué dans la section 3.2.4.

Une fois que les coefficients de la courbe constitués de morceaux d'hélices circulaires ont été générés, nous calculons sa position dans l'espace. Le résultat est une courbe discontinue en G^1, car elle ne garantit pas la continuité des tangentes aux jonctions entre les différentes hélices composant la courbe. Ainsi, nous modifions les coefficients des hélices de façon à minimiser les discontinuités entre les tangentes aux jonctions des différents morceaux de la courbe tout en maintenant une faible erreur de correspondance. Cette dernière étape est expliquée dans la section 3.2.5.

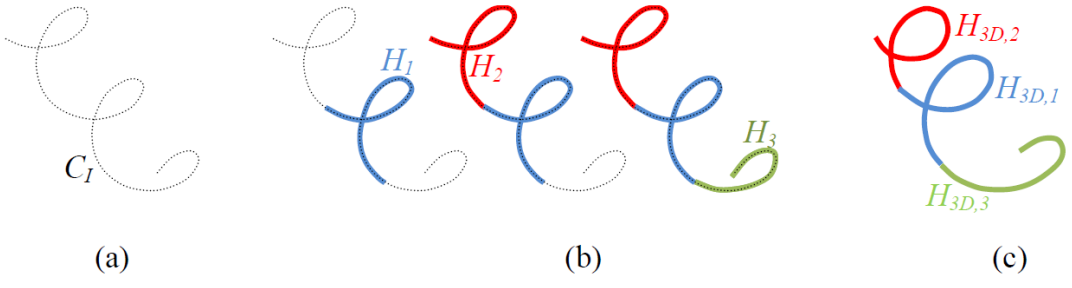

FIGURE 3.1 : Vue d'ensemble de l'approche : la première étape (b) consiste à ajuster un ensemble d'hélices projetées $\{H_1, H_2, H_3\}$ à différentes parties de la courbe polygonale d'entrée C_I (a). Dans la seconde étape (c), les hélices sont calculées en 3D depuis leurs projections. Ensuite, nous utilisons un procédé d'optimisation pour réduire les discontinuités entre les tangentes aux jonctions de ces hélices

L'utilisateur dessine une courbe 2D polygonale C_I sur le plan $(z = 0)$, ce plan est appelé plan du croquis. Cette courbe polygonale 2D C_I est constituée d'une séquence de p sommets $\{v_1, v_2, \ldots, v_p\}$ et $p-1$ segments de droite connectant successivement les sommets entre eux. Cette courbe est la projection orthogonale sur le plan du croquis $(z = 0)$ d'une courbe constituée de morceaux d'hélices circulaires (voir figure 3.1). En conséquence, les coordonnées x et y des sommets de la courbe constituée de morceaux d'hélices sont connus. Les coordonnées z restent à calculer.

3.2.2 Propriétés de l'hélice

3.2.2.1 Hélice

On considère une hélice avec une orientation arbitraire et sa projection sur le plan (z=0). Parmi les trois rotations possibles (selon les axes x,y et z), seulement la rotation selon l'axe x change la courbure de l'hélice projetée (voir figure 3.2).

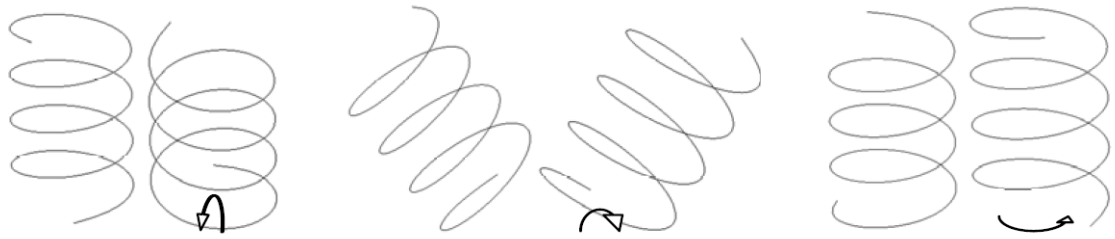

FIGURE 3.2 : Rotation autour des axes x,y et z

Nous allons donc seulement prendre en compte la rotation selon cet axe. Soit θ l'angle de rotation selon l'axe x. La matrice de rotation R_x selon l'axe x est la suivante :

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{pmatrix} \tag{3.2}$$

L'équation paramétrique $H_{3D,R}(t)$ de l'hélice avec rotation est la suivante :

$$H_{3D,R}(t) = \begin{pmatrix} r(cos(t)-1) \\ pt \\ rsin(t) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{pmatrix} = \begin{pmatrix} r(cos(t)-1) \\ ptcos(\theta) - rsin(\theta)sin(t) \\ ptsin(\theta) + rcos(\theta)sin(t) \end{pmatrix} \tag{3.3}$$

l'équation paramétrique de l'hélice après projection sur le plan $(z = 0)$ est :

$$H(t) = \begin{pmatrix} r(cos(t) - 1) \\ ptcos(\theta) - rsin(\theta)sin(t) \\ 0 \end{pmatrix} \tag{3.4}$$

La figure 3.3 permet d'illustrer le fait que le paramètre θ qui est l'angle de rotation selon l'axe x influe sur la courbure de l'hélice projetée.

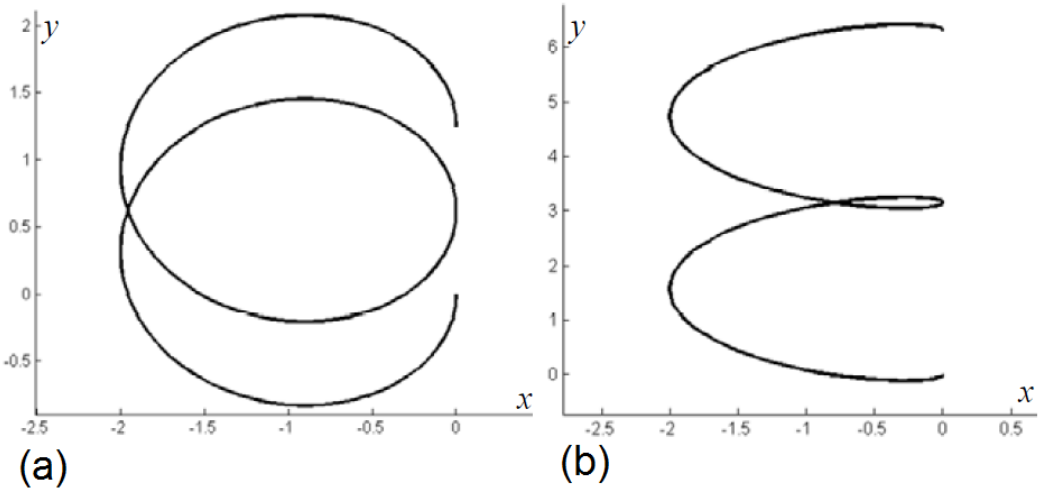

FIGURE 3.3 : Courbe H(t) générée avec différents coefficients r,p et θ. (a) $r = 1, p = 0.5$ et $\theta = 0.2$ avec $t \in [0, 4\pi]$. (b) $r = 1$, $p = 0.7$ et $\theta = 0.8$ avec $t \in [0, 4\pi]$.

3.2.2.2 Point de courbure maximal de l'hélice projetée

La courbure de l'hélice projetée H_t est donnée par l'équation suivante :

$$\kappa_H(t) = \frac{|r(pcos(\theta)cos(t) - rsin(\theta))|}{r^2 sin(t)^2 + (pcos(\theta) - rsin(\theta)cos(t))^2)^{\frac{3}{2}}} \tag{3.5}$$

Nous formulons la proposition suivante :

Proposition 1. La valeur de $\kappa_H(t)$ est maximale quand t est un multiple de π (c'est à dire $t = 0$ ou $t = pi$).

Cette proposition est très importante car elle nous permet de trouver les origines des hélices que nous allons ajuster à la courbe C_I. En effet il suffira ensuite de trouver le point de courbure maximal, d'y ajuster une hélice et de répéter cette opération autant de fois qu'il le faut pour que les projections des morceaux d'hélices couvrent l'intégralitée de la courbe C_I. Nous donnons miantenant la démonstration de cette proposition. Afin de prouver la proposition, on analyse la courbure de l'ellipse O qui est la projection du

cercle osculateur O_{3D} sur la courbe $H_{3D}(t)$ 3.6. Pour déterminer l'orientation du cercle osculateur O_{3D}, nous calculons le repère de Frenet le long de la courbe $H_{3D}(t)$.

Sois $(\overrightarrow{X}, \overrightarrow{Y}, \overrightarrow{Z})$ le système de coordonnée orthonormal dans lequel l'hélice $H_{3D}(t)$ est définie. Sois $(\hat{T}_{H3D}(t), \hat{N}_{H3D}(t), \hat{B}_{H3D}(t))$ le système de coordonnées de Frenet au point $H_{3D}(t)$, ce système est orthonormal et son origine est le point $H_{3D}(t)$. Nous définissons maintenant l'équation paramétrique du cercle O_{3D} dans le repère de Frenet :

$$O_{3D}(u) = \begin{pmatrix} \frac{(a^2+b^2)}{a} sin(u) \\ \frac{a^2+b^2}{a}(1 - cos(u)) \\ 0 \end{pmatrix} \tag{3.6}$$

L'equation paramétrique $O_{3D}(u)$ est définit tel que :

- Le point $O_{3D}(0)$ est l'origine du repère de Frenet, cela implique que $O_{3D}(0)$ se trouve à la même position que $H_{3D}(t)$.
- La tangente de $O_{3D}(u)$ à $u = 0$ est parallèle à l'axe x du repère de Frenet.

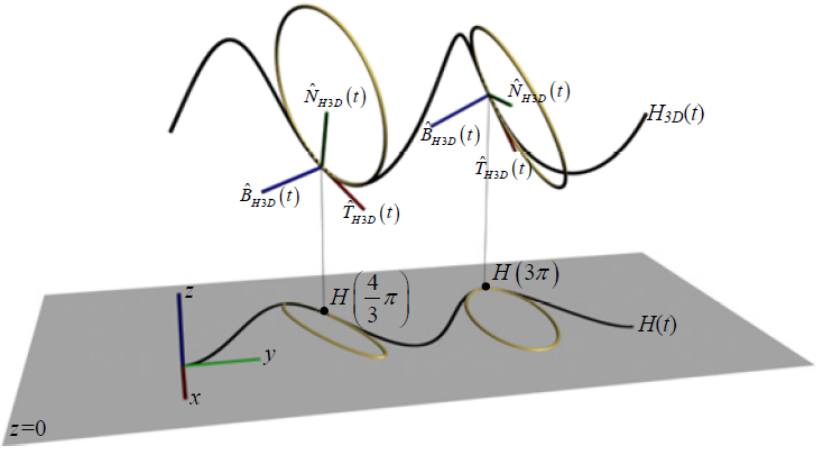

FIGURE 3.4 : Le cercle osculateur et sa projection pour $t = \frac{4}{3}\pi$ et $t = 3\pi$. Quand $t = 3\pi$, le point $H(3\pi)$ est localisé sur l'axe principal de l'éllipse O

Sois $\hat{T}_{H3D}(t)$,$\hat{N}_{H3D}(t)$ et $\hat{B}_{H3D}(t)$ respectivement les vecteurs tangente, normale et binormale au point $H_{3D}(t)$. Afin de démontrer la proposition, nous mettons en évidences les cinq propriétes suivantes :

- (P1) : Le cercle osculateur O_{3D} est dans le plan osculateur $(\hat{T}_{H3D}(t), \hat{N}_{H3D}(t))$ de $H_{3D}(t)$ et sa courbure est identique à celle de $H_{3D}(t)$ (voir figure 3.4).
- (P2) : La matrice $M_{F\to G}$ qui définit la transformation du repère de Frenet au repère globale est :

$$M_{F\to G} = \begin{pmatrix} T_x & N_x & B_x \\ T_y & B_y & B_y \\ T_z & N_z & B_z \end{pmatrix} \tag{3.7}$$

Ces coefficients sont les composants x, y et z de respectivement $\hat{T}_{H3D}(t)$,$\hat{N}_{H3D}(t)$ et $\hat{B}_{H3D}(t)$, ce qui nous donne

$$M_{F\to G} = \begin{pmatrix} -\frac{asin(t)}{\sqrt{a^2+b^2}} & -cos(t) & -\frac{bsin(t)}{\sqrt{a^2+b^2}} \\ \frac{bcos(c)-asin(c)cos(t)}{\sqrt{a^2+b^2}} & sin(c)sin(t) & -\frac{acos(c)+bsin(c)cos(t)}{\sqrt{a^2+b^2}} \\ \frac{bsin(c)+acos(c)cos(t)}{\sqrt{a^2+b^2}} & -cos(c)sin(t) & -\frac{asin(c)-bcos(c)cos(t)}{\sqrt{a^2+b^2}} \end{pmatrix} \quad (3.8)$$

— (P3) : La matrice $M_{F\to S}(t)$ qui permet de passer du repère de Frenet à la projection orthogonale sur le plan $z = 0$ est défini de la façon suivante :

$$M_{F\to S} = \begin{pmatrix} -\frac{asin(t)}{\sqrt{a^2+b^2}} & -cos(t) & -\frac{bsin(t)}{\sqrt{a^2+b^2}} \\ \frac{bcos(c)-asin(c)cos(t)}{\sqrt{a^2+b^2}} & sin(c)sin(t) & -\frac{acos(c)+bsin(c)cos(t)}{\sqrt{a^2+b^2}} \\ 0 & 0 & 0 \end{pmatrix} \quad (3.9)$$

$M_{F\to S}(t)$ est le résultat de deux transformations successives, la première étant $M_{F\to G}(t)$ et la seconde la projection sur le plan $z = 0$.

— (P4) : La projection du cercle osculateur O_{3D} sur le plan $z = 0$ est l'ellipse O. La longueur de l'axe majeur de O est constant, elle est égale au diamètre du cercle osculateur. La longueur du petit axe est, elle, fonction de l'orientation du plan osculateur par rapport au plan $z = 0$. L'équation paramétrique de l'ellipse O est $O(u,t) = M_{F\to S}(t)O_{3D}(u)$ ce qui nous donne :
l'équation paramétrique du cercle O_{3D} dans le repère de Frenet :

$$O(u,t) = \begin{pmatrix} \frac{cos(t)(cps(u)-1)(a^2+b^2)}{a} - sin(t)sin(u)\sqrt{a^2+b^2} \\ \frac{sin(u)(bcos(c)-asin(c)cos(t))\sqrt{a^2+b^2}}{a} - \frac{sin(c)sin(t)(cos(u)-1)(a^2+b^2)}{a} \\ 0 \end{pmatrix} \quad (3.10)$$

— (P5) : La courbure de l'ellipse $O(u,t)$ pour $u = 0$ est égale à la courbure de l'hélice projetée $H(t)$. En effet, en attribuant à u la valeur 0 dans l'équation de calcul de la courbure de l'ellipse $O(u,t)$, $\frac{||O'(u,t)\times O''(u,t)||}{||O'(u,t)||^3}$, nous obtenons l'expression de la courbure de l'hélice projetée . $O\prime(u,t)$ et $O\prime\prime(u,t)$ sont les dérivées première et seconde de u.

Afin de prouver la proposition 1, nous faisons les déclarations suivantes. Premièrement, la projection orthogonale du point $O_{3D}(0)$ situé sur le cercle osculateur $O_{3D}(u)$ est le point $O(0,t)$ qui est situé sur l'ellipse $O(u,t)$. $O(u,t)$ étant la projection orthogonale de $O_{3D}(u)$ tel que $0(u,t) = M_{F\to S}(t)O_{3D}$.

Deuxièmement, le point $O(0,t)$ est situé à une des extrémités de l'axe majeur de l'ellipse $O(u,t)$ si le vecteur $M_{F\to S}(t)\begin{pmatrix}0 & 1 & 0\end{pmatrix}^T$ est un vecteur unitaire. Ceci est vrai lorsque $t = 0$ ou quand $t = \pi$. Puisque la courbure de l'ellipse est maximale aux extrémités de

l'axe majeur, il en découle que la courbure au point $O(0,t)$ est maximale pour $t=0$ ou $t=\pi$.

La propriétée (P5) indiquant que la courbure de l'éllipse $O(u,t)$ pour $u=0$ étant égale à la courbure de l'hélice projetée $H(t)$. Nous concluons que la courbure pour $H(t)$ est maximale pour $t=0$ ou $t=\pi$.

3.2.2.3 Définition du repère local

Afin d'évaluer la correspondance entre l'hélice projetée et la courbe C_I, nous devons calculer la distance entre les points de C_I et les points correspondants de l'hélice. Ainsi, une étape préliminaire est d'aligner les deux courbes afin de comparer les coordonnées de leurs points. Pour ce faire, nous définissons un repère de coordonnées locales F_H au point $t=0$ (voir figure 3.5).

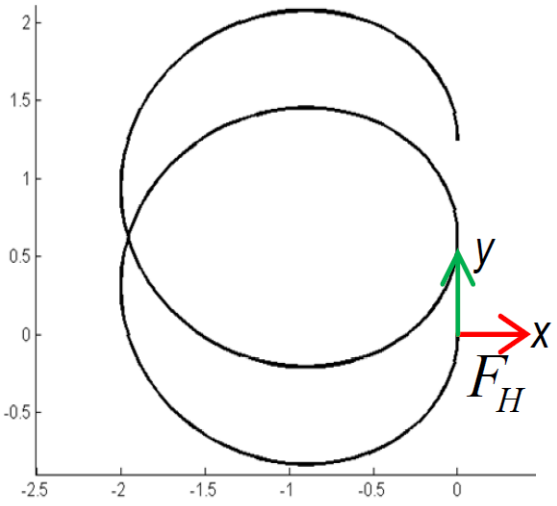

FIGURE 3.5 : Le repère local F_h à $t=0$

L'axe y est le vecteur unitaire parallèle à la tangente à la courbe à t=0 et l'axe x est normal à l'axe y tel que (x,y) soit orienté dans le sens des aiguilles d'une montre. Le vecteur unitaire tangent de l'hélice projetée $H(t)$ est :

$$\hat{T}_H(t) = \begin{pmatrix} -\frac{rsin(t)}{\sqrt{r^2sin(t)^2+(pcos(\theta)-rsin(\theta)cos(t))^2}} \\ \frac{pcos(\theta)-rsin(\theta)cos(t)}{\sqrt{r^2sin(t)^2+(pcos(\theta)-rsin(\theta)cos(t))^2}} \\ 0 \end{pmatrix} \tag{3.11}$$

3.2.3 Estimation de la courbure et de la tangente de la courbe polygonale

De même que pour l'hélice projetée, il est aussi nécessaire de calculer la courbure et le vecteur unitaire tangent de la courbe polygonale C_I. Nous utilisons la méthode décrite par [57]. Soit v_i un sommet de C_I pour lequel nous voulons estimer la courbure. L'idée principale est de calculer une courbe paramétrique quadratique $C_i(t)$ passant au plus près des sommets de la courbe C_I au voisinage de v_i. L'équation de cette courbe est :

$$C_i(t) = \begin{pmatrix} x_i(t) \\ y_i(t) \end{pmatrix} = \begin{pmatrix} x_i + x'_i t + \frac{1}{2}x''_i t^2 \\ y_i + y'_i t + \frac{1}{2}y''_i t^2 \end{pmatrix} \tag{3.12}$$

x_i et y_i sont les positions des sommets v_i. x'_i, x''_i , y'_i et y''_i sont respectivement les dérivés 1^{re} et 2^{nd} des coordonnées x et y de v_i ; t est le paramètre de la courbe. Étant donné la courbe quadratique paramétrique $C_i(t)$ qui passe par les sommets de la courbe C_I dans le voisinage de v_i, la courbure κ_{vi} et le vecteur unitaire tangent $\hat{T}_{vi}$ au point v_i sont exprimés par les équations suivantes :

$$\hat{T}_{v_i} = \begin{pmatrix} \frac{x'_i}{\sqrt{x'^2_i + y'^2_i}} \\ \frac{y'_i}{\sqrt{x'^2_i + y'^2_i}} \end{pmatrix} et \kappa_{vi} = \frac{x'_i y''_i - x''_i y'_i}{(x'^2_i + y'^2_i)^{\frac{3}{2}}} \tag{3.13}$$

3.2.4 Ajuster l'hélice projetée à la courbe C_I

Notre algorithme prend en entrée une courbe polygonale C_I uniformément échantillonnée. La courbe C_I est composée de p sommets $v_1, ...v_p$ connectés par des segments de droite de longueur approximativement égales. Premièrement, nous calculons la courbure pour chaque sommet de C_I en utilisant la méthode décrite dans la section 3.2.3. Nous identifions ensuite le sommet v_s dont la courbure est maximale. Ce sommet est le sommet initial depuis lequel nous commençons l'ajustement de l'hélice projetée. L'objectif est de calculer une hélice dont la projection orthogonale passe au plus près des sommets de la courbe polygonale au voisinage de v_s.

3.2.4.1 Alignement de C_I à l'hélice

Ensuite, nous calculons le vecteur tangent unitaire $\hat{T}_v$, au sommet v_s de C_I en utilisant la méthode décrite dans la section 3.2.3 et nous construisons un repère local F_{v_s} dont l'origine est v_s, et son axe y est $\hat{T}_{v_s}$, son axe x est normal à l'axe y et son orientation est défini tel que (x,y) soit dans le sens antihoraire. Ensuite, nous alignons le repère

de coordonnées F_{v_s} au repère F_H, F_H étant le repère de coordonnées locale de l'hélice projetée pour $t = 0$. Cet alignement requiert d'appliquer une translation, une rotation et possiblement une réflexion aux coordonnées des sommets de C_I. Après cette transformation, les sommets de C_I et les points de l'hélice projetée sont maintenant dans le même système de coordonnées.

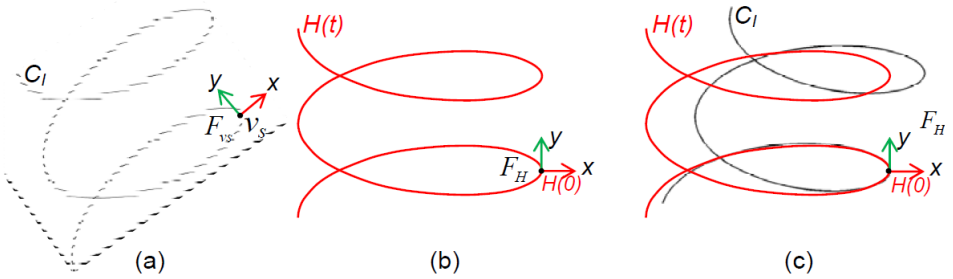

FIGURE 3.6 : (a) le sommet de courbure maximale v_s et le repère local F_{vs} de la courbe C_I sont calculés. (b) le repère local F_H de l'hélice au point $H(0)$ est aussi calculé. (c) Une transformation est appliquée sur C_I afin que F_{vs} et F_H soient alignés.

3.2.4.2 Estimation des coefficients de l'hélice en utilisant deux sommets de C_I

L'étape suivante consiste à estimer la valeur des coefficients r, p et θ de l'hélice projetée de façon à ce qu'elle passe au plus près des sommets de la courbe C_I dans le voisinage de v_s. Notons que t est aussi inconnu sauf pour le point v_s et sa valeur change le long de l'hélice. Nous calculons ces coefficients en deux étapes. Premièrement, nous choisissons un sommet v_n dans le voisinage de v_s et calculons les coefficients r, p et θ avec v_s et v_n. Dans un second temps, nous estimons l'erreur de correspondance entre l'hélice projetée et la courbe polygonale C_I avec les coefficients r,p et θ calculés à l'étape précédente. Nous répétons ces deux étapes plusieurs fois et nous choisissons les valeurs de r, p et θ qui donnent l'erreur de correspondance la plus faible.

Comme le repère local de C_I au sommet v_s a été aligné avec le repère de l'hélice projetée $H(t)$ à $t = 0$, les points $H(0)$ et v_s ont leurs coordonnées égales à 0. κ_{v_s} est la courbure de C_I au sommet v_s. Nous sélectionnons un sommet $v_n = \begin{pmatrix} X_{v_n} \\ Y_{v_n} \end{pmatrix}$ dans le voisinage de v_s, son vecteur unitaire tangent est $\hat{T}_{v_n} = \begin{pmatrix} X_{\hat{T}_{v_n}} \\ Y_{\hat{T}_{v_n}} \end{pmatrix}$. Nous supposons que l'hélice projetée passe par les sommets v_s et v_n. En particulier, v_n est localisé sur l'hélice projetée à $t = \alpha$. Ensuite, nous écrivons le système d'équations :

$$\begin{cases} v_n = H(\alpha) \\ \hat{T}_{v_n} = \hat{T}_H(\alpha) \end{cases} \tag{3.14}$$

La première équation implique que les les points $H(\alpha)$ et v_n doivent avoir les mêmes coordonnées et la seconde implique que $\hat{T}_{v_n}$ et $\hat{T}_H(\alpha)$, le vecteur unitaire tangent à $H_(\alpha)$ doivent être égaux.

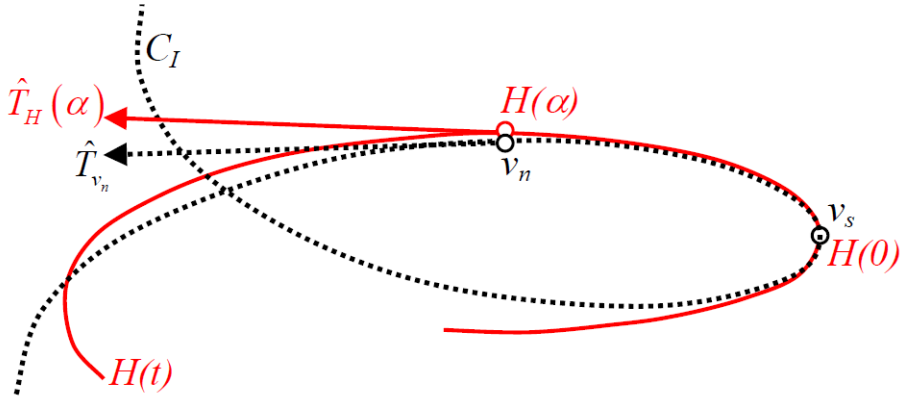

FIGURE 3.7 : L'hélice projetée (en rouge) en cours d'ajustement sur C_I.

Le système d'équations ci-dessus n'a pas de solution ; il est résolu en utilisant une méthode d'optimisation dont les inconnues sont r , p et θ et la fonction objectif est donnée par l'équation suivante :

$$E(a,b,c) = \beta||v_n - H(\alpha)||^2 + (1-\beta)||\hat{T}_{v_n} - \hat{T}_H(\alpha)||^2 \tag{3.15}$$

beta est un paramètre compris entre 0 et 1 qui permet de privilégier soit l'égalité entre les deux points, soit l'égalité entre les deux tangentes. Notons aussi que v_n peut être n'importe quel sommet dans le voisinage de v_s. Dans notre implémentation, nous fixons α à $\pi/4$. Les variables inconnues r, p et θ ne sont pas linéairement liés. Des méthodes telles que le simplex proposé par [58] sont utilisées pour résoudre de tels problèmes d'optimisation. La figure 3.8 montre différentes hélices projetées calculées pour différents sommets v_n.

3.2.4.3 Estimation de l'ajustement de l'hélice aux autres sommets de C_I

Maintenant que nous avons calculé les coefficients de l'hélice pour les deux sommets v_s et v_n, la prochaine étape est de vérifier l'ajustement de l'hélice projetée aux autres sommets. L'objectif est de trouver le nombre de sommets consécutifs le long de C_I pour

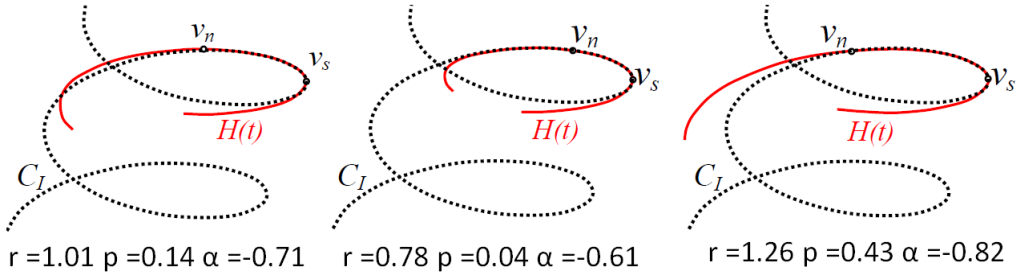

FIGURE 3.8 : L'hélice (en rouge) générée pour différents sommets v_n. Les coefficients r, p et θ sont montrés en dessous de chaque hélice.

lesquels l'ajustement est satisfaisant. Soit v_{s+j} un sommet dans le voisinage de v_s dont l'index est $s+j$ (j démarrant à 1). Le sommet d'index $s+j$ est le j^{me} sommet après v_s dans le sens des aiguilles d'une montre.

Pour chaque sommet v_{s+j}, nous calculons d'abord le paramètre t_{s+j} tel que le point sur l'hélice projetée $H(t_{s+j})$ est le point le plus proche de v_{s+j}. Ensuite, nous calculons l'erreur de correspondance qui mesure les différences de coordonnées entre V_{s+j} et le point correspondant $H(t_{s+j})$ sur l'hélice projetée :

$$E(v_{s+n}, ts+n) = ||v_{s+n} - H(t_{s+n})||^2 \tag{3.16}$$

Les itérations sur v_{s+j} stoppent lorsque l'erreur de correspondance $E(v_{s+n}, t_{s+n})$ est plus grande que le seuil E_{Max} défini par l'utilisateur. Ce seuil E_{Max} doit être suffisamment faible considérer que les deux points ont la même position. Typiquement, cette valeur est définie à 0.1 % de la longueur totale de la courbe C_I.

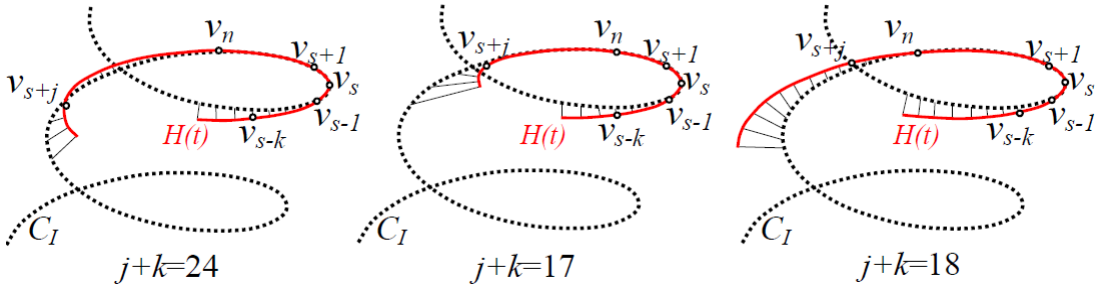

FIGURE 3.9 : Calcul de l'erreur pour trois les hélices de la figure 3.8. Le nombre de sommets $(j+k)$ dont l'erreur est inférieure à E_{Max} est montré sous chaque courbe.

Le même processus est répété pour les sommets adjacents v_{s-k} de l'autre côté de de v_s avec l'indice k commençant à 1. Finalement, nous calculons la valeur $j+k$ qui est le nombre de sommets consécutifs pour lesquels l'erreur de correspondance est en dessous du seuil E_{max}. Ce nombre de sommets indique la portion de la courbe C_I qui est bien

ajustée avec l'hélice projetée dont les coefficients ont été calculés (voir figure 3.9). Le pseudo-code de l'algorithme est donné dans la figure 3.10.

```
1: Compute_Num_Of_Matched_Vert( r, p, θ , Vs )
2:   Input: r, p, θ , Vs
3:   j=0
4:   do
5:   |   j=j+1
6:   |   compute ts+j such that ||v_{s+j} − H(t_{s+j})||^2 is minimal
7:   while ||v_{s+j} − H(t_{s+j})|| < E_Max
8:   k=0
9:   do
10:  |   k=k+1
11:  |   compute ts-k such that ||v_{s−k} − H(t_{s−k})||^2 is minimal
12:  while ||v_{s−k} − H(t_{s−k})|| < E_Max
13:  return j+k
```

FIGURE 3.10

Tout le processus d'ajustement requiert l'utilisation de façon alternative des deux algorithmes décrits dans 3.2.4.3 et 3.2.4.2. Nous choisissons un sommet v_n parmi les voisins de v_s et calculons les coefficients de l'hélice en utilisant v_s et v_n. Ensuite, nous calculons à quel point l'hélice projetée, avec les coefficients calculés, peut correspondre au reste de la courbe. Nous choisissons les coefficients r, p et θ pour lesquelles l'hélice couvre la plus grande partie de la courbe polygonale C_I. le pseudo-code de cet algorithme est donné ci-dessous :

3.2.4.4 Approximation de C_I avec plusieurs hélices

Une seule hélice ne suffit pas en général pour approximer la courbe C_I en entier. Dans ce cas, nous construisons une courbe constituée de plusieurs hélices afin d'approcher C_I.

Le processus est récursif, après avoir ajusté une hélice à C_I, nous identifions la partie de C_I dont les sommets n'ont pas été traités par l'algorithme. L'ajustement de la prochaine hélice est effectué sur ces sommets. (figure 3.12).

La sortie de l'algorithme est un ensemble d'hélices projetées : $H_1, H_2, ..., H_n$ triées selon leurs positions sur C_I. Chaque hélice projetée H_I a un sommet en commun avec ses deux voisins H_{i-1} et H_{i+1}. Ces sommets en commun sont nommés sommet de jonctions.

```
1: Compute_Best_Helix_Coefficients(vs)
2:   Input: vs
3:   maxNbMatchV=0
4:   for i=-NMAX to +NMAX
5:     n=s+i
6:     compute the Helix coefficients rn, pn and θn
                  using the two vertices vs and vn
7:     nbMatchV=Compute_Num_Of_Matched_Vert( rn,pn,θn,vn)
8:     if nbMatchV > maxNbMatchV
9:       maxNbMatchV= nbMatchV
10:      r=rn
11:      p=pn
12:      θ=θn
13:    end
14:  end
15:  return r, p and θ
```

FIGURE 3.11

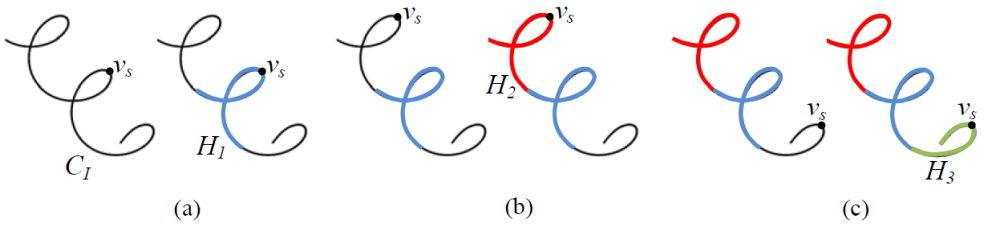

FIGURE 3.12 : Ajustement de plusieurs hélices à la courbe C_I : en (a), après avoir trouvé v_s, le sommet de courbure maximal de la courbe C_I, l'hélice projetée H_1 est ajustée à la portion de C_I contenant v_s. en (b), le sommet suivant de courbure maximale est trouvé dans la partie de la courbe CI qui n'a pas été ajustée avec des hélices. Ce processus est répété jusqu'à ce que tout CI soit approximé avec des hélices. Le résultat est une courbe constituée de morceaux d'hélices (H_2, H_1, H_3).

De plus, les orientations des hélices projetées sont cohérentes entre elles ; c'est à dire la courbe H_I commence au dernier point de la courbe H_{I-1}.

3.2.5 Reconstruction 3D et optimisation de l'ajustement

Une fois que toute la courbe C_I a été approximée à l'aide d'hélices projetées, l'étape suivante est de calculer les coordonnées dans l'espace des sommets de C_I. Nous commençons par la première hélice localisée à une extrémité de C_I et nous continuons ce processus de reconstruction en traitant les hélices adjacentes jusqu'à atteindre l'autre extrémité de

C_I. Les coordonnées z sont calculées en utilisant l'équation 3.17 et les coefficients des hélices calculés précédemment.

Notons qu'il existe deux hélices qui correspondent à la même hélice projetée sur le plan (z=0). Ces deux hélices sont symétriques selon le plan ($z = 0$). Les équations de l'hélice 3D $H_{3D}(t)$ et de son symétrique$H_{3D,S}(t)$ sont :

$$H_{3D}(t) = \begin{pmatrix} r(cos(t) - 1) \\ ptcos(\theta) - rsin(\theta)sin(t) \\ ptsin(\theta) + rcos(\theta)sin(t) \end{pmatrix} \tag{3.17}$$

$$H_{3D,S}(t) = \begin{pmatrix} r(cos(t) - 1) \\ ptcos(\theta) - rsin(\theta)sin(t) \\ -ptsin(\theta) - rcos(\theta)sin(t) \end{pmatrix} \tag{3.18}$$

Ces deux hélices ont les mêmes coordonnées à l'exception des coordonnées z pour lesquelles les signes sont opposés.

Quand nous traitons l'hélice suivante $H_{3D,i+1}$, la translation le long de l'axe z est calculée tel que le premier point H_{3Di+1} a la même coordonnée z que le dernier point de $H_{3D,j}$. Ceci est pour assurer que les extrémités des hélices voisines ont la même position. De plus, nous choisissons l'hélice (équation 3.17) ou sa symétrie (équation 3.18) tels que les vecteurs tangentes de $H_{3D,I}$ et $H_{3D,i+1}$ à la jonction entre les deux hélices soient les plus parallèles possible.

Comme nous pouvons le remarquer, la courbure de la courbe constituée de morceaux d'hélices est discontinue aux points de jonction entre les différentes hélices. Ceci est dû au fait que les coefficients des hélices ont été calculés indépendamment pour chaque hélice. Pour résoudre ce problème, nous modifions les coefficients des hélices de manière à rendre les tangents des hélices aux points de jonctions aussi parallèles que possible tout en gardant la distance entre la projection des hélices et la courbe C_I petite. Nous formulons ce problème comme un problème d'optimisation pour lesquels la fonction objectif mesure (1) la différence entre les tangentes aux points de jonction de l'hélice par morceaux et (2) l'erreur totale de correspondance qui est la distance entre les hélices projetées et les sommets de C_I.

Soit $H_{3D,1}, H_{3D,2}, ..., H_{3D,n}$ l'ensemble des n hélices composant l'hélice par morceaux. soit $H_1, H_2, ..., H_n$ l'ensemble des n hélices projetées correspondantes qui approximent la courbe C_I. Soit r_i,p_i et θ_i les coefficients de l'hélice projetée H_i. Chaque hélice projetée H_i correspond à une portion de la courbe C_I. Soit $V_i = \{v_{i,1}, v_{i,2}, ..., v_{i,j}, ...v_{i,n}\}$ l'ensemble des n_i sommets localisés sur une partie de la courbe C_I qui est approximée par H_i. À

chaque ensemble de sommets V_i on associe une matrice de transformation F_i. L'objectif de cette matrice de transformation est d'aligner les sommets V_i avec l'hélice projetée H_i dans l'espace en deux dimensions (x, y).

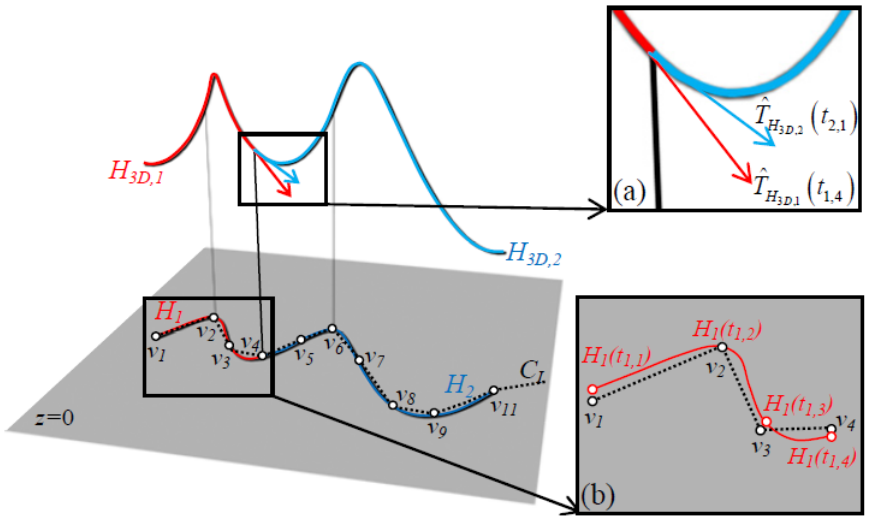

FIGURE 3.13 : Deux hélices $H_{3D,1}$,$H_{3D,2}$ ainsi que leurs projections H_1 et H_2. L'ensemble V_1 qui sont les sommets sur lesquelles H_1 est ajusté est $\{v_1, v_2, v_3, v_4\}$. L'ensemble V_2 correspondant à H_2 est composé de$\{v_4, v_5, v_6, v_7, v_8, v_9\}$. Le sommet qui effectue la jonction entre H_1 et H_2 est v_4

Le premier sommet $v_{i,1}$ de H_i est commun avec le dernier sommet $v_{i-1,n_{j-1}}$ de H_{i-1}. Ce sommet est appelé sommet de jonction (v_4 dans la figure 5.14). De façon similaire, le dernier sommet V_{i,n_i} de H_i est commun avec le premier sommet $v_{i+1,1}$ de H_{i+1}. Soit $\{t_{i,1}, t_{i,2}, ..., t_{i,n_i}\}$ les n_i paramètres des points sur l'hélice projetée H_i telle que $||F_i \cdots v_{i,j} - H_i(t_{i,j})||$ est minimal pour tout $v_{i,j} \in V_i$.

La fonction objectif à minimiser est la différence entre les vecteurs tangents aux sommets de jonctions, elle est définie de la façon suivante :

$$E_T(r_1, p_1, \theta_1, ..., r_n, p_n, \theta_n) = \sum_{i=2}^{n} ||\hat{T}_{H_{3D,i-1}}(t_{i-1,n_{i-1}}) - \hat{T}_{H_{3D,i}}(t_{i,1})||^2 \tag{3.19}$$

$\hat{T}_{H_{3D,i-1}}(t_{i-1,n_{i-1}})$ et $\hat{T}_{H_{3D,i}}(t_{i,1})$ sont les tangentes des deux hélices $H_{3D,i}$ et $H_{3D,i-1}$ au point de jonction (voir figure 5.14).Leurs expressions sont les suivantes :

$$\hat{T}_{H_{3D,i-1}}(t_{i-1,n_{i-1}}) = \begin{pmatrix} -\frac{r_{i-1} sin(t_{i-1,n_{i-1}})}{\sqrt{r_{i-1}^2 + p_{i-1}^2}} \\ \frac{p_{i-1} cos(c_{i-1}) - r_{i-1} sin(\theta_{i-1}) cos(t_{i-1,n_{i-1}})}{\sqrt{r_{i-1}^2 + p_{i-1}^2}} \\ \frac{p_{i-1} sin(c_{i-1}) - r_{i-1} cos(\theta_{i-1}) cos(t_{i-1,n_{i-1}})}{\sqrt{r_{i-1}^2 + p_{i-1}^2}} \end{pmatrix} \tag{3.20}$$

$$\hat{T}_{H_{3D,i}}(t_{i,1}) = \begin{pmatrix} -\frac{r_i sin(t_{i,1})}{\sqrt{r_i^2+p_i^2}} \\ \frac{p_i cos(\theta_i) - r_i sin,(\theta_i) cos(t_{i,1}}{\sqrt{a_i^2+b_i^2}} \\ \frac{p_i sin(\theta_i) - r_i cos,(\theta_i) cos(t_{i,1}}{\sqrt{a_i^2+b_i^2}} \end{pmatrix} \tag{3.21}$$

La fonction objectif qui permet de calculer l'erreur entre C_I et l'hélice projetée est :

$$E_D(r_1, p_1, \theta_1, ..., r_n, p_n, \theta_n) = \sum_{i=2}^{n} (\sum_{j=1}^{n_j} ||F_i \cdot v_{i,j} - H_i(t_{i,j})||^2) \tag{3.22}$$

$||F_i \cdot v_{i,j} - H_i(t_{i,j})||$ est la distance entre un sommet de C_I et le point le plus proche le long de l'hélice projetée $H_i(t_{i,j})$ correspondante (voir figure 5.14 (b)). La fonction objectif à minimiser est la suivante :

$$E(r_1, p_1, \theta_1, ..., r_n, p_n, \theta_n) = \gamma_T E_T(r_1, p_1, \theta_1, ..., r_n, p_n, \theta_n) + (1 - \gamma_T) E_D(r_1, p_1, \theta_1, ..., r_n, p_n, \theta_n) \tag{3.23}$$

Le poids γ_T est un paramètre dont la valeur est comprise entre 0 et 1 qui permet à l'utilisateur de privilégier la réduction des discontinuités des tangentes, soit de donner plus d'importance à la réduction de la distance entre les hélices projetées et C_I.

La fonction objectif n'est pas linéaire, afin de résoudre ce problème nous utilisons la méthode proposée par [58] pour calculer les variables inconnues telles que $E(r_1, p_1, \theta_1, ..., r_n, p_n, \theta_n)$ soit minimal.

3.2.6 Résultats

Nous avons implémenté notre algorithme à l'aide de scripts Matlab et nous avons utilisé 3DS Max pour dessiner les courbes 2D polygonales et pour la visualisation des hélices reconstruites. Nous avons testé notre méthode de reconstruction sur différentes courbes. Les résultats sont montrés dans les figures 3.14 et 3.15.

Le temps de calcul est d'environ 30 secondes pour une courbe composée de 500 sommets. La plus grande partie du temps de calcul est consacrée à l'ajustement des hélices (Section 3.5). Notons que le temps d'exécution d'un algorithme implémenté sous Matlab est généralement plus long que pour le même algorithme implémenté en C++.

Les résultats de notre algorithme sont invariants par translation, rotation et changement d'échelle uniforme ; c'est à dire, la reconstruction de deux courbe 2D identiques à une

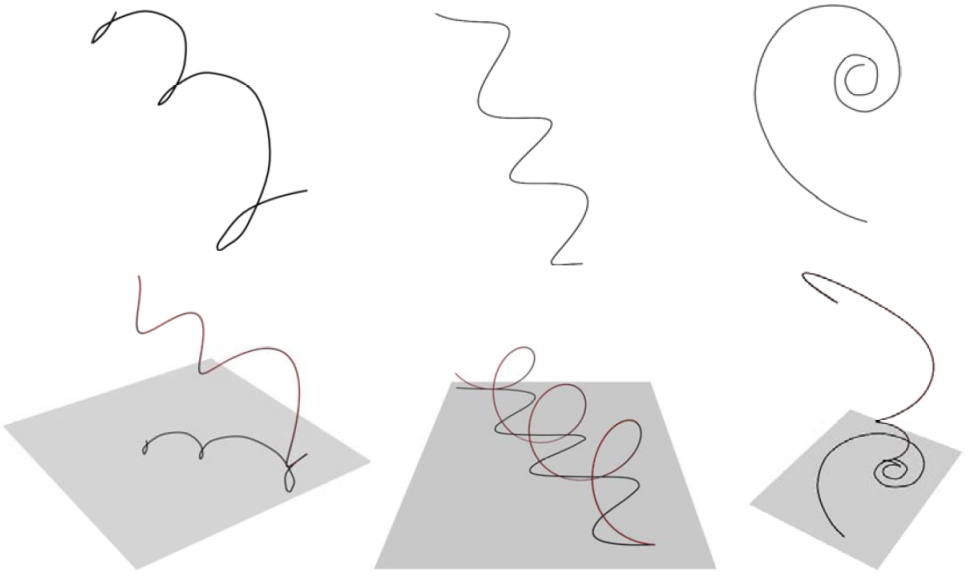

FIGURE 3.14 : Exemples d'hélices par morceaux générées par notre algorithme

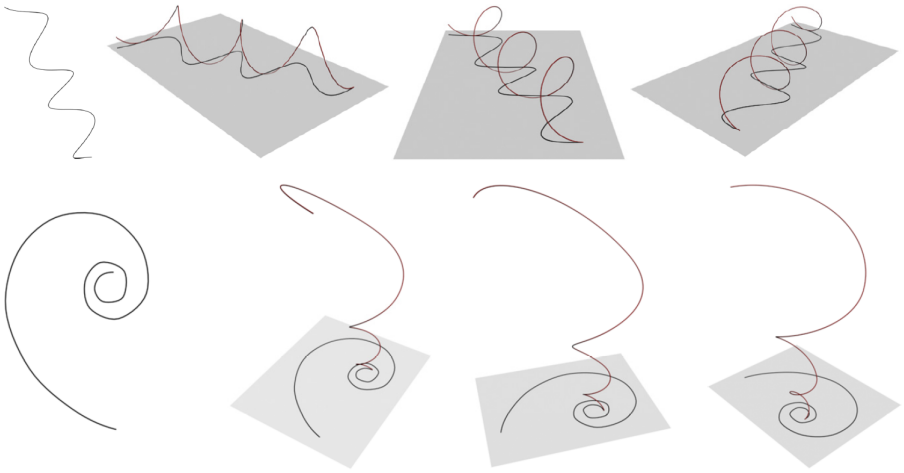

FIGURE 3.15 : Exemples d'hélices par morceaux générées par notre algorithme

transformation rigide et changement d'échelle près, donnera la même courbe 3D. Ceci est dû au fait que notre algorithme aligne l'hélice projetée avec la courbe d'entrée C_I.

3.2.6.1 Cas spéciaux

Dans cette section, nous analysons comment notre algorithme se comporte sur des lignes droites, cercles, ellipses et courbes présentant des angles aigus. Dans les cas des lignes droites, celles-ci ont une courbure qui est égale à 0 et leur vecteur tangent constant. La première étape de l'algorithme d'ajustement est la recherche du sommet de courbure maximale v_s. Dans le cas des lignes droites, n'importe quel sommet peut être choisi

puisqu'ils ont tous la même courbure. L'étape suivante est de calculer les coefficients r, p, α de l'hélice. Sur une ligne droite, le rayon r est égal à 0, ce qui fait de l'hélice une ligne droite. Notre algorithme se comporte de façon similaire sur les cercles, en effet le sommet de courbure maximal peut être n'importe quel sommet et l'étape de calcul des coefficients donne p égal à 0 et α à $\pi/2$. Ce qui implique que le pas de l'hélice est nul et le cercle reconstruit est sur un plan parallèle au plan de dessin ($z = 0$). On peut effectuer une analyse similaire dans le cas de la reconstruction d'ellipse.

Nous analysons maintenant comment notre algorithme se comporte pour les courbes présentant des angles aigus, ce qui correspond à un changement abrupt de la direction de la courbe où la courbure y est plus importante que la courbure des sommets voisins. Les hélices, projetées sur un plan peuvent produire des courbes présentant des angles aigus. Comme nous l'avons vu dans la section 3.2.2 la courbure $\kappa_H(t)$ de l'hélice projetée atteint sa valeur maximale quand le paramètre t est un multiple de π (0, π, etc.) et le dénominateur de $\kappa_H(t)$ est proche de 0, quand $bcos(\theta)$ est environ égal à $rsin(\theta)$ (voir équation 3.2.3). On peut remarquer sur l'expression de la tangente à la courbe (voir équation 3.11) que la tangente pour t proche de 0 ou π est parallèle à l'axe y. En d'autres termes, les tangentes à la courbe des deux côtés du point $H(0)$ sont parallèles entre elles. Ainsi, il existe deux cas possibles pour l'ajustement d'une hélice projetée à une courbe possédant des angles aigus. Soit v_{s-1} et v_{s+1} les deux sommets adjacents à v_s et $\hat{(T)}_{v_{s-1}}$ et $\hat{(T)}_{v_{s+1}}$ leur vecteur tangent respectivement. Dans le premier cas, ces deux vecteurs sont parallèles entre eux : les deux côtés de la courbe sont ajustés à l'aide de la même hélice projetée (voir figure 4.14 (a)). Dans le second cas, les deux vecteurs tangents ont une orientation différente. Deux hélices projetées différentes sont utilisées pour approximer les deux côtés de C_i au sommet v_s (voir figure 4.14 (b)). La courbe reconstruite présentera alors une discontinuité aux tangentes.

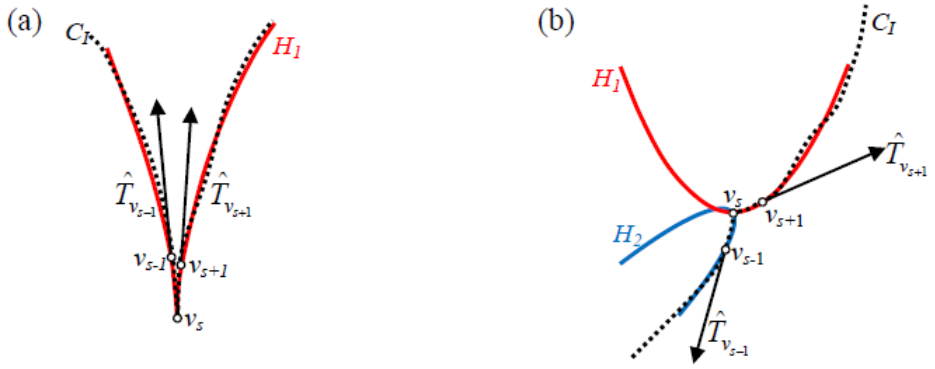

FIGURE 3.16 : (a) la courbe C_I est ajustée à l'aide d'une seule hélice projetée. (b) Les tangentes des deux cotés de v_s forment un angle supérieur à $\pi/2$. L'ajustement à C_I requiert deux hélices H_1 et H_2

3.2.7 Limitations

La courbe 3D générée par notre algorithme n'est pas G^0 continue, bien qu'en pratique, la distance entre les sommets de jonctions de deux hélices adjacentes est suffisamment petite pour considérer qu'ils ont les mêmes positions.

Notre algorithme ne produit pas non plus de courbe G^1 continue bien qu'il essaye de trouver une courbe composée de morceaux d'hélices telle que la discontinuité de tangente entre les hélices soit la plus faible possible. La plupart du temps, cette discontinuité est suffisamment petite pour qu'elle ne soit pas visible. Il existe des cas pour lesquelles l'hélice par morceaux reconstruite présente d'importantes discontinuités aux tangentes (voir figure 3.17).

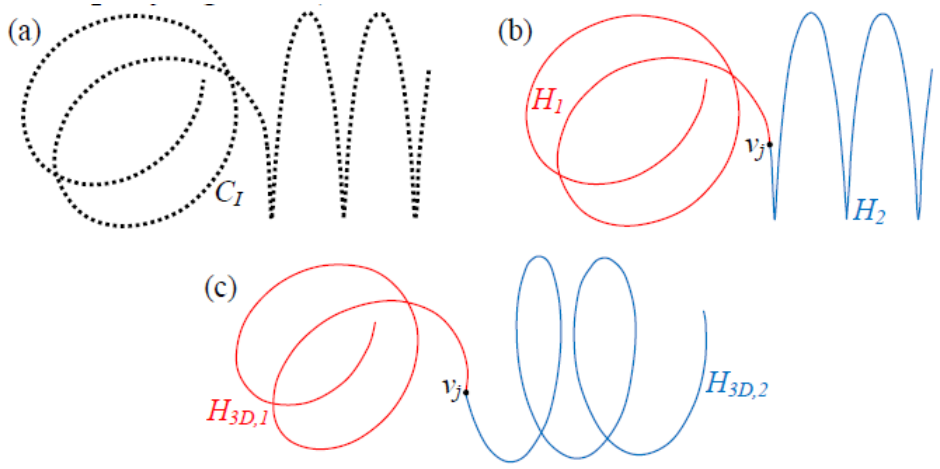

FIGURE 3.17 : Cas où l'hélice par morceaux reconstruite présente des discontinuités aux tangentes. (a) La courbe d'entrée C_I.(b) la courbe est approximée à l'aide de deux hélices projetées H_1 et H_2. v_j est le sommet de jonction. (c) Les tangentes au sommet de jonction ont différentes orientations. L'hélice par morceaux reconstruite n'est pas continue en G^1.

La plus grande faiblesse de cette méthode est qu'elle ne trouve pas de bons résultats si la courbe d'entrée C_I est bruitée. C'est pour cela que nous avons développé une deuxième méthode qui prend en compte l'intégralité des points de la courbe d'entrée pour calculer une hélice en trois dimensions.

Chapitre 4

Modélisation d'hélices

4.1 Vue d'ensemble de la méthode

Cette méthode prend en entrée une courbe 2D polygonale C dans le plan (x, y). Elle génère un segment d'hélice tel que sa projection orthogonale dans le plan (x, y) correspond à la courbe C. Au lieu de calculer l'ajustement de plusieurs morceaux d'hélices projetées avec la courbe C, cette fois nous simplifions le problème en échantillonnant l'hélice et en calculant l'alignement des points des points de l'hélice projetée avec ceux de la courbe C. Le problème revient à estimer la matrice de transformation qui aligne au mieux les points de l'hélice échantillonnée avec ceux de la courbe polygonale. En utilisant cette matrice de transformation, nous estimons les paramètres de l'hélice dont la projection correspond à la courbe polygonale C. Cette méthode est expliquée dans la section 4.2.

Pour que notre algorithme fonctionne, il faut qu'il y ait correspondance entre les points de l'hélice échantillonnée et ceux de la courbe polygonale. Ce qui implique que l'hélice et la courbe C doivent être échantillonnées de telle façon que cette correspondance soit valide. Dans la section 4.3, nous présentons une méthode d'échantillonnage adaptatif qui satisfait cette propriété.

Dans la section 4.5 nous produisons une comparaison détaillée entre cette méthode et la précédente et nous évaluons la robustesse de cette méthode. Dans la section 4.5.3 nous comparons les performances de notre algorithme aux performances d'un algorithme d'optimisation non linéaire.

4.2 Calcul des paramètres de l'hélice et de l'erreur d'ajustement

Sois C une courbe polygonale dans le plan (x, y). C est composée de n points $\{p_1, p_2, ..., p_i, ..., p_n\}$; chaque point p_i à pour coordonnées (x_{p_i}, y_{p_i}). Nous rappelons l'équation paramétrique de l'hélice :

$$H(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} r(cos(t) - 1) \\ pt \\ rsin(t) \end{pmatrix} \tag{4.1}$$

Où les coefficients r et p sont respectivement le rayon et le pas de l'hélice. De façon similaire à une courbe polygonale qui possède deux points terminaux, nous définissons deux points terminaux pour le segment d'hélice. Ces deux points sont $H(-\alpha)$ et $H(\alpha)$ où $-\alpha$ et α sont les valeurs que prends t aux points terminaux du segment d'hélice.

Nous formulons le problème de la façon suivante : étant donnée une courbe C dans le plan (x, y), le but est d'estimer les paramètres du segment d'hélice (le rayon r et le pas p) ainsi que sa matrice de rotation R telle que la projection orthogonale du segment d'hélice sur le plan (x, y) correspond à la courbe C.

L'idée principale de notre approche est d'échantillonner le segment d'hélice ainsi que la courbe C et de calculer la correspondance entre les deux ensembles de points obtenus de l'échantillonnage. En utilisant ces résultats, nous estimons ensuite les paramètres de l'hélice. Cette approche qui est basée sur l'échantillonnage de l'hélice et de la courbe d'entrée C implique que :

- La longueur du segment d'hélice (le paramètre α) soit définie. Nous discuterons par la suite du choix du paramètre α.
- Le segment d'hélice et la courbe C doivent être échantillonnés tel qu'il y ait une correspondance un à un entre les points de l'hélice et ceux de la courbe C. Dans le reste de cette section, nous assumons que cette correspondance existe.

L'ajustement de l'hélice à la courbe et l'estimation de ses paramètres est fait en trois étapes :

Dans la première étape, nous calculons un ensemble de points en échantillonnant de façon uniforme le segment d'hélice dont le rayon et le pas ont pour valeur initiale 1. Ensuite nous calculons la matrice de transformation optimale L qui aligne l'ensemble des points de l'hélice échantillonnée avec ceux de la courbe C. Puisque la matrice L n'est pas contrainte à une matrice orthonormale, la transformation n'est pas seulement

une rotation . Par conséquent, après transformation par L la courbe obtenue peut ne plus être une hélice. Afin de préserver les propriétés des hélices, la matrice L doit être seulement équivalente à une matrice de rotation. C'est pourquoi L doit avoir des colonnes orthonormales.

Dans un second temps, nous modifions les paramètres de l'hélice (rayon et pas) tel que la matrice de transformation linéaire L devient proche d'une matrice avec des colonnes orthonormales.

Une fois que les paramètres des hélices ont été estimés, la troisième étape est de calculer la matrice de rotation optimale R qui aligne les points de l'hélice avec les points de la courbe d'entrée C.

La dernière étape consiste à calculer les coordonnée z des points de l'hélice, pour cela nous calculons une matrice de rotation en 3D dimensions à partir de celle calculée dans l'étape précédente.

4.2.1 Calcul de la matrice de transformation L

Dans cette section nous allons montrer comment calculer la matrice de transformation L qui permet d'aligner les points d'un segment d'hélice sur les points de la courbe C.

Nous prenons comme exemple de courbe d'entrée la figure suivante (figure 4.1), obtenue en projetant de façon orthogonale une hélice avec les paramètres suivants : $r = 2, p = 2, \alpha = 2\pi$, cette hélice avant projection a subi une rotation selon l'axe x de 1.25 radian. Celle-ci a été ensuite légèrement bruitée.

Sois $H_u(t)$ l'hélice dont le rayon et le pas sont égaux à 1.

$$H_U(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} = \begin{pmatrix} cos(t) - 1 \\ t \\ sin(t) \end{pmatrix} \tag{4.2}$$

Nous calculons un échantillonnage uniforme de la courbe $H_U(t)$ avec un ensemble de n valeurs de t distribuées de façon égale sur l'intervalle $[-\alpha, +\alpha]$, n étant le nombre de points de la courbe polygonale C. Cet échantillonnage uniforme est un ensemble de valeur $\{\alpha_0, \alpha_1, ..., \alpha_i, ...\alpha_n\}$ avec $\alpha_i = \frac{2\alpha}{n-1} i - \alpha$ et i variant de 0 à $n-1$.

Le résultat de cet échantillonnage est un ensemble de points $\{H_U(\alpha_0), H_U(\alpha_1), ..., H_U(\alpha_{n-1})\}$. Le choix de α qui définit les points terminaux du segment d'hélice est discuté dans la section 4.2.4.

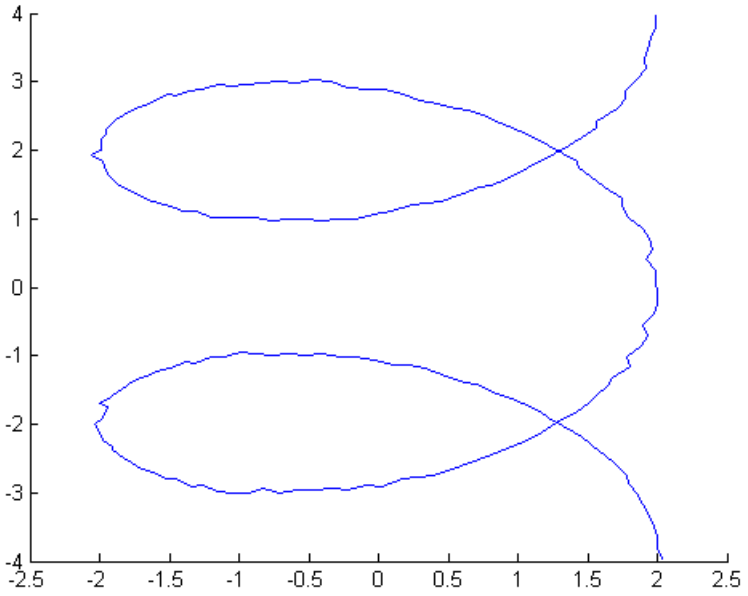

FIGURE 4.1 : Courbe d'entrée C

Pour le moment nous considérons que α est une valeur fixée

Sans perte de généralité, nous considérons que les centroids des deux ensembles de points (les points de l'hélice et les points de la courbe C) coïncident avec l'origine du système de coordonnées.

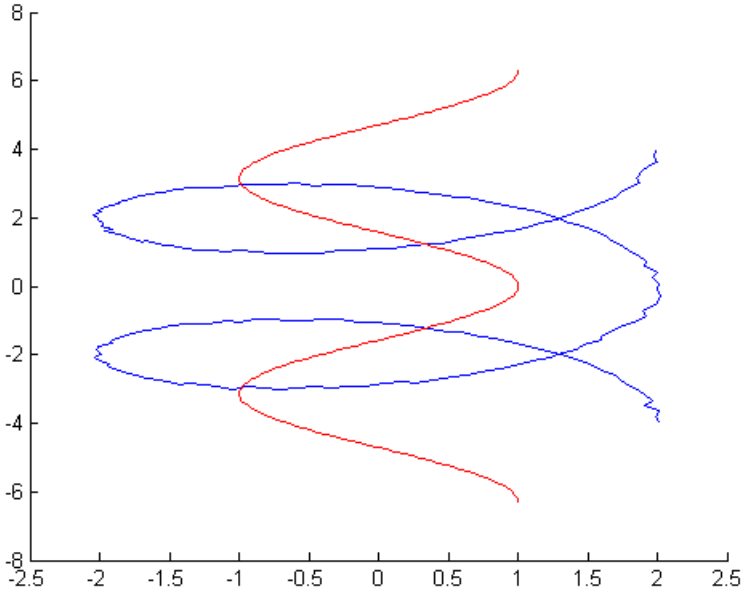

FIGURE 4.2 : En bleu courbe d'entrée C, en rouge la courbe $H_{U,\alpha}$

Étant donné les ensembles des points de l'hélice et de la courbe C, nous calculons une matrice de transformation linéaire L qui aligne au mieux les points de l'hélice échantillonnée avec ceux de C. Cette matrice est la matrice de transformation optimale L qui minimise la distance entre les points de la courbe C et les points de l'hélice. La matrice de transformation L est calculée minimisant la fonction objectif suivante 4.3 :

$$||M_{U,\alpha}L - M_C||_F^2 \tag{4.3}$$

Où $||.||_F$ est la norme de Frobenius.

$M_{U,\alpha}$ et M_C sont respectivement les matrices représentant les coordonnées obtenues par l'échantillonnage du segment d'hélice et de la courbe C.

$$M_{U,\alpha} = \begin{pmatrix} cos(-\alpha) & -\alpha & sin(-\alpha) \\ cos(\frac{2\alpha}{n-1} - \alpha) & \frac{2\alpha}{n-1} - \alpha & sin(\frac{2\alpha}{n-1} - \alpha) \\ . & . & . \\ . & . & . \\ cos(\alpha) & \alpha & sin(\alpha) \end{pmatrix} M_C = \begin{pmatrix} x_{P,1} & y_{P,1} \\ x_{P,2} & y_{P,2} \\ . & . \\ . & . \\ . & . \\ x_{P,n} & y_{P,n} \end{pmatrix} \tag{4.4}$$

La matrice de transformation L qui est solution du problème d'optimisation ci-dessus est :

$$L = (M_{U,\alpha}^T M_{U,\alpha})^{-1} M_{U,\alpha}^T M_C \tag{4.5}$$

Comme L est calculée à l'aide des points en trois dimensions de l'hélice et les points en deux dimensions de la courbe C, les dimensions de la matrice L sont donc de trois lignes par deux colonnes :

$$L = \begin{pmatrix} l_{1,1} & l_{1,2} \\ l_{2,1} & l_{2,2} \\ l_{3,1} & l_{3,2} \end{pmatrix} \tag{4.6}$$

4.2.1.1 Estimation du rayon et du pas de l'hélice

L'idée est de modifier les coefficients de l'hélice (rayon et pas) de façon à rendre la matrice de transformation proche d'une matrice ayant des colonnes orthonormales. Sois r et p respectivement le rayon et le pas de l'hélice. La matrice représentant les coordonnées de l'hélice avec r et p est :

$$\begin{pmatrix} rcos(-\alpha) & -p\alpha & rsin(-\alpha) \\ rcos(\frac{2\alpha)}{n-1} - \alpha) & p(\frac{2\alpha}{n-1} - \alpha) & rsin(\frac{2\alpha}{n-1} - \alpha) \\ . & . & . \\ . & . & . \\ . & . & . \\ rcos(\alpha) & p\alpha & rsin(\alpha) \end{pmatrix} = M_{U,\alpha} M_{rp} \tag{4.7}$$

avec

$$M_{rp} = \begin{pmatrix} r & 0 & 0 \\ 0 & p & 0 \\ 0 & 0 & r \end{pmatrix} \tag{4.8}$$

Nous pouvons réécrire la fonction objectif de la façon suivante :

$$||(M_{U,\alpha} M_{rp})(M_{rp}^{-1} L) - M_C||_F^2 \tag{4.9}$$

Le but est de calculer les paramètres inconnus r et p tel que la matrice $(M_{rp}^{-1} L)$ devient proche d'une matrice avec des colonnes orthonormales. Si $(M_{rp}^{-1} L)$ est une matrice avec des colonnes orthonormales, alors l'expression $(M_{rp}^{-1} L)^T (M_{rp}^{-1} L)$ est égale à la matrice identitée. La fonction objectif à minimiser est donc la suivante :

$$||(M_{rp}^{(-1)} L)^T (M_{rp}^{(-1)} L) - I||_F^2 \tag{4.10}$$

Où I est la matrice identité.

$$(M_{rp}^{(-1)} L)^T (M_{rp}^{(-1)} L) = \begin{pmatrix} \frac{l_{2,1}^2}{p^2} + \frac{l_{1,1}^2 + l_{3,1}^l}{r^2} & \frac{l_{2,1} l_{2,2}}{p^2} + \frac{l_{1,1} l_{1,2} + l_{3,1} l_{3,2}}{r^2} \\ \frac{l_{2,1} l_{2,2}}{p^2} + \frac{l_{1,1} l_{1,2} + l_{3,1} l_{3,2}}{r^2} & \frac{l_{2,2}^2}{p^2} + \frac{l_{1,2}^2 + l_{3,2}^l}{r^2} \end{pmatrix} \tag{4.11}$$

Après simplification, la fonction objectif devient :

$$||AX - B||_F^2 \tag{4.12}$$

Avec

$$A = \begin{pmatrix} l_{2,1}^2 & l_{1,1}^2 + l_{3,1}^2 \\ l_{2,1}l_{2,2} & l_{1,1}l_{1,2} + l_{3,1}l_{3,2} \\ l_{2,1}l_{2,2} & l_{1,1}l_{1,2} + l_{3,1}l_{3,2} \\ l_{2,2}^2 & l_{1,2}^2 + l_{3,2}^2 \end{pmatrix}, X = \begin{pmatrix} \frac{1}{p^2} \\ \frac{1}{r^2} \end{pmatrix} B = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{4.13}$$

r et p étant les variables inconnues. Le vecteur X qui est solution du problème d'optimisation est :

$$X = (A^T A)^{-1} A^T B \tag{4.14}$$

Les paramètres de l'hélice r et p sont facilement calculés en utilisant le vecteur X.

4.2.2 Estimation de la matrice de rotation

Une fois que les paramètres r et p ont été calculés, la prochaine étape est d'estimer la matrice de rotation R qui minimise la différence entre les points de l'hélice et les points de la courbe polygonale C. La matrice contenant les coordonnées de l'hélice avec les paramètres r, p et α est :

$$M_{U,\alpha} = \begin{pmatrix} rcos(\alpha) & -p\alpha & rsin(\alpha) \\ rcos(\frac{2\alpha}{n-1} - \alpha) & p(\frac{2\alpha}{n-1} - \alpha) & rsin(\frac{2\alpha}{n-1} - \alpha) \\ . & . & . \\ . & . & . \\ . & . & . \\ rcos(\alpha) & p\alpha & rsin(\alpha) \end{pmatrix} \tag{4.15}$$

La matrice de rotation R est estimée en résolvant le problème d'optimisation suivant :

$$||(M_{U,\alpha} M_{rp})R - M_C||_F^2 \tag{4.16}$$

Avec la contrainte $R^T R = I$. Ce problème est le problème de Procuste orthogonal où une matrice R est contrainte à une matrice orthonormale. Dans notre cas, ce problème est non-balancé, car $(M_U M_{rp})$ et M_C n'ont pas les mêmes dimensions. Pour résoudre ce problème, nous calculons la matrice orthogonale la plus proche de la matrice $M_{rp}^{-1} L$. Pour trouver une matrice orthonormale Q qui soit la plus proche d'une matrice M nous utilisons la formule :

$$Q = M(M^T M)^{-\frac{1}{2}} \tag{4.17}$$

La matrice R est donc égale à :

$$R = M_{rp}^{-1} L((M_{rp}^{-1} L)^T M_{rp}^{-1} L)^{-\frac{1}{2}} \tag{4.18}$$

L'erreur entre les points de l'hélice et les points de la courbe polygonale C est :

$$Err(r, p, \alpha, R) = ||(M_{U,\alpha} M_{rp})R - M_C||_F^2 \tag{4.19}$$

Comme nous fixons α, r et p pour calculer la matrice de rotation R, la solution que nous trouvons n'est pas optimale. Malgré cela, nous montrons dans la section 4.5.3 que les solutions trouvées par notre méthode sont bonnes.

4.2.3 Calcul de l'hélice en trois dimensions

Afin de calculer les coordonnées 3D des points de l'hélice, il est impératif de transformer la matrice de rotation R en une matrice carrée de dimension 3. Pour préserver la rotation, il faut que celle-ci soit orthonormale.

$$R_{3D} = \begin{pmatrix} R_{1,1} & R_{1,2} & a \\ R_{2,1} & R_{2,2} & b \\ R_{3,1} & R_{3,2} & c \end{pmatrix} \tag{4.20}$$

Afin de calculer les coefficients a, b, c de la troisième colonne de cette nouvelle matrice de rotation R_{3D}, nous posons un système de trois équations. Comme la matrice R_{3D} est orthonormale nous pouvons écrire :

$$R_{3D}^T R_{3D} = I \tag{4.21}$$

Nous pouvons en déduire les équations suivantes :

$$S = \begin{cases} aR_{1,1} + bR_{2,1} + cR_{3,1} = 0 \\ aR_{1,2} + bR_{2,2} + cR_{3,2} = 0 \\ R_{1,1}^2 + R_{1,2}^2 + a^2 = 1 \end{cases} \tag{4.22}$$

La troisième équation provient du fait que la matrice R_{3D} possède des lignes orthonormales, donc le produit scalaire d'une ligne par elle-même est égal à 1. L'hélice en 3D se calcule de la façon suivante :

$$H_{3D} = (M_{U,\alpha} M_{rp}) R_{3D} \tag{4.23}$$

La figure 4.3 montre le résultat du calcul de l'hélice en 3D (en vert) pour notre exemple.

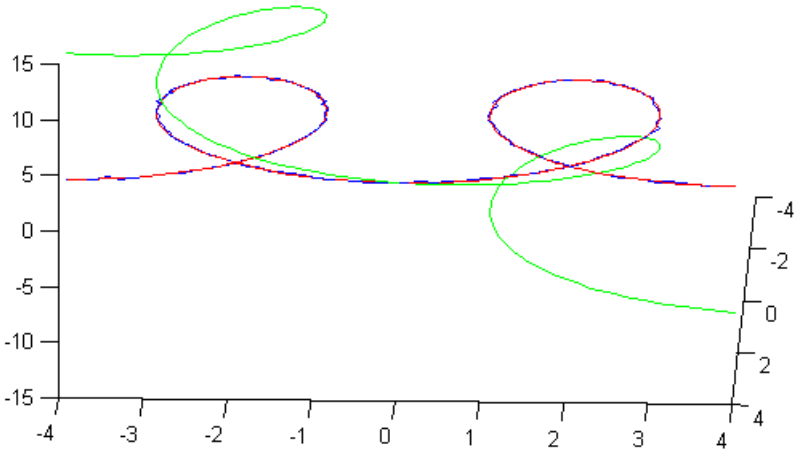

FIGURE 4.3 : L'hélice en 3D apparait en vert

4.2.4 Estimation du paramètre α

Dans les sections précédentes, nous avions considéré que le paramètre α des deux points terminaux de l'hélice ($H(-\alpha)$ et $H(\alpha)$) était fixé. Il est possible d'obtenir différents résultats en modifiant la valeur de α (voir figure 4.4).

Nous cherchons la valeur de α tel que l'erreur soit minimale(équation 4.19) . Nous calculons donc les paramètres des hélices pour un certain nombre de valeurs de α comprises dans un intervalle $]A, B]$. Dans notre implémentation nous supposons que la longueur de la courbe C est assez petite, nous avons donc choisi de calculer les paramètres de l'hélice ainsi que sa matrice de rotation R pour 500 valeurs de α comprises dans l'intervalle $]0, 10]$.

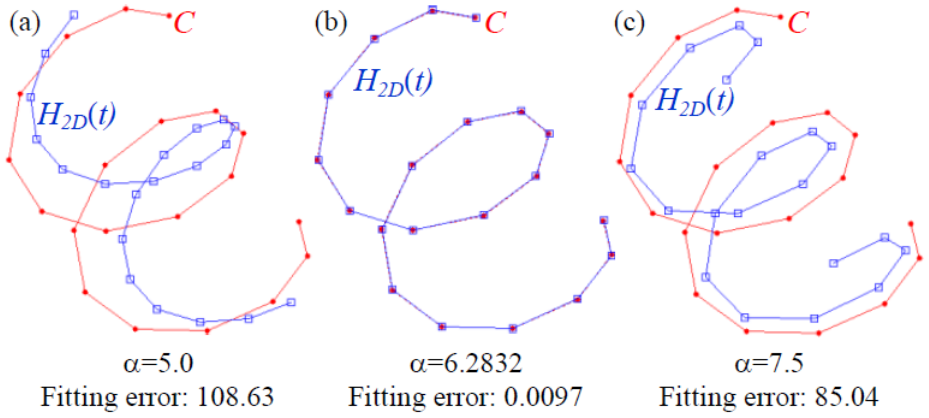

FIGURE 4.4 : Différents résultats (en rouge) pour différentes valeurs de $ij\alpha$

4.3 Échantillonnage adaptatif de la courbe polygonale C

Notre méthode suppose qu'il existe une correspondance entre chaque point de l'hélice et chaque point de la courbe C. Cela implique que l'hélice et la courbe C ont le même échantillonnage. La difficulté est donc d'échantillonner la courbe C de façon à établir cette correspondance.

Comme nous l'avons mentionné dans la section 4.2, le segment d'hélice est échantillonné de façon uniforme, c'est-à-dire que la distance entre deux points consécutifs est toujours la même. L'échantillonnage de la courbe C est un peu plus compliqué. Contrairement à l'hélice, l'échantillonnage uniforme de la courbe C ne fonctionne pas. En effet, nous supposons que la courbe C est la projection orthogonale d'une hélice et cette projection ne conserve pas la distance entre les points en 3D (voir figure 4.5). Les points sont plus rapprochés quand la courbure est importante sur la projection de l'hélice et inversement quand la courbure est faible.

Notre solution à ce problème est de calculer l'échantillonnage de la courbe C en utilisant la longueur d"arc et l'échantillonnage de l'hélice projetée qui a été ajustée à la courbe C. Notre algorithme d'échantillonnage fonctionne de façon itérative sur les points de l'échantillonnage de l'hélice. Étant donné un point $H_{2D}(t_j)$ de l'hélice projetée, nous calculons sa longueur d'arc. Le point correspondant à $H_{2D}(t_j)$ sur C est alors localisé le long de C en utilisant la longueur d'arc de $H_{2D}(t_j)$.

Dans un premier temps, nous fournissons l'expression analytique de la longueur d'arc du segment d'hélice projetée. Sois $H(t)$ une hélice et R sa matrice de rotation. Sans perte de généralité, nous supposons que R inclus seulement une rotation selon l'axe x les rotations selon les deux autres axes n'ont pas d'effet sur l'échantillonnage.

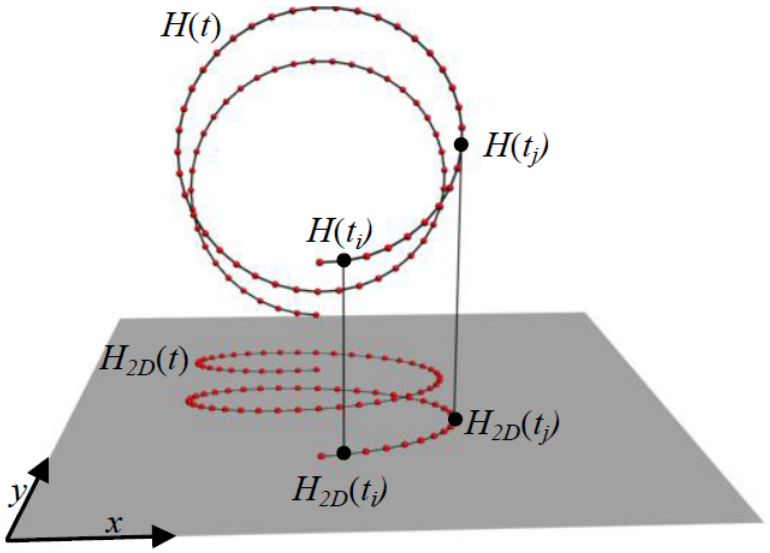

FIGURE 4.5 : Le segment d'hélice $H(t)$ échantillonné de façon uniforme et sa projection orthogonale $H_{2D}(t)$ sur le plan (x, y). Contrairement au segment d'hélice, l'échantillonnage du segment d'hélice n'est pas uniforme.

$$H(t) = \begin{pmatrix} rcos(t) \\ pt \\ rsin(t) \end{pmatrix} R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & sin(\theta) \\ 0 & -sin(\theta) & cos(\theta) \end{pmatrix} \tag{4.24}$$

θ, r et p sont respectivement l'angle de rotation autour de l'axe x, le rayon et le pas de l'hélice léquation de l'hélice après rotation et projection sur le plan (x, y) est la suivante :

$$H_{2D}(t) = \begin{pmatrix} rcos(t) \\ ptcos(\theta) + rsin(\theta)sin(t) \\ 0 \end{pmatrix} \tag{4.25}$$

L'expression analytique de la longueur d'arc $l_{H2D}(t_i, t_j)$ de la projection du segment d'hélice dont les points terminaux sont $H_{2D}(t_i)$ et $H_{2D}t_j$ est :

$$l_{H2D}(t_i, t_j) = \int_{t_i}^{t_j} ||\frac{d}{dt} H_{2D}(t)||dt \tag{4.26}$$

Ce qui équivaut à :

$$\int_{t_i}^{t_j} sqrt(\frac{d}{dt}(rcos(t)))^2 + (\frac{d}{dt}(ptcos(\theta) + rsin(\theta)sin(t)))^2 dt \tag{4.27}$$

Assumons qu'une projection d'un segment d'hélice a été ajustée à la courbe C et que les paramètres r, p, θ ainsi que les deux points terminaux $H_{2D}(-\alpha)$ et $H_{2D}(\alpha)$ soient connus.

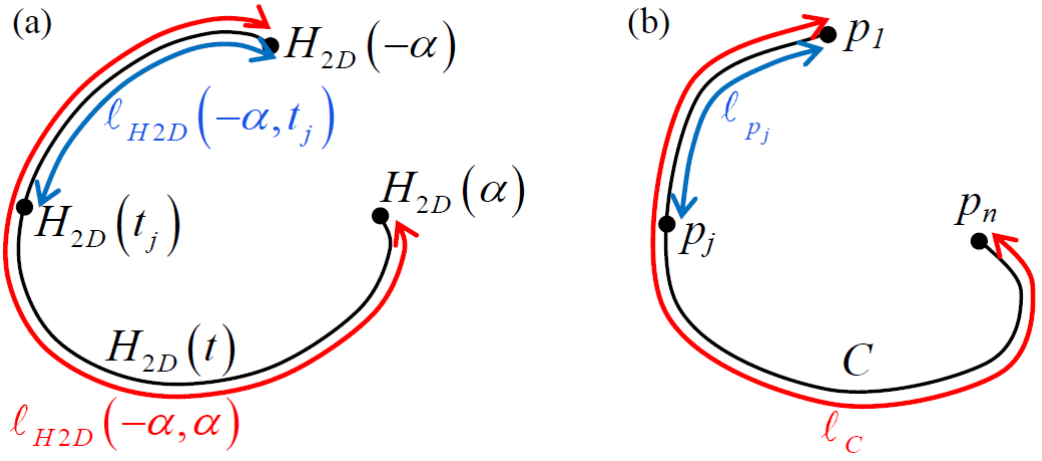

FIGURE 4.6 : Le point p_j correspondant à $H_{2D}(t_j)$ est localisé le long de C tel que sa longueur d'arc sois égale à l_{p_j}.

L'échantillonnage adaptatif de C est calculé de façon itérative sur les n points de l'échantillonnage de l'hélice projetée. Étant donné un point $H_{2D}(t_j)$ de l'hélice projetée, nous calculons sa longueur d'arc $l_{H2D}(-\alpha, t_j)$. Le point p_j de C et qui correspond à $H_{2D}(t_j)$ est localisé le long de C tel que sa longueur d'arc l_{p_j} sois égale à

$$l_{p_j} = l_{H2D}(-\alpha, t_j)\frac{l_c}{l_{H2D}(-\alpha, \alpha)} \tag{4.28}$$

$l_{H2D}(-\alpha, \alpha)$ et l_c sont respectivement la longueur d'arc de l'hélice projeté et la longueur d'arc de la courbe.

4.4 Utilisation de l'échantillonnage adaptatif en combinaison avec l'ajustement de l'hélice à la courbe

Comme mentionné précédemment, l'échantillonnage adaptatif est possible seulement si l'hélice projetée a été initialement ajustée à la courbe C et si les paramètres (r, p, α, et R) ont été calculés. En fait, notre méthode d'échantillonnage adaptatif et la méthode d'ajustement de l'hélice à la courbe sont utilisées alternativement afin de calculer des approximations successives des paramètres de l'hélice. Nous générons dans un premier temps une courbe C_{samp} en échantillonnant de façon uniforme la courbe C (ligne 4 dans le pseudo-code ci-dessous). Les paramètres de l'hélice sont estimés en ajustant

l'hélice à C_{samp} (ligne 9). Ces paramètres sont ensuite utilisés afin de calculer un nouvel échantillonnage de C_{samp} (ligne 10).

Ce processus est itéré jusqu'a ce que la différence entre deux valeurs successives de l'erreur d'ajustement est inférieur à un seuil ϵ. L'erreur d'ajustement est calculée à l'aide de l'équation 4.19.

Le pseudo-code de l'échantillonnage adaptatif est donné ci-dessous :

```
1: Compute_Adaptive_Sampling_and_Helix_Matching(C)
2:   Input: C
3:   Output: r,p,α,R,T
4:   C_samp ← Uniform_Sampling_Curve(C)
5:   T ← Compute_Translation_Component(C_samp)
6:   err ← +∞;
7:   do
8:   |   prevErr ← err;
9:   |   r,p,α,R,err ← Helix_Fitting(C_samp);
10:  |   C_samp ← Adaptive_Sampling_Curve(C,r,p,α,R);
11:  while (prevErr-err)>ε
12:  return r, p, α, R and T
```

4.5 Résultats

Dans un premier temps nous allons comparer cette méthode avec la précédente, puis dans un second temps nous allons regarder comment se comporte notre algorithme sur des courbes d'entrées bruitées. Ensuite nous comparons notre méthode avec un algorithme d'optimisation non linéaire.

4.5.1 Comparaison avec la méthode précédente

Dans cette section nous comparons les résultats de cette méthode avec ceux de la méthode précédente.

La différence majeure qui existe entre ces deux méthodes est la suivante : dans la méthode précédente l'ajustement de l'hélice à la courbe C est calculée de façon locale, c'est-à-dire que l'hélice est calculée au voisinage du point de courbure maximale. Au contraire, notre deuxième méthode prend en compte tous les points de la courbe C pour calculer l'hélice.

Comme le montre la figure 4.7, la méthode précédente n'arrive pas à reconstruire une courbe dont la projection orthogonale est proche de la courbe C. En effet une petite

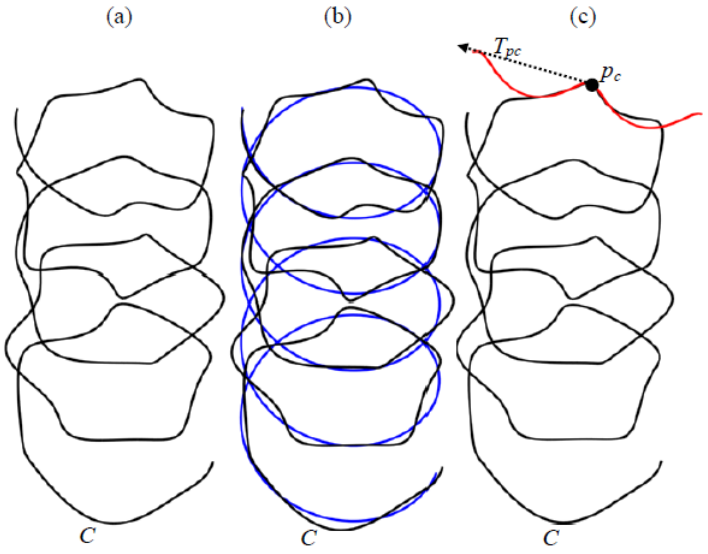

FIGURE 4.7 : (a) Courbe d'entrée C. (b) Hélice produite par la deuxième méthode. (c) Hélice produite par la méthode précédente

modification des coordonnées de la courbe au point de courbure maximale entraine un changement de l'orientation de la tangente T_{pc} et ainsi génère une erreur importante de l'orientation de l'hélice.

Nous comparons les temps d'exécution de nos deux algorithmes sur un exemple : nous générons une courbe d'entrées à l'aide d'hélices que nous projetons dans le plan et mesurons le temps d'exécution de nos deux algorithmes. La première courbe d'entrée a été générée à l'aide d'une hélice possédant les paramètres suivants : $r = 1.5$, $p = 1.5$, l'angle de rotation selon x est $\theta = 1.35$.

La courbe d'entrée C est montrée dans la figure suivante (figure 4.8).

Les temps de calcul sont d'environ 5.1578 secondes pour la première méthode et de 0.19 seconde pour la deuxième méthode pour un résultat équivalent (voir figure 4.9).

4.5.2 Reconstitution de courbes bruitées

Afin de démontrer la robustesse de notre méthode de reconstruction par rapport au bruit, nous avons conduit un ensemble d'expérience avec une courbe planaire C qui a été généré en projetant de façon orthogonale un segment d'hélice puis en bruitant cette projection.

Dans les expériences qui suivent le nombre de points de la courbe C est 600. Nous mesurons l'erreur de la façon suivante : Nous cherchons le point de C dont la distance au point correspondant sur l'hélice est la plus grande. L'erreur est ensuite calculée en

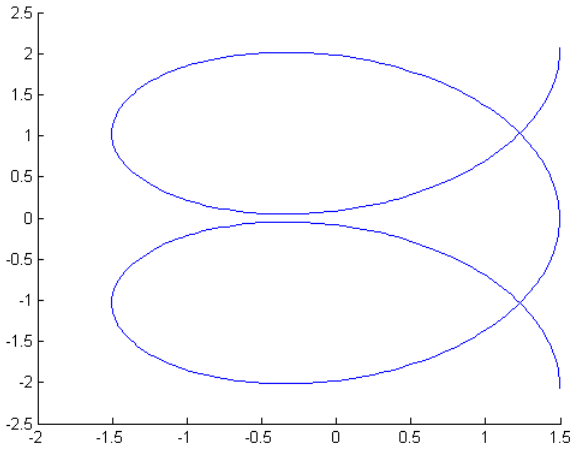

FIGURE 4.8 : Courbe d'entrée C utilisée pour comparer le temps d'exécution des deux algorithmes

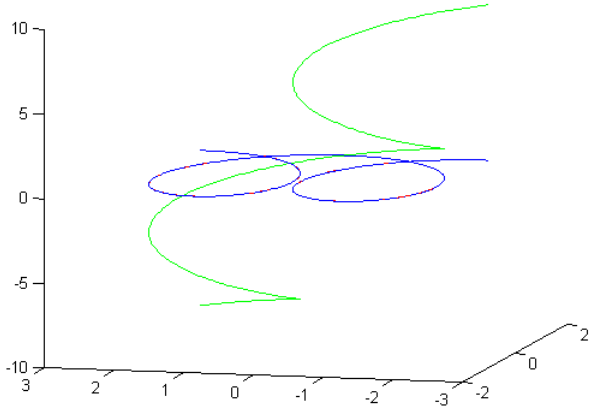

FIGURE 4.9 : Résultat

divisant cette distance par la longueur de la courbe C. Pour chaque figure l'erreur est montrée en pourcentage.

Les courbes sont modifiées en faisant varier la force du bruit (magnitude) sur deux types de fréquences : haute et basses. Dans les figures qui suivent la première ligne, montre l'utilisation de bruits haute fréquence et la deuxième ligne montre l'utilisation de bruits basse fréquence.

Pour la première expérience nous générons une hélice avec les paramètres suivants : $r = 4$, $p = 1$ et $\alpha = 4\pi$. Les résultats de l'expérience sont montrés dans les figures 4.10 et 5.32 . Comme nous pouvons le remarquer, notre algorithme réussit à reconstituer l'hélice et à retrouver les paramètres initiaux.

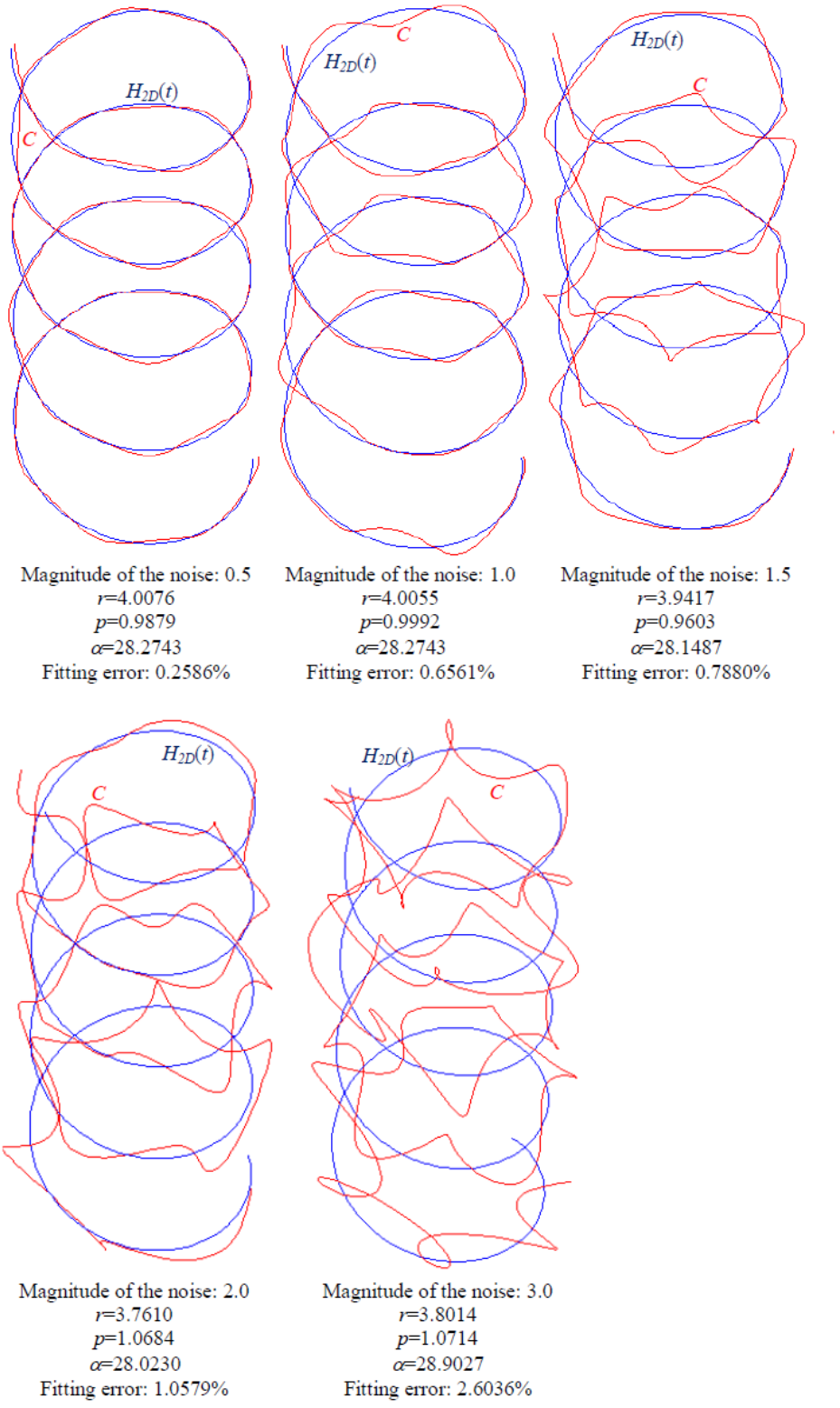

FIGURE 4.10 : En rouge la courbe d'entrée C pour différent niveau de bruits. La courbe bleue est le résultat de notre algorithme

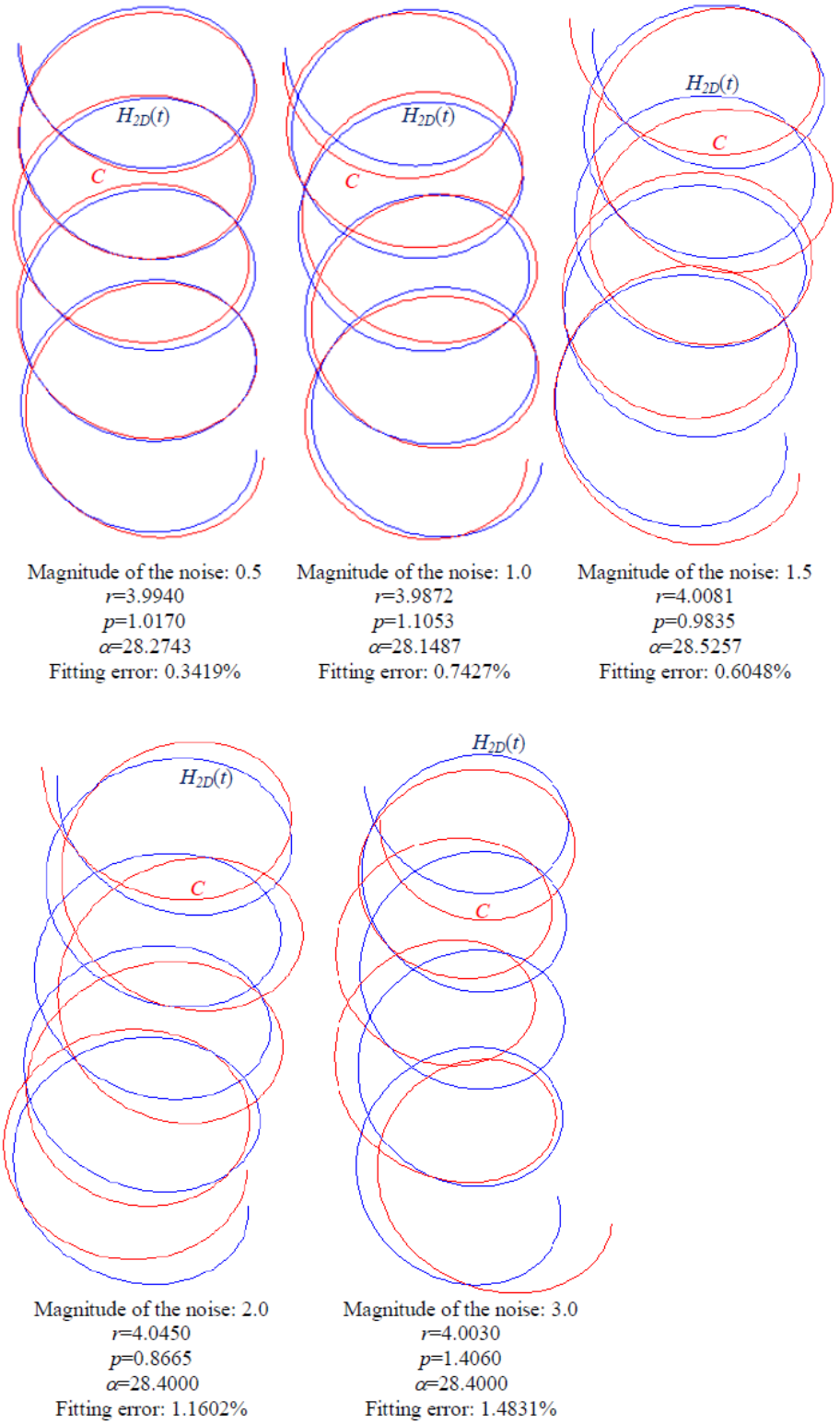

FIGURE 4.11 : En rouge la courbe d'entrée C pour différent niveau de bruits. La courbe bleue est le résultat de notre algorithme

La seconde expérience (voir figures4.12 et 4.13) est la même que précédemment sauf que la courbe C est bien plus courte, en effet dans cette expérience le paramètre α est égal à 1.5π. Comme nous pouvons l'observer, la qualité de reconstruction est inférieure à l'expérience précédente. Cela s'explique par le fait que la longueur de l'hélice à reconstruire est plus petite.

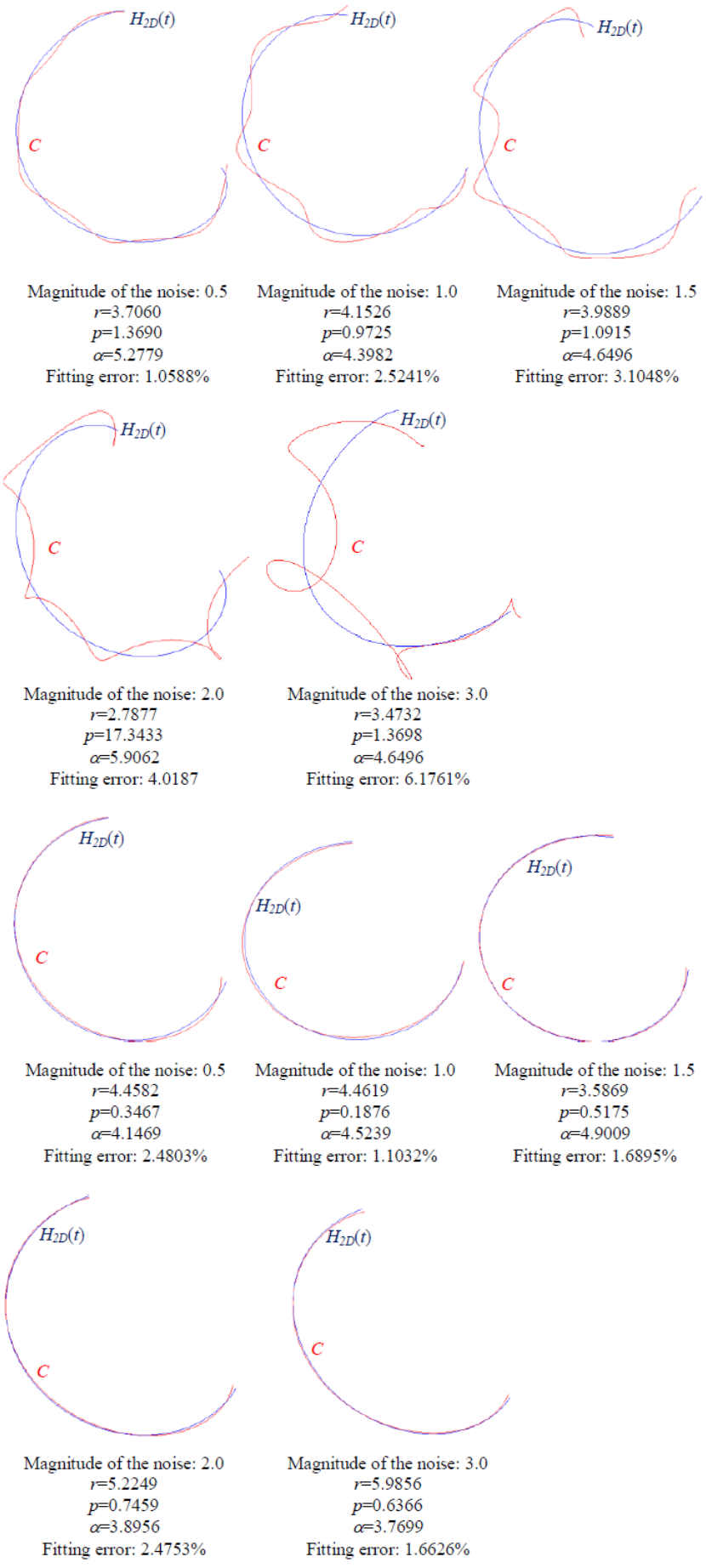

FIGURE 4.12 : En rouge la courbe d'entrée C pour différent niveau de bruits. La courbe bleue est le résultat de notre algorithme

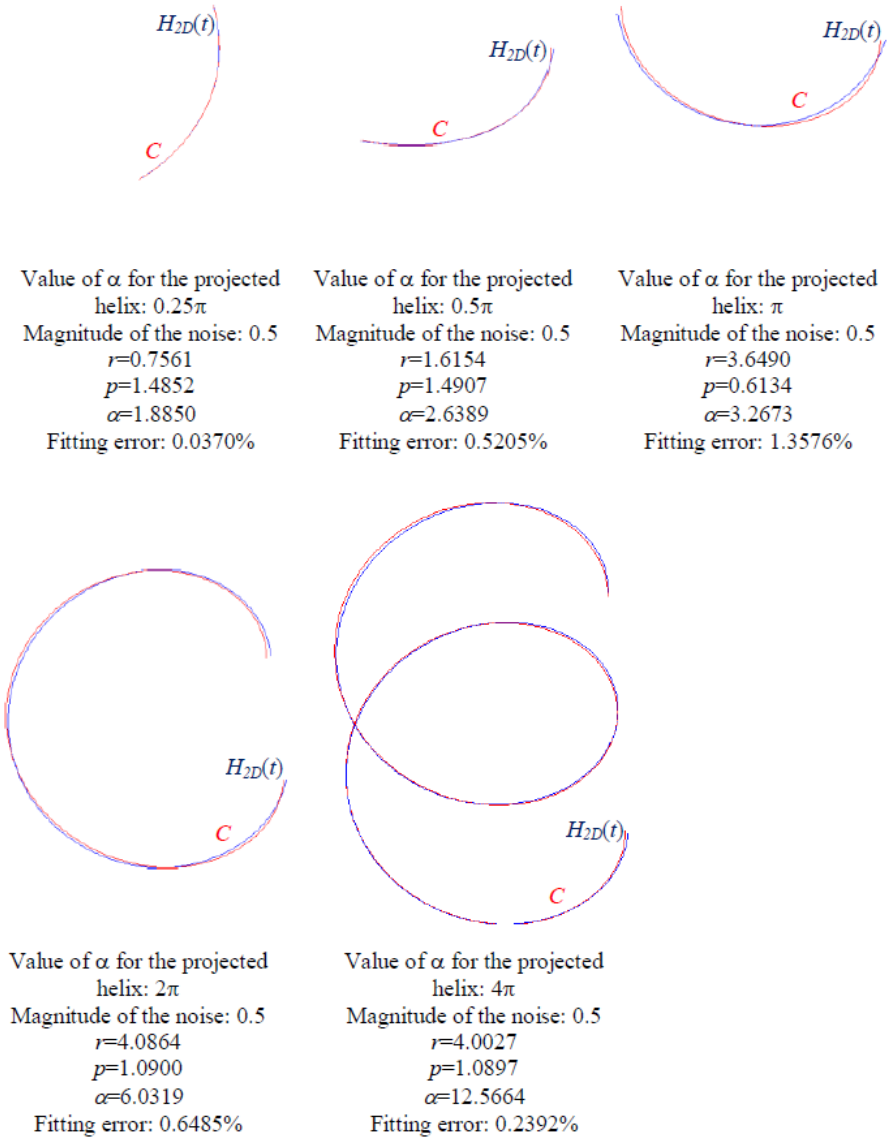

FIGURE 4.13 : En rouge la courbe d'entrée C pour différent niveau de bruits. La courbe bleue est le résultat de notre algorithme

4.5.3 Comparaison avec une méthode d'optimisation non linéaire

Dans cette section nous allons montrer que les solutions trouvées par notre algorithme peuvent être utilisées comme solutions initiales pour un algorithme d'optimisation non linéaire. Pour cela nous allons le comparer avec la méthode d'optimisation de Levenberg-Marquardt. L'algorithme de Levenberg-Marquardt est un algorithme qui permet d'obtenir une solution numérique à un problème de minimisation d'une fonction (linéaire ou non) dépendant de plusieurs variables. Nous utilisons cet algorithme, car c'est un algorithme commun d'optimisation non linéaire et que celui-ci est intégré dans matlab .

Nous allons procéder comme suis, dans un premier temps nous utilisons notre méthode pour fournir une solution initiale à l'algorithme de Levenberg-Marquardt, puis dans un second temps nous utiliserons seulement l'algorithme de Levenberg-Marquardt pour trouver une solution à notre problème, nous comparerons les valeurs trouvées par les deux approches ainsi que les temps de calcul.

Nous reformulons le problème (équation 4.16) qui consiste à déterminer la matrice orthonormale Z de taille 3×2 ainsi que les paramètres r et p qui minimisent la fonction objectif suivante :

$$||M_{U,\alpha} M_{rp} Z - M_c||^2 \tag{4.29}$$

Avec $Z^T Z = I$. Nous allons exprimer la matrice de rotation Z à l'aide de 3 paramètres : ϕ, θ et β qui sont respectivement les angles de rotations autour des axes x,y et z. Les matrices R_x, R_z et R_y qui sont les matrices de rotations autour des axes x,y et z s'écrivent de la façon suivante :

$$R_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{pmatrix} R_y(\theta) = \begin{pmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{pmatrix} R_z(\beta) = \begin{pmatrix} cos(\beta) & -sin(\beta) & 0 \\ sin(\beta) & cos(\beta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.30}$$

Nous pouvons écrire la matrice suivante Z_{3D} qui est le produit des 3 matrices de rotations autour des axes x, y et z :

$$Z_{3D} = \begin{pmatrix} cos(\beta)cos(\theta) & -sin(\beta) & cos(\beta)sin(\theta) \\ sin(\phi)sin(\theta) + cos(\phi)sin(\beta)cos(\theta) & cos(\beta)cos(\phi) & cos(\phi)sin(\beta)sin(\theta) - cos(\theta)sin(\phi) \\ sin(\beta)cos(\theta)sin(\phi) - cos(\phi)sin(\theta) & cos(\beta)sin(\phi) & cos(\phi)cos(\theta) + sin(\beta)sin(\phi) * sin(\theta) \end{pmatrix} \tag{4.31}$$

Comme la matrice Z est de dimension 3×2 nous ne conservons que les deux premières colonnes :

$$Z = \begin{pmatrix} cos(\beta)cos(\theta) & -sin(\beta) \\ sin(\phi)sin(\theta) + cos(\phi)sin(\beta)cos(\theta) & cos(\beta)cos(\phi) \\ sin(\beta)cos(\theta)sin(\phi) - cos(\phi)sin(\theta) & cos(\beta)sin(\phi) \end{pmatrix} \tag{4.32}$$

Le problème devient donc un problème d'optimisation sans contrainte où nous cherchons les meilleurs paramètres $r, p, \phi, \theta, \beta$ qui minimisent :

$$||M_{U,\alpha} M_{rp} Z - M_c||^2 \tag{4.33}$$

Nous avons effectué une série d'expériences sur 5000 courbes générées aléatoirement : les courbes C sont obtenues en projetant une hélice sur le plan $(z = 0)$ et en bruitant la courbe obtenue. La figure 4.1 montre une partie de nos résultats. Dans ce qui suit, M_1 désigne notre méthode seule, M_2 désigne la méthode d'optimisation seule, $M_1 M_2$ désigne l'approche où notre méthode est utilisée pour initialiser l'algorithme d'optimisation. Les erreurs sont calculées en utilisant l'équation 4.19. r_0, p_0 et α_0 désignent les paramètres de la courbe C, γ désigne l'angle de rotation selon l'axe x avant projection de la courbe C sur le plan $(z = 0)$. Les temps sont donnés en secondes.

En comparant les résultats (tableau 4.2) nous observons qu'utiliser la méthode de Levenberg-Marquardt en utilisant notre méthode pour trouver une solution initiale permet de diminuer l'erreur.

Dans un deuxième temps, nous n'utilisons pas notre méthode pour fournir des paramètres initiaux à l'algorithme d'optimisation. Le but est de trouver les paramètres r ,p,ϕ, θ, β ainsi que le paramètre α en utilisant seulement l'algorithme de $Levenberg - Marquardt$. Les valeurs suivantes sont utilisées afin d'initialiser l'algorithme d'optimisation : $r = 1$, $p = 1$. ϕ, θ et β sont calculés tel que la matrice Z initiale soit :

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \tag{4.34}$$

Z est une matrice dont les colonnes sont orthonormales. Le tableau suivant (4.3) montre que l'algorithme de Levenberg-Marquardt non initialisé avec une solution trouvée par notre méthode obtiens une erreur plus importante dans la plupart des cas. En effet dans certains cas la méthode de Levenberg-Marquardt n'arrive pas à converger vers une solution, ce qui peut être expliqué par le fait que les valeurs initiales données à l'algorithme

r_0	p_0	α_0	γ initiale	bruit	erreur M_1	erreur M_1M_2	erreurM_2 seule	temps M_1 en secondes	tempsM_1M_2 en secondes	temps M_2 en secondes
4.17	4.63	3.14	0.96	0.00	21.47	21.47	21.47	0.95	1.55	9.02
4.17	4.63	3.14	0.96	0.20	20.91	20.91	20.93	0.29	0.30	9.08
4.17	4.63	3.14	0.96	0.40	19.75	19.75	19.75	0.29	0.29	8.81
4.17	4.63	3.14	0.96	0.60	17.89	17.88	17.89	0.30	0.31	7.31
4.17	4.63	3.14	0.96	0.80	15.96	15.96	15.97	0.32	0.34	8.27
0.20	4.91	7.85	0.21	0.00	0.11	0.11	0.11	0.28	0.30	48.07
0.20	4.91	7.85	0.21	0.20	3.96	3.94	3.76	0.26	0.27	46.28
0.20	4.91	7.85	0.21	0.40	3.02	1.88	4.31	0.26	0.28	44.50
0.20	4.91	7.85	0.21	0.60	4.09	4.17	6.56	0.29	1.43	56.60
0.20	4.91	7.85	0.21	0.80	15.21	14.90	15.00	0.31	0.34	51.13
4.45	4.02	4.08	0.54	0.00	71.26	71.26	71.26	0.28	0.29	7.97
4.45	4.02	4.08	0.54	0.20	77.49	77.40	77.46	0.29	0.31	8.76
4.45	4.02	4.08	0.54	0.40	71.89	71.89	71.89	0.29	0.32	7.79
4.45	4.02	4.08	0.54	0.60	73.52	73.52	73.53	0.29	0.31	8.36
4.45	4.02	4.08	0.54	0.80	68.08	68.08	68.08	0.27	0.31	8.27
2.58	0.79	13.19	0.76	0.00	3.42	3.39	17.46	0.25	0.27	5.81
2.58	0.79	13.19	0.76	0.20	4.19	4.09	16.51	0.26	0.27	5.99
2.58	0.79	13.19	0.76	0.40	6.75	6.47	21.85	0.31	0.32	5.59
2.58	0.79	13.19	0.76	0.60	15.14	15.10	18.10	0.26	0.31	6.06
2.58	0.79	13.19	0.76	0.80	22.15	22.08	22.08	0.30	0.33	5.64
4.73	4.22	10.68	0.29	0.00	52.37	41.46	212.95	0.26	0.28	29.28
4.73	4.22	10.68	0.29	0.20	59.08	41.01	223.95	0.35	0.37	30.48
4.73	4.22	10.68	0.29	0.40	94.95	52.44	220.95	0.27	0.30	33.24
4.73	4.22	10.68	0.29	0.60	184.23	87.69	220.07	0.27	0.32	28.77
4.73	4.22	10.68	0.29	0.80	213.18	213.13	213.46	0.27	0.28	33.10
0.79	0.60	11.31	0.45	0.00	9.96	3.44	10.56	0.32	0.42	8.64
0.79	0.60	11.31	0.45	0.20	10.47	10.47	10.47	0.27	0.28	8.53
0.79	0.60	11.31	0.45	0.40	10.45	10.45	10.45	0.42	0.43	7.92
0.79	0.60	11.31	0.45	0.60	10.34	10.33	10.33	0.29	0.30	8.13
0.79	0.60	11.31	0.45	0.80	10.03	10.03	10.03	0.29	0.30	7.88
2.83	2.21	13.19	0.57	0.00	72.61	72.54	72.56	0.53	0.54	14.73
2.83	2.21	13.19	0.57	0.20	68.29	68.13	68.14	0.26	0.27	14.89
2.83	2.21	13.19	0.57	0.40	59.41	59.40	59.43	0.26	0.31	15.43
2.83	2.21	13.19	0.57	0.60	65.06	65.00	65.01	0.31	0.32	17.59
2.83	2.21	13.19	0.57	0.80	55.14	55.14	55.29	0.32	0.36	15.86
2.02	4.40	6.60	0.75	0.00	3.11	1.93	21.78	0.25	0.30	15.24
2.02	4.40	6.60	0.75	0.20	4.05	1.94	21.62	0.28	0.33	16.58
2.02	4.40	6.60	0.75	0.40	9.84	2.40	21.92	0.26	0.37	15.55
2.02	4.40	6.60	0.75	0.60	20.44	15.73	21.45	0.26	0.31	14.15
2.02	4.40	6.60	0.75	0.80	21.55	21.10	21.10	0.26	0.29	14.90

TABLE 4.1 : Ce tableau montre une partie des résultats que nous avons collectés sur 5000 exemples différents. M_1désigne notre méthode seule, M_2 désigne la méthode d'optimisation, M_1M_2 désigne l'approche où notre méthode est utilisée pour initialiser l'algorithme d'optimisation

	M_1	M_1M_2
Erreur moyenne	43.19	40.71
Temps moyen(s)	0.28	0.33

TABLE 4.2 : Résultats de la première expérience

peuvent être très éloignées de la solution. Notre méthode cherche un minimum global, alors que la méthode M_2 parcourt localement la fonction objectif et peut rester bloquer dans des minimums locaux.

	M1	M2
Erreur moyenne	43.19	58.50
Temps moyen(s)	0.28	13.36

TABLE 4.3 : Résultats de la deuxième expérience

On observe un temps de calcul important. Ceci peut s'expliquer par le fait qu'il faut beaucoup d'itérations de l'algorithme pour trouver une solution étant donné que la solution initiale n'est pas très bonne dans la plupart des cas. Il est possible d'augmenter le nombre d'itérations maximal de l'algorithme de levenberg-Marquardt ce qui peut dans

certainscas permettre de trouver de meilleures solutions, mais augmente encore le temps de calcul.

En conclusion ces deux expériences montrent que notre méthode peut être utilisée pour trouver une bonnz solution initiale. Cette solution peut être améliorée à l'aide d'une méthode d'optimisation non linéaire. Cette méthode d'optimisation non linéaire nécessite une solution initiale pour fonctionner correctement, cette solution peut être trouvée par notre algorithme.

4.5.4 Cas spéciaux

Dans le cas où la courbe d'entrée s'approche d'un cercle (voir figure 4.14) notre algorithme ne peut pas calculer de façon certaine le pas de l'hélice, car il existe une infinité de solutions possibles.

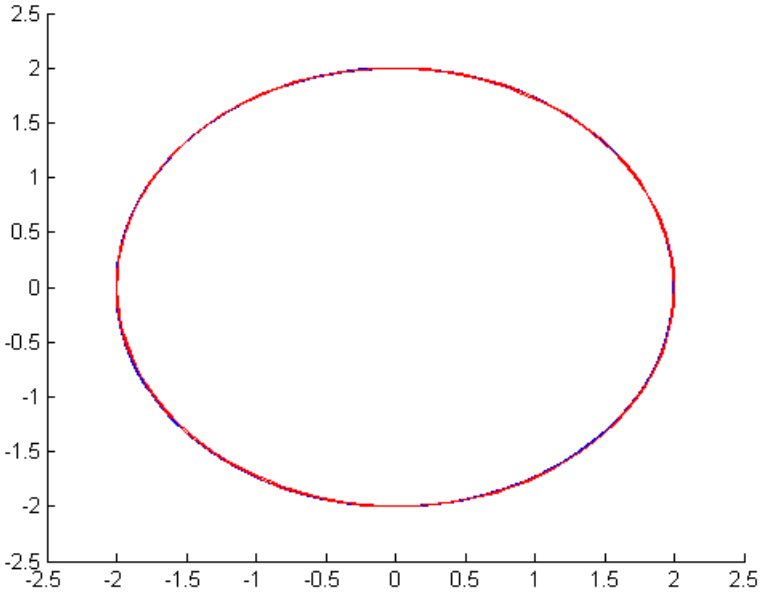

FIGURE 4.14 : Cas où la courbe C est un cercle, il existe une infinité de solutions pour le paramètre p

Un autre cas spécial est lorsque la courbe d'entrée C est un segment de droite (voir figure 4.15). Dans ce cas notre algorithme est incapable de calculer une solution. Reprenons l'équation (4.35) permettant de calculer r et p. Quand C est une ligne droite, le deuxième composant du vecteur X est égal à 0, il est donc impossible de calculer ,r car cela provoque une division par 0.

$$A = \begin{pmatrix} l_{2,1}^2 & l_{1,1}^2 + l_{3,1}^2 \\ l_{2,1}l_{2,2} & l_{1,1}l_{1,2} + l_{3,1}l_{3,2} \\ l_{2,1}l_{2,2} & l_{1,1}l_{1,2} + l_{3,1}l_{3,2} \\ l_{2,2}^2 & l_{1,2}^2 + l_{3,2}^2 \end{pmatrix}, X = \begin{pmatrix} \frac{1}{p^2} \\ \frac{1}{r^2} \end{pmatrix} B = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{4.35}$$

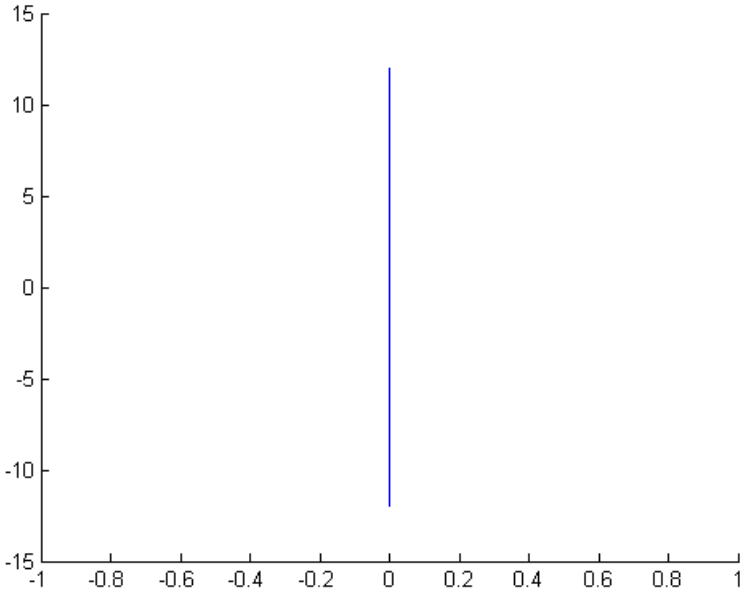

FIGURE 4.15 : Cas où la courbe d'entrée est un segment de droite

Chapitre 5

Modélisation de surfaces à partir de croquis

Une autre catégorie de modélisation par croquis est la modélisation d'un objet dont la silhouette est représentée par un dessin. L'objectif est de reconstruire la surface telle que la projection orthogonale de sa silhouette corresponde au croquis. Une des difficultés majeures de la modélisation de surface à partir de croquis est le traitement des parties cachées, c'est à dire lorsqu'un élément du dessin passe derrière un autre. Quand ce cas se présente, la difficulté est de proposer une reconstruction plausible en recréant les parties non dessinées.

Afin d'éviter de devoir traiter ce cas, nous choisissons de reconstruire des objets sous forme de bas-reliefs. Nous avons donc développé un algorithme qui construit des surfaces sous forme de bas-reliefs à partir d'un dessin en deux dimensions.

Dans un premier temps nous présentons ce que sont les bas-reliefs ainsi que quelques traveaux qui concerne leurs générations à partir de modèles 3D. Puis dans un second temps nous décrivons notre algorithme et enfin nous montrons quelques résultats obtenus à l'aide de celui-ci.

5.1 Les bas-relief

Un bas-relief est un type de sculpture ou de modelage pouvant être peint qui a pour particularité de ne présenter qu'un faible relief. Il est possible de créer un effet de profondeur en simulant une perspective. À l'origine les bas-reliefs sont des gravures sculptées dans la roche, la figure 5.1 montre un exemple de ces bas-reliefs.

FIGURE 5.1 : Bas-relief sculpté dans la roche, représentant une vache sacrée à Mamallapuram (extraite de http ://fr.wikipedia.org/wiki/Bas-relief)

Les bas-reliefs peuvent être exécutés en terre cuite, en pierre, en marbre, en ivoire, en bois, sur toutes sortes de métaux, sur des vases, bijoux, pierres fines, etc. Les bas-reliefs servent à décorer les batiments, en effet ils sont présents en très grand nombre sur les monuments et édifices religieux. Nous pouvons les trouver sur les colonnes, les autels, les tombeaux, les arcs de triomphe, etc. La figure 5.2 montre un autre exemple de bas-reliefs.

FIGURE 5.2 : Exemple de Bas-relief au moyen age (tirée de http ://www.cosmovisions.com/artBasRelief.htm)

Il existe des travaux qui concerne la génération de bas-relief à partir de modèles 3D. Par exemple l'article [59] présente une méthode qui génère un bas-relief à partir d'un modèle 3D en se basant sur une technique appellée Adaptive Histogram Equalisation (AHE). L'AHE est une méthode permettant d'améliorer les contrastes d'une image. La figure 5.3 montre un exemple de résultat.

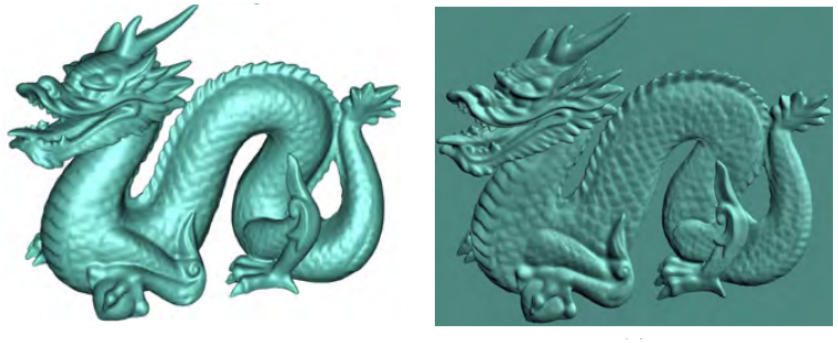

FIGURE 5.3 : A gauche le modele 3D et à droite le bas-relief généré à l'aide de la méthode de [59]

Il existe d'autres méthodes pour générer des bas-reliefs comme par exemple [60] ou [61]. Contrairement à ces méthodes, notre objectif est de générer des bas-reliefs à partir de croquis en deux dimensions.

Pour générer notre bas-relief nous allons utiliser une carte de hauteur (heightmap en anglais). Les heightmap sont des structures de données (un tableau) qui contiennent les données d'élévations d'un ensemble de sommets. Souvent elles sont stockées sous forme d'images (voir figure 5.4) et sont en général utilisées pour générer des terrains (par exemple dans le milieu du jeux vidéo).

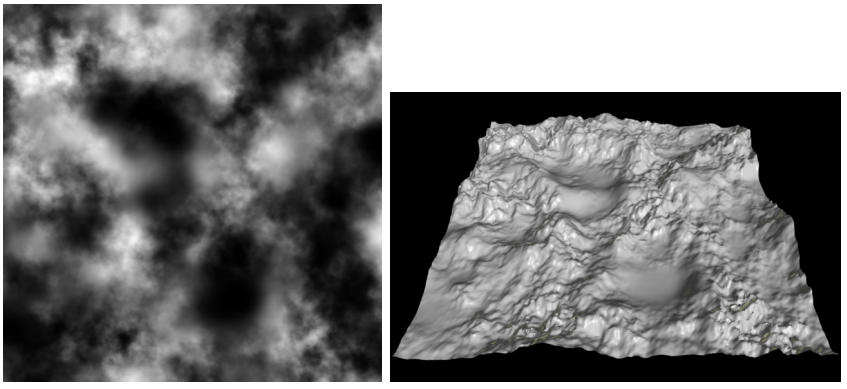

FIGURE 5.4 : Exemple de carte de hauteur et de terrain (à droite) généré à l'aide de celle-ci)

Nous avons un ensemble de n^2 sommets dont les coordonnées x, y sont attribués selon une grille 2D (n est la taille de la grille). L'objectif est de calculer l'élévation (la hauteur), la coordonée z de chaque sommets (figure 5.5). Ainsi les bas-reliefs générés ne présentent pas de surfaces cachées par une autre surface selon l'axe z.

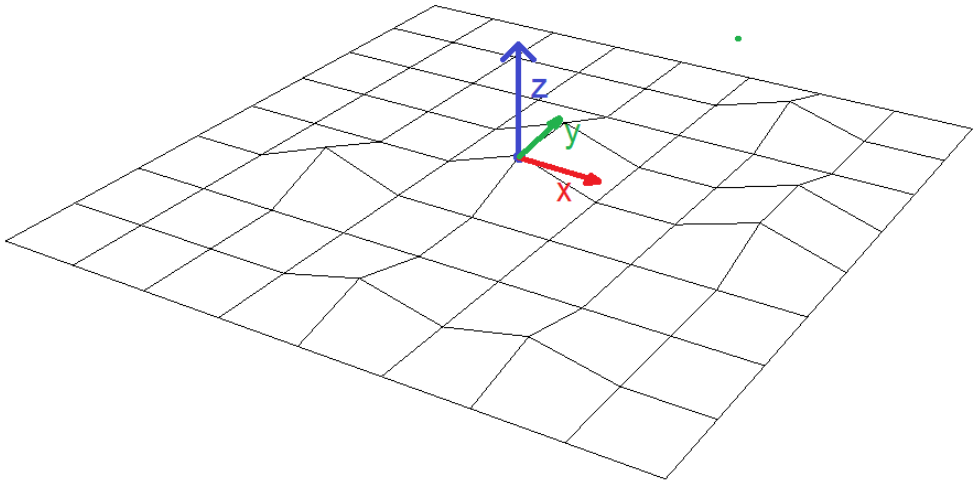

FIGURE 5.5 : L'objectif est de calculer la coordonnée z de chaque sommet

5.2 Reconstruction de bas-relief à partir de croquis

Nous présentons une méthode pour reconstruire une surface sous forme de bas-relief à partir d'un croquis de celle-ci. L'algorithme est composé de plusieurs étapes, la première est d'acquérir le dessin. La deuxième étape de l'algorithme consiste à convertir le dessin alors constitué de courbes (voir 5.2.1) en une représentation sous forme de grilles en 2D. Nous utilisons deux grilles pour contenir les informations sur le dessin. En effet une première grille contient les informations sur la forme du dessin et une deuxième grille contient la hauteur de chaque case. L'étape suivante est de calculer les hauteurs de chacune des cases de la deuxième grille (5.2.2) . On obtient alors une forme rugueuse et l'étape suivante est de lisser cette forme afin de l'adoucir (section 5.2.3). La grille 2D qui contient les hauteurs peut être considérée comme une carte de hauteurs, notre algorithme convertit ensuite cette carte des hauteurs en objet 3D exploitable sous un logiciel de modélisation (par exemple 3DS Max). C'est-à-dire, nous créons un maillage à partir des deux grilles contenant les hauteurs et le dessin. Nous présentons ensuite différents résultats obtenus à l'aide de notre algorithme et discutons des limitations de celui-ci (section 5.2.5). Ces étapes sont résumées par la figure 5.6.

5.2.1 Discrétisation du dessin

La première étape de l'algorithme consiste à transformer le dessin donné en entrée par l'utilisateur. L'espace est divisé à l'aide d'une grille de n cases de côtés. Plus n est grand plus la reconstruction sera détaillée. Dans notre implémentation nous choisissons $n = 256$. La figure 5.8 montre un exemple de morceau de courbe provenant d'un dessin et sa discrétisation. Dans notre implémentation le dessin est constitué de courbes polygonales, créées à l'aide d'une interface graphique (voir figure 5.7) .Comme nous l'avons vu dans le chapitre 2, Il est possible d'utiliser d'autres techniques pour acquérir les données du dessin (voir section 2.2.1) . Nous utilisons une deuxième grille de mêmes dimensions afin de stocker les hauteurs de chaque case du dessin.

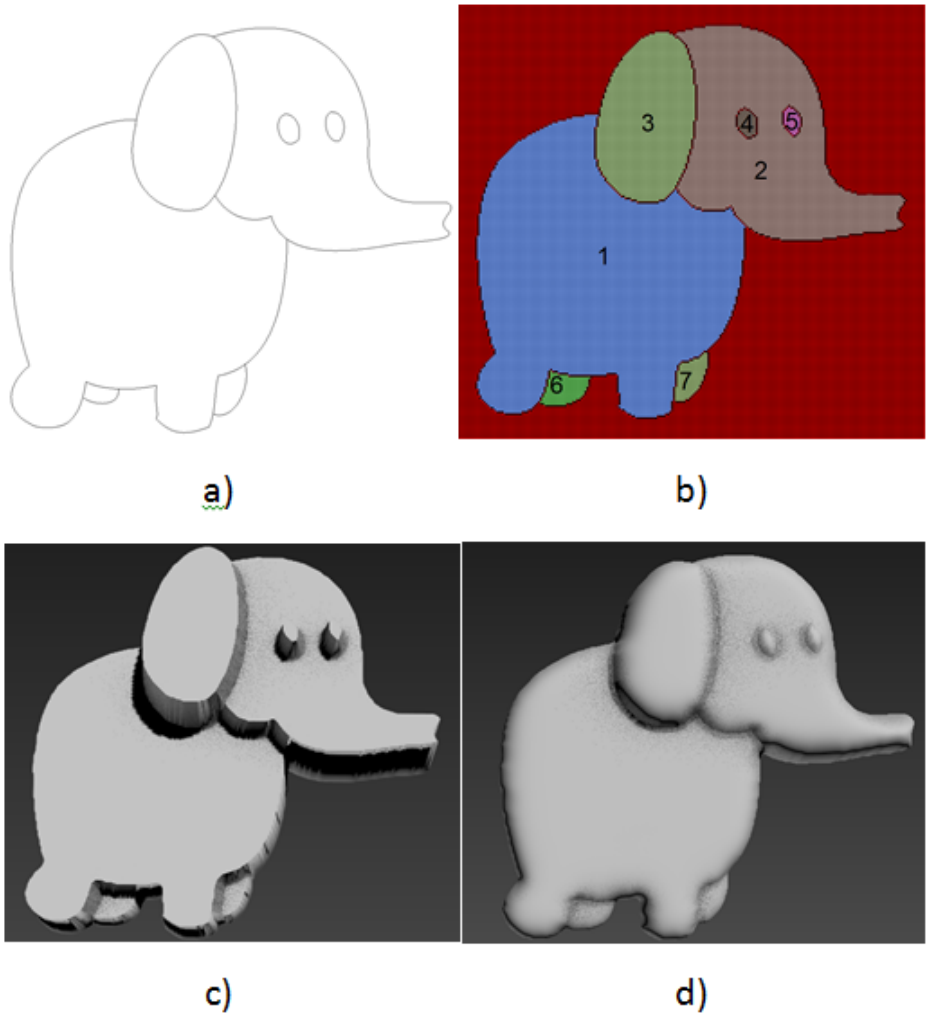

FIGURE 5.6 : Apperçu de la méthode, l'utilisateur fournit un dessin à notre algorithme (a), qui est ensuite discrétisé selon une grille 2D. Les groupes de cases sont ensuite regroupés (b) et élévation initiale est calculée (c). La forme est ensuite lissée (d)

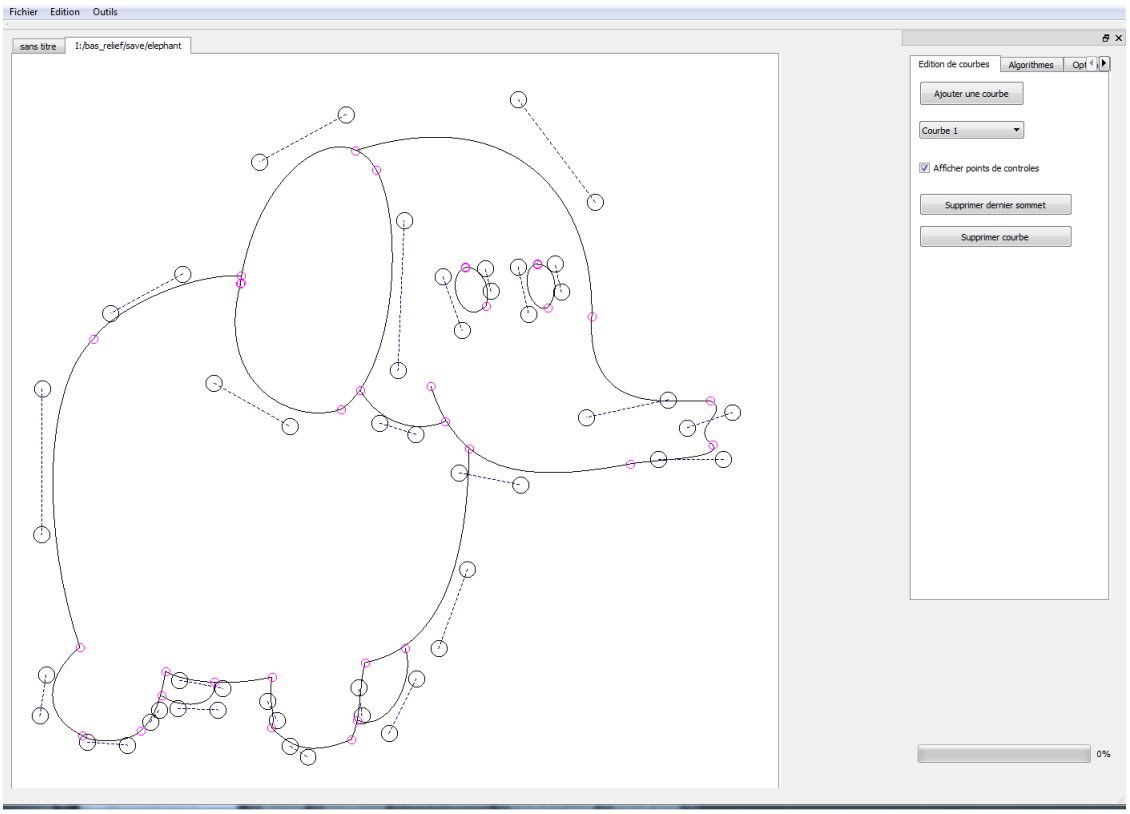

FIGURE 5.7 : Interface graphique utilisée pour dessiner les dessins donnés en entrée de notre algorithme

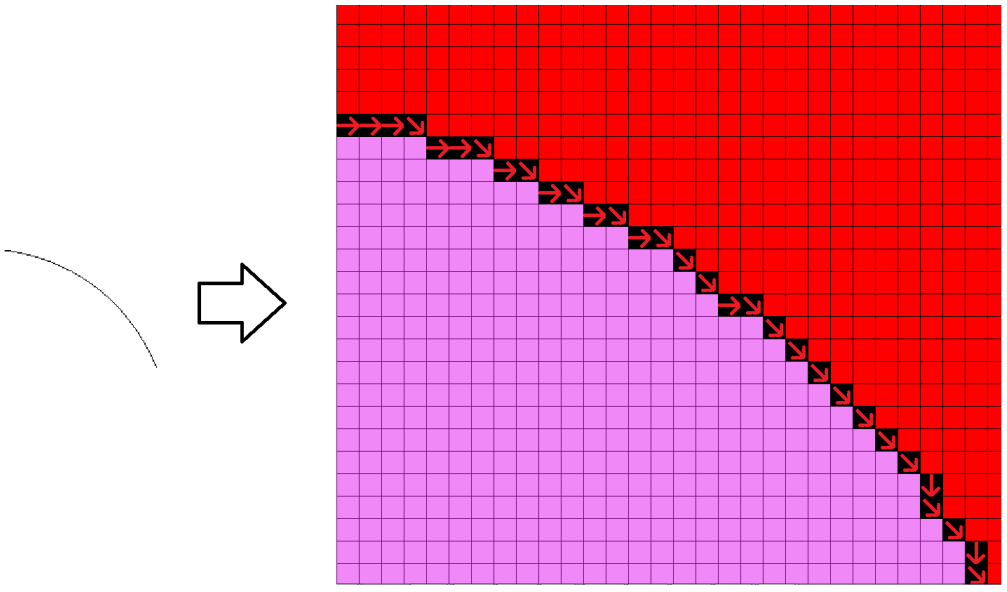

FIGURE 5.8 : Exemple de courbe provenant d'un dessin (à gauche) et sa discrétisation en utilisant une grille 2D (à droite). Les flèches représentent le sens de la courbe, les cases en roses appartiennent au dessin contrairement aux cases en rouges qui ne font pas partie du dessin.

Avant d'importer le dessin, l'utilisateur attribue un sens aux traits de celui-ci. L'intérieur de la surface décrite par la courbe se situe à la droite de celle-ci. S'il y a deux surfaces de part et d'autre de la courbe, celle de droite est plus élevée que celle de gauche. Notre algorithme utilise deux grilles de même taille (n^2). La première contient les informations sur le dessin (directions des cases représentants un trait) et la deuxième l'élévation de chaque case.

5.2.2 Calcul des hauteurs

L'objectif de cette étape est de calculer la hauteur initiale (l'élévation) de chaque case, l'idée ici est de regrouper les cases qui font partie d'une même zone du dessin ensemble et de leur attribuer une hauteur initiale commune. Cette structure de groupe permet d'avoir une cohérence dans les hauteurs initiales, par exemple (figure 5.10) les cases qui représente une patte se verront attribuer la même hauteur initiale. Dans un premier temps nous décrivons comment constituer ces groupes de cases puis dans un second temps comment les hauteurs initiales de ces groupes sont calculées.

5.2.2.1 Groupes de cases

Afin de calculer les hauteurs initiales des cases, nous regroupons les cases au sein de différents groupes. Les groupes sont numérotés de 1 à m, où m est le nombre de groupes différents. Par exemple, prenons le cas d'un dessin représentant un éléphant (figure 5.9). L'idée est de regrouper ensemble les cases connexes qui sont du même côté d'une courbe du dessin. La figure 5.10 montre sur l'exemple de l'éléphant la répartition des différentes cases en groupe.

Dans la section suivante, nous décrivons comment répartir les cases en différents groupes.

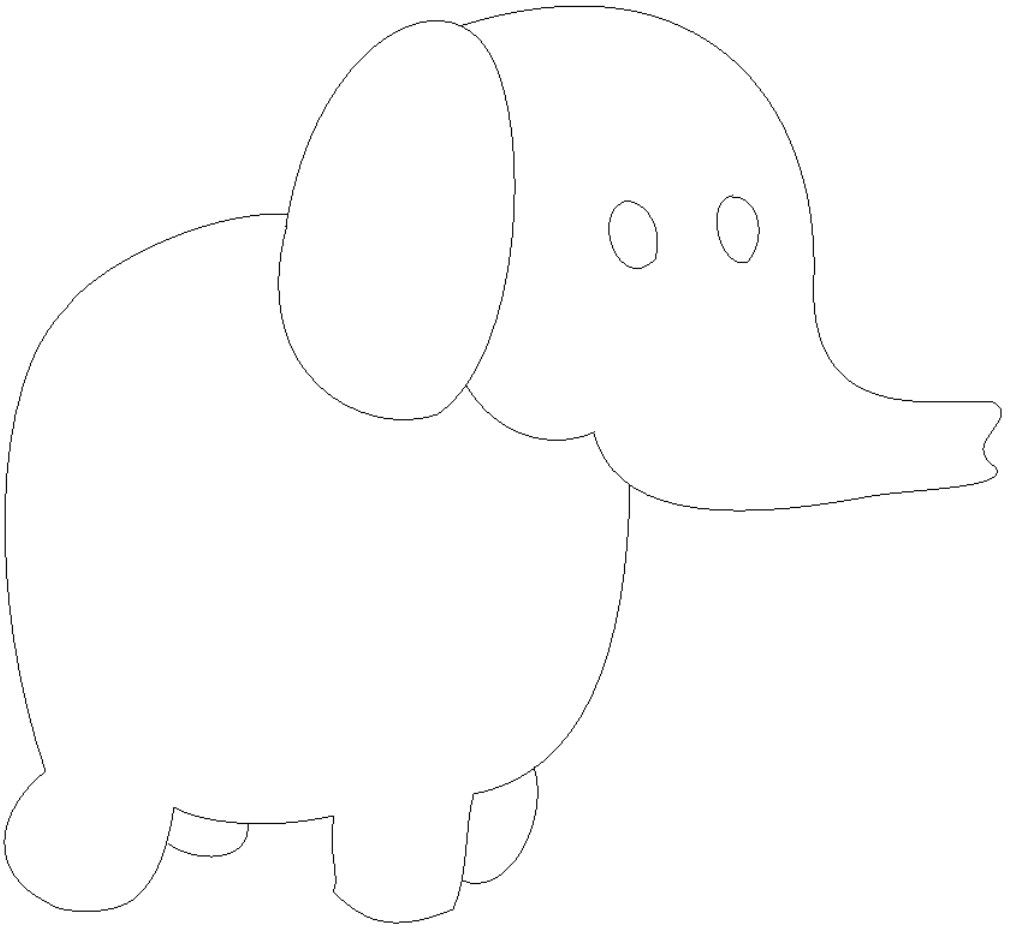

FIGURE 5.9 : Un dessin d'un éléphant

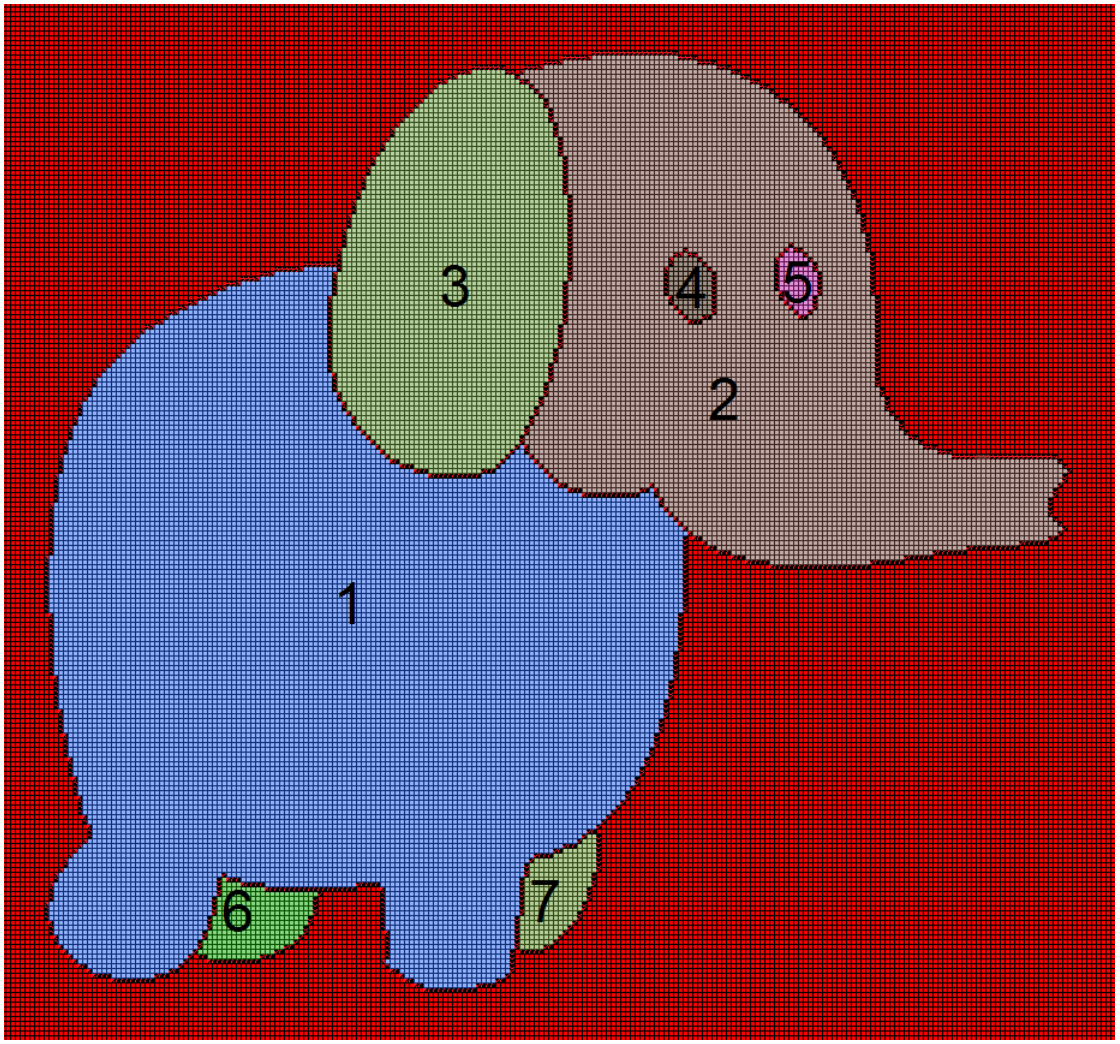

FIGURE 5.10 : Regroupements des cases en différents groupes

5.2.2.2 Calcul des groupes

Les traits du dessin forment différentes zones, il existe deux types de zones : les zones fermées et les zones ouvertes. Les zones sont créées par les traits du dessin. Une zone fermée peut être crée par une seule courbe qui se ferme sur elle-même (l'oeil de l'éléphant par exemple, figure5.9) où par un ensemble de courbe qui sont connectées les unes aux autres (par exemple le ventre de l'éléphant). Si on reprend l'exemple de l'éléphant, on remarque que celui-ci n'est constitué que de zones fermées, c'est-à-dire qu'il n'y a aucune extrémité d'un trait du dessin non connecté à un autre trait. Au contraire si l'on prend le dessin suivant (figure 5.11), il y a des traits qui ne sont pas connectés (identifiés par un cercle rouge), créant ainsi des zones ouvertes.

La méthode pour calculer les groupes consiste donc à :

— Creer un ensemble de zones fermées.

— Grouper ensemble les cases de chaque zone fermées.

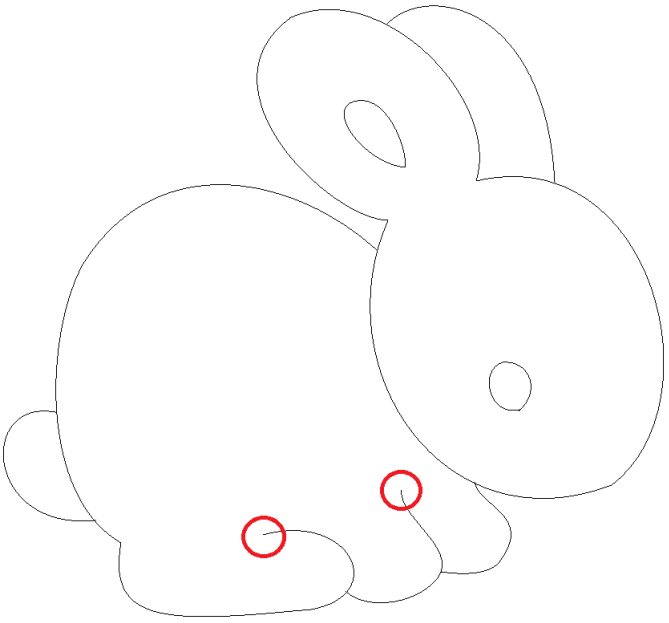

FIGURE 5.11 : Exemple de dessin. Les extrémités libres sont entourées en rouge

La première étape consiste donc à fermer ces zones, la figure 5.12 montre la fermeture d'une zone ouverte (case en blanc) dans l'exemple précédant. Une fois ces zones fermées chacune d'entre elles constituent un groupe. Pour fermer les zones, nous calculons la normale à l'extrémité libre et nous fermons la zone en suivant cette normale jusqu'à la rencontre d'une des courbes du dessin.

Cette méthode présente une limitation , en effet la zone doit pouvoir être fermée à l'aide d'un seul segment de droite. Par exemple la figure 5.13 illustre cette limitation, en effet

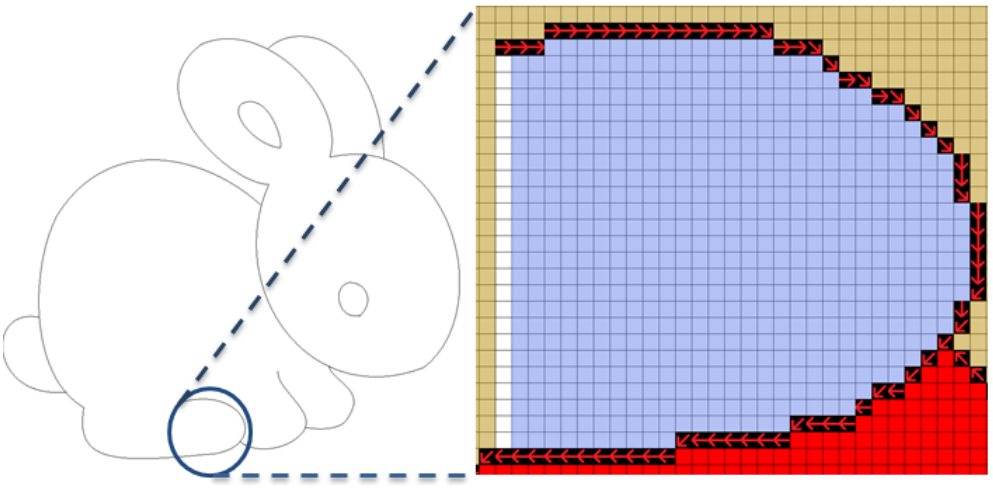

FIGURE 5.12 : Fermeture d'une zone ouverte

en utilisant un seul trait pour fermer les zones il est impossible d'isoler les cases sous la tête du chat et les cases de la tête du chat.

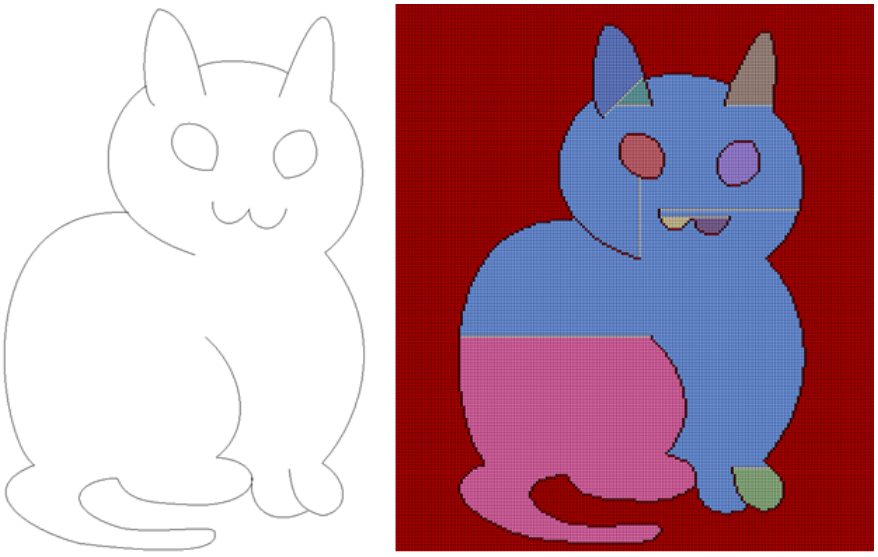

FIGURE 5.13 : Ce dessin montre une limitation de notre méthode. En effet les cases sous la tête du chat ne devraient pas être groupées avec les cases de la tête (bleu clair)

5.2.2.3 Calcul des hauteurs des groupes

Dans cette section nous montrons comment est calculée l'élévation initiale des cases de chaque groupe. Nous effectuons un balayage vertical et horizontal des cases pour détecter les différences d'élévation entre les groupes. C'est-à-dire que nous parcourons verticalement et horizontalement les cases de la grille contenant les informations sur le dessin. La hauteur (ou l'élévation) d'un groupe g_1 est inférieure à la hauteur d'un groupe g_2 s'il existe une case contenant un trait du dessin qui sépare les deux groupes et que le groupe g_2 est à droite de cette case. Par exemple sur la figure 5.10 le groupe 6 est inférieur au groupe 1, car les cases du groupe 6 se trouvent à gauche du trait séparant les deux groupes.

À partir de ces relations entre les groupes nous établissons un graphe orienté où les sommets représentent les différents groupes et les arcs les relations d'infériorité entre les groupes.

Reprenons l'exemple de l'éléphant, le calcul des groupes est montré dans la figure 5.10. Les balayages verticaux et horizontaux nous permettent de déduire les relations de hauteur entre les groupes que nous résumons à l'aide du tableau ci-dessous :

	1	2	3	4	5	6	7
1	-	$<$	$<$	-	-	-	-
2	-	-	$<$	$<$	$<$	-	-
3	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-
6	$<$	-	-	-	-	-	-
7	$<$	-	-	-	-	-	-

TABLE 5.1 : Ce tableau résume les relations de hauteur entre les groupes

Le tableau précédent 5.1 se lit de la façon suivante : il montre les relations de hauteurs entre les groupes. Par exemple l'élévation du groupe 1 est plus faible que celle du groupe 2 et 3. À partir de cela nous pouvons établir un graphe orienté (voir figure 5.14). Si le dessin est éclaté, c'est-à-dire qu'il existe des morceaux de dessins séparés les uns des autres, notre algorithme crée alors plusieurs graphes indépendants.

Il est ensuite assez simple de calculer les hauteurs initiales de chaque groupe en parcourant le graphe G_h. Nous calculons la profondeur du graphe, les sommets les plus profonds ont la hauteur la plus faible (1) et les sommets les moins profonds ont la hauteur la plus importante (la profondeur du graphe). Dans notre exemple les groupes 3, 4 et 5 ont une hauteur initiale de 4, le groupe 2 a une hauteur de 3, le groupe 1 une hauteur de 2 et enfin les groupes 6 et 7 une hauteur initiale de 1. Les cases qui ne font pas partie du dessin ont une hauteur de 0.

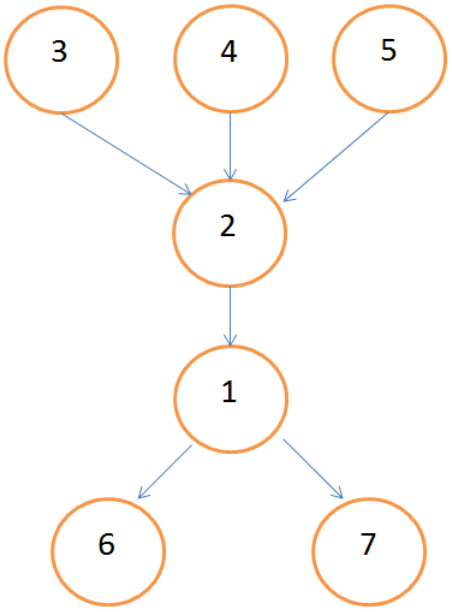

FIGURE 5.14 : Graphe représentant les relations de hauteurs entre les différents groupes

Pour résumer, le calcul des hauteurs des groupes s'effectue de la façon suivante : dans un premier temps nous déduisons les relations de hauteurs entre les groupes à l'aide de balayages verticaux et horizontaux de la grille contenant le dessin . Dans un second temps un graphe orienté est généré à l'aide de ces relations. Enfin, l'élévation de chaque groupe est calculée à l'aide de ce graphe.

5.2.3 Lissage

Après l'étape de calcul des hauteurs initiales, chaque case d'un même groupe ayant la même hauteur la figure possède un aspect non lisse (voir figure 5.15). La figure 5.16 montre le calcul des hauteurs initiales dans le cas ou il ya des zones ouvertes. Dans ce cas les étapes de lissage permettent par la suite d'établir une pente douce entre les cases du corps et les cases des pattes (figure 5.17).

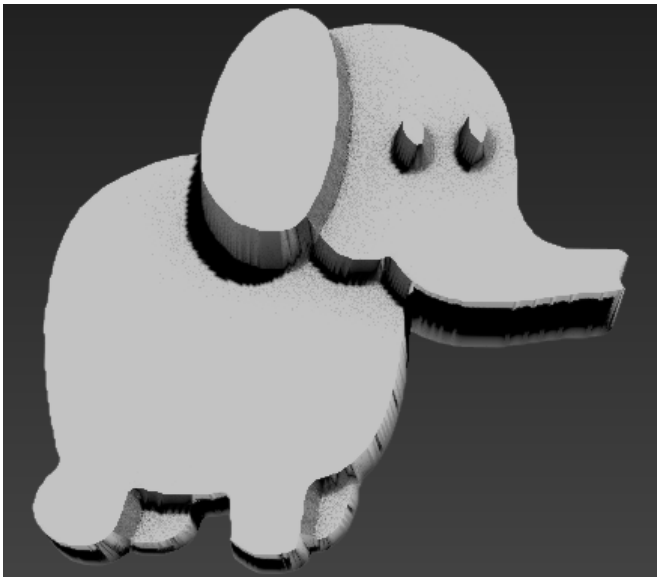

FIGURE 5.15 : Resultat sur l'exemple de l'éléphant après calcul des hauteurs initiales

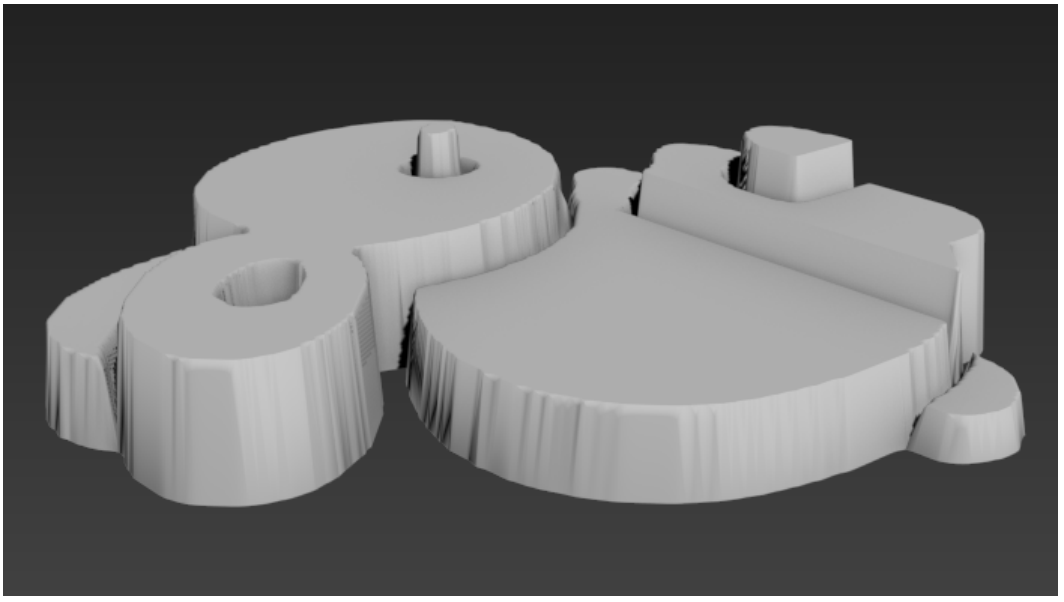

FIGURE 5.16 : Calcul des hauteurs initiales sur l'exemple du lapin

Afin de rendre le modèle 3d plus agréable à regarder, il est nécessaire de lisser celui-ci. Cependant le lissage doit respecter certaines règles afin de ne pas perdre les détails du dessin. Le lissage est effectué en deux étapes.

La première étape de lissage à pour but de lisser les hauteurs des cases non séparées par un élément du dessin. Elle consiste à parcourir chaque case du dessin et à calculer la moyenne des hauteurs entre cette case et les cases adjacentes. La figure (figure 5.17) suivante permet d'observer l'effet de la première étape de lissage. Cette étape n'a pour but que de créer des jonctions douces là où il y a des zones ouvertes dans le dessin. Elle n'a aucun effet à l'intérieur des zones fermées.

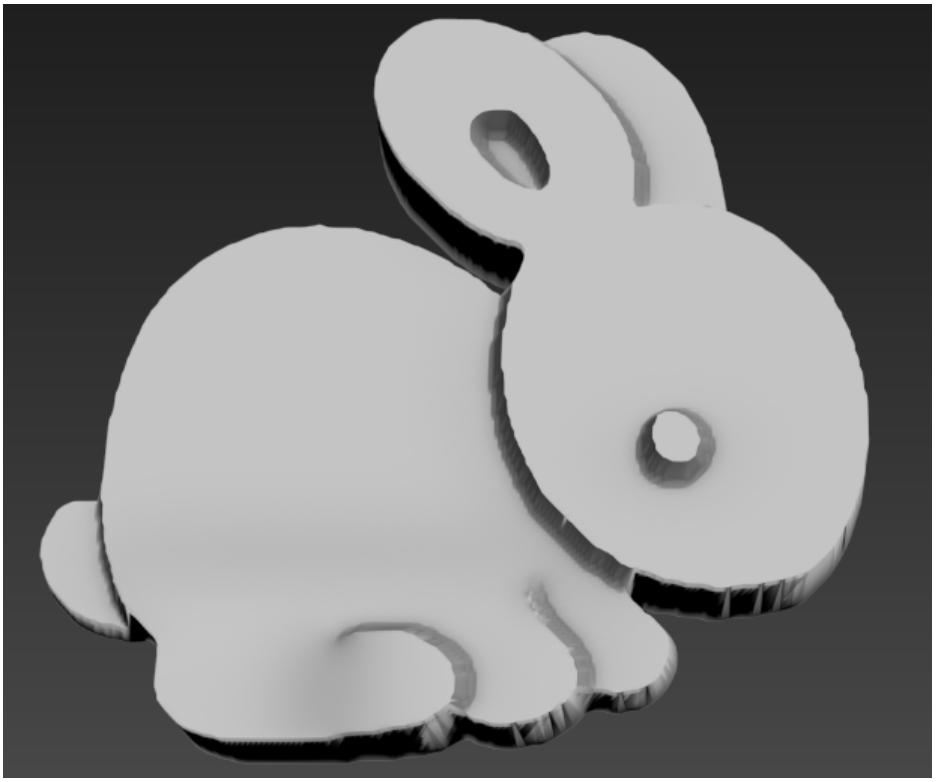

FIGURE 5.17 : Première étape de lissage

La deuxième étape fonctionne en deux temps. Elle consiste dans un premier temps à calculer l'élévation des cases représentant les courbes du dessin. Cette élévation est calculée en effectuant la moyenne des hauteurs des cases de part et d'autre de celle-ci. Puis dans un second temps une nouvelle élévation est calculée pour les cases à l'intérieur du dessin, en effectuant la moyenne des hauteurs avec les cases adjacentes. Ces deux opérations sont répétées pour un certain nombre d'itérations. Ce nombre d'itérations est fixé par l'utilisateur. Ceci a pour effet d'abaisser progressivement la hauteur des cases proches des traits du dessin et donc d'arrondir la forme 3D. Par exemple, si nous regardons l'oreille de l'éléphant (figure 5.18), les hauteurs des cases (en violet) à proximité du trait de l'oreille

(ici les flèches) sont abaissées vers la hauteur des cases du corps. Le résultat en 3D est montré dans la figure 5.19.

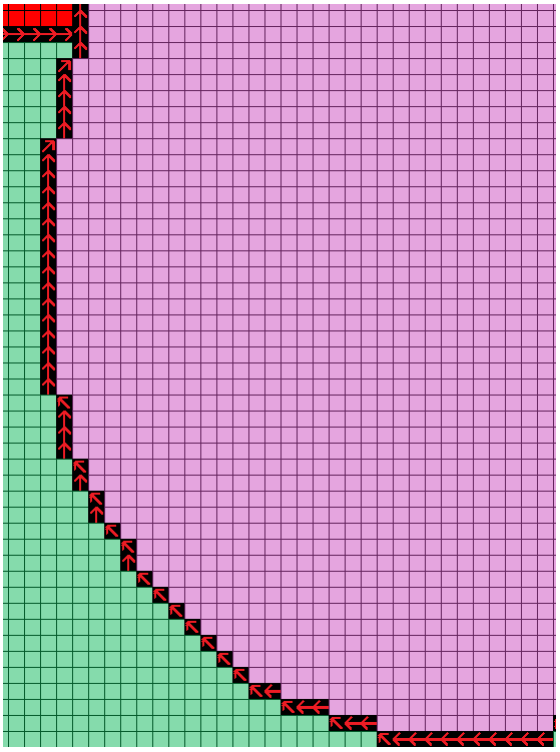

FIGURE 5.18 : Transition entre l'oreille et le corps de l'éléphant

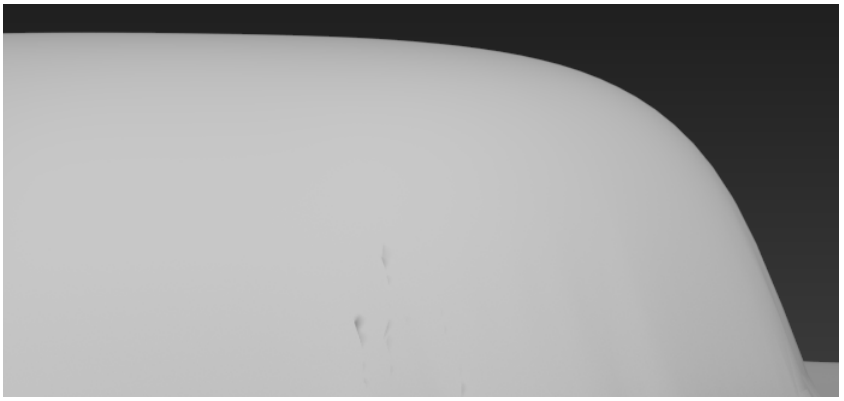

FIGURE 5.19 : Vue de côté de l'oreille de l'éléphant après lissage

5.2.4 Calcul du maillage

Une fois la figure lissée, nous créons à partir de la carte des hauteurs finale, un maillage que nous pouvons exporter vers un logiciel de modélisation 3D (par exemple 3DS Max). Le maillage est créé de la façon suivante, chaque case est le sommet d'un triangle du

maillage. Pour qu'un triangle du maillage existe, il faut que la hauteur d'un des sommets soit supérieure à 0.

Une fois le maillage crée, nous remarquons qu'il existe un effet de crénelage qui concerne les sommets à gauche des courbes du dessin (voir figure 5.20).

FIGURE 5.20 : Maillage de l'éléphant, nous pouvons observer que les bords de la forme (les sommets à gauche des traits du dessin) ne sont pas lisses

Pour remédier à ceci, nous ajoutons une étape où les sommets à gauche des traits du dessin sont attirés les uns aux autres. Cette étape fonctionne de façon itérative. A chaque itération, les sommets du contour du dessin se rapprochent des sommets voisins comme l'illustre la figure 5.21. Soit $v_{i,j}$ un sommet à la gauche d'un trait du dessin et soit $N_{i,j}$ l'ensemble des sommets voisins de $v_{i,j}$ dans le maillage. Nous calculons la nouvelle position de chaque sommet du voisinage $N_{i,j}$ en appliquant la formule suivante :

$$v_{k,l} = v_{k,l} + \gamma(v_{i,j} - v_{k,l}) \tag{5.1}$$

Où $v_{k,l}$ est un sommet voisin de $v_{i,j}$ et γ un paramètre entre 0 et 1. i et j ainsi que k et l sont les coordonnées des sommets dans la grille. Plus γ est faible, moins le déplacement est important. Nous parcourons l'ensemble des sommets $v_{i,j}$ qui sont à la gauche d'un trait du dessin et nous appliquons cette formule sur chacun de leurs voisins. Cette étape est répétée un certain nombre de fois, nombre fixé par l'utilisateur (64 dans notre cas).

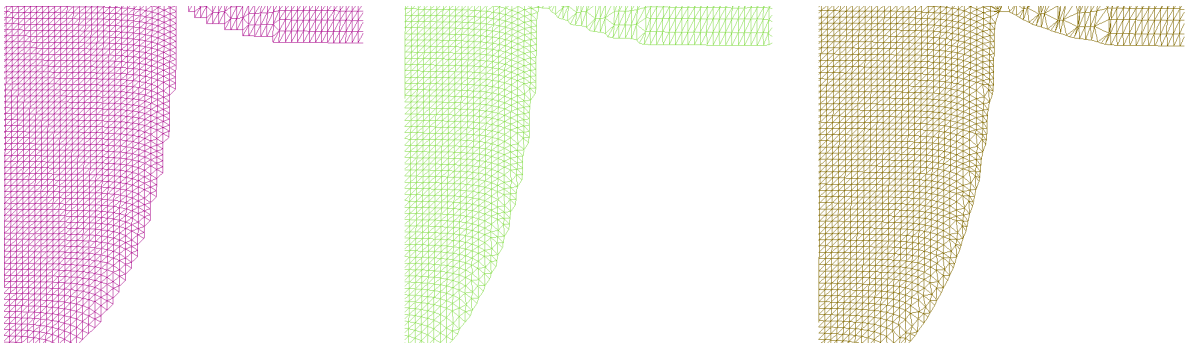

FIGURE 5.21 : La figure à gauche montre le maillage sans traitement additionnel après la création de celui-ci. La figure du milieu montre ce même maillage après 16 itérations du traitement additionnel, celle de droite montre le maillage après 64 itérations.

5.2.5 Résultats et limitations

Dans cette section, nous présentons quelques résultats obtenus à l'aide de notre algorithme, puis dans un second temps nous discutons des limitations de celui-ci.

5.2.5.1 Résultats

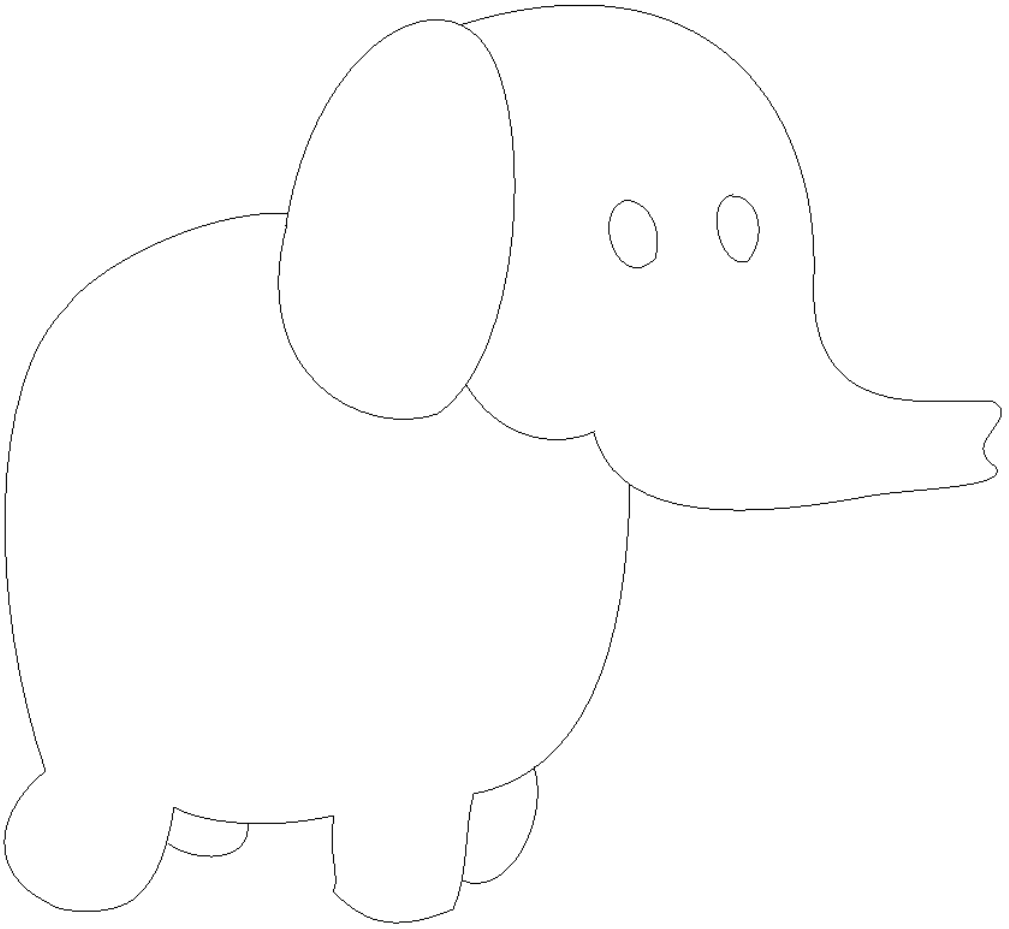

FIGURE 5.22 : Un dessin d'un éléphant

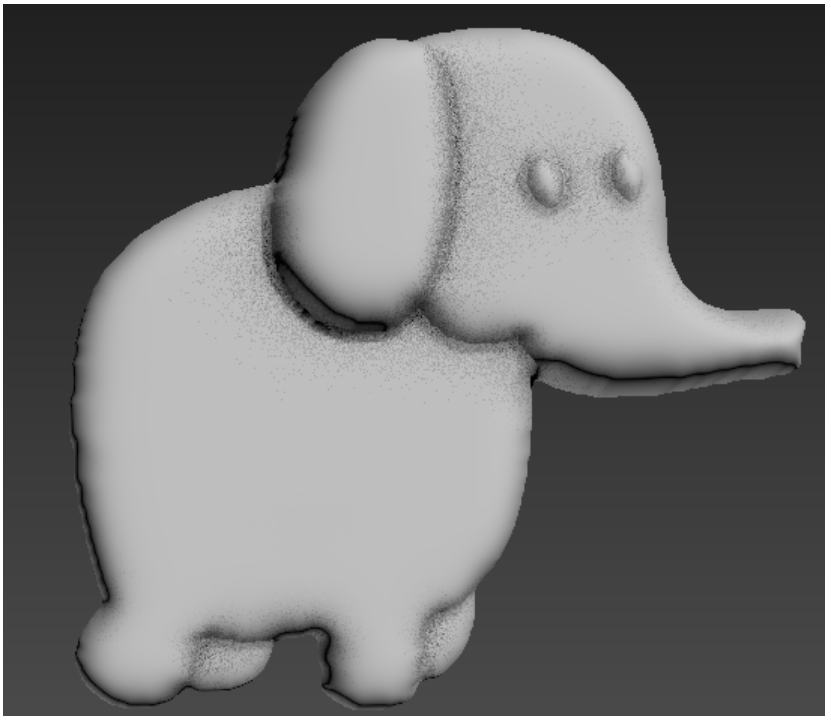

FIGURE 5.23 : Modèle 3D de l'éléphant

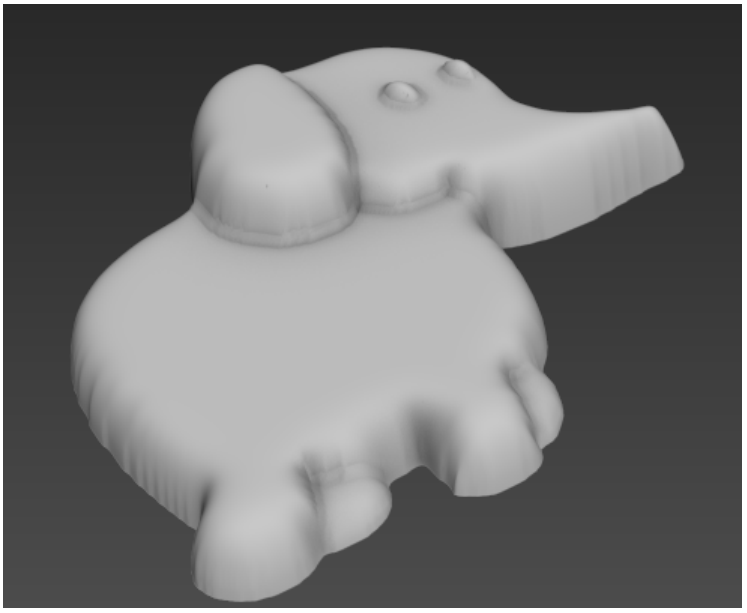

FIGURE 5.24 : Modèle 3D de l'éléphant

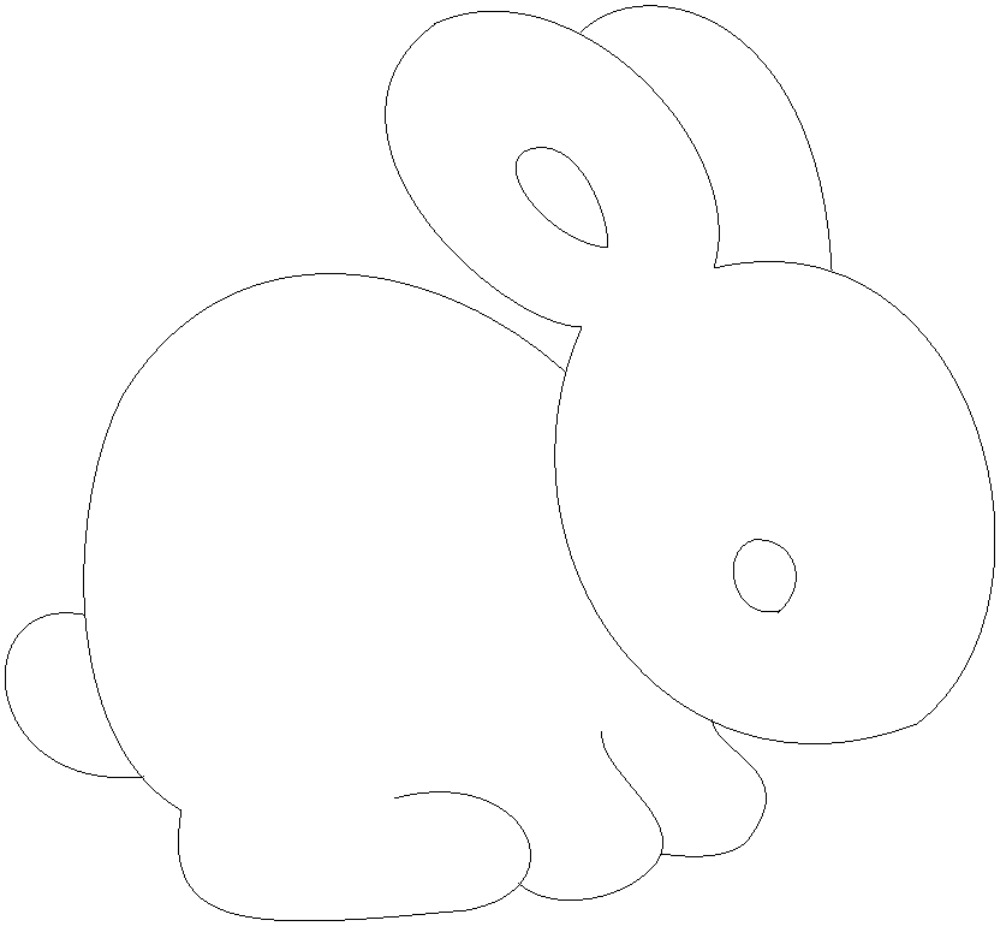

FIGURE 5.25 : Dessin d'un lapin

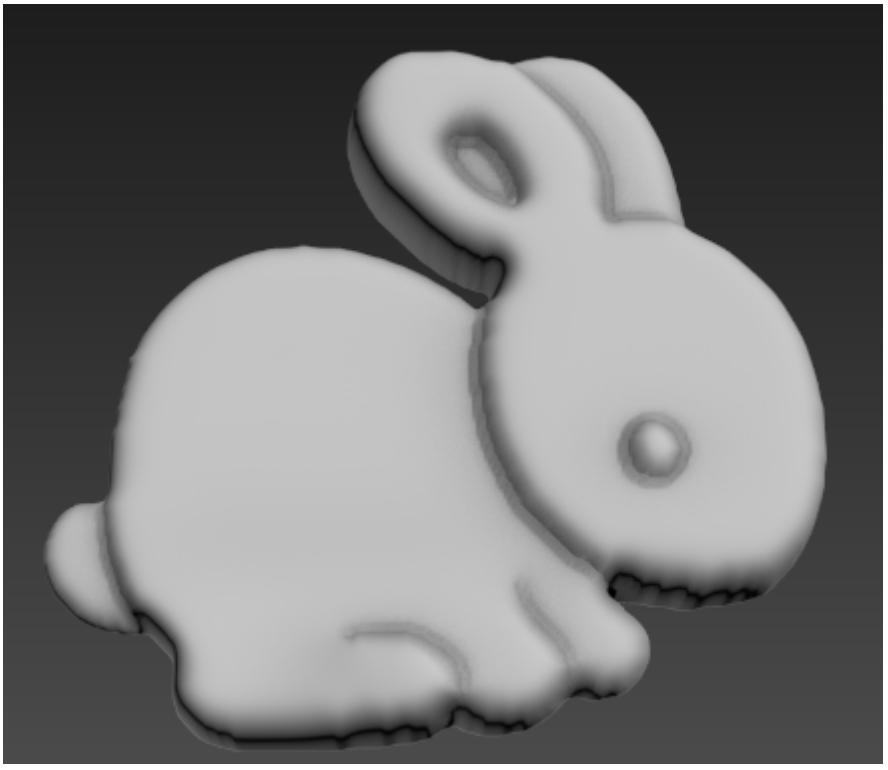

FIGURE 5.26 : Modèle 3D du lapin

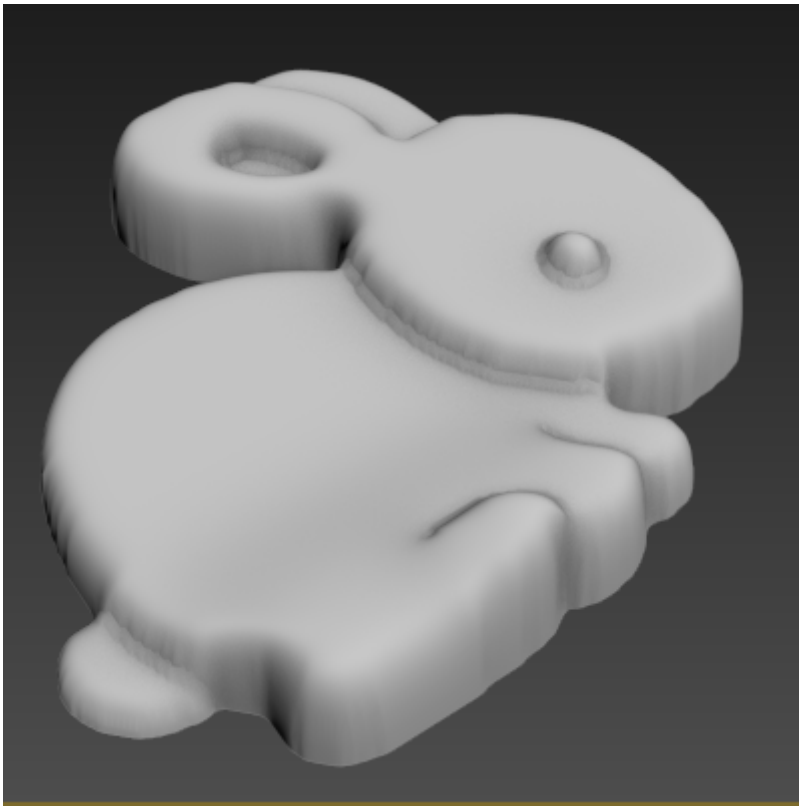

FIGURE 5.27 : Modèle 3D du lapin

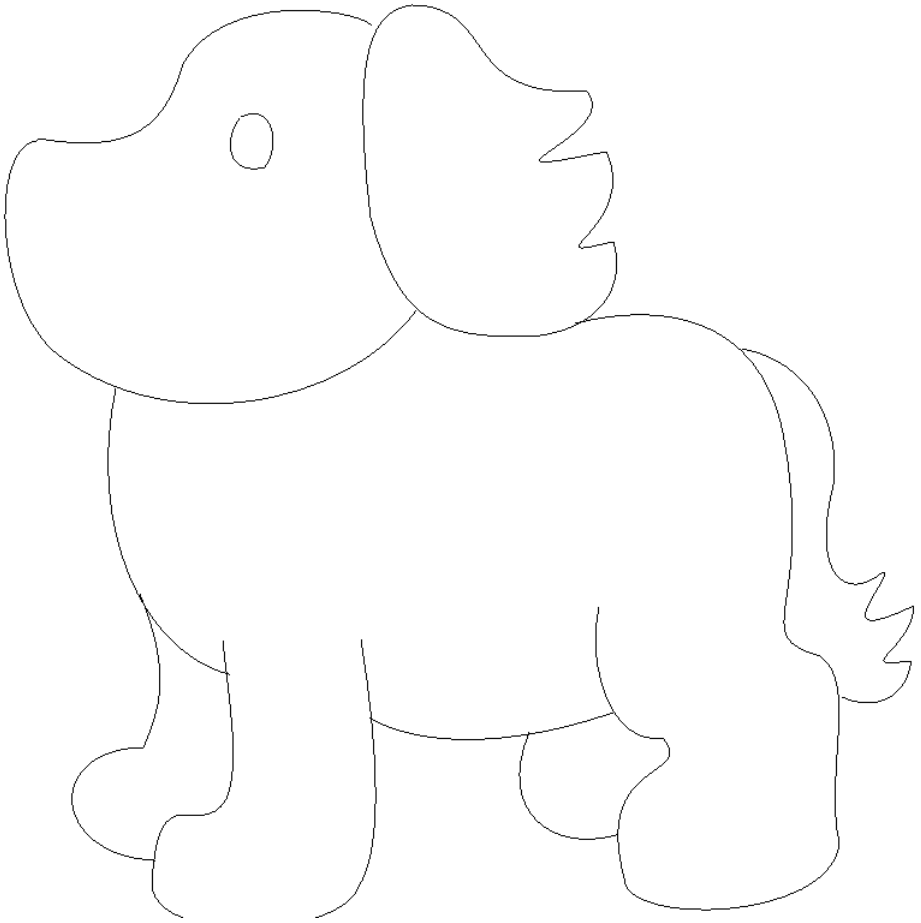

FIGURE 5.28 : Autre exemple de dessin

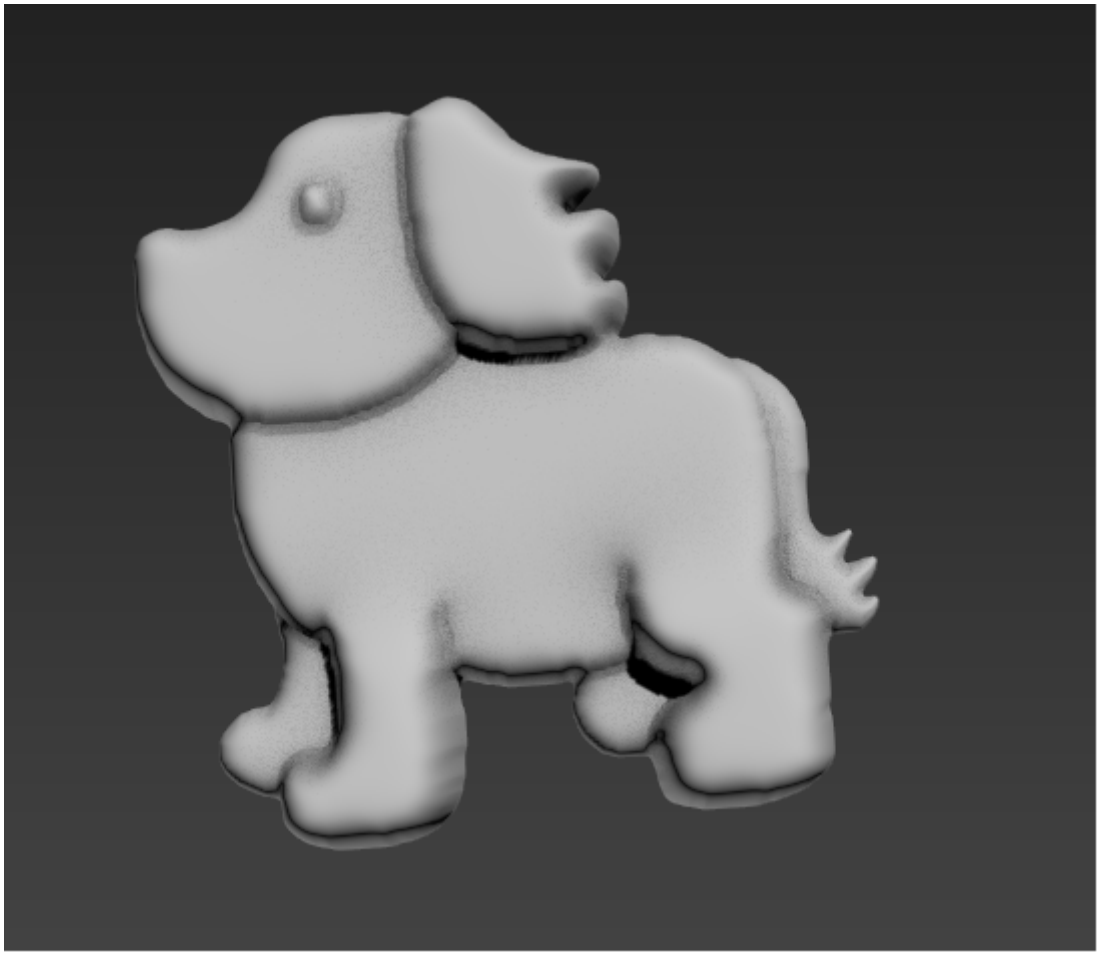

FIGURE 5.29 : Modèle 3D du chien

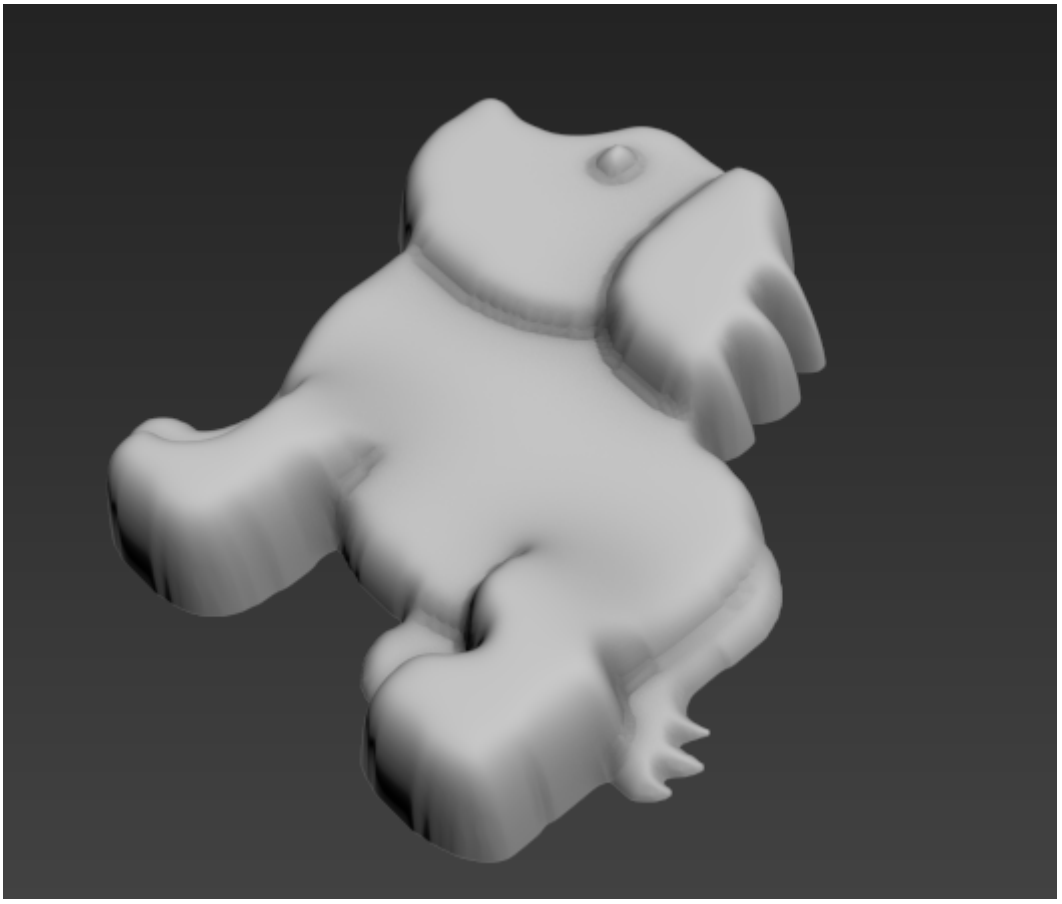

FIGURE 5.30 : Modèle 3D du chien

5.2.6 Limitations

Comme nous pouvons le constater sur les résultats précédent, les formes générées peuvent être un peu plates. Il serait sans doute possible de donner plus de relief en corrélant la hauteur de la forme avec sa superficie. Un deuxième point à améliorer est l'étape de lissage, en effet les petits détails (comme les yeux par exemple) peuvent être effacés par le lissage .

Certaines formes ne sont pas reconstructibles avec cet algorithme, en effet notre méthode pour générer les groupes ne permet pas de gérer les cas où des cases peuvent être dans le même groupe, mais se trouver de part et d'autre d'un trait du dessin. L'exemple de la spiral illustre ce cas (voir figure 5.31). Cette limitation est due à la structure de groupe. Enfin, une autre limitation est la méthode permettant de fermer les groupes, celle-ci ne permet que des gérer des formes simples qui peuvent être refermées à l'aide d'une seule ligne.

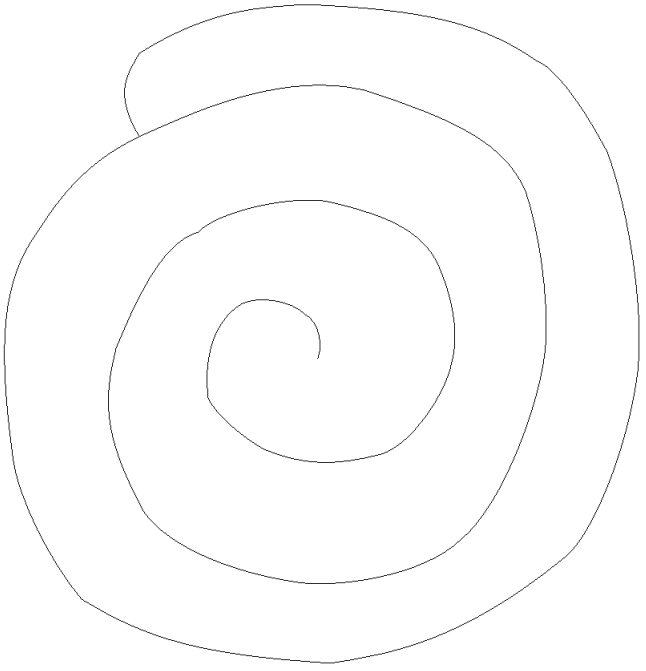

FIGURE 5.31 : Exemple de cas où notre algorithme ne fonctionne pas. Dans ce cas notre algorithme groupe ensemble des cases qui sont de part et d'autre d'un trait ce qui rend impossible le calcul de la hauteur du groupe.

Conclusion

En ce qui concerne la modélisation de courbes à partir de croquis, nous avons créé deux méthodes qui permettent de reconstruire des courbes en 3D à partir de courbes 2D. Ces deux méthodes sont basées sur l'utilisation d'hélices. Nous utilisons des hélices, car ce sont des courbes qui ont une courbure et une torsion constante. De surcroît celles-ci sont des courbes qui peuvent se retrouver dans la nature, par exemple il est possible de modéliser un cheveu à l'aide d'une hélice.

La première méthode a pour but de reconstruire une courbe en 3D constituée de plusieurs morceaux d'hélices à partir d'un croquis. Un des problèmes de cette méthode est qu'elle génère des courbes qui peuvent posséder des discontinuités. Ces discontinuités se trouvent au niveau des tangentes entre les jonctions des différents morceaux d'hélice. De plus comme cette méthode ne prend pas en compte l'intégralité des points, mais seulement les points de courbures maximales, celle-ci est très sensible au bruit. C'est pour cela que nous avons développé une deuxième méthode qui approxime l'ensemble de la courbe à l'aide d'une seule projection d'hélice. Comme cette méthode prend en compte l'ensemble des points de la courbe d'entrée, celle-ci est moins sensible au bruit dans la courbe. Enfin nous avons montré que les solutions trouvées par cette méthode peuvent être utilisées pour initialiser une méthode d'optimisation non linéaire. Ainsi, grâce aux solutions fournies par notre algorithme cette méthode d'optimisation est capable de trouver de meilleures solutions.

Un des développements possibles dans le futur serait d'utiliser ces méthodes pour reconstruire une forme 3D à partir d'un dessin constitué de plusieurs courbes avec comme contraintes que les courbes qui se touchent en 2D se touchent aussi en 3D.. Par exemple la figure suivante montre une sphère qui est constituée de cercles qui peuvent chacun être modélisés à l'aide d'une hélice dont le pas a pour valeur 0.

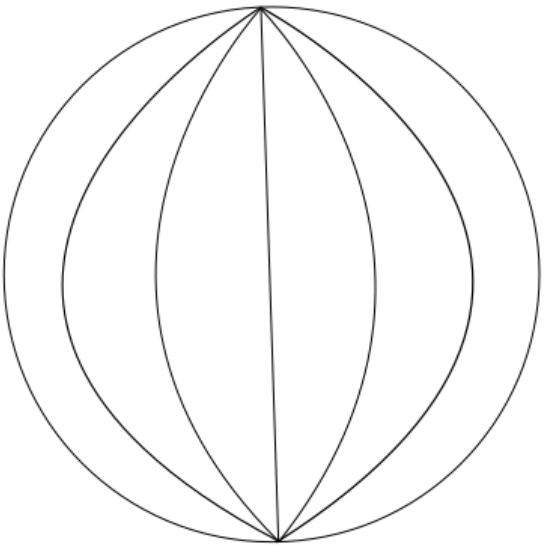

FIGURE 5.32 : La sphère suivante peut être modélisée à l'aide d'hélices

Nous avons dans un second temps développé une méthode permettant de modéliser des surfaces en 3D sous forme de bas-reliefs à partir de croquis . Nous utilisons cette représentation sous forme de bas-reliefs, car celle-ci à l'avantage de ne pas présenter de parties cachées.

L'utilisateur dessine un croquis qui est ensuite transformé par notre algorithme en une grille de deux dimensions. Le but est de calculer la hauteur de chacune des cases cette grille. Notre méthode repose principalement sur le regroupement des différentes cases qui constituent le dessin en différents groupes. L'algorithme calcule ensuite les hauteurs initiales de ces groupes de cases, puis à l'aide de plusieurs étapes de lissages la forme reconstruite est adoucie. Enfin, un maillage en 3D est calculé et la forme peut être utilisée dans un logiciel de modélisation 3D.

Cet algorithme présente plusieurs limitations, une première limitation concerne la façon dont les zones ouvertes sont fermées, ne permettant pas de gérer certains cas (voir figure 5.33.

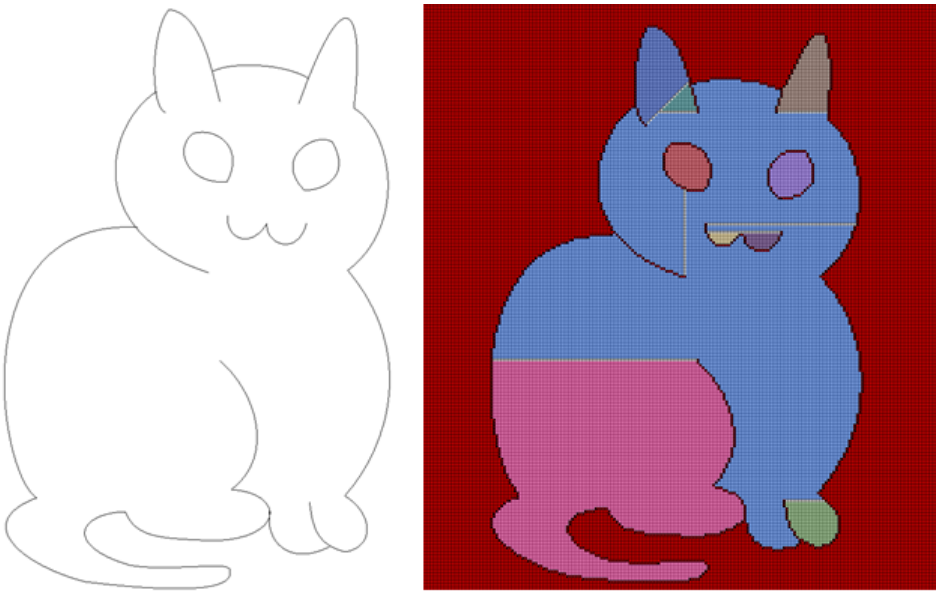

FIGURE 5.33 : Ce dessin montre une limitation de notre méthode. En effet les cases sous la tête du chat ne devraient pas être groupées avec les cases de la tête (bleu clair). Chaque couleur représente un groupe de case différent

Une amélioration possible de cette méthode concerne la génération des formes qui peuvent être trop plates. Une piste à explorer serait de mettre en corrélation la hauteur de chaque groupe de cases avec la surface de celui-ci. D'autre part, l'algorithme ne permet pas de gérer les cas où des cases d'un groupe se trouvent de part et d'autre d'une courbe du dessin 5.34. Une autre amélioration possible consisterait à prendre ceci en compte et diviser les groupes qui présentent ces cas de figure.

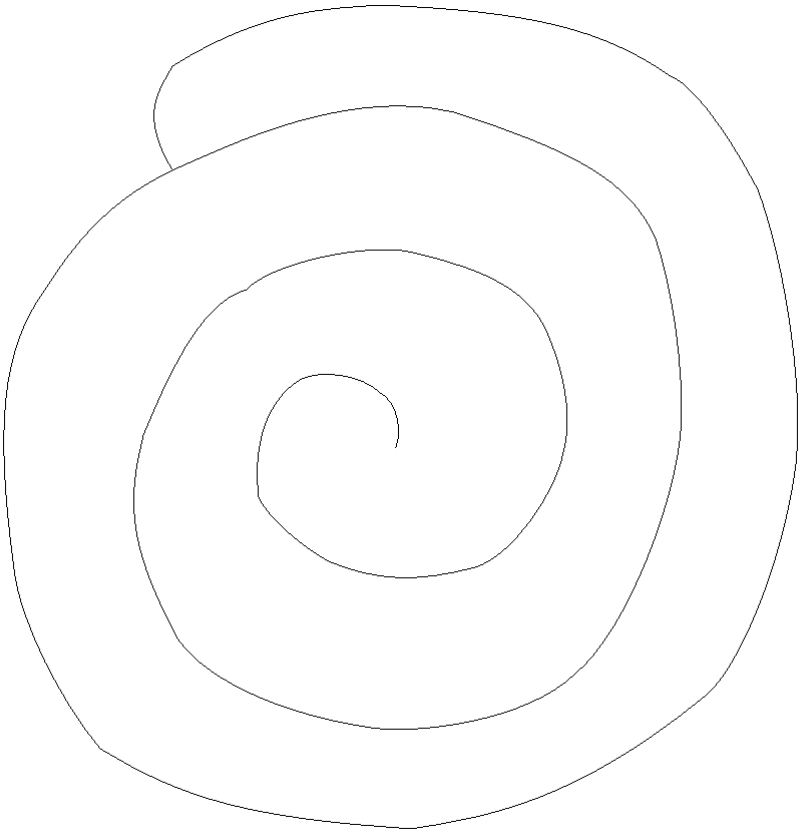

FIGURE 5.34 : Ce dessin montre un cas qui ne peut être reconstruit par notre algorithme, en effet un tel dessin produirait un groupe qui contiendrait des cases de part et d'autre d'une même courbe du dessin.

Bibliographie

[1] F. Cordier, Hyewon Seo, Jinho Park, and Junyong Noh. Sketching of Mirror-Symmetric Shapes. *IEEE Transactions on Visualization and Computer Graphics*, 17(11) :1650–1662, November 2011.

[2] Nicolas Cherin, Frederic Cordier, and Mahmoud Melkemi. Modeling piecewise helix curves from 2d sketches. *Computer-Aided Design*, 46 :258–262, January 2014.

[3] Frederic Cordier and Hyewon Seo. Free-Form Sketching of Self-Occluding Objects. *IEEE Comput. Graph. Appl.*, 27(1) :50–59, January 2007.

[4] M. Masry and H. Lipson. A Sketch-based Interface for Iterative Design and Analysis of 3d Objects. In *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[5] Donald D Hoffman. *Visual intelligence : How we create what we see*. WW Norton & Company, 2000.

[6] Yunfeng Li, Zygmunt Pizlo, and Robert M. Steinman. A computational model that recovers the 3d shape of an object from a single 2d retinal representation. *Vision Research*, 49(9) :979–991, May 2009.

[7] Vincent Hayward, Oliver R. Astley, Manuel Cruz-Hernandez, Manuel Cruz-Hernandez, Danny Grant, and Gabriel Robles-de-la Torre. *Haptic interfaces and devices*. 2004.

[8] L. Piegl. Interactive Data Interpolation by Rational Bezier Curves. *IEEE Computer Graphics and Applications*, 7(4) :45–58, April 1987.

[9] Lynn Eggli, Ching-yao Hsu, Beat D Brüderlin, and Gershon Elber. Inferring 3d models from freehand sketches and constraints. *Computer-Aided Design*, 29(2) :101–112, February 1997.

[10] Michael Banks and Elaine Cohen. Real Time Spline Curves from Interactively Sketched Data. In *Proceedings of the 1990 Symposium on Interactive 3D Graphics*, I3D '90, pages 99–107, New York, NY, USA, 1990. ACM.

[11] Levent Burak Kara and Kenji Shimada. Construction and Modification of 3d Geometry Using a Sketch-based Interface. In *Proceedings of the Third Eurographics Conference on Sketch-Based Interfaces and Modeling*, SBM'06, pages 59–66, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[12] Tevfik Metin Sezgin, Thomas Stahovich, and Randall Davis. Sketch Based Interfaces : Early Processing for Sketch Understanding. In *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[13] Faramarz Famil and Samavati Nezam Mahdavi-amiri. *A Filtered B-spline model of Scanned Digital Images*. 1998.

[14] Rangachar Kasturi, Lawrence OGorman, and Venu Govindaraju. Document image analysis : A primer. *Sadhana*, 27(1) :3–22, February 2002.

[15] Luke Olsen, Faramarz F. Samavati, Mario Costa Sousa, and Joaquim A. Jorge. Sketch-based modeling : A survey. *Computers & Graphics*, 33(1) :85–103, February 2009.

[16] Bruno Rodrigues De Araújo, Rua Alves Redol, Joaquim Armando, and Pires Jorge. Blobmaker : Free-form modelling with variational implicit surfaces. In *In Proc. of the 12th Portuguese Computer Graphics Meeting*, pages 17–26, 2003.

[17] John F. Hughes, Joaquim A. Jorge (editors, Timo Fleisch, Florian Rechel, Pedro Santos, and André Stork. *Constraint Stroke-Based Oversketching for 3D Curves ABSTRACT*.

[18] Levent Burak Kara, Chris M. D'Eramo, and Kenji Shimada. Pen-based Styling Design of 3d Geometry Using Concept Sketches and Template Models. In *Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling*, SPM '06, pages 149–160, New York, NY, USA, 2006. ACM.

[19] Richard Pusch, Faramarz Samavati, Ahmad Nasri, and Brian Wyvill. Improving the Sketch-based Interface : Forming Curves from Many Small Strokes. *Vis. Comput.*, 23(9) :955–962, August 2007.

[20] Amit Shesh and Baoquan Chen. SMARTPAPER : An Interactive and User Friendly Sketching System. *Computer Graphics Forum*, 23(3) :301–310, September 2004.

[21] Levent Burak Kara and Thomas F. Stahovich. An image-based, trainable symbol recognizer for hand-drawn sketches. *Computers & Graphics*, 29(4) :501–517, August 2005.

[22] David A Huffman. Impossible objects as nonsense sentences. *Machine intelligence*, 6(1) :295–323, 1971.

[23] Takeo Kanade. Recovery of the Three-Dimensional Shape of an Object from a Single View. *Artificial Intelligence*, 17 :409 – 460, 1981.

[24] H Lipson and M Shpitalni. Optimization-based reconstruction of a 3d object from a single freehand line drawing. *Computer-Aided Design*, 28(8) :651–663, August 1996.

[25] Pedro Company, Manuel Contero, Julian Conesa, and Ana Piquer. An optimisation-based reconstruction engine for 3d modelling by sketching. *Computers & Graphics*, 28(6) :955–979, December 2004.

[26] Jianzhuang Liu, Liangliang Cao, Zhenguo Li, and Xiaoou Tang. Plane-Based Optimization for 3d Object Reconstruction from Single Line Drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2) :315–327, February 2008.

[27] Makoto Okabe, Shigeru Owada, and Takeo Igarashi. Interactive Design of Botanical Trees Using Freehand Sketches and Example-based Editing. In *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[28] Xuejin Chen, Boris Neubert, Ying-Qing Xu, Oliver Deussen, and Sing Bing Kang. Sketch-based Tree Modeling Using Markov Random Field. In *ACM SIGGRAPH Asia 2008 Papers*, SIGGRAPH Asia '08, pages 109 :1–109 :9, New York, NY, USA, 2008. ACM.

[29] J. Wither, F. Boudon, M.-P. Cani, and C. Godin. Structure from silhouettes : a new paradigm for fast sketch-based design of trees. *Computer Graphics Forum*, 28(2) :541–550, April 2009.

[30] Steven Longay, Adam Runions, Frédéric Boudon, and Przemyslaw Prusinkiewicz. TreeSketch : Interactive Procedural Modeling of Trees on a Tablet. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, SBIM '12, pages 107–120, Aire-la-Ville, Switzerland, Switzerland, 2012. Eurographics Association.

[31] Malik Shahzad. A sketching interface for modeling and editing hairstyles.

[32] Jamie Wither, Florence Bertails, and Marie-paule Cani. Realistic hair from a sketch. In *in Shape Modeling International*, 2007.

[33] Hongbo Fu, Yichen Wei, Chiew-Lan Tai, and Long Quan. Sketching Hairstyles. In *Proceedings of the 4th Eurographics Workshop on Sketch-based Interfaces and Modeling*, SBIM '07, pages 31–36, New York, NY, USA, 2007. ACM.

[34] Emmanuel Turquin, Jamie Wither, Laurence Boissieux, Marie-Paule Cani, and John F. Hughes. A Sketch-Based Interface for Clothing Virtual Characters. *IEEE Computer Graphics and Applications*, 27(1) :72–81, 2007.

[35] Emmanuel Turquin, Marie-Paule Cani, and John F. Hughes. Sketching Garments for Virtual Characters . In *Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 175–182, August 2004.

[36] Philippe Decaudin, Dan Julius, Jamie Wither, Laurence Boissieux, Alla Sheffer, and Marie-Paule Cani. Virtual Garments : A Fully Geometric Approach for Clothing Design. *Computer Graphics Forum*, 25(3) :625–634, September 2006.

[37] Cody Robson, Ron Maharik, Alla Sheffer, and Nathan Carr. Context-aware garment modeling from sketches. *Computers & Graphics*, 35(3) :604–613, June 2011.

[38] Chen Yang, Dana Sharon, and Michiel van de Panne. Sketch-based Modeling of Parameterized Objects. In *ACM SIGGRAPH 2005 Sketches*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[39] Sang-Uk Cheon and Soonhung Han. A Template-based Reconstruction of Plane-symmetric 3d Models from Freehand Sketches. *Comput. Aided Des.*, 40(9) :975–986, September 2008.

[40] Xiaohua Xie, Kai Xu, Niloy J. Mitra, Daniel Cohen-Or, Wenyong Gong, Qi Su, and Baoquan Chen. Sketch-to-Design : Context-Based Part Assembly. *Computer Graphics Forum*, 32(8) :233–245, 2013.

[41] Chen Mao, Sheng Feng Qin, and David Wright. A sketch-based approach to human body modelling. *Computers & Graphics*, 33(4) :521–541, August 2009.

[42] Xuejin Chen, Sing Bing Kang, Ying-Qing Xu, Julie Dorsey, and Heung-Yeung Shum. Sketching Reality : Realistic Interpretation of Architectural Designs. *ACM Trans. Graph.*, 27, 2008.

[43] Jamie Wither, Antoine Bouthors, and Marie-Paule Cani. Rapid Sketch Modeling of Clouds. In *Proceedings of the Fifth Eurographics Conference on Sketch-Based Interfaces and Modeling*, SBM'08, pages 113–118, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[44] Adeline Pihuit, Marie-Paule Cani, and Olivier Palombi. Sketch-Based Modeling of Vascular Systems : a First Step Towards Interactive Teaching of Anatomy. In Marc Alexa, Ellen Yi-Luen Do, editor, *SBIM 2010 - Sketch-Based Interfaces and Modeling*, pages 151–158, Annecy, France, June 2010. Eurographics Association.

[45] A. Bernhardt, A. Pihuit, M. P. Cani, and L. Barthe. Matisse : Painting 2d Regions for Modeling Free-form Shapes. In *Proceedings of the Fifth Eurographics Conference on Sketch-Based Interfaces and Modeling*, SBM'08, pages 57–64, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.

[46] A. Alexe, V. Gaildrat, and L. Barthe. Interactive Modelling from Sketches Using Spherical Implicit Functions. In *Proceedings of the 3rd International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, AFRIGRAPH '04, pages 25–34, New York, NY, USA, 2004. ACM.

[47] Chiew-lan Tai, Hongxin Zhang, and Jacky Chun-kin Fong. Prototype Modeling from Sketched Silhouettes based on Convolution Surfaces. *Computer Graphics Forum*, 23 :71–83, 2004.

[48] Anca Alexe, Loic Barthe, Marie Paule Cani, and Véronique Gaildrat. Shape modelling by sketching using convolution surfaces. In *In Pacific Graphics (Short Papers*, 2005.

[49] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. SKETCH : An Interface for Sketching 3d Scenes. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA, 2006. ACM.

[50] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. FiberMesh : Designing Freeform Surfaces with 3d Curves. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[51] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy : A Sketching Interface for 3d Freeform Design. In *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[52] Ryan Schmidt, Azam Khan, Karan Singh, and Gord Kurtenbach. Analytic Drawing of 3d Scaffolds. *ACM Transactions on Graphics*, 28(5) :(to appear), 2009. Proceedings of SIGGRAPH ASIA 2009.

[53] Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. ILoveSketch : As-natural-as-possible Sketching System for Creating 3d Curve Models. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*, UIST '08, pages 151–160, New York, NY, USA, 2008. ACM.

[54] Alec Rivers, Frédo Durand, and Takeo Igarashi. 3d Modeling with Silhouettes. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, pages 109 :1–109 :8, New York, NY, USA, 2010. ACM.

[55] John F. Hughes, Joaquim A. Jorge (editors, Olga Karpenko, John F. Hughes, and Ramesh Raskar. Epipolar methods for multi-view sketching. In *In Eurographics Workshop on Sketch-Based Interfaces and Modeling*, pages 167–173, 2004.

[56] Olga A. Karpenko and John F. Hughes. SmoothSketch : 3d Free-form Shapes from Complex Sketches. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 589–598, New York, NY, USA, 2006. ACM.

[57] Thomas Lewiner, João D. Gomes Jr., Hélio Lopes, and Marcos Craizer. Curvature and torsion estimators based on parametric curve fitting. *Computers & Graphics*, 29(5) :641–655, October 2005.

[58] Jeffrey C. Lagarias, James A. Reeds, Margaret H. Wright, and Paul E. Wright. Convergence Properties of the NelderMead Simplex Method in Low Dimensions. *SIAM J. on Optimization*, 9(1) :112–147, May 1998.

[59] Xianfang Sun, Paul L. Rosin, Ralph R. Martin, and Frank C. Langbein. *Bas-Relief Generation Using Adaptive Histogram.*

[60] Wenhao Song, A. Belyaev, and H.-P. Seidel. Automatic Generation of Bas-reliefs from 3d Shapes. In *IEEE International Conference on Shape Modeling and Applications, 2007. SMI '07*, pages 211–214, June 2007.

[61] Zhongping Ji, Xianfang Sun, Shi Li, and Yigang Wang. Real-time Bas-Relief Generation from Depth-and-Normal Maps on GPU. *Computer Graphics Forum*, 33(5) :75–83, August 2014.