Single-view 3D reconstruction of curves

Ali Fakih

Université De Haute-Alsace

Université De Strasbourg

Thèse

Pour l'obtention du grade de
Docteur de l'Université de Haute-Alsace
École Doctorale: Mathématiques, Sciences de l'Information et de l'Ingénieur (ED 269)
Discipline : Informatique

Présentée et soutenue publiquement
par

Ali Fakih

Le 21 Novembre 2023

Single-View 3D Reconstruction Of Curves

Sous la direction du Dr. Frédéric Cordier

Jury

Dimitry Sokolov, Associate professor, Université de Lorraine (Rapporteur)
Guillaume Lavoué, Full professor, École Nationale d'Ingénieurs de Saint-Étienne (Rapporteur)
Caroline Essert, Full professor, Université de Strasbourg (Examinateur)
Amal Parakkat, Associate professor, Telecom Paris (Examinateur)
Lhassane Idoumghar, Full professor, Université de Haute-Alsace (Examinateur)
Frédéric Cordier, Associate professor, Université de Haute-Alsace (Directeur de thèse)
Yvan Maillot, Associate professor, Université de Haute-Alsace (Encadrant de thèse)

Abtract

The computer graphics field has made significant advancements in areas such as rendering and animation. However, a key challenge remains in accurately capturing users' intentions for 3D modeling in a simple and intuitive manner. This is due to the limitations of two-dimensional interfaces, which do not properly accommodate our natural perception and thinking in a three-dimensional environment. One particularly challenging aspect is drawing 3D curves within a 2D input device. These curves are crucial in various applications such as defining the trajectories of moving objects or modeling 3D objects. Several works have been proposed to address this problem, such as changing perspectives or requiring the artist to draw the shadow of the curve. However, most of these methods pose some constraints to the artists, leading to reduced efficiency. In this thesis, our ultimate goal is the 3D reconstruction of planar polygonal curves, including circular helices, Euler spirals, and hand-drawn curves, using multiple approaches. Motivated by the advancements of machine learning techniques, we employ these methods to perform the 3D reconstruction of circular helices. Building on this, we then extended our investigation to a more general family of curves, specifically, the Euler spirals. The limitation of machine learning techniques in the reconstruction of such curves led us to adopt a different methodology. We employ a piecewise curve-matching approach to generate the reconstructed 3D Euler spiral curve. Finally, we introduced a novel approach that can perform the 3D reconstruction of various input curves by minimizing the curvature variance of the reconstructed curve. All methods used in the different approaches were evaluated against synthetic and hand-drawn curves.

Résumé

L'infographie a fait des progrès significatifs dans des répétition tels que le rendu et l'animation. Cependant, un défi majeur demeure dans la capture précise des intentions des utilisateurs pour la modélisation 3D d'une manière simple et intuitive. Cela est dû aux limites des interfaces bidimensionnelles, qui ne s'adaptent pas correctement à notre perception et à notre pensée naturelle dans un environnement tridimensionnel. Un aspect particulièrement difficile est de dessiner des courbes 3D d'un périphérique d'entrée 2D. Ces courbes sont cruciales dans diverses applications telles que la définition de trajectoires d'objets en mouvement ou la modélisation d'objets 3D. Plusieurs travaux ont été proposés pour résoudre ce problème, comme changer de perspective ou demander à l'utilisateur de dessiner l'ombre de la courbe. Cependant, la plupart de ces méthodes posent certaines contraintes qui affectent leur efficacité. Dans cette thèse, notre objectif ultime est la reconstruction 3D de courbes polygonales planes, y compris les hélices circulaires, les spirales d'Euler et les courbes dessinées à la main, en utilisant plusieurs approches. Motivés par les progrès des techniques d'apprentissage automatique, nous utilisons ces méthodes pour effectuer la reconstruction 3D d'hélices circulaires. Sur cette base, nous avons ensuite étendu notre enquête à une famille plus générale de courbes, en particulier les spirales d'Euler. La limitation des techniques d'apprentissage automatique dans la reconstruction ces dernières nous a fait passer à une méthodologie différente. Nous utilisons une approche d'appariement de courbes par morceaux pour générer la courbe en spirale d'Euler 3D reconstruite. Enfin, nous avons introduit une nouvelle approche qui peut effectuer la reconstruction 3D de diverses courbes d'entrée en minimisant la variance de courbure de la courbe reconstruite. Toutes les méthodes utilisées dans les différentes approches ont été évaluées par rapport à des courbes synthétiques et dessinées à la main.

Declaration

I'm submitting this thesis for the degree of Doctor of Philosophy in Computer Science at the Université de Haute-Alsace. My research was conducted under the supervision of Dr. Frederic Cordier and Dr. Yvan Maillot. This work is original, with references to previous studies where applicable. I confirm that this thesis is not being submitted for any other degree, diploma, or qualification at any other university.

Publications

- A. Fakih, F. Cordier, Y. Maillot. (October 2023). 'Piecewise Reconstruction of 3D Euler Spirals From Planar Polygonal Curve', In: International Journal of Computer Science, Engineering and Information Technology (IJCSEIT), Vol.13, No.5 (https://doi.org/10.5121/ijcseit.2023.13501).
- A. Fakih, N. Wilser, Y. Maillot, F. Cordier (2023).'Single-view 3D reconstruction of curves'. In: CGI 2023. Lecture Notes in Computer Science, Springer (**Accepted**).

Acknowledgement

I want to express my heartfelt thanks to the people who've been there for me during this journey.

First, I'm incredibly grateful to my supervisors, Dr. Frederic Cordier and Dr. Yvan Maillot. They've been my guides, sharing their knowledge and support every step of the way. Whenever I had questions or faced challenges, they were there for me.

I also want to thank Dr. Dimitry Sokolov and Prof. Guillaume Lavoué for reviewing my Ph.D. thesis, and the jury members, Prof. Caroline Essert and Dr. Amal Parakkat, for evaluating my work.

A big shout-out to Prof. Lhassane Idoumghar, the head of our laboratory, for his constant support and guidance.

I'm thankful for all my colleagues at the Université de Haute-Alsace. We've worked together professionally and formed great personal bonds.

To my Lebanese friends in Mulhouse, your encouragement and support mean the world to me. I would like to thank them for standing by my side.

Lastly, I owe a special thanks to my parents, sisters, and girlfriend. They've supported me throughout these three years and all the years before this journey began.

I couldn't have done it without all of you.

Ali Fakih

Contents

List of Figures

List of Tables

List of Abbreviations

ML Machine Learning

RFR Random forest regressor

GBR Gradient boosting regressor

GPR Gaussian process regressor

SVR Support vector regressor

KNN-R K-nearest neighbors regressor

ANN Artificial neural network

ED-CNN Encoder-decoder convolutional neural network

Résumé des chapitres en français

Chapitre 1: Introduction

La reconstruction tridimensionnelle (3D) est une technique permettant de créer un modèle ou une représentation 3D d'un objet en utilisant des informations provenant de diverses sources, telles que des images 2D, des données de nuages de points, des scans laser ou d'autres sources de données. Cette représentation 3D a de nombreuses applications dans différents domaines. En infographie, par exemple, les modèles 3D créés par reconstruction peuvent être utilisés pour créer des jeux vidéo, des films et des animations. La vision par ordinateur peut utiliser des modèles 3D pour la reconnaissance, tandis que la robotique peut utiliser des modèles 3D pour la navigation, l'évitement d'obstacles et la manipulation d'objets. De plus, l'ingénierie peut utiliser des modèles 3D pour la conception, la simulation et le prototypage de produits.

Le but de la reconstruction 3D est de restituer la structure 3D d'un objet à partir d'un ensemble de mesures ou d'observations. Diverses techniques peuvent être utilisées à cette fin, notamment la photogrammétrie, la lumière structurée, la vision stéréo, le LiDAR. De plus, l'optimisation et l'apprentissage en profondeur sont également des approches couramment employées.

Les approches d'apprentissage profond et d'apprentissage automatique sont devenues de plus en plus populaires dans le domaine de la reconstruction 3D. Ces techniques peuvent être utilisées pour extraire des informations d'images ou d'autres sources de données afin de générer des représentations 3D précises. Une approche consiste à utiliser des réseaux de neurones convolutifs (CNN) pour extraire des caractéristiques d'images ou de nuages de points, qui sont ensuite

utilisées pour estimer la structure 3D de l'objet (Yang, Cui, Belongie & Hariharan 2018, Wu et al. 2016, Knyaz et al. 2018, Smith & Meger 2017, Yang, Rosa, Markham, Trigoni & Wen 2018, Li et al. 2018). Cette approche est souvent utilisée pour la reconstruction 3D à vue unique, où un modèle 3D est généré à partir d'une seule image. Par exemple, un CNN peut être formé pour prédire la carte de profondeur d'une image, qui peut ensuite être convertie en un modèle 3D à l'aide de techniques stéréo photométriques. Une autre approche consiste à utiliser des réseaux antagonistes génératifs (GAN) pour générer des modèles 3D à partir d'un ensemble d'images 2D ou de nuages de points (Zhang et al. 2021, Nozawa et al. 2022). Les GAN peuvent être formés pour générer des modèles 3D réalistes en apprenant la distribution des données de formation. Cette approche a été utilisée pour des applications telles que la génération de modèles 3D de meubles ou d'autres objets à partir d'images 2D. D'autres techniques d'apprentissage automatique, telles que les forêts aléatoires et les machines à vecteurs de support, ont également été utilisées pour la reconstruction 3D. Ces techniques peuvent être utilisées pour l'extraction et la classification de caractéristiques, ainsi que pour estimer la structure 3D d'un objet. Dans l'ensemble, les approches d'apprentissage en profondeur et d'apprentissage automatique se sont révélées prometteuses dans le domaine de la reconstruction 3D et sont susceptibles de jouer un rôle de plus en plus important dans les développements futurs dans ce domaine.

Dans cette thèse, nous nous concentrons sur la reconstruction de courbe 3D à partir de courbes polygonales planes, en privilégiant les courbes en hélice circulaire et en spirale d'Euler. Ce choix est motivé par le fait qu'il existe une pénurie dans la littérature de travaux visant à réaliser la reconstruction 3D de courbes. De plus, alors que certains travaux ont été réalisés sur la reconstruction 3D d'hélices circulaires à partir de courbes polygonales planes, il existe un manque de littérature sur l'utilisation des courbes en spirale d'Euler à cette fin. Pour commencer notre étude, nous avons d'abord travaillé sur la reconstruction 3D d'hélices circulaires à partir de courbes polygonales planes. Motivés par la popularité croissante des techniques Machine Learning (ML), nous avons choisi d'utiliser ces techniques pour notre reconstruction et avons obtenu des résultats acceptable par rapport aux véritables hélices circulaires.

Sur cette base, nous avons ensuite étendu notre enquête à une famille plus

générale de courbes, en particulier les spirales d'Euler. Nous avons dans un premier temps tenté d'appliquer les mêmes techniques ML utilisées pour les hélices circulaires aux spirales d'Euler. Cependant, cette approche n'a pas donné de résultats prometteurs en raison de la complexité inhérente des spirales d'Euler par rapport aux hélices circulaires. Les spirales d'Euler se caractérisent par une évolution linéaire continue de la courbure et de la torsion par rapport à la longueur de l'arc, les rendant mathématiquement plus complexes que les hélices circulaires dont la courbure et la torsion sont constantes sont constantes. Cette complexité accrue a posé des défis aux algorithmes ML, qui s'efforçaient de capturer et de modéliser efficacement les variations complexes de courbure et de torsion le long des spirales d'Euler sans donner de résultat satisfaisant. Pour résoudre ce problème, nous avons changé de méthode et traité le problème en utilisant une approche d'appariement de courbes par morceaux. En substance, ce processus de reconstruction consiste d'abord à diviser la courbe 2D d'entrée en petits morceaux. Pour chaque morceau, nous recherchons les correspondances les plus proches à partir d'un ensemble de morceaux de spirales d'Euler préalablement choisis. Ensuite, nous connectons les pièces appariées et appliquons un algorithme de lissage pour générer la courbe en spirale d'Euler 3D reconstruite par morceaux. Bien que cette méthode ait permis d'obtenir une bonne reconstruction 3D, sa portée était limitée aux courbes en spirale d'Euler. Pour élargir l'applicabilité de notre approche de reconstruction, nous avons développé une nouvelle méthode capable de reconstruire plusieurs types de courbes, y compris les hélices circulaires et les spirales d'Euler. Cette approche a minimisé la variance de courbure de la courbe en ajustant des ellipses à la courbe d'entrée, ce qui nous a permis de déterminer les cercles osculateurs et les tangentes à chaque point de la courbe pour la reconstruction 3D.

Chapitre 2: Apprentissage automatique pour la reconstruction 3D d'hélices circulaires à partir de courbes 2D

Les hélices circulaires sont un type de courbe 3D qui apparaît dans diverses structures naturelles et artificielles. Ils sont caractérisés par une courbure et une torsion

constante le long de leur arc. Leurs formes géométriques peuvent être observées dans certaines structures naturelles, en particulier les structures biologiques, et ont également été utilisées dans un large éventail d'applications de la technologie humaine. À l'échelle microscopique, des structures hélicoïdales peuvent être observées en biologie et en nanotechnologie, comme dans le sperme (Saggiorato et al. 2017), l'ADN (Dickerson 1983) et les nanoressorts (McIlroy et al. 2001). Alors qu'à l'échelle macro, ils peuvent être trouvés dans les plantes (Goriely & Tabor 1998), les coquillages (Forterre & Dumais 2011) et divers composants d'ingénierie tels que les ressorts (Ke et al. 2020) et les cordes en fibres synthétiques (He et al. 2020).

Bien que de nombreux chercheurs travaillent sur la reconstruction 3D d'hélices circulaires à partir de courbes polygonales planes, la plupart de leurs approches reposaient sur des algorithmes d'optimisation. Cependant, au cours de la dernière décennie, les algorithmes de ML/deep learning ont prouvé leur efficacité dans le domaine de la reconstruction 3D. À notre connaissance, aucun travail antérieur n'a utilisé ces méthodes dans le cadre de la reconstruction d'hélice. Ce chapitre présente une nouvelle approche qui comble cette lacune en utilisant des algorithmes ML/deep learning pour reconstruire des hélices circulaires 3D à partir de courbes polygonales planes. Plus précisément, nous avons formé différents modèles ML, y compris le régresseur de forêt aléatoire (RFR)(Breiman 2001), le régresseur de gradient boostant (GBR) (Friedman 2001), le régresseur de processus gaussien (GPR)(Rasmussen et al. 2006), prend en charge le régresseur vectoriel (SVR) (Smola & Schölkopf 2004), le réseau de neurones artificiels (ANN) (McClelland et al. 1987) et le réseau de neurones convolutionnels codeur-décodeur (ED-CNN) (LeCun et al. 1998) utilisant un ensemble de données composé de morceaux d'hélices circulaires 3D. Nous reconstruisons les hélices circulaires 3D de manière par morceaux et comparons les résultats de chaque algorithme ML dans la section expérimentale. Enfin, nous avons analysé les résultats et tiré des conclusions concernant l'efficacité de l'utilisation des algorithmes ML pour la reconstruction 3D des hélices circulaires.

Notre objectif est de développer un modèle ML qui reconstruit des hélices circulaires 3D à partir de courbes polygonales planes. Plus précisément, la courbe d'entrée doit se présenter sous la forme d'une projection polygonale d'une hélice ou d'une hélice dessinée à la main dans le plan (x, y). En raison du fait que la courbe

d'entrée peut être de n'importe quelle longueur et de la forme spéciale d'une hélice, qui se répète le long de la longueur de l'arc, il est pratique de diviser la courbe d'entrée en segments plus petits. Par conséquent, la reconstruction est effectuée de manière fragmentaire, où nous effectuons une segmentation de courbe pour diviser la courbe d'entrée en segments plus petits. Chaque segment est ensuite reconstruit individuellement. Une fois tous les segments reconstruits, ils sont assemblés pour former l'hélice 3D approximative dont la projection sur le plan 2D correspond le mieux à la courbe polygonale d'entrée. Pour améliorer le lissage global de la courbe, en particulier aux points de connexion de ses segments, un algorithme de lissage post-traitement est appliqué. Cet algorithme de lissage permet d'affiner la courbe reconstruite et d'améliorer sa qualité esthétique.

Pour entraîner les modèles ML, nous créons un ensemble de données composé de segments d'hélices circulaires, chacun comprenant 100 points dans le plan (x, y, z). Nous appliquons un échantillonnage uniforme pour nous assurer que chaque segment est représenté de manière cohérente. Il s'ensuit une mise à l'échelle uniforme qui supprime le facteur d'échelle de l'ensemble de données, nous permettant de gérer les courbes d'entrée de n'importe quelle échelle. Pendant la phase d'entraînement, le modèle utilise les coordonnées (x, y) des segments d'hélice en entrée pour générer leurs coordonnées z correspondantes en sortie. Notons que nous expérimentons différents modèles ML tels que RFR, GBR, GPR, SVR, ANN et ED-CNN. Pour chaque modèle, nous effectuons un réglage hyperparamétrique pour trouver son architecture optimale et obtenir les meilleures performances possibles.

Une fois le modèle ML formé, il peut être utilisé pour la reconstruction d'une courbe polygonale d'entrée donnée. Une condition de notre approche de reconstruction est que la courbe d'entrée ne peut pas être n'importe quelle courbe aléatoire telle que des lignes droites ou des courbes avec des angles vifs. Nous ciblons spécifiquement la reconstruction de courbes qui peuvent être la projection orthogonale de véritables hélices circulaires et même des courbes d'hélices circulaires dessinées à la main avec une certaine précision.

Après avoir construit un modèle ML capable d'effectuer la reconstruction 3D de l'hélice à partir de la courbe polygonale d'entrée, nous appliquons une approche de segmentation de courbe, qui consiste à diviser la courbe d'entrée en segments

plus petits. Chaque segment est ensuite reconstruit indépendamment. Une fois que tous les segments sont reconstruits, ils sont combinés pour former l'hélice 3D reconstruite approximative complète. Afin d'améliorer le lissage global de la courbe, en particulier aux points de connexion entre les segments, un algorithme de lissage post-traitement est utilisé. Cet algorithme joue un rôle essentiel dans l'affinement de la courbe reconstruite et l'amélioration de sa qualité esthétique.

Pour déterminer la meilleure segmentation de la courbe d'entrée, nous utilisons l'algorithme de programmation dynamique utilisé dans (McCrae & Singh 2011). Un paramètre essentiel dans cet algorithme est noté E_{cost}, qui est la variance de courbure d'une courbe donnée. Nous calculons la partie triangulaire supérieure d'une matrice M, qui a des dimensions $n \times n$, et n représente le nombre de points dans la courbe polygonale d'entrée P. Les entrées $M(i,j)$, avec $1 \leq i < j \leq n$, sont calculées de manière ascendante, à partir des éléments les plus proches de la diagonale, en utilisant l'équation suivante :

$$M(i,j) = \min \left\{ E_{\text{cost}}(i,j), \min_{i<k<j} \{M(i,k) + M(k,j)\} \right\}$$

L'équation ci-dessus permet de déterminer s'il est préférable de reconstruire une seule courbe ($E_{\text{cost}}(i,j)$) pour toute la section $p_i \dots p_j$, ou d'en subdiviser et d'en utiliser une (ou more) morceaux pour chacun des intervalles $p_i \dots p_k$ et $p_k \dots p_j$. soit de diviser la courbe en un ou plusieurs morceaux, chacun correspondant aux intervalles $p_i \dots p_k$ et $p_k \dots p_j$. Les informations concernant l'endroit où diviser la courbe d'entrée sont stockées au fur et à mesure que la matrice M est remplie.

Une fois le processus terminé, l'élément $M(1,n)$ représente le coût de la configuration optimale pour les points $p_1 \dots p_n$ dans la courbe polygonale P. Pour reconstruire la solution, nous naviguons dans la matrice M à partir de l'élément $(1,n)$ et faisons les divisions nécessaires si nécessaire.

Après avoir divisé la courbe polygonale d'entrée en segments et reconstruit chaque segment à l'aide de l'algorithme ML, nous assemblons les pièces reconstruites pour former l'hélice reconstruite en 3D. Cependant, ce procédé d'assemblage introduit des discontinuités G^1 et G^2 (discontinuités de tangente et de courbure), notamment aux points de raccordement des pièces reconstruites. Pour résoudre ce problème, nous échantillonnons d'abord uniformément la courbe hélicoïdale recon-

struite, puis nous appliquons le filtre de Savitzky-Golay (Savitzky & Golay 1964) pour améliorer son lissage. Ce filtre réduit efficacement le bruit et les irrégularités de la courbe en ajustant une fonction polynomiale aux segments locaux de la courbe, ce qui donne une représentation plus fluide et plus attrayante visuellement.

Après avoir divisé la courbe polygonale d'entrée en segments et reconstruit chaque segment à l'aide de l'algorithme ML, nous assemblons les pièces reconstruites pour former l'hélice reconstruite en 3D. Cependant, ce procédé d'assemblage introduit des discontinuités G^1 et G^2 (discontinuités de tangente et de courbure), notamment aux points de raccordement des pièces reconstruites. Pour résoudre ce problème, nous échantillonnons d'abord uniformément la courbe hélicoïdale reconstruite, puis nous appliquons le filtre de Savitzky-Golay (Savitzky & Golay 1964) pour améliorer son lissage. Ce filtre réduit efficacement le bruit et les irrégularités de la courbe en ajustant une fonction polynomiale aux segments locaux de la courbe, ce qui donne une représentation plus fluide et plus attrayante visuellement.

Enfin, nous avons testé la reconstruction de nos modèles ML formés contre deux types de courbes. Courbes d'entrée synthétiques, qui sont la projection orthogonale des vraies hélices circulaires, et courbes dessinées à la main. Les résultats montrent que tous les algorithmes ML ont fourni des résultats visuellement acceptables. Cependant, il convient de noter que KNN et ANN ont systématiquement obtenu les meilleures performances de reconstruction en fonction du paramètre d'erreur $E_{\text{cost}(P)} = \text{Var}(\kappa(P))$, qui quantifie la variance de la courbure, caractéristique importante de l'hélice.

Chapitre 3: Reconstruction 3D de courbes

Dans le chapitre précédent, les modèles ML, en particulier ANN et KNN, ont démontré leur efficacité pour reconstruire une hélice 3D approximative à partir de courbes polygonales planes. Cependant, la simplicité de la courbe d'hélice, avec sa courbure et sa torsion constantes le long de sa longueur d'arc, a joué un rôle crucial dans le succès des modèles ML pour la reconstruction d'hélice. Néanmoins, lorsqu'ils sont confrontés à des familles de courbes plus complexes comme les

courbes en spirale d'Euler, qui présentent une courbure et une torsion évoluant linéairement, ou des courbes de forme libre, ces modèles ne peuvent pas fournir une bonne reconstruction. Une autre limitation des modèles ML est leur incapacité à fournir plusieurs reconstructions pour chaque segment de la courbe d'entrée, ce qui entraîne une discontinuité entre les segments reconstruits assemblés. Cette limitation provient du fait que nous ne pouvons avoir qu'une seule reconstruction candidate pour chaque segment, ce qui nous empêche de poser des restrictions de continuité de courbure et de torsion lors du processus d'assemblage. Il est important de noter qu'une courbe polygonale en 2D peut correspondre à plusieurs courbes dans l'espace 3D.

Motivés par les limitations susmentionnées, nous explorons des méthodes alternatives pour relever les défis posés par des familles de courbes plus complexes. Dans ce chapitre, nous présentons une nouvelle approche pour reconstruire des spirales d'Euler 3D en utilisant une technique d'appariement de courbes par morceaux. De plus, nous introduisons une méthode plus généralisée capable de reconstruire différents types de courbes. Cette méthode est basée sur l'idée d'ajuster un ensemble d'ellipses à la courbe d'entrée, ce qui nous permet de déterminer les cercles osculateurs et, par la suite, la tangente en chaque point de la courbe, facilitant le processus de reconstruction 3D.

Les spirales d'Euler 3D sont des courbes esthétiques dont la courbure et la torsion évoluent linéairement avec la longueur de l'arc (Kimia et al. 2003, Knuth 1979, Ullman 1976). Ils possèdent des propriétés souhaitables, telles que l'invariance aux transformations de similarité (translation, rotation et mise à l'échelle), la symétrie, l'extensibilité et le lissage (Gur Harary 2012).

Dans ce travail, nous nous concentrons sur la reconstruction 3D de spirales d'Euler à partir de courbes polygonales planes qui peuvent être utilisées dans différentes applications car il existe de nombreux domaines dans lesquels les spirales d'Euler 3D sont utiles comme les véhicules aérospatiaux (en créant des transitions douces entre différents régimes de vol) , les turbomachines (en créant des voies d'écoulement lisses et incurvées en continu), les dispositifs médicaux (pour concevoir des dispositifs médicaux nécessitant des voies lisses et incurvées à travers le corps, comme les cathéters et les endoscopes).

Malgré la présence de toutes les propriétés mentionnées, à notre connaissance, il

n'existe aucun travail antérieur dans la littérature qui vise à reconstruire les spirales d'Euler 3D à partir de courbes polygonales planes. Le but de notre méthode est de reconstruire des spirales d'Euler 3D approximatives par morceaux à partir de courbes polygonales planes. L'entrée de notre méthode est une courbe polygonale 2D dans le plan (x, y) et la sortie est une spirale d'Euler 3D par morceaux telle que sa projection orthogonale sur le plan (x, y) est proche de la courbe polygonale d'entrée.

L'approche courante pour l'appariement de courbe consiste à utiliser l'équation qui calcule les coordonnées des points le long de la courbe à travers la paramétrisation de la longueur de l'arc. Cependant, cette forme n'existe pas pour la spirale d'Euler 3D. Alternativement, les coordonnées des points peuvent être calculées par optimisation, mais cela nécessiterait un temps de calcul important. Au lieu d'appliquer les algorithmes d'appariement de courbes typiques, nous créons un ensemble de données contenant de courts segments de spirales d'Euler 3D avec leur projection polygonale plane. Pour permettre une couverture complète des diverses formes qui peuvent être rencontrées dans la courbe polygonale d'entrée, nous faisons en sorte que l'ensemble de données ait des segments de différentes longueurs et nous leur appliquons différentes rotations en plus d'effectuer un échantillonnage et une mise à l'échelle uniformes pour améliorer la représentation du segment. et pour supprimer le facteur d'échelle de notre ensemble de données.

Dans cette direction, la reconstruction commence par diviser la courbe polygonale 2D d'entrée S en segments $\{s_1, s_2, ...\}$; la division est basée sur l'approche de dichotomie (Lien 1981) pour trouver un certain nombre de segments de spirale d'Euler 3D appariés à partir de l'ensemble de données pour chaque segment de la courbe d'entrée. $C_{s_i} = \{c_{i,1}, c_{i,2}, ...\}$ représente l'ensemble des segments appariés candidats pour le segment s_i. À ce stade, pour chaque segment de la courbe d'entrée, nous avons un pool de segments de spirale d'Euler correspondants possibles (candidats) à partir de l'ensemble de données. Nous implémentons un nouvel algorithme qui sélectionne la connexion optimale des candidats (un candidat de chaque groupe de candidats) en utilisant l'algorithme de Dijkstra (Dijkstra 1959). La sélection de deux candidats quelconques qui seront liés l'un à l'autre est basée sur plusieurs critères tels que leur continuité de courbure, continuité de torsion, continuité tangente, continuité normale et la distance entre leur projection polyg-

onale et la courbe d'entrée. Le résultat de cette étape est une spirale d'Euler par morceaux représentée sous la forme d'une courbe polygonale 3D. Un lissage est alors appliqué sur cette courbe polygonale pour améliorer la continuité C^0 et C^1 afin d'obtenir une courbe reconstruite plus agréable à l'oeil.

Nous testons notre modèle contre des courbes d'entrée synthétiques de véritable spirale d'Euler et contre des courbes dessinées à la main. Les résultats montrent que notre algorithme de reconstruction produit une courbe visuellement agréable et un résultat assez similaire aux spirales d'Euler de vérité terrain. Notre reconstruction fournit des courbes 3D avec C^0 et C^1-continuité. Même les continuités G^2 et G^3 peuvent être considérées comme acceptables car elles sont principalement continues à l'exception de quelques pics de courbure et de torsion se produisant principalement aux points de jonction des segments. En effet, la courbure et la torsion reposent sur la dérivée seconde, ce qui les rend très sensibles aux petits changements de la courbe qui peuvent être difficiles à discerner visuellement.

La première approche s'est concentrée sur la reconstruction des courbes en spirale d'Euler en utilisant une technique d'appariement de courbes par morceaux. Bien qu'il ait donné des résultats visuellement attrayants, sa portée était limitée aux courbes en spirale d'Euler. Pour élargir l'applicabilité de notre approche de reconstruction, nous avons développé une nouvelle méthode capable de reconstruire plusieurs types de courbes, y compris les hélices circulaires et les spirales d'Euler.

L'idée principale de la nouvelle approche est motivée par l'observation suivante. Soit C_{3D} une courbe polygonale 3D et $O_{3D,i}$ le cercle osculateur au milieu d'un segment $e_{3D,i}$ de cette courbe. Le cercle osculateur est le cercle qui a la même tangente et la même courbure qu'au milieu de $e_{3D,i}$. Soit C_{2D} , $e_{2D,i}$ et $O_{2D,i}$ la projection orthogonale sur le plan (x,y) de C_{3D}, $e_{3D,i}$ et $O_{3D,i}$ respectivement. $O_{2D,i}$ est une ellipse dont la tangente et la courbure sont égales à celles de la courbe C_{2D} au milieu du segment $e_{2D,i}$. Maintenant, seule la courbe plane C_{2D} nous est fournie et notre objectif est de reconstruire la courbe C_{3D}. Étant donné un segment $e_{2D,i}$ de C_{2D} , nous calculons un ensemble d'ellipses dont la courbure et la tangente correspondent à celles au milieu de $e_{2D,i}$. Si l'on suppose que cet ensemble d'ellipses est suffisamment grand, l'une de ces ellipses doit être la projection d'un cercle osculateur très proche du cercle osculateur réel de la courbe C_{3D} à reconstruire. Nous répétons ce processus de calcul de l'ensemble des candidats

au cercle osculateur pour chaque segment de C_{3D} . Ensuite, le problème revient à sélectionner le meilleur cercle osculateur pour chaque segment de sorte que la variation de courbure soit minimisée le long de la courbe. Il s'agit d'un problème de graphe simple qui est résolu à l'aide de l'algorithme de Dijkstra. Ces cercles osculateurs sont ensuite utilisés pour déterminer la tangente de chaque segment et pour calculer les coordonnées 3D des points de C_{3D}.

L'apport de ce travail est double. Tout d'abord, nous proposons l'idée d'ajuster des ellipses à la courbe plane d'entrée ; ces ellipses permettent ensuite de déterminer les cercles osculateurs et tour à tour l'orientation de la tangente au milieu de chaque segment de la courbe 3D à reconstruire. La deuxième contribution est une méthode pour résoudre le problème d'optimisation qui vise à trouver les cercles osculateurs optimaux tels que la variation de la courbure le long de la courbe 3D reconstruite soit minimisée. L'approche habituelle nécessiterait de résoudre une grande optimisation non linéaire en utilisant la méthode de descente de gradient. Au lieu de cela, l'espace des solutions est discrétisé pour former un graphe, chaque nœud de ce graphe étant un candidat pour les cercles osculateurs au milieu de chaque segment de la courbe. Les bords du graphique correspondent à la relation de contiguïté entre les segments. La solution de ce problème d'optimisation est ensuite résolue en utilisant l'algorithme de Dijkstra.

Notre méthode a été implémentée en Python et a été testée avec une variété de courbes. Le temps de calcul varie de 10 à 60 secondes selon le nombre de points de la courbe d'entrée. Le résultat montre que notre algorithme est capable de générer une courbe 3D dont la variation de courbure est faible le long de la courbe.

Chapter 1

Introduction

Three-dimensional (3D) reconstruction involves creating a 3D model or representation of an object by utilizing information from various sources, such as 2D images, point cloud data, laser scans, or other data sources. This 3D representation has numerous applications in different fields. In computer graphics, for instance, 3D models created through reconstruction can be used to create video games, movies, and animations. Computer vision can use 3D models for object recognition, tracking, and pose estimation, while robotics can employ 3D models for navigation, obstacle avoidance, and object manipulation. Furthermore, engineering can utilize 3D models for product design, simulation, and prototyping.

The goal of 3D reconstruction is to recover the 3D structure of an object from a set of measurements or observations. Various techniques can be used for this purpose, including photogrammetry, structured light, stereo vision, LiDAR, optimization, and deep learning.

Photogrammetry (Do & Nguyen 2019) is a technique that uses multiple images of an object taken from different angles to reconstruct the 3D structure. The images are processed using computer vision algorithms (feature detection and matching, bundle adjustment, structure from motion, and dense reconstruction) to extract features, such as key points and edges, which are used to estimate the 3D structure.

Structured light (Farsangi et al. 2020) is a technique that uses a projector and a camera to capture 3D information about an object. The projector projects a

pattern of light onto the object, and the camera captures the reflected pattern. The 3D structure of the object can be computed by analyzing the deformation of the pattern.

Stereo vision (Ham et al. 2019, Tian et al. 2022) is a technique that uses two or more cameras to capture images of an object from different viewpoints. The images are processed using computer vision algorithms to extract correspondences between the images, which are used to estimate the 3D structure.

LiDAR (Tachella, Altmann, Ren, McCarthy, Buller, Mclaughlin & Tourneret 2019, Tachella, Altmann, Mellado, McCarthy, Tobin, Buller, Tourneret & McLaughlin 2019) is a technique that uses a laser scanner to measure the distance between the scanner and the object. The laser scanner emits a laser beam that reflects off the object and returns to the scanner. The distance between the scanner and the object can be computed from the time of flight of the laser beam.

Optimization (Zhang & Wonka 2022, Han et al. 2020) is a technique that uses mathematical optimization algorithms to estimate the 3D structure of an object from a set of measurements that represent the 2D object such as points coordination or pixel values. Optimization can be used in combination with other techniques, such as photogrammetry and stereo vision, to refine the estimated 3D structure.

Deep learning and machine learning approaches have become increasingly popular in the field of 3D reconstruction. These techniques can be used to extract information from images or other data sources to generate accurate 3D representations. One approach is to use convolutional neural networks (CNNs) to extract features from images or point clouds, which are then used to estimate the 3D structure of the object (Yang, Cui, Belongie & Hariharan 2018, Wu et al. 2016, Knyaz et al. 2018, Smith & Meger 2017, Yang, Rosa, Markham, Trigoni & Wen 2018, Li et al. 2018). This approach is often used for single-view 3D reconstruction, where a 3D model is generated from a single image. For example, a CNN can be trained to predict the depth map of an image, which can then be converted into a 3D model using photometric stereo techniques. Another approach is to use generative adversarial networks (GANs) to generate 3D models from a set of 2D images or point clouds (Zhang et al. 2021, Nozawa et al. 2022). GANs can be trained to generate realistic 3D models by learning the distribution of the

training data. This approach has been used for applications such as generating 3D models of furniture or other objects from 2D images. Other machine learning techniques, such as random forests and support vector machines, have also been used for 3D reconstruction. These techniques can be used for feature extraction and classification, as well as for estimating the 3D structure of an object. Overall, deep learning and machine learning approaches have shown promise in the field of 3D reconstruction, and are likely to play an increasingly important role in future developments in this area.

In this thesis, we focus on the 3D reconstruction of curves from single-view drawings. We begin with circular helix and Euler spiral curves and extend our approach to free-form curves. Notably, our proposed approach does not require any other input or user interaction than the planar curve of the drawing. Moreover, while some work has been done on 3D reconstruction of helices from planar polygonal curves, there is a lack of literature on using Euler spiral curves for this purpose. To begin our study, we first worked on the 3D reconstruction of circular helices from planar polygonal curves. Motivated by the growing popularity of ML techniques, we opted to use these techniques for our reconstruction and achieved challenging results compared with the synthetic helices.

Building on this, we then extended our investigation to a more general family of curves, specifically, the Euler spirals. We initially attempted to apply the same ML techniques used for the circular helices to the Euler spirals. However, this approach did not yield promising results, as the shape of the Euler spirals caused conflicts in ML's learning process. To address this, we shifted our methodology and dealt with the problem using a piecewise curve-matching approach. When reconstructing a 2D input curve in 3D, we first divide it into smaller pieces. For each piece, we search for the closest matches from a dataset we created that contains pieces of Euler spirals. Then, we connect the matched pieces and apply a smoothing algorithm to generate the reconstructed piecewise 3D Euler spiral curve. Although this method achieved a good 3D reconstruction, its scope was limited to Euler spiral curves. To broaden the applicability of our reconstruction approach, we developed a novel method capable of reconstructing multiple curve types, including helices and spirals. This method minimizes the curvature variance of the curve by fitting ellipses to the input curve, allowing us to determine the osculating circles and

tangents at each curve point for 3D reconstruction.

This thesis manuscript is composed as follows. In Chapter 2 we review the previous work on the 3D reconstruction domain. Chapter 3 describes the ML techniques that we used for the 3D reconstruction of circular helices from planar polygonal curves. In Chapter 4, we introduce the two approaches that deal with a more complex family of curves. Firstly, the piecewise matching approach that we used to reconstruct the 3D Euler spirals from planar polygonal curves. Then, the novel method, which is capable of reconstructing multiple curve types, including helices and spirals. Finally, in Chapter 5, we conclude our work and discuss potential avenues for future research.

Chapter 2

The state of the art for 3D reconstruction

The field of 3D reconstruction has attracted significant attention from researchers in recent years. In this Chapter, we provide an overview of previous work in various domains of 3D reconstruction. We begin with a global perspective, highlighting research in image-based reconstruction and sketch-based modeling. Subsequently, we narrowed our focus to the specific topic of interest, exploring the state of the art for 3D reconstruction of circular helices, Euler spiral curves. Finally, we review the state of the art for 3D modeling of general curves.

2.1 Image-based 3D reconstruction

Image-based 3D reconstruction is a challenging problem that has gathered significant interest in computer vision, computer graphics, and ML communities. The earliest techniques dealt with the problem from a geometrical perspective. They focused on mathematically formalizing the projection process from 3D to 2D in order to find the solution. Efficient techniques such as stereo-based (Hartley & Zisserman 2003) reconstruction necessitate multiple images captured by calibrated cameras, which can be impractical in some situations. With the availability of large datasets and rapid advancements in ML algorithms, researchers have increasingly turned to convolutional neural networks to address these challenges, treating 3D

reconstruction as a recognition problem. Two applications of these techniques have emerged, with some works focusing on generic shapes (Yang, Cui, Belongie & Hariharan 2018, Wu et al. 2016, Knyaz et al. 2018, Smith & Meger 2017, Yang, Rosa, Markham, Trigoni & Wen 2018, Li et al. 2018) and others on specific shapes like 3D Human Body Reconstruction (Bhatnagar et al. 2019, Dibra et al. 2016, 2017, Alldieck et al. 2019) or 3D Face Reconstruction (Feng et al. 2018, Lin et al. 2020, Gecer et al. 2021, Wood et al. 2022). To improve the quality of 3D reconstruction, some approaches (Sun et al. 2018, Wu et al. 2018, Yang, Rosa, Markham, Trigoni & Wen 2018) decompose the problem into sequential steps that estimate normal maps or depth maps, followed by traditional techniques such as backpropagation and filtering to recover the final 3D object. Another approach (Choy et al. 2016, Xie et al. 2019) is to process each image independently and then merge the reconstructions using registration techniques to exploit spatio-temporal correlations, which is only used in 3D reconstruction from multiple images.

2.2 Sketch-based modeling

The idea behind the sketch-based modeling is to start from a drawn shape in the (x,y) plane which is composed of lines, and then try to reconstruct the 3D shape whose projection onto the (x,y) plane matches the input sketch. Some of the first reconstruction methods were done by solving an optimization whose unknown variables are the third coordinates of the input sketch references (Brown & Wang 1996, Shoji et al. 2001) but the main drawback was that these methods only deal with rectilinear shapes. With the rise of ML algorithms, several researchers have worked on solving the sketch-based modeling problem using deep learning (Nozawa et al. 2020, Yang et al. 2021, Wang et al. 2020, Shen et al. 2021, Yang et al. 2022) but these algorithms are tailored to specific shapes, such as faces, cars, and chairs, making them limited in their applicability. We refer the reader to the state-of-the-art paper (Xu et al. 2022) for more details about sketch-based modeling using deep learning.

2.3 3D reconstruction of circular helices

There exist some works in the literature concerned with the 3D reconstruction of circular helices from 2D curves.

In (Cherin et al. 2014), a method is presented for computing and fitting projected helix curves to 2D polygonal curves. The approach involves identifying the optimal parameters of a projected helix that best matches a portion of the input polygonal curve. This fitting process is performed iteratively, and multiple projected helix curves are employed to approximate the entire input curve. The key steps include computing helix coefficients based on local curvature, aligning coordinate frames, estimating fitting errors, and determining the portion of the curve that matches well with the projected helix. This piecewise fitting strategy allows for a more accurate approximation of complex input curves. The result is a set of projected helices, each with junction vertices and consistent orientations, providing a piecewise representation of the original polygonal curve. Followed by that they they used optimization techniques to fine-tune their reconstructed curves. However, it's worth noting that despite achieving impressive visual reconstructions, this method has certain limitations concerning curve continuity. Specifically, the piecewise helix curves lack both G^0 and G^1 continuity, which may lead to subtle discontinuities that are usually inconspicuous but, in some cases, more noticeable tangent discontinuities can occur at junction vertices.

Similarly, in (Piuze et al. 2011), the authors introduce a method centered on the use of generalized helicoids for modeling and simulating hair patterns. These generalized helicoids characterize hair strands and their local volumetric neighborhoods using four key parameters that control curvature and elevation angles. One of the significant contributions of this approach is the synthesis of diverse hair types, including waviness effects, by sampling from a generalized helicoid-based representation. Additionally, their method interpolates between guide hair strands, each parametrized by a generalized helicoid, ensuring a smooth variation in curvature between them. Furthermore, the authors describe a process for fitting these generalized helicoids to hair strand data, even when dealing with sparse samples. They achieve this by optimizing a parameter vector to minimize the difference between the trace generated by the helicoid and the target hair strand,

using a measure based on Fréchet distance. This fitting process enables hairstyle reconstruction, even from limited data, making it a valuable tool for applications involving real hair data.

In (Cordier et al. 2016), they introduce a method for fitting orthogonally-projected helix segments to 2D polygonal curves. Their approach begins by recognizing the underdetermined nature of this problem and subsequently employs iterative transformations to resolve it effectively. Firstly, their method aligns sampled helix points with the polygonal curve, calculating an optimal transformation matrix that encapsulates scale, shear, and rotation components. This matrix must retain the essential helix properties. Subsequently, they estimate the helix's parameters, namely radius and pitch, in a manner that preserves its core characteristics. Afterward, they focus on estimating the optimal rotation matrix, by minimizing the root mean squared deviation between helix and curve points. Following that, they employed an adaptive sampling technique to ensure a consistent one-to-one correspondence between helix and curve points. Finally, through an iterative process, the adaptive sampling and helix fitting stages refine one another, aiming to minimize the fitting error. In summary, this method iteratively aligns, estimates, and optimizes to obtain a helix segment that meticulously conforms to the polygonal curve in the (x, y) plane.

These previous methods are the most similar to ours in Chapter 3. The main difference is that they used optimization techniques to perform their reconstructions. In our work, we use ML algorithms for this purpose.

2.4 3D reconstruction of Euler spirals

The 2D Euler spiral, also known as the Clothoid or Cornu spiral, is a curve whose curvature evolves linearly with arc length. It was independently discovered by several researchers including Bernoulli, Euler, and Talbot (Levien 2008). Many researchers have used 2D Euler spirals in computer-aided design.

In (Walton & Meek 2005), they introduce a control polyline method for guiding complex curves composed of clothoids, straight lines, and circular arcs. Control polylines, commonly used in design applications, define key curve points, facilitating interactive adjustments. Their method employs blending curves using clothoid

pairs, with options for symmetric or unsymmetric blending.

The work most closely related to ours in Chapter 4 is that of (Baran et al. 2010, McCrae & Singh 2009, 2011). In (Baran et al. 2010), they implement an algorithm for approximating sketched strokes with visually pleasing clothoid splines, known for their piecewise linear curvature profiles. The algorithm segments strokes into curve primitives like lines, arcs, and clothoids, treating this as a shortest path problem on a weighted graph. The graph's nodes represent a comprehensive set of curve primitives, while edges signify continuity transitions. The shortest path yields an ideal segmentation. Post-segmentation, non-linear constrained optimization fits primitives to the curve, ensuring a high-quality result close to the user's sketch. In (McCrae & Singh 2009), they present a method for sketching 2D curves, by minimizing curvature variation through the use of piecewise clothoids. Their approach works by fitting piecewise linear curves to the curvature of the segmented input curve. Each linear piece corresponds to a line, circular arc, or clothoid curve segment. In the subsequent phase, they determine a singular 2D rigid transformation aligning this composite curve with the sketched stroke, minimizing the error between them. Moreover, by locally blending adjacent clothoid segments, they achieve G^3 continuity where curvature predominantly varies linearly with arc length. Similarly, in (McCrae & Singh 2011), they introduce an approach to tidy up sketched strokes using piecewise French curves, seamlessly blending traditional bimanual curve modeling techniques with French curves to automatically refine sketched strokes. The algorithm begins by computing curvature profiles for the input polyline and one or more French curves. It then employs a dynamic programming algorithm to identify sections of the French curve that optimally match sections of the input curve, with a focus on minimizing the number of sections used. Subsequently, the input curve is reconstructed using sections of the French curve represented piecewise in a clothoid form. Finally, disjoint sections of the reconstructed curve are skillfully interpolated to produce a connected curve, ensuring G2 continuity throughout. Despite of the good results of these previous works, their method is limited to 2D plane and cannot be used for 3D reconstruction, as opposed to our method which produces a piecewise 3D Euler spiral that fits a planar polygonal curve.

On the other side, the 3D Euler spiral is the curve whose curvature and torsion

evolve linearly with arc length. Several works have been done on the generation of 3D Euler spirals. In (Guiqing et al. 2001), the authors aimed to generate a 3D Euler spiral, starting by refining a polygon, such that the polygon satisfies the linearity evolution of the curvature along the arc by introducing the concept of discrete Frenet frames and binormal planes but they ignored the torsion. In (Frego 2022), the authors defined the closed form parametrization of 3D Euler spirals by modeling the problem as a linear time-variant system and studied its stability with Lyapunov techniques (Lyapunov 1992). Their most significant achievement was defining the closed form of the 3D Euler spiral in terms of the standard Fresnel integrals that satisfy the property of both curvature and torsion evolving linearly with arc length. They also present numerical methods of order four to represent the clothoids based on techniques such as Lie algebra, Magnus Expansions, and Commutator-Free Expansions. In (Gur Harary 2012), the authors extend the Euler spirals from 2D to 3D by solving an optimization problem and proving many of their properties, including their invariance to similarity transformation (translation, rotation, and scaling), symmetry, extensibility, smoothness, and roundness. Furthermore, in this article, they used the 3D Euler spiral for the archaeological reconstruction such as the completion of the shape of some broken ancient sculptures and objects. However, the proposed algorithm in this paper only works in 3D space, thus it cannot be used to create 3D Euler spirals from planar polygonal curves.

2.5 3D modeling of curves

Modeling a 3D curve can be done in two different ways, either using a 3D drawing interface or a 2D drawing interface.

2.5.1 Curve modeling using 3D drawing interfaces

A 3D drawing interface enables the artist to create the curve directly in 3D using a device that captures the movements of the hand in the 3D space. In (Ye et al. 2021, Kwan 2019), they proposed the use of mobile phone as a 3D pen. Notably, they identify seven input points and six phone grip styles preferred by users, revealing 21

unique combinations. Furthermore, the top three pairs were selected for in-depth analysis, wherein a quantitative experiment was performed to assess four distinct types of motion tracking errors during controlled in-situ 3D drawing. In (Yu, Arora, Stanko, Bærentzen & Singh 2021), they proposed a HTC Vive controller to create the 3D curves. After the user has completed their initial freehand sketch. Their methods starts by identifying candidate hard intersection constraints by assessing the distances between the sketched stroke's points and existing 3D curves or elements within the virtual space. Subsequently, a subset of discrete constraints is determined through a greedy linear search to optimize the model's geometric consistency. Following this, a least squares minimization approach is formulated to establish continuous constraints that ensure smooth and coherent surfaces within the 3D model. In (Li 2021), the authors introduce the use of VR headsets to create the 3D curves. Specifically, when wearing VR headsets, users can engage in 3D drawing within a virtual environment by employing VR controllers to directly create geometric elements such as polylines through intuitive clicks and mid-air dragging gestures. Additionaly, they proposed an optimization-based approach that automatically refines the initial 3D input provided by the user, transforming it into polished and aesthetically pleasing 3D drawings.

Other researchers used different types of virtual reality controller (Arora & Singh 2021, Yu, DiVerdi, Sharma & I 2021), or a hand tracking sensor (Kim 2016) or a 3D pen (Yue et al. 2017). The main limitation of this class of approaches is the use of a tracking device and/or a head-mounted display. This makes the modeling process not easy to setup and not as convenient as making a drawing, which only requires a pen and paper.

2.5.2 Curve modeling using 2D drawing interfaces

The second category is the use of a 2D drawing interface such as a tablet or a mouse. Sketching using a tablet is preferred by many designers since sketching is natural for them. In addition, these devices are usually cheaper than virtual reality devices. However, the main difficulty is to generate a 3D curve from the 2D curve drawn by the user; the third dimension is missing and the goal is to compute this missing information in a way to generate a 3D curve that looks natural and that

matches the user's expectation. Researchers have proposed different approaches to recover this missing information.

One of the most common approaches is to process altogether the curves of the drawing. It is actually much easier to reconstruct in 3D a set of curves representing an object rather than a single curve. When all the curves are taken together, additional criteria such as parallelism, orthogonality, and symmetry between the curves can be used to facilitate the 3D reconstruction (Iarussi et al. 2015, Gryaditskaya et al. 2020, Xu et al. 2014). Specifically, in (Iarussi et al. 2015), the authors present a novel technique for extrapolating curvature lines within rough concept sketches, effectively recovering the intended 3D curvature field and surface normals at each pixel of the sketch. Their method introduces the concept of regularized curvature lines, encompassing various types of curvature lines and their extensions over flat or umbilical regions. Furthermore, they establish an orthogonal cross field that assigns two regularized curvature lines to every 3D surface point. The algorithm skillfully estimates the projection of this cross field onto the sketch, even when faced with non-orthogonality due to foreshortening. This estimation process relies on scattered interpolation of the sketch's strokes, which enhances the method's robustness in handling the characteristic sketchy lines commonly found in design sketches. In (Gryaditskaya et al. 2020), the authors propose a method for 3D sketch reconstruction from an initial input sketch. Their approach begins by identifying all straight strokes within the sketch that are parallel to one of the major axes. Subsequently, they construct the 3D scaffold progressively, handling one stroke at a time and determining its 3D intersections with already reconstructed strokes. Once the scaffold is established, it serves as essential geometric context for reconstructing curved strokes. For each curved stroke, their algorithm considers multiple potential reconstructions and selects the one that best adheres to a predefined set of geometric regularities. In instances where no single reconstruction clearly stands out as the best choice, the algorithm postpones the decision until additional context emerges to resolve any ambiguity. In (Xu et al. 2014), the authors begin with a 2D vector sketch as their input in order to create a 3D Curve Networks. They introduce a set of localized 3D regularity properties, which their algorithm selectively identifies and applies to elevate the curves from the 2D page into a 3D representation. Initially, the algorithm produces a baseline 3D

result, which may contain inaccuracies. However, these inaccuracies are progressively rectified through the selective regularization process. It's important to note that the angles between potentially parallel and locally symmetric curves start in a random distribution but ultimately converge to a globally consistent state where these properties are either disregarded or precisely enforced. These methods have been proven to work very well. However, they fail when they are given a single curve; such a method cannot be used to reconstruct the trajectory of a dynamic object from a drawing.

Another common strategy to make the 3D reconstruction problem tractable is to make some assumptions about the curves to reconstruct. Some works have been done for the reconstruction of mirror-symmetric curves (Cordier et al. 2013, Hähnlein et al. 2022). For instance, in (Cordier et al. 2013), their approach initiates by calculating all potential orientations of the symmetry lines, where these lines connect pairs of vertices that exhibit mirror symmetry, implying that these vertices remain unchanged when reflected. Subsequently, they determine the turning vertices for the polygonal curves associated with each feasible orientation of the symmetry line. The subsequent phase involves establishing the symmetry relationship for each orientation candidate by devising a method that utilizes the turn vertices. The final part of their algorithm incorporates two strategies to resolve any ambiguity. Firstly, they leverage the connectivity among the curves to identify and discard symmetry relationships that are invalid. If any ambiguity persists, they define the symmetry relationship in a manner that maximizes the compactness of the reconstructed model. Other works focuses on the reconstruction of curves around existing objects (Krs et al. 2017). In this article, the user draws strokes in front of an object. Their approach begins by identifying potential 3D vertices in 3D space for each 2D point based on their distances from the geometry, termed as vertex height. Next, they construct a graph of vertex segments to optimize 3D curve reconstruction. These segments, representing points on the curve, serve as nodes in the segment graph, while the edges denote stroke portions not intersecting with the geometry. Finally, they determine the best path through the segment graph, forming a smooth cubic spline, with nodes and edges scored using a curvature criterion.

Finally, some works have been done for the curve reconstruction using multiple

images or multiple viewpoints. The first strategy is to enable the artist to construct the curve interactively by drawing an initial version of the curve. The user then can change the viewpoint so that she/he can make some modifications in case the initial curve is not satisfactory (Bae et al. 2008). Other researchers have proposed a method to reconstruct a curve given its drawing and its shadow on the floor plane (Cohen et al. 1999). Some work has also been done for the reconstruction of a curve given its two images (Fei Mai 2010). The strategy of using multiple viewpoints or images has been proven to work very well. However, they require the user to provide multiple images or some interaction during the reconstruction process.

2.6 Conclusion

In this chapter, we conducted a comprehensive review of prior research in the field of 3D reconstruction. Our exploration spanned from a global perspective, which is the reconstruction of entire objects from various visual sources, to a more focused area, which aligns with our specific research interest—the reconstruction of 3D curves. We reviewed previous works on the 3D reconstruction of circular helices, followed by an examination of research concerning 2D and 3D Euler spirals, including both their theoretical definitions and practical applications. Additionally, we explored earlier work related to the general 3D modeling of curves, covering techniques for sketching, and methodologies for achieving 3D curve reconstruction.

In conclusion, earlier studies have offered a variety of approaches for reconstructing 3D curves from 2D polygonal representations. However, it is noteworthy that, as of our review, no research has explored the application of machine learning ML techniques to the task of reconstructing 3D circular helices. Similarly, there appears to be a gap in prior work addressing the 3D reconstruction of 3D Euler spiral curves.

Chapter 3

3D reconstruction of circular helices

Helices are a type of 3D curve that appears in various natural and artificial structures. They are characterized by constant curvature along their arc-length. Their geometric shapes can be observed in natural structures, especially biological ones, and have also been utilized in a wide range of applications in human technology. At the micro-scale, helical structures can be observed in biology and nanotechnology, such as in sperm (Saggiorato et al. 2017), DNA (Dickerson 1983), and nanosprings (McIlroy et al. 2001). Meanwhile, at the macro-scale, they can be found in plants (Goriely & Tabor 1998), seashells (Forterre & Dumais 2011), and various engineering components such as springs (Ke et al. 2020) and synthetic fiber ropes (He et al. 2020).

Although many researchers work on the 3D reconstruction of helices from planar polygonal curves, most of their approaches are based on optimization algorithms. However, in the last decade, ML/deep learning algorithms have proven their efficiency in the 3D reconstruction domain. To the best of our knowledge, no previous work has employed these methods in the context of helix reconstruction. This chapter presents a novel approach that addresses this gap in the literature by utilizing ML/deep learning algorithms to reconstruct 3D helices from planar polygonal curves. Specifically, we trained different ML models, including random forest regressor (RFR)(Breiman 2001), gradient boosting regressor (GBR)

(Friedman 2001), gaussian process regressor (GPR)(Rasmussen et al. 2006), support vector regressor (SVR) (Smola & Schölkopf 2004), artificial neural network (ANN) (McClelland et al. 1987), and encoder-decoder convolutional neural network (ED-CNN) (LeCun et al. 1998) using a dataset composed of pieces of 3D helices. We reconstruct the 3D circular helices in a piecewise manner and compare the results of each ML algorithm in the experimental section. Finally, we analyzed the results and drew conclusions regarding the effectiveness of using ML algorithms for 3D reconstruction of circular helices.

3.1 Motivation

Circular helices are characterized by constant curvature along their arc-length, they appear in various natural and artificial structures. As described in the previous paragraph, their geometric shapes can be observed in natural structures and utilized in a wide range of applications in human technology. A formal representation of a 3D helical curve is very useful because it helps to fully describe the object, including its local pitch and radius. This representation can be particularly useful in a variety of real-life applications, as highlighted in the following examples:

- In quality control on manufacturing lines, 3D reconstruction of helical structures can be helpful in ensuring that products are manufactured to the desired specifications (Aloisi et al. 2016).
- In high-throughput experimentation in plant sciences: 3D reconstruction of helical structures can be used to analyze plant growth and development (Kawagoe & Murai 1996).
- In modeling hair in cosmetics: 3D reconstruction of helical structures can be used to create realistic and accurate models of hair fibers, which might be used to design and test new hair care products (Bertails et al. 2006).
- In the geometric modeling and understanding of microscopic structures in medicine, biology, and nanotechnology (Wittung et al. 1994).

Overall, the 3D reconstruction of circular helices has many practical applications across various fields, and accurate reconstruction of their 3D structure can

provide valuable insights and improve the design and performance of products and processes.

3.2 Problem definition

Our objective is to develop a ML model that reconstructs 3D circular helices from planar polygonal curves. Specifically, the input curve should be in the form of the coordination of the circular helix in the (x,y) plane. Given the potential variability in the length of input curves and the repetitive nature of circular helices along their arc length, it is practical to divide the input curve into smaller segments. Therefore, the reconstruction is done in a piecewise manner, where we perform curve segmentation to divide the input curve into smaller segments. Each segment is then reconstructed individually. Once all segments are reconstructed, they are assembled together to form the approximate 3D circular helix whose projection onto the 2D plane best fits the input polygonal curve. To enhance the overall smoothness of the curve, particularly at the connecting points of its segments, a post-processing smoothing algorithm is applied. This smoothing algorithm helps refine the reconstructed curve and improve its aesthetic quality.

To train the ML models, we create a dataset consisting of segments of circular helices, each comprising 100 points in the (x, y, z) plane. We apply uniform sampling to ensure that each segment is consistently represented. Followed by a uniform scaling to remove the scale factor from the dataset, allowing us to handle input curves of any scale. In the training process, we use the (x, y) coordinates of the circular helix segments as input, and their respective z-coordinates as output. Noting that we experiment with different ML models such as RFR, GBR, GPR, SVR, ANN, and ED-CNN. For each model, we perform a hyperparameter tuning to find its optimal architecture and achieve the best possible performance.

Once the ML model is trained, it can be utilized for the reconstruction of a given input polygonal curve. One condition of our reconstruction approach is that the input curve can not be any random curve such as straight lines or curves with sharp corners. We specifically target the reconstruction of curves which are the orthogonal projection of true circular helices or hand-drawn circular helix curves.

It is important to note that fitting an orthogonal-projected 3D Euler spiral

to a polygonal curve is widely recognized as an underdetermined problem, which implies that a unique solution may not always exist.

3.3 Dataset setup

To train the ML models for generating the z-coordinates of the circular helix segments from the planar polygonal curve, we create a diverse dataset. The dataset comprises 650,536 segments of 3D circular helices. This dataset size was chosen to cover a broad range of lengths, scales, and orientations. Each segment is defined as a polygonal curve within the (x, y, z) plane and consists of precisely 100 data points. The rationale behind choosing this specific point count will be elaborated upon in the forthcoming subsection on Dataset sampling. In this section, we describe the steps taken to set up the dataset. First, we specify the equation used for dataset creation. Second, we determine the lengths of dataset segments. Third, we define the orientation of the segments in 3D space. Fourth, we apply a scaling mechanism to remove the scale factor from the dataset and make it suitable for input curves of any scale. Finally, we apply a sampling mechanism to consistently represent each segment in the dataset. The purpose of this diversity of the dataset is to ensure that the ML models are trained on a comprehensive range of input circular helix shapes, lengths, and orientations, enabling them to accurately generate a 3D circular helix from any planar polygonal curve.

3.3.1 Equation

The equation used for creating the dataset segments is based on the parametric equations for a circular helix in 3D space, expressed parametrically in Cartesian coordinates. Specifically, we used the equations:

$$\begin{aligned} x(t) &= r\cos(t), \\ y(t) &= r\sin(t), \quad t \in \mathbb{R} \\ z(t) &= pt. \end{aligned} \tag{3.1}$$

Here the parameter t is proportional to the arc length of the circular helix and is used to express the parametric equations of the dataset segments. The

radius of the circular helix is denoted by r, while the pitch is denoted by p. The pitch represents the distance that the circular helix advances along its axis after completing one full turn around the center, and is given by $2\pi p$.

3.3.2 Pieces length

The input of our algorithm can be a planar polygonal curve of any length. However, it is not practical to train an ML model to generate the z-coordinates of a long circular helix composed of many loops. This is because the input size required to represent it accurately would be too high. Additionally, the dataset needed for training the ML models in such a case will have an enormous size, as it would need to contain circular helix shapes that vary from short (with one loop) to very long (with a high number of loops). To address this issue, we reconstruct the circular helix in a piecewise manner by creating a dataset composed of circular helix segments with a length varying between half a loop and a loop and a half. This is possible due to the property of the circular helix, which is a curve that repeats itself.

Practically, to achieve this, when we create the dataset we uniformly distribute the ending point t_{end} such that we make the difference between the variable t of the Equation 3.1 at the starting point t_{start} and the ending point t_{end} of the circular helix segment to be within the range of $\in [\pi, 3\pi]$. As illustrated in Figure 3.1, t_{end} must fall within the range of $\in [t_a, t_b]$ for any given dataset segment with a starting point t_{start}.

3.3.3 Pieces orientation in 3D space

Our method takes as input a planar polygonal curve of a circular helix, which could be in any orientation. In order to train our models can accurately handle circular helices from various orientations in 3D, we rotate each segment of the dataset along the x, y, and z axes. As the orientation of a circular helix changes in 3D, its orthogonal projection changes in 2D. By representing all possible projections of the 3D Euler spiral segments, we can ensure that our models are able to handle circular helices from any orientation. Let M_i, $R_{x,i}$, $R_{y,i}$, and $R_{z,i}$ be respectively

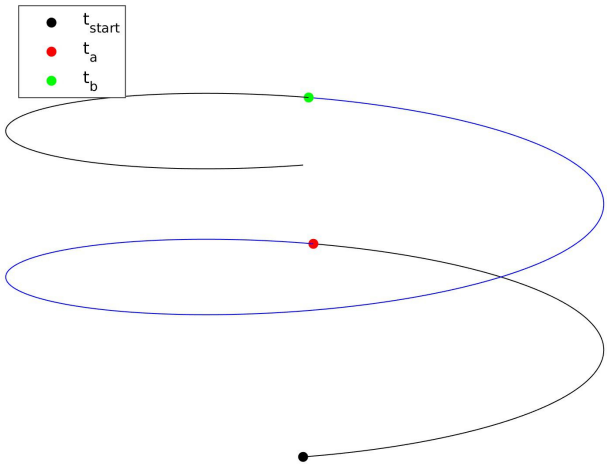

Figure 3.1: The ending point t_{end} of the any dataset segment with a starting point t_{start} must fall within the range of $[t_a, s_b]$.

the matrix that represents the point's coordinates of the segment i of the dataset, the rotation matrices along the x, y, and z-axis.

$$M_i = \begin{pmatrix} x_{i,1} & y_{i,1} & z_{i,1} \\ x_{i,2} & y_{i,2} & z_{i,2} \\ \vdots & \vdots & \vdots \\ x_{i,100} & y_{i,100} & z_{i,100} \end{pmatrix}, R_{x,i}(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix}$$

$$R_{y,i}(\beta) = \begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{pmatrix}$$

$$R_{z,i}(\lambda) = \begin{pmatrix} \cos(\lambda) & -\sin(\lambda) & 0 \\ \sin(\lambda) & \cos(\lambda) & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

where α, β, and λ are respectively the rotation angle along x, y, and z axis. During the dataset creation, we uniformly distributed these rotation angles in the range $[0, 2\pi[$ in order to cover all possible rotation angles. The rotated 3D circular helix

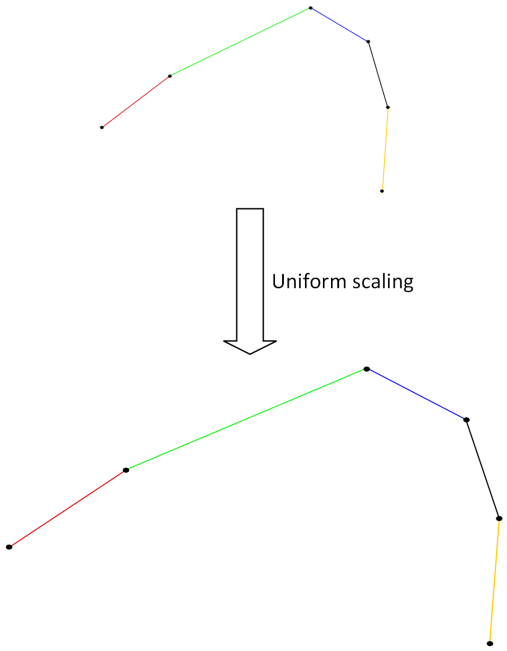

Figure 3.2: Example of the uniform scaling applied on some circular helix segments (colored segments). Note that we are presenting the orthogonal projection of the circular helix segments.

segment c_i is defined as:

$$c_i = M_i R_{x,i}(\alpha) R_{y,i}(\beta) R_{z,i}(\lambda).$$

3.3.4 Dataset scaling

The input of our method is a planar polygonal curve which is the orthogonal projection of a circular helix. This circular helix could come with various values of radius and pitch. Hence, it is practically impossible to train our models with circular helices of all radius and pitch values. Instead of that, we train our models with circular helices that have a radius and pitch uniformly distributed in the interval $[1, 5]$ but also have the same length in 2D. This interval was selected to cover a variety of circular helix configurations, including those with a small radius and large pitch (e.g., radius = 1, pitch = 5), large radius and small pitch (e.g., radius = 5, pitch = 1), and cases where the radius and pitch are equal (e.g., radius

= 2, pitch = 2). Subsequently, we scale all these segments in a way that they will all have the same length in 2D while preserving their original aspect ratio. This removes the scaling factor from our dataset. The uniform scaling mechanism extends the applicability of our approach to a wider range of radius and pitch values beyond what was explicitly included in the training data. An example of this uniform scaling is illustrated in Figure 3.2. Therefore, when we split the input polygonal curve into segments, we first scale each segment to the same length as dataset segments, and then we give it to the model in order to predict its z-coordinates. After that, we rescale each one of the 3D reconstructed segments to its original scale. Finally, we connect the reconstructed segments to obtain the final approximate reconstruction of the circular helix.

Concerning the scaling process, when scaling a dataset circular helix segment, we aim to make the **average** 2D distance between every two consecutive points equal to a desired value α. Let's consider the input segment as $a = \{a_1, a_2, ..., a_i, ..., a_n\}$, where a_i is the coordinates of the point of index i in the curve and n is the number of points of that curve. The coordination of its first two points given by $a_1 = (x_1, y_1, z_1)$ and $a_2 = (x_2, y_2, z_2)$. We first calculate its original average 2D distance between every two consecutive points, denoted by β. Let's consider the scaled segment as $a_s = \{a_{s,1}, a_{s,2}, ..., a_{s,i}, ..., a_{s,n}\}$, to get the coordination of this segment, we calculate the coordination of it first point as $a_{s,1} = (x_{s,1}, y_{s,1}, z_{s,1}) = (x_1, y_1, z_1)$. To determine the coordination of the second point of the scaled segment, we calculate the ratio to which every two consecutive points of the scaled segment will be scaled, which is $r = \alpha/\beta$. The coordination of the second point of the scaled segment is denoted by $a_{s,2} = (x_{s2}, y_{s2}, z_{s2})$, and can be calculated as follows:

$$x_{s2} = x_1 + r \times (x_2 - x_1),$$

$$y_{s2} = y_1 + r \times (y_2 - y_1).$$

To calculate z_{s2} while preserving the original aspect ratio of a, we first compute the Euclidean distance between p_1 and p_2 in the (x,y)-plane, given by $d = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}$. Then, we can compute the z-coordinate of a_{s2} as:

$$z_{s2} = z_1 + r \times d(z_2 - z_1).$$

We repeat the same procedure to calculate the remaining points coordination of a_s. By the end of the scaling, the **average** 2D distance between every two consecutive points of the scaled segment a_s will be equal to α. A real example of the result of applying the uniform scaling on a circular helix segment is presented in Figure 3.4 (1, 2).

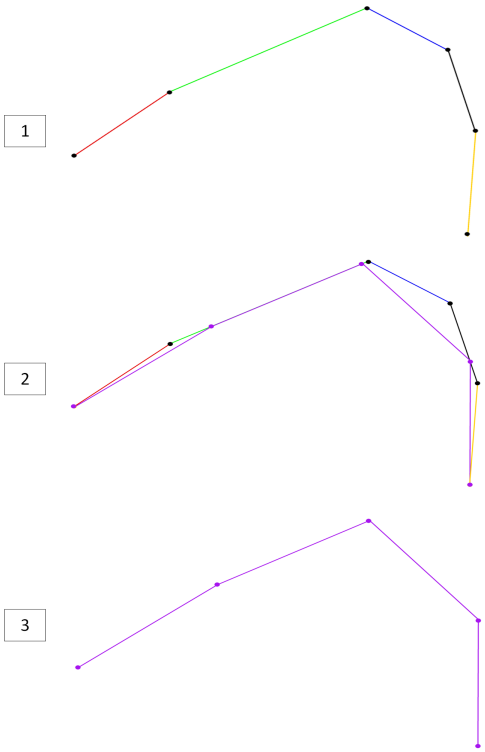

Figure 3.3: (1) Segments of various lengths. (2) The results of applying the uniform sampling on the segments (Purple), and the original segments (Colored). (3) The uniform sampled segments (All segments with the same length). Note that we are presenting the orthogonal projection of the segments.

3.3.5 Dataset sampling

After applying the scaling procedure, we ensure that all dataset segments have the same length in 2D. However, the distance between each pair of consecutive points

in a given segment may not be the same. This can introduce a noisy factor that can affect the learning of the ML models, as the model may focus on capturing variations in distances rather than the underlying patterns in the data. To address this issue, we performed uniform sampling to ensure that the **exact** distance between each pair of consecutive points is the same in every segment. Specifically, given the scaled segment, we sampled it uniformly to obtain the sampled segment such that the **exact** distance between each pair of consecutive points is equal to α. This ensures that the distance between each pair of consecutive points is equal, reducing the impact of variations in distances on the ML model's learning process. Additionally, when later reconstructing hand-drawn curves, we must employ the same sampling procedure as used with the dataset segments.

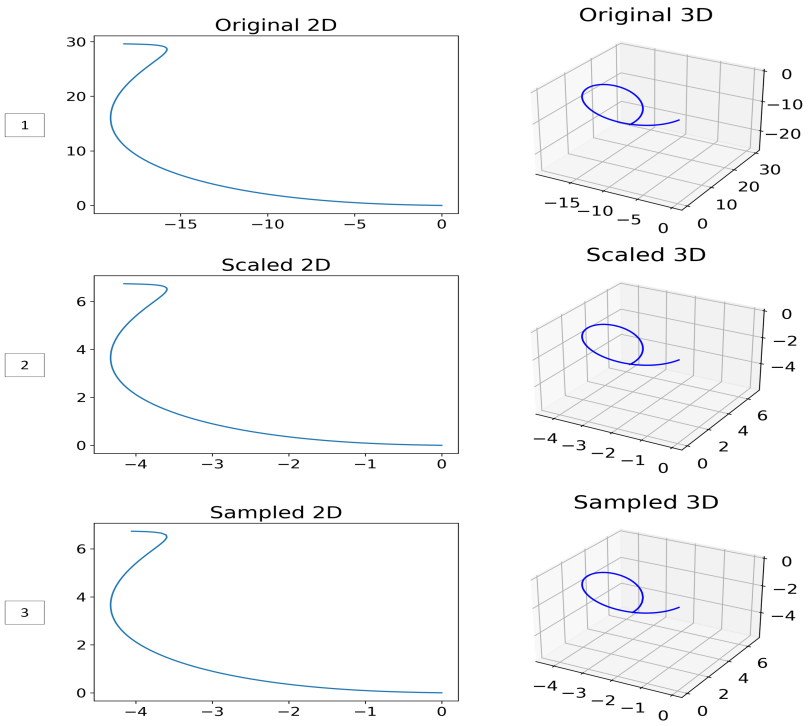

Figure 3.4: (1) The original 3D circular helix with its orthogonal projection. (2) The resulting circular helix after applying the uniform scaling algorithm on the circular helix of (1). (3) The resulting circular helix after applying the uniform sampling algorithm on the circular helix of (2).

Regarding the number of sampled points that our dataset segment is composed of, we opt to make it 100. If the sampling rate is too low, the sampled points may not be sufficient to fully capture the characteristics of the original curve and valuable information may be lost. On the other hand, if the sampling rate is too high, the sampled points may contain redundant information, which can lead to computational inefficiencies and overfitting in ML models. Therefore, it is important to carefully select the sampling rate to balance between capturing enough information and avoiding redundancy. An example of applying the sampling algorithm is shown in Figure 3.3, noting that this is just an example figure to explain the sampling process and does not exactly reflect the sampling of the dataset. A real example of how our scaling and sampling algorithm affect the circular helix segments is presented in Figure 3.4. We can also see in Figure 3.5 that after applying the uniform sampling the distance between each two consecutive points of the orthogonal projection of the circular helix segment became equal.

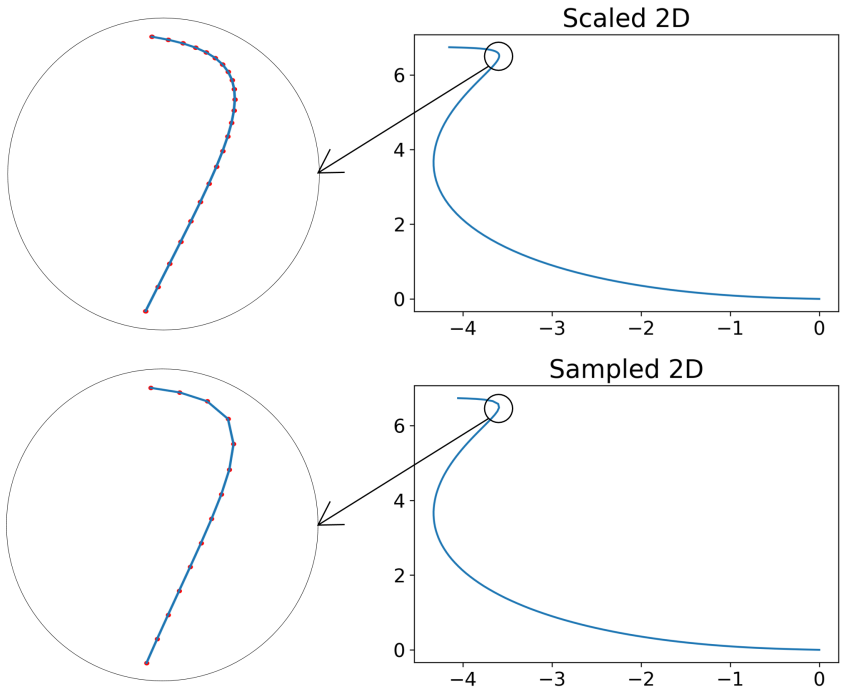

Figure 3.5: On top, the distance between consecutive points of the orthogonal projection of the scaled circular helix segment is not the same. On the bottom, the distance between consecutive points of the orthogonal projection of the sampled circular helix segment is the same.

3.3.6 Dataset summary

We create a dataset composed of 650,536 segments of the circular helix. Covering various orientations of the segment with a circulation length between a half loop and a loop and half. The radius and pitch (as defined in Equation 3.1) used in creating the dataset have values in the interval [1, 5] (uniformly distributed in this interval). This is because we applied a uniform scaling technique, which removes the scaling factor and thereby covers an additional variety of radius and pitch values. Additionally, we applied a uniform sampling on the circular helix segments making the distance between each two consecutive points of the orthogonal projection of the circular helix segment equal. This standardizes the dataset and simplifies the learning process of the ML models. Figure 3.6 shows some segments

of the dataset, where all segments have a uniform scaling and sampling applied.

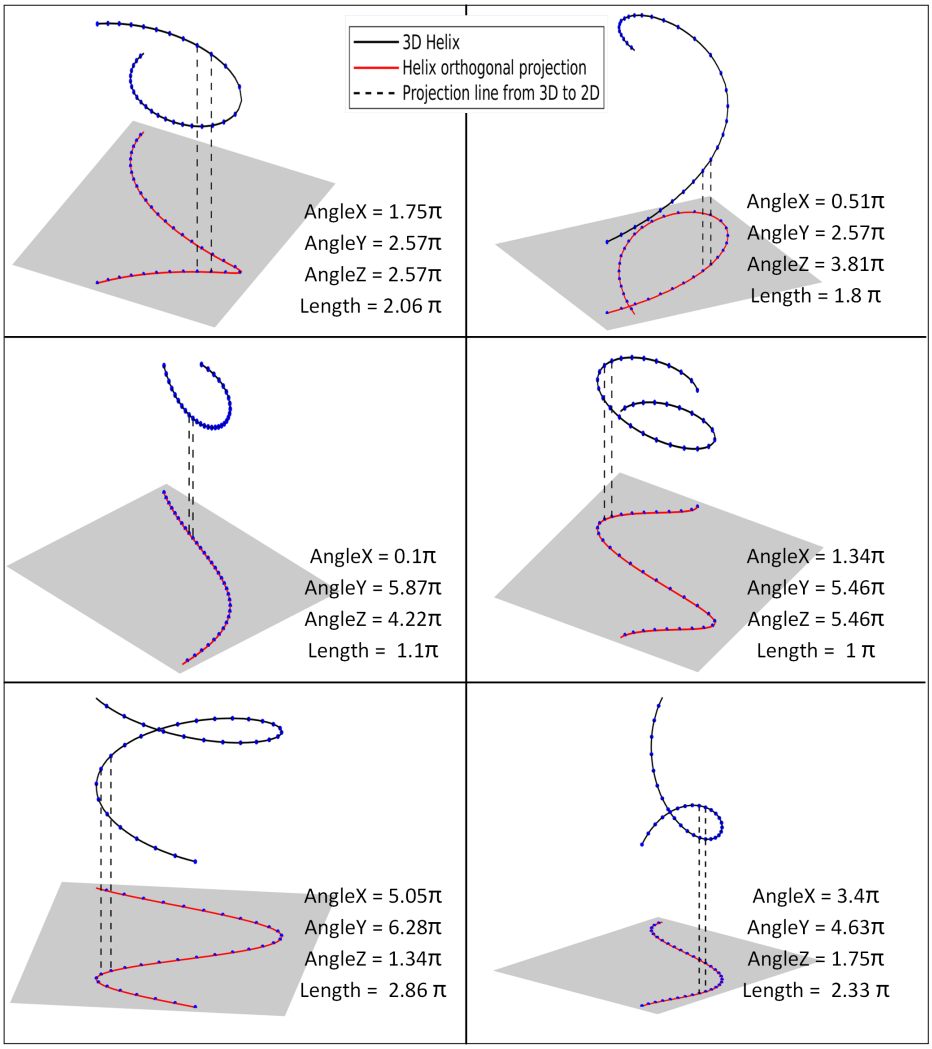

Figure 3.6: Some examples of dataset segments with their orthogonal projections. AngleX, AngleY, and AngleZ represent the rotation angles applied to the segments. Length represents the circular length of the segment.

3.4 Models training

ML algorithms have demonstrated remarkable success in 3D reconstruction, particularly when working with images. However, our case is distinct as we are dealing

with the 3D Cartesian coordinates of a circular helix, which significantly differs from image pixels. To the best of our knowledge, there is a lack of literature addressing the use of ML algorithms in such scenarios. This motivates us to test different ML algorithms in order to perform the 3D reconstruction of circular helices from planar polygonal curves.

3.4.1 Models training methods

The selection of ML algorithms in our study was driven by several factors. Firstly, we aimed to explore a diverse set of algorithms that have been widely used and have shown promising results in regression tasks. Secondly, we considered the specific requirements and characteristics of our 3D circular helix reconstruction problem. Additionally, each algorithm has its own strengths and weaknesses, and we wanted to assess their suitability for our task. Hence, we chose to employ RFR, GBR, GPR, SVR, KNN-R,ANN, and ED-CNN in our study. These algorithms cover a wide range of methodologies and offer diverse modeling capabilities, including ensemble-based learning (RFR and GBR), Bayesian non-parametric method (GPR), kernel-based method (SVR), instance based method (KNN-R), and deep learning (ANN and ED-CNN). By exploring multiple algorithms, we aim to evaluate their performance and identify the most effective approach for 3D circular helix reconstruction.

Note that meanwhile the training process of all models, to ensure reliable and accurate 3D reconstruction of circular helices, the dataset is divided into training and testing sets using K-Fold cross-validation. This technique is used to evaluate the performance and generalization ability of a ML model. It involves dividing the dataset into K subsets, or folds, of approximately equal size. The model is then trained and evaluated K times, with each fold serving as the testing set once while the remaining K-1 folds are used for training. This ensures that each data point is used for both training and testing. The results from each fold are averaged to obtain an overall performance metric, providing a more robust estimate of the model's performance. In our case, we divided the dataset into 5 folds.

To identify the optimal combination of hyperparameters, a grid search is performed. In this search, various hyperparameter combinations are exhaustively

explored based on cross-validation. The chosen values for hyperparameter tuning were initially guided by the documentation recommendations of each method and further refined through trial and error.

To measure the model's performance we use the coefficient of determination (Draper & Smith 1998):

$$R^2 = 1 - \frac{u}{v},$$

where u represents the residual sum of squares $(y_{\text{true}} - y_{\text{pred}})^2$ and v represents the total sum of squares $(y_{\text{true}} - \bar{y}_{\text{true}})^2$. The value of R^2 ranges between 0 and 1, with 1 indicating a perfect fit between the predicted and true values.

3.4.2 Random forest regressor (RFR)

RFR is an ensemble ML algorithm used for regression tasks. It combines multiple decision trees to make predictions. The basic idea behind RFR is to create an ensemble of decision trees that independently learn from random subsets of the training data and features. Each decision tree is constructed independently from the others. During the training process, RFR creates multiple subsets of the training data by random sampling with replacement, a process known as bootstrap aggregating or bagging. Each subset contains a random selection of the original data, allowing different decision trees to learn from slightly different perspectives.

In addition to subsampling the training data, RFR also performs random feature selection. For each split in a decision tree, only a subset of features is considered. This introduces further randomness and diversity among the decision trees.

With the subsampled data and randomly selected features, each decision tree is constructed using an algorithm such as the Classification and Regression Tree (CART) algorithm. The decision tree splits the data based on different features and thresholds to create a hierarchical structure of nodes and leaves.

Once the decision trees are constructed, they can be used to make predictions. Each decision tree independently produces its own prediction, and in the case of regression, the predictions from all the decision trees are averaged to obtain the final prediction.

The power of RFR lies in its ensemble learning approach. By combining the

predictions from multiple decision trees, RFR can improve the accuracy and generalization of the overall model. It helps to mitigate the risk of overfitting and increases robustness to noise and outliers in the data. An example of RFR is illustrated in Figure 3.7 where the number of trees is 100.

Motivation

The motivation behind using RFR for the 3D reconstruction of circular helices lies in its ability to handle nonlinear relationships and capture complex patterns in the data. In the case of circular helix reconstruction, where the input is the (x, y) coordinates of the circular helix and the output is the z-coordinate, RFR can effectively learn the relationship between the 2D coordinates and the corresponding z-coordinate. By training the model on a dataset that includes known (x, y) coordinates and their corresponding z-coordinates, the RFR can learn the underlying patterns and variations in the data. The algorithm's ensemble nature and ability to handle nonlinearities make it suitable for capturing the intricate geometry and shape of circular helices. Furthermore, RFR combines the power of decision trees with ensemble learning techniques, making it robust against overfitting and capable of capturing intricate patterns in the data.

Training

To train the RFR a hyperparameter tuning is crucial for optimizing the model. RFR has several hyperparameters that can be tuned to optimize its performance, such as:

- Number estimators: This parameter determines the number of decision trees in the random forest. Increasing the number of estimators can improve the model's performance but also increase computational complexity. The values chosen for training include [5, 100, 150].
- Max depth: It specifies the maximum depth of each decision tree in the forest. A higher value allows the trees to capture more complex relationships but can lead to overfitting if not carefully tuned. The values chosen for training include [None, 5, 10].

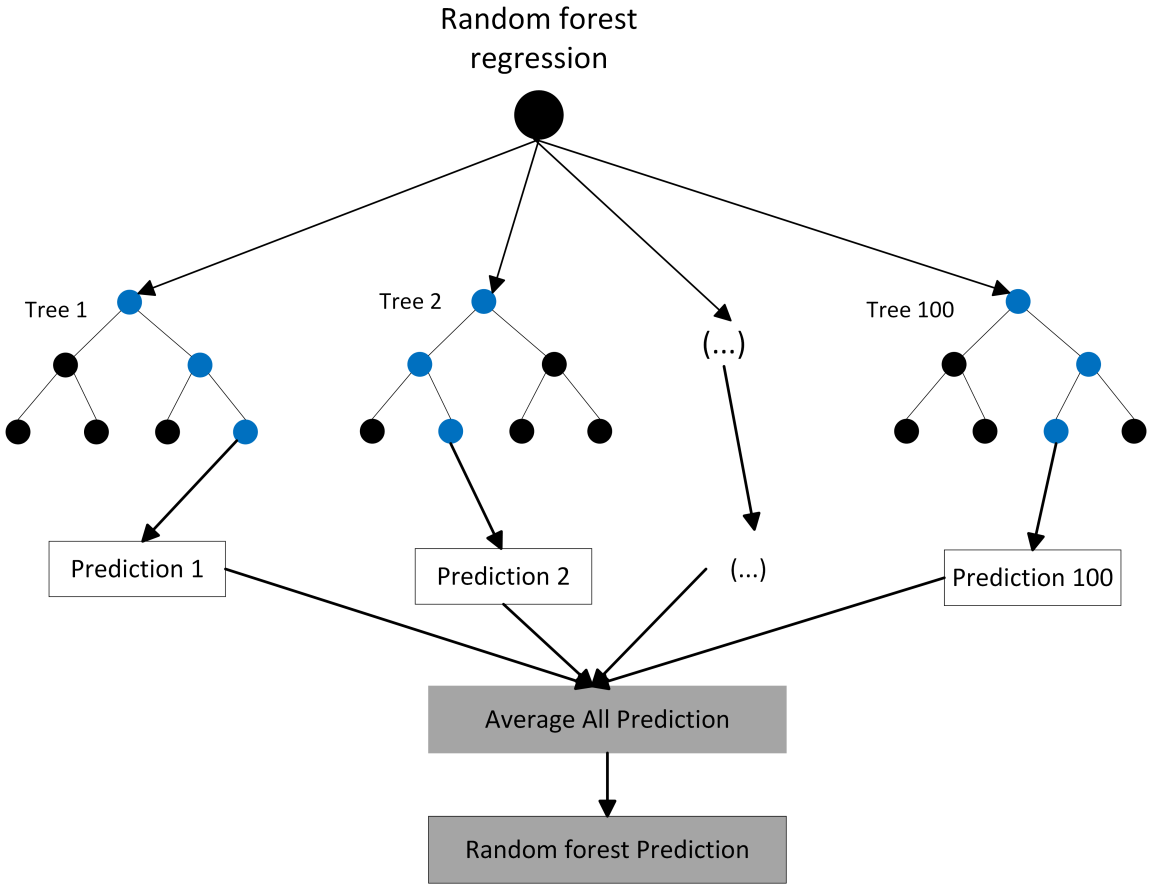

Figure 3.7: The architecture of RFR with the number of trees equal to 100.

- Min samples split: This parameter sets the minimum number of samples required to split an internal node. It controls the tree's ability to make further splits based on the number of samples available, preventing overfitting by limiting the tree's growth. The values chosen for training include [2, 4, 6, 8].

- Min samples leaf: It specifies the minimum number of samples required to be at a leaf node. Similar to 'Min samples split', this parameter helps control the tree's growth and prevents overfitting by requiring a minimum number of samples at the terminal nodes. The values chosen for training include [1, 2, 3, 4, 5].

- Max features: This parameter determines the number of features to consider when looking for the best split. It can be set to 'auto', which considers all features, or 'sqrt' or 'log2', which considers a square root or logarithm of the total number of features, respectively. The values chosen for training include ['auto', 'sqrt', 'log2'].

Combination ID	Parameters				
	Number of trees	Max depth	Min samples split	Min samples leaf	Max features
Combination 1	50	5	4	2	'sqrt'
Combination 2	150	None	2	1	'auto'
Combination 3	50	None	8	4	'sqrt'
Combination 4	100	5	2	1	'auto'
⋮	⋮	⋮	⋮	⋮	⋮

Table 3.1: Some of the parameter combinations for RFR.

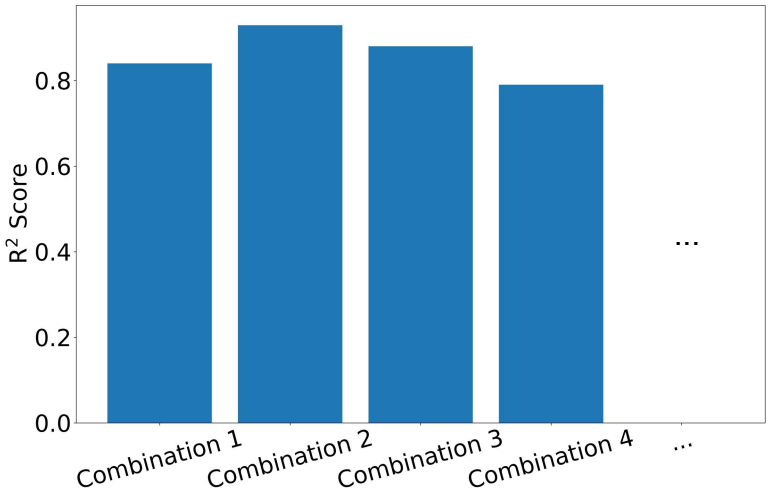

Figure 3.8: The R^2 score for some of the parameter combinations.

Table 3.1 shows some parameter combinations from the grid search hyperparameters process. Figure 3.8 illustrates the R^2 scores of the models trained with different parameter combinations. As a result of the grid search, we identify the combination [Number of tree = '150', Max depth = 'None', Min samples split =

'2', Min samples leaf = '1', Max features = 'auto'] as the one that provides the best model performance $R^2 = 0.929$.

3.4.3 Gradient boosting regressor (GBR)

GBR is a powerful ensemble ML algorithm commonly used for regression tasks. It belongs to the family of boosting algorithms and is known for its ability to create a strong predictive model by combining weak learners, typically decision trees.

Unlike Random Forest Regression, GBR uses a different approach called gradient boosting to build the ensemble of decision trees. The algorithm starts by creating a simple decision tree, also known as a weak learner, and makes predictions on the training data. It then calculates the residuals, which represent the differences between the predicted values and the true values.

In the subsequent iterations, GBR focuses on learning the residuals by training additional decision trees. Each new tree is trained to predict the residuals of the previous trees, rather than the original target variable. This iterative process continues, with each new tree attempting to correct the errors made by the ensemble of trees built so far.

The final prediction from the GBR is obtained by summing the predictions from all the individual trees in the ensemble. Each tree contributes to the final prediction by a certain weight, which is determined during the training process.

The strength of the GBR lies in its ability to sequentially learn from the mistakes of previous models. By continually focusing on the residuals, the algorithm can improve the accuracy of the predictions with each iteration. Figure 3.9 shows an example of the training process of the GBR.

Motivation

Similarly to RFR, GBR is able to handle complex relationships and capture intricate patterns in the data by learning the nonlinear mapping between the 2D coordinates and the corresponding z-coordinate using the decision trees. Especially since the ensemble nature of GBR allows it to combine multiple weak learners, typically decision trees, to form a strong predictive model. This ensemble approach enables GBR to capture the intricate geometry and shape of circular helices by

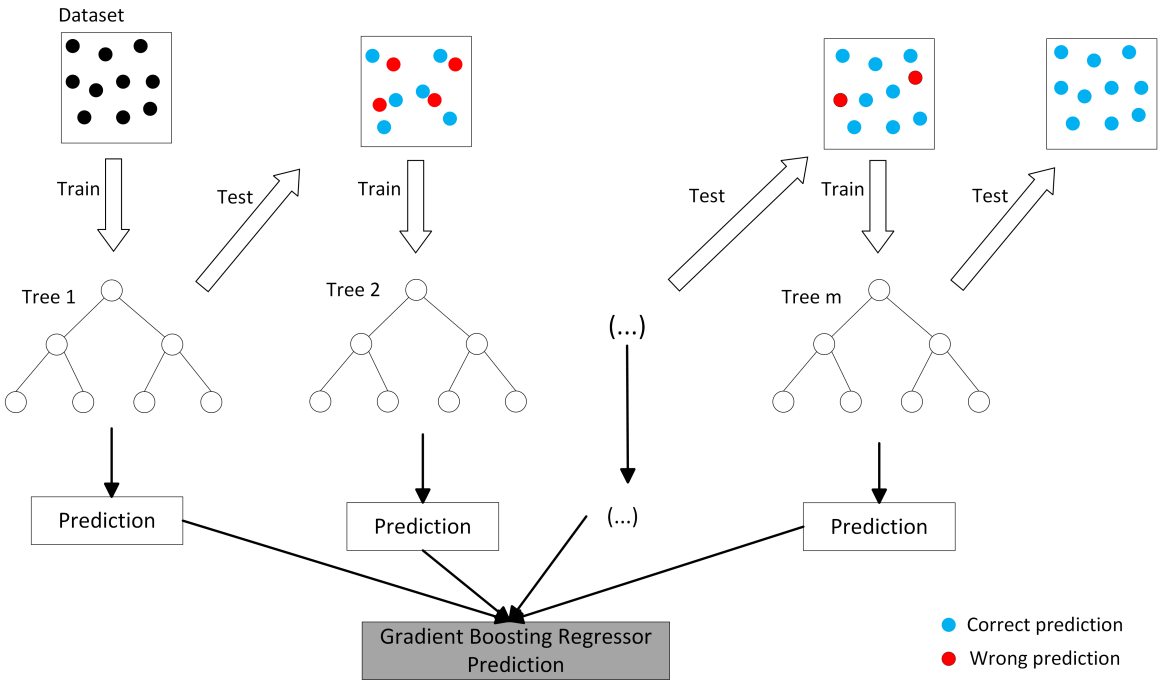

Figure 3.9: The architecture of GBR.

leveraging the collective knowledge of the individual decision trees. By iteratively focusing on the residuals and correcting the errors made by the ensemble, GBR can adapt and refine its predictions to improve accuracy.

Training

Training the GBR model involves crucial hyperparameter tuning to optimize its performance. GBR has several hyperparameters that can be tuned to achieve the best results. Some important hyperparameters to consider are:

- Number of estimators: This parameter determines the number of boosting stages or decision trees in the gradient boosting ensemble. Increasing the number of estimators can improve the model's performance, but it also increases computational complexity. The values chosen for training include [100, 200, 300, 400].

- Learning rate: The learning rate controls the contribution of each tree in the ensemble. A smaller learning rate makes the model more robust but requires more estimators to achieve similar performance. It is essential to find an optimal balance between the learning rate and the number of estimators. The values chosen for training include [0.01, 0.05, 0.1].
- Max depth: Similar to RFR. The values chosen for training include [1, 2, 3, 4, 5].
- Max features: Similar to RFR. The values chosen for training include ['auto', 'sqrt', 'log2'].

Combination ID	Parameters			
	Nb of Estimators	Learning Rate	Max depth	Max features
Combination 1	100	0.05	10	'sqrt'
Combination 2	300	0.1	None	'auto'
Combination 3	200	0.1	5	'sqrt'
Combination 4	400	0.01	5	'auto'
⋮	⋮	⋮	⋮	⋮

Table 3.2: Some of the parameter combinations for GBR.

Table 3.2 shows some parameter combinations from the grid search hyperparameters process. Figure 3.10 illustrates the R^2 scores of the models trained with different parameter combinations. As a result of the grid search, we identify the combination [Number of estimators = '400', Learning rate ='0.01', Max depth = '5', Max features = 'auto'] as the one that provides the best model performance $R^2 = 0.59$. Note that this R^2 score indicates that the model is not able to perform a good reconstruction. One possible reason for that is the complexity of GBR, which is based on an ensemble of weak learners (decision trees) that are trained sequentially to correct the mistakes of previous models. The increased complexity of GBR can make it prone to overfitting.

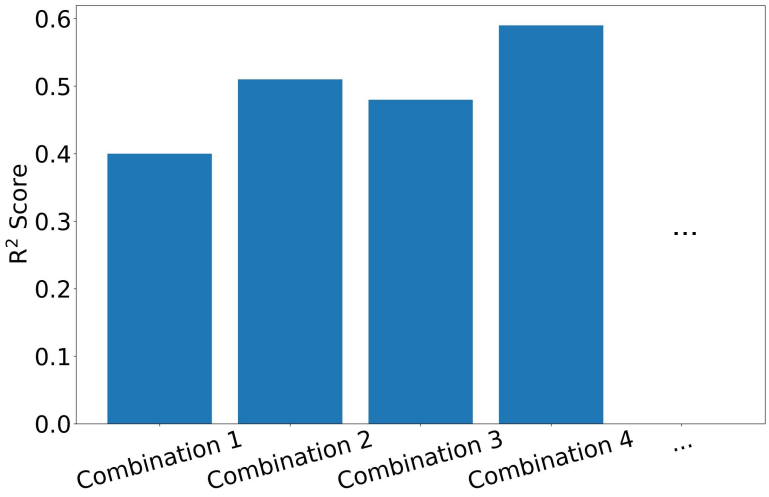

Figure 3.10: The R^2 score for some of the parameter of GBR.

3.4.4 Gaussian process regressor (GPR)

GPR is a powerful probabilistic ML algorithm used for regression tasks. Unlike other regression models that assume a specific functional form for the relationship between input and output variables, GPR models the underlying function as a distribution over functions. It is based on the concept of Gaussian processes, which define a prior distribution over functions, and uses Bayesian inference to learn the posterior distribution of functions given observed data.

In contrast to many popular supervised ML algorithms that determine exact parameter values in a function, the Bayesian approach takes a probabilistic perspective by inferring a probability distribution over all possible parameter values. Consider a linear function: $y = wx + \sigma$. By incorporating observed data, the Bayesian approach updates the distribution using Bayes' Rule:

$$\begin{aligned} p(w|y,X) &= \frac{p(y|X,w)p(x)}{p(y|X)} \\ posterior &= \frac{likelihood \times prior}{marginal\ likelihood} \end{aligned} \tag{3.2}$$

The posterior ('updated') distribution $p(w|y,X)$ combines information from both the prior distribution and the dataset. To make predictions at unseen points

of interest (x^*), the Bayesian approach calculates the predictive distribution by weighting all possible predictions based on their corresponding posterior distribution (Rasmussen et al. 2006):

$$p(f^*|x^*, y, X) = \int_w p(f^*|x^*, w)p(w|y, x)\, dw, \tag{3.3}$$

where f^* is the prediction label. In order for the integration to be manageable, the prior and likelihood are typically considered to be Gaussian. With that presumption, we can solve for the predictive distribution and obtain a Gaussian distribution, from which we can derive a point prediction using the mean and uncertainty quantification using the variance. Therefore, the Bayesian approach not only allows us to make predictions but also provides a measure of confidence or uncertainty in those predictions by leveraging the properties of the Gaussian distribution.

Motivation

The motivation behind using GPR for the 3D reconstruction of circular helices from planar polygonal curves lies in its flexibility and ability to provide probabilistic predictions. GPR models capture uncertainty in the data by representing the target variable as a distribution over functions rather than a single point estimate. This makes GPR suitable for situations where the data may exhibit complex patterns or where the underlying relationship between inputs and outputs is unknown. Furthermore, GPR provides a natural way to incorporate prior knowledge or assumptions about the data through the choice of covariance function or kernel. This allows the model to adapt to different data patterns and can help regularize the model to prevent overfitting.

Training

Training GPR involves tuning important hyperparameters that strongly influence the model's behavior and predictive accuracy. These key hyperparameters include:

- Kernel function: The choice of kernel function determines the type of spatial correlation between the input features (planar polygonal projection) and the

target variable (z-coordinate). Different kernel functions, such as radial basis function, Matérn, or rational quadratic, offer varying levels of flexibility in capturing complex patterns and correlations in the data. The values chosen for training include ['Matérn', 'Radial basis', 'Rational quadratic'].

- Length-scale parameter: This parameter controls the length scale of the kernel function and influences the smoothness and flexibility of the GPR model. Larger length-scale values result in smoother predictions, while smaller values allow for more intricate and localized variations. The values chosen for training include ['0.1', '1.0', '10'].

- Noise level: The noise level parameter accounts for the level of noise present in the training data. It allows the GPR model to account for measurement errors and uncertainties in the target variable. Adjusting the noise level parameter helps improve the model's robustness to noisy data and prevents overfitting. The values chosen for training include ['0.1', '5', '10'].

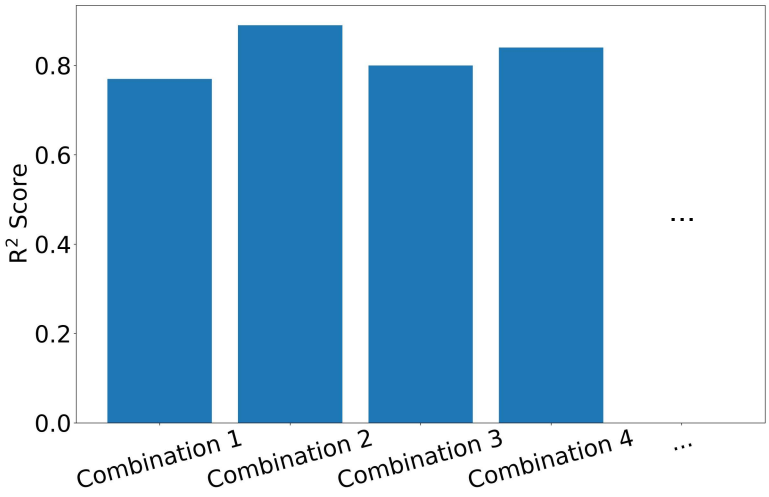

Figure 3.11: The R^2 score for some of the parameter combinations of GPR.

Table 3.3 shows some parameters combinations. Figure 3.11 illustrates the R^2 scores of the models trained with different parameter combinations. As a result of the grid search, we identify the combination [Kernel function = 'Radial basis',

Combination ID	Parameters		
	Kernel function	Length-scale parameter	Noise level
Combination 1	Matérn	1.0	1.0
Combination 2	Radial basis	1.0	5.0
Combination 3	Rational quadratic	0.1	0.1
Combination 4	Radial basis	10	0.1
⋮	⋮	⋮	⋮

Table 3.3: Some of the parameter combinations for GPR.

Length-scale parameter ='1.0', Noise level = '5.0'] as the one that provides the best model performance $R^2 = 0.89$.

3.4.5 Support vector regressor (SVR)

SVR is a powerful regression algorithm that belongs to the family of Support Vector Machines (SVMs). SVR builds upon the principles of SVMs, which were originally developed for classification tasks but have been adapted for regression as well. SVR excels at handling complex, non-linear relationships between the input variables and the target variable.

One of the key concepts behind SVR is the use of kernel functions. These functions allow SVR to implicitly transform the input data into a higher-dimensional feature space, where the data becomes more amenable to linear separation. By projecting the data into this higher-dimensional space, SVR can effectively model non-linear patterns in the data using linear techniques. Figure 3.12 illustrates the mapping from input space to feature space where it would be easier to separate the data. The choice of the kernel function, such as the radial basis function (RBF) or polynomial kernel, plays a crucial role in determining the flexibility and expressiveness of the SVR model.

Motivation

SVR is known for its capability to handle nonlinear relationships, robustness to outliers, theoretical foundations in statistical learning theory, and its wide applicability across various domains. SVR employs kernel functions to capture nonlinear

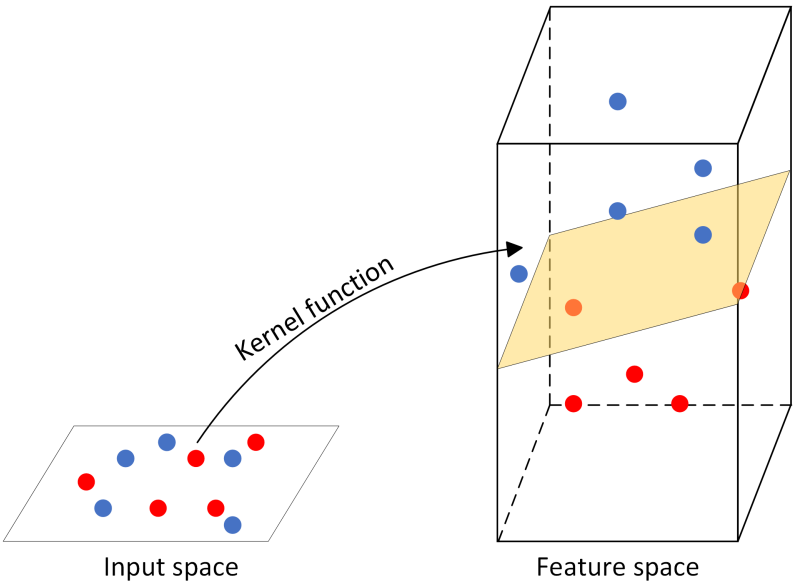

Figure 3.12: Mapping the data from input space to feature space using the kernel function.

patterns, focuses on support vectors for increased robustness, and balances empirical risk and model complexity. These features enable SVR to accurately model complex systems, make reliable predictions, and find applications in diverse fields. Therefore it is worth trying a kernel-based algorithm to test its performance in the 3D reconstruction of circular helices.

Training

The training of SVR involves tuning several key hyperparameters that significantly influence the model's behavior and performance. These hyperparameters include:

- Kernel function: The choice of the kernel function in SVR determines the type of non-linear relationship that can be modeled between the input features (planar polygonal projection) and the target variable (z-coordinate). Different kernel functions, such as radial basis function, polynomial, or sigmoid, offer varying degrees of flexibility in capturing complex patterns. The values chosen for training include ['Radial basis', 'Sigmoid', 'Polynomial'].

- C parameter: This parameter, often referred to as the penalty parameter,

controls the trade-off between minimizing the error and maximizing the margin. A smaller C value allows for more errors but larger margins, while a larger C value aims to minimize errors but can lead to overfitting. The values chosen for training include ['0.1', '1', '3', '5'].

- Epsilon parameter: Epsilon specifies the margin of tolerance for errors in SVR. It determines the width of the epsilon tube within which errors are not penalized. Larger epsilon values allow for a wider margin of tolerance, providing more robustness to noise and outliers. The values chosen for training include ['0.01', '0.05', '0.1'].

Combination ID	Parameters		
	Kernel function	C parameter	Epsilon parameter
Combination 1	Radial basis	1.0	0.01
Combination 2	Sigmoid	0.1	0.1
Combination 3	Polynomial	1.0	0.05
Combination 4	Radial basis	5.0	0.1
⋮	⋮	⋮	⋮

Table 3.4: Some of the parameter combinations for SVR.

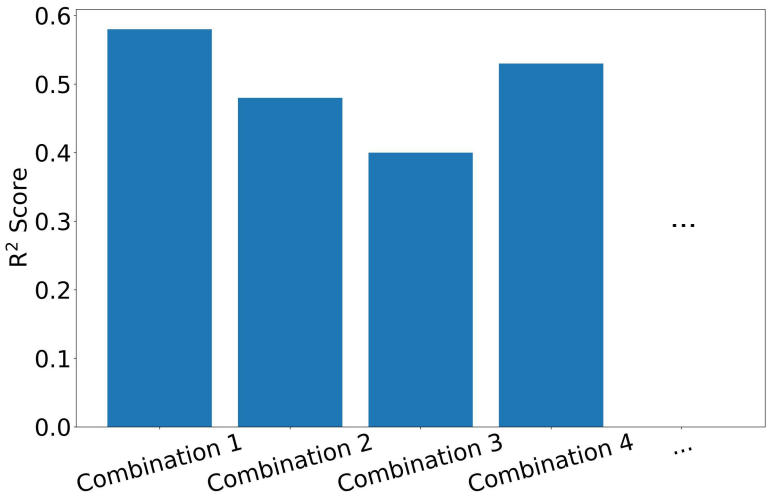

Figure 3.13: The R^2 score for the some of the parameter combinations of SVR.

Table 3.4 shows some of the parameter combinations. Figure 3.13 illustrates the R^2 scores of some of the models trained with different parameter combinations. As a result of the grid search, we identify the combination [Kernel function = 'Radial basis', C parameter ='1.0', Epsilon parameter = '0.01'] as the one that provides the best model performance $R^2 = 0.58$. It is important to note that while the SVR model may have achieved a relatively lower R^2 score, it exhibited successful reconstruction in some instances and less successful reconstruction in others.

3.4.6 K-nearest neighbors regression (KNN-R)

KNN-R is a non-parametric algorithm that estimates the target variable by considering the average or weighted average of the values of its K nearest neighbors in the feature space. KNN-R operates on the principle that similar data points tend to have similar target values. It is a simple yet effective algorithm for regression tasks, particularly when the underlying relationships in the data are not easily characterized by a mathematical function.

In KNN-R, the choice of K determines the number of neighbors considered for estimating the target value. A smaller K value captures local information and is more susceptible to noise, while a larger K value captures global information and may oversmooth the predictions. Additionally, the choice of distance metric, such as Euclidean distance or Manhattan distance, affects the calculation of similarity between data points and can influence the performance of KNN-R.

Motivation

The flexibility of KNN-R allows it to adapt to the local characteristics of the data, enabling accurate reconstructions of helical structures. By considering the nearest neighbors in the feature space, KNN-R can effectively model the non-linear mappings between the planar polygonal projection (input) and the corresponding z-coordinate (target), leading to faithful 3D reconstructions.

Furthermore, KNN-R is known for its simplicity and ease of implementation. It does not require assumptions about the underlying data distribution or complex mathematical computations. This makes KNN-R a suitable choice when the focus is on the intrinsic structure of the data and when interpretability is desired.

Training

Similar to RFR and SVR, the training process for KNN-R involves several key considerations. Firstly, the choice of K must be determined. This can be done through hyperparameter tuning techniques, such as grid search or cross-validation, to find the optimal K value that balances model complexity and predictive performance. The values chosen for training include ['5', '10', '15', '25'].

Secondly, the distance metric used to measure the similarity between data points needs to be selected. Euclidean distance is commonly used in KNN-R, but other metrics like Manhattan distance can be employed based on the characteristics of the data and the underlying problem. The values chosen for training include ['Euclidean distance', 'Manhattan distance'].

Combination ID	Parameters	
	K value	Distance metric
Combination 1	10	Euclidean distance
Combination 2	10	Manhattan distance
Combination 3	15	Manhattan distance
Combination 4	5	Euclidean distance
⋮	⋮	⋮

Table 3.5: Some of the parameter combinations for KNN-R.

Table 3.5 shows some of the parameter combinations. Figure 3.14 illustrates the R^2 scores of some of the models trained with different parameter combinations. As a result of the grid search, we identify the combination [K value = '25', Distance metric ='Euclidean distance'] as the one that provides the best model performance $R^2 = 0.88$.

3.4.7 Artificial neural networks (ANN)

ANN is a class of ML models inspired by the structure and functionality of the human brain. ANN consists of interconnected nodes, called neurons, organized in layers. Each neuron takes inputs, applies a mathematical transformation, and produces an output. Figure 3.15 show the architecture of ANN. The strength of ANN lies in their ability to learn complex patterns and relationships in the data through a process known as training.

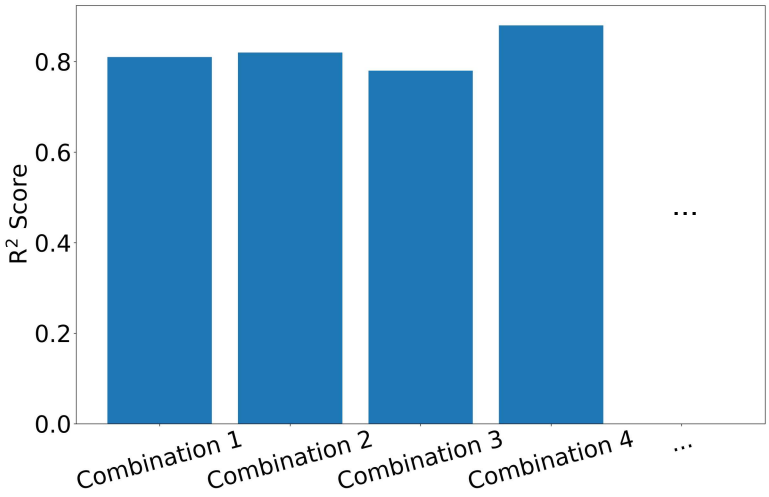

Figure 3.14: The R^2 score for some of the parameter combinations of KNN-R.

ANN is particularly suited for tasks that involve non-linear mappings and intricate relationships between input variables and the target variable. Traditional linear models may struggle to capture and represent these complex functions accurately. ANN, on the other hand, can leverage multiple layers and nonlinear activation functions to capture intricate patterns and relationships in the data. The neurons in each layer receive inputs, perform calculations using weights and biases, and apply an activation function to produce an output. By stacking multiple layers with increasing complexity, ANN can learn and model highly nonlinear relationships, making them a powerful tool for regression tasks.

The training process of ANN involves adjusting the weights and biases of the neurons to minimize a predefined loss function. This adjustment is performed iteratively using optimization algorithms such as stochastic gradient descent or Adam. Backpropagation, a fundamental algorithm in neural networks, is used to compute the gradients of the loss function with respect to the weights and biases, enabling the network to learn from data and optimize its parameters. By iteratively updating the weights and biases based on the computed gradients, the ANN gradually improves its performance and becomes better at generalizing to unseen data.

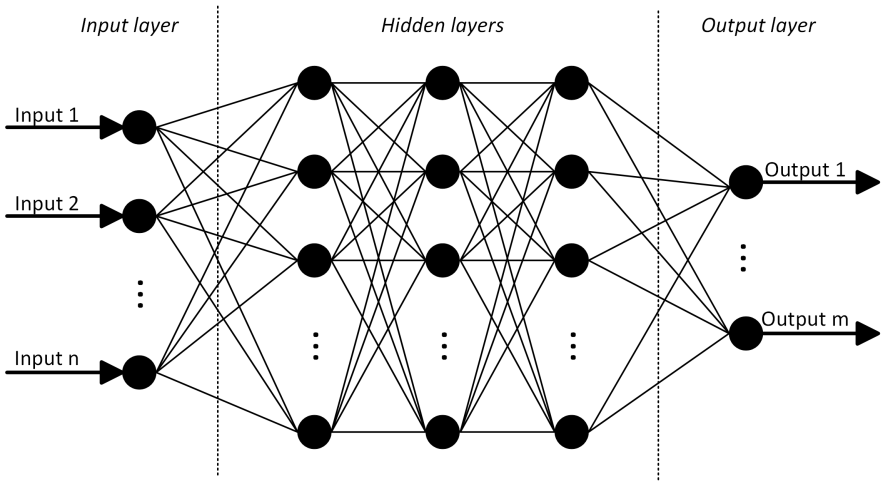

Figure 3.15: The network architecture of ANN.

Motivation

Traditional methods for helical reconstruction often rely on simplified mathematical models or heuristic approaches, which may not capture the full complexity of helical structures. These methods may struggle to account for variations in curvature, twist, and other intricate characteristics that are essential for accurate 3D reconstruction. ANN, with their capacity to learn complex patterns and relationships, provides a more flexible and data-driven approach to helical reconstruction.

Another motivation for using ANN in helical reconstruction is their ability to handle high-dimensional data and learn from large amounts of training data. Moreover, ANN offers the advantage of being able to learn from diverse datasets. By training on a wide range of helical structures with different characteristics, ANN can generalize their learning and adapt to variations in the data. This flexibility allows ANN to handle different helical geometries, variations in helical parameters, and even new helical structures that were not present in the training dataset. As a result, ANN provides a robust and adaptable framework for helical reconstruction, accommodating the inherent diversity and complexity of helical structures.

Additionally, the training process of ANN allows for iterative refinement and optimization. By adjusting the weights and biases during training, ANN continuously improve their performance and fine-tune their ability to capture the intricate

details of helical structures. This iterative learning process ensures that the ANN becomes increasingly accurate in reconstructing circular helices as it receives more training examples.

Training

Hyperparameter tuning plays a crucial role in optimizing the behavior and performance of the ANN model. Several key hyperparameters significantly influence the model's behavior, including:

- Number of hidden layers: The number of hidden layers in the ANN determines the complexity and depth of the model. Increasing the number of hidden layers allows the ANN to learn more complex relationships but may also lead to overfitting if not carefully tuned. The values chosen for training include ['2', '3', '5', '10'].

- Number of neurons per hidden layer: The number of neurons in each hidden layer affects the model's capacity to capture and represent complex patterns. A larger number of neurons increases the model's flexibility but may also increase the risk of overfitting. The values chosen for training include ['120', '130', '150', '170'].

- Activation function: The choice of activation function determines the nonlinear transformation applied to the inputs of each neuron. Common activation functions used for regression problems are hyperbolic tangent (tanh), and rectified linear unit (ReLU) for the inner layers (Note that the performance of both activation functions is similar if the network is not very deep). Since the output of the network are the z-coordinates which can be any real value, the activation function of the output layer must be linear. The values chosen for training include ['ReLU', 'tanh'].

- Learning rate: The learning rate controls the step size during the optimization process. A higher learning rate allows for faster convergence but may lead to overshooting the optimal solution, while a lower learning rate requires more iterations to converge but may yield more accurate results. The values chosen for training include ['0.01', '0.05', '0.1'].

- Number of training iterations: The number of training iterations determines the length of the training process. ANN typically requires a large number of iterations to converge and generate high-quality samples. It is important to train the ANN for a sufficient number of iterations to achieve desirable results. The values chosen for training include ['200', '450', '600'].

- Regularization techniques: Regularization techniques such as L1 or L2 regularization can be applied to prevent overfitting in the ANN. These techniques introduce penalty terms to the loss function, discouraging the model from relying too heavily on any single input feature or neuron. Dropout is another regularization technique that randomly disables a fraction of neurons during training, forcing the remaining neurons to learn more robust and independent representations. The values chosen for training include ['Dropout', 'L1', 'L2'].

- Optimizer: The optimizer determines the algorithm used to update the weights and biases of the ANN during training. Popular optimization algorithms include stochastic gradient descent (SGD), Adam, and RMSprop. The values chosen for training include ['SGD', 'Adam', 'RMSprop'].

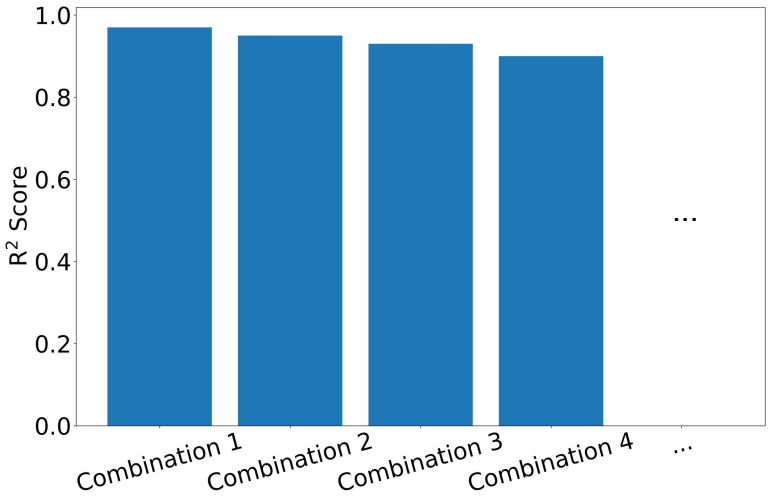

Figure 3.16: The R^2 score for some of the parameter combinations of ANN.

Combination ID	Parameters				
	NB of hidden layers	NB of neurons	Activation function	Learning rate	Regularization techniques
Combination 1	2	150	ReLU	0.01	Dropout
Combination 2	3	120	tanh	0.1	L2
Combination 3	5	170	ReLU	0.01	L1
Combination 4	10	130	tanh	0.05	L2
⋮	⋮	⋮	⋮	⋮	⋮

Table 3.6: Some of the parameter combinations for ANN.

Table 3.6 shows some of the parameter combinations, note that the table does not show all the parameters used during the hyperparameters tuning. Figure 3.16 illustrates the R^2 scores of the models trained with different parameter combinations. As a result of the grid search, we identify the combination [NB of hidden layers = '2', NB of neurons = '150', Activation function = 'RelU', Learning rate = '0.01', Number of training iterations = '450', Regularization techniques = 'Dropout', Optimizer = 'Adam'] as the one that provides the best model performance $R^2 = 0.97$.

3.4.8 Encoder-Decoder convolutional neural networks (ED-CNN)

ED-CNN are a class of neural network architectures specifically designed for tasks involving structured or variable-sized data, such as image segmentation, image generation, and 3D reconstruction. These networks combine the power of convolutional neural networks in capturing spatial information with the encoder-decoder structure in transforming high-dimensional input into a lower-dimensional representation and then reconstructing the output.

The architecture of ED-CNN consists of an encoder network followed by a decoder network (Figure 3.17). The encoder network is responsible for extracting relevant features from the input data and compressing it into a lower-dimensional representation, while the decoder network takes this compressed representation

and reconstructs the output.

The encoder network typically consists of several convolutional layers, which convolve the input data with learned filters to capture local patterns and spatial relationships. These convolutional layers are often followed by pooling layers, such as max pooling or average pooling, which downsample the feature maps, reducing their spatial dimensionality and extracting the most salient information. Batch normalization, a technique commonly employed in ED-CNN, may also be incorporated after the convolutional layers. Batch normalization helps to normalize the activations of each layer, making the network more stable during training and allowing for faster convergence.

After the encoder network, the decoder network takes the compressed representation and aims to reconstruct the output in its original format or domain. The decoder network often employs upsampling operations, such as transposed convolutions or bilinear upsampling, to increase the spatial resolution of the feature maps. These upsampling operations are typically accompanied by convolutional layers, which help refine the features and generate the final output.

During the training process of ED-CNN, the model's parameters are optimized using backpropagation and gradient descent-based algorithms. The model is trained on input-output pairs, where the input is fed through the encoder network to obtain the compressed representation, and the output is compared with the ground truth. The discrepancy between the predicted output and the ground truth is quantified by a loss function, such as mean squared error or cross-entropy loss. The gradients of the loss function with respect to the model's parameters are computed through backpropagation, allowing for parameter updates that gradually improve the model's performance.

The strength of ED-CNN networks lies in their ability to handle structured or variable-sized data while capturing spatial dependencies and contextual information. These networks have demonstrated exceptional performance in tasks such as image segmentation, where the input and output have a spatial structure and varying resolutions. By leveraging convolutional layers, pooling operations, upsampling techniques, and batch normalization, ED-CNN networks can effectively model complex patterns, normalize activations, and generate detailed and contextually meaningful outputs.

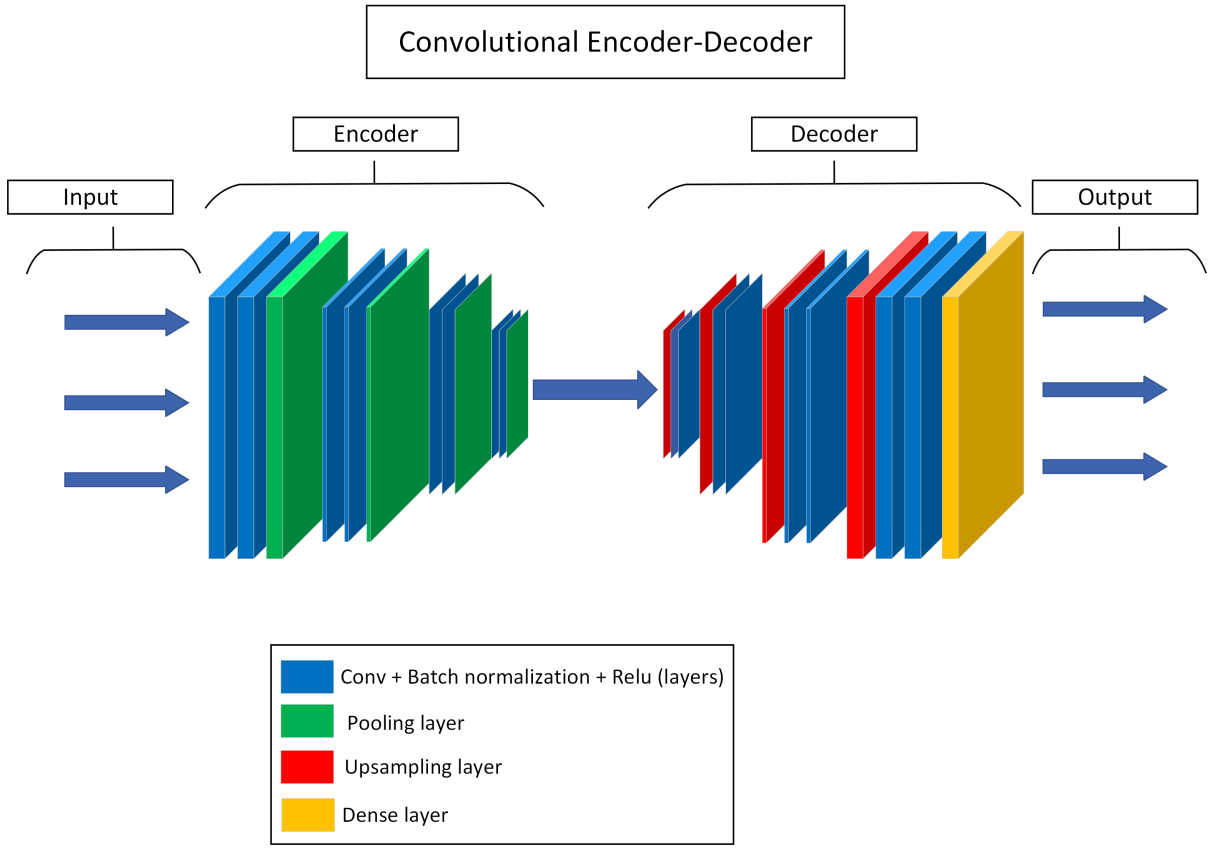

Figure 3.17: The network architecture of Convolutional Encoder-Decoder.

Motivation

ED-CNN models have demonstrated remarkable performance in various domains, particularly in tasks involving image analysis and generation. However, their efficacy is not limited to image-related problems. These models have shown promising results in handling non-image inputs as well (Bing et al. 2021), making them a compelling choice for a wide range of applications.

By testing the effectiveness of ED-CNN models in the context of helical 3D reconstruction, we aim to leverage their capabilities beyond the realm of images. The objective is to explore their potential for capturing the intricate details of circular helices, overcoming the limitations of traditional approaches, and providing a more accurate and data-driven solution for circular helix reconstruction.

Training

Hyperparameters play a crucial role in optimizing the behavior and performance of ED-CNN models. Some key hyperparameters that significantly influence the model's behavior include:

- Number of convolutional layers: The number of these layers affects the depth and complexity of the model. Increasing the number of layers allows the model to capture more intricate features but may also increase the risk of overfitting if not properly tuned. Note that in our model, each convolutional layer in the Encoder is followed by a pooling layer to decrease the spatial dimensions, while in the Decoder, each convolutional layer is followed by an UpSampling layer to restore the original dimensions of the data. The values chosen for training include ['6', '8', '10', '12'].
- Filter size: The size of the filters used in the convolutional layers determines the receptive field of the model. Larger filters can capture more global features, while smaller filters excel at capturing local details. The values chosen for training include ['24', '36', '48'].
- Learning rate: Similar to ANN. The values chosen for training include ['0.01', '0.05', '0.1'].
- Regularization techniques: Similar to ANN. The values chosen for training include ['Dropout', 'BatchNormalization'].
- Number of training iterations: Similar to ANN. The values chosen for training include ['200', '450', '600'].
- Optimizer: Similar to ANN. The values chosen for training include ['SGD', 'Adam', 'RMSprop'].

Table 3.7 shows some of the parameter combinations. Figure 3.18 illustrates the R^2 scores of some of the models trained with different parameter combinations. As a result of the grid search, we identify the combination [NB of convs layer = '8', Filter size = '150', Learning rate = '0.01', Number of training iterations =

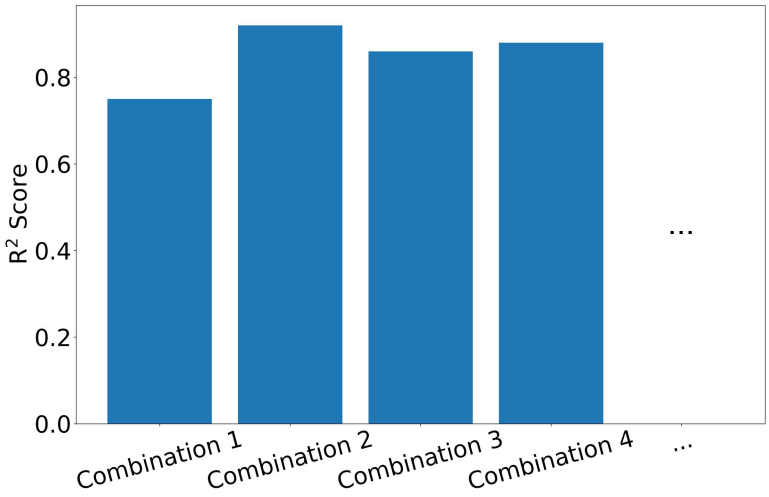

Figure 3.18: The R^2 score for some of the parameter combinations of ED-CNN.

Combination ID	Parameters			
	NB of Convs layers	Filter size	Learning rate	Regularization techniques
Combination 1	12	24	0.1	Dropout
Combination 2	8	48	0.01	BatchNormalization
Combination 3	6	48	0.05	Dropout
Combination 4	10	36	0.01	BatchNormalization
⋮	⋮	⋮	⋮	⋮

Table 3.7: Some of the parameter combinations for ED-CNN.

'600', Regularization techniques = 'BatchNormalization', Optimizer = 'Adam'] as the one that provides the best model performance $R^2 = 0.92$.

3.5 Curve segmentation and reconstruction

After training the ML model, we divide the input curve into smaller segments using a curve segmentation approach. Each segment is independently reconstructed, and afterward, we combine them to form the complete 3D circular helix approximation. To enhance the overall smoothness of the curve, especially at the connecting points between segments, a post-processing smoothing algorithm is employed. This al-

gorithm plays a vital role in refining the reconstructed curve and enhancing its aesthetic quality.

3.5.1 Curve segmentation

Before discussing how we divide the input curve ($P_{2D} = p_{2D,0}, p_{2D,1}, ..., p_{2D,n}$) into sections, we first describe how for a given section of the input curve ($p_{2D,i}, ..., p_{2D,j}$) we measure the smoothness of its corresponding 3D reconstructed curve ($p_{3D,i}, ..., p_{3D,j}$). n represents the number of points of the input polygonal curve P_{2D}, with $1 \leq i < j \leq n$.

In order to compare the smoothness of two 3D curves, we employed the concept of variance curvature. Where the average curvature of a curve provides information about its overall smoothness. It represents the average amount of bending or curvature along the entire length of the curve. A higher variance curvature indicates a more tightly curved or jagged curve, while a lower variance curvature suggests a smoother and more gradual curve.

Therefore, to do the smoothness comparison between the two 3D reconstructed curves, firstly, starting from a set of points representing each curve we calculate the curvature at each point of the curves. The curvature $\kappa(t)$ is defined as the absolute value of the cross product of the second derivatives of the x and y coordinates with t being the arc length:

$$\kappa(t) = |\frac{x''(t) * y'(t) - y''(t) * x'(t)}{(x'(t)^2 + y'(t)^2)^{(3/2)}}|.$$

Next, we compute the variance of curvature $\sigma^2(\kappa)$ for each curve. The variance is calculated as the average of the squared differences between each curvature value and the mean curvature value:

$$\sigma^2(\kappa) = \frac{1}{n}\sum_{i=1}^{n}(\kappa_{t_i} - \bar{\kappa})^2$$

where $\bar{\kappa}$ is the mean curvature value. Thus for a given section of the input curve ($p_{2D,i}, ..., p_{2D,j}$) we calculate a term for error cost in reconstruction as follows: $E_{\text{cost}(p_{3D,i}, p_{3D,j})} = \sigma^2(\kappa_{p_{3D,i}, p_{3D,j}})$ where this term determines the smoothness of the

reconstructed curve. Hence, the curve with the lowest E_{cost} value is the smoother curve.

Note that we chose the variance of curvature as a measure of smoothness because it captures the extent of variability and irregularity in the bending of a curve, providing a comprehensive assessment beyond just the average curvature. Furthermore, since the curvature of a circular helix remains constant with arc length, a lower variance of curvature indicates a smoother reconstructed curve that closely resembles a circular helix.

To determine the best segmentation of the curve, we utilize the dynamic programming algorithm used in (McCrae & Singh 2011). An essential parameter in this algorithm is denoted as E_{cost}. We calculate the upper triangular part of a matrix M, which has dimensions $n \times n$, and n represents the number of points in the input polygonal curve P_{2D}. The entries $M(i, j)$, with $1 \leq i < j \leq n$, are computed in a bottom-up manner, starting from elements closest to the diagonal, using the following equation:

$$M(i,j) = \min \left\{ E_{\text{cost}(p_{3D,i}, p_{3D,j})}, \min_{i<k<j} \{M(i,k) + M(k,j)\} \right\}$$

The above equation helps determine whether it is better to reconstruct a single curve ($E_{\text{cost}(p_{3D,i}, p_{3D,j})}$) for the entire section ($p_{2D,i} \ldots p_{2D,j}$), or to divide the curve into one or more pieces, each corresponding to the intervals ($p_{2D,i} \ldots p_{2D,k}$) and ($p_{2D,k} \ldots p_{2D,j}$). The information regarding where to split the input curve is stored as the matrix M is populated.

Once the process is done, the element $M(1, n)$ represents the cost of the optimal configuration for points ($p_{3D,1} \ldots p_{3D,n}$) in the polygonal curve P. To reconstruct the solution, we navigate through the matrix M starting from the element $(1, n)$ and make necessary splits when required.

3.5.2 Curves assembling and post-processing smoothing

After dividing the input polygonal curve into segments and reconstructing each segment using the ML algorithm, we assemble the reconstructed pieces to form the 3D reconstructed circular helix. However, this assembly process introduces G^1 and G^2 discontinuities (tangent and curvature discontinuities), particularly

at the connecting points of the reconstructed pieces. To address this issue, we first uniformly sample the reconstructed circular helix curve and then apply the Savitzky-Golay filter (Savitzky & Golay 1964) to enhance its smoothness. This filter employs a moving window and polynomial fitting approach, where it fits a low-degree polynomial to local segments of the curve within the selected window. By replacing each data point with a calculated polynomial coefficient, it effectively creates a smoother and visually appealing representation of the curve, preserving essential features while minimizing noise. This results in a reconstructed circular helix that not only maintains continuity but also offers improved aesthetic quality.

3.6 Experiments results

Our method has been implemented in Python and has been tested with three types of input curves. Firstly we apply our reconstruction to polygonal curves, which are the orthogonal projection of synthetic circular helices subjected to different rotations. Secondly, we tested the effectiveness of our reconstruction method on noisy input curves. Lastly, we evaluated our method on hand-drawn curves. In all experiments, we compared the 3D reconstructed curves generated using various models including SVR, RFR, KNN-R, GPR, ED-CNN, and ANN. The results of the GBR model are not presented due to its poor performance. For each reconstruction, we provide visual comparisons before and after applying the post-processing smoothing algorithm. To quantify the reconstruction quality, we used the error parameter $E_{\text{cost}(P_{3D})} = \sigma^2(\kappa_{P_{3D}})$, where P_{3D} is the reconstructed curve.

3.6.1 Results using synthetic input data

We test our approach using synthetic input data, which is the orthogonal projection of true circular helices. Figure 3.19, Figure 3.20, and Figure 3.21 shows the reconstructed curves generated using SVR, RFR, KNN-R, GPR, ED-CNN, and ANN models before and after applying the post-processing smoothing algorithm in (a) and (b) respectively. The results show that all ML algorithms provided visually accepted results. However, it is noteworthy that KNN-R and ANN consistently achieved the best reconstruction performance based on the error parameter

$E_{\text{cost}(P_{3D})} = \sigma^2(\kappa_{(P_{3D})})$, which quantifies the variance of the curvature, an important characteristic of the circular helix. Figure 3.22 shows the curvature of the reconstructed 3D circular helices using KNN-R algorithm shown in Figure 3.19, Figure 3.20, and Figure 3.21 respectively.

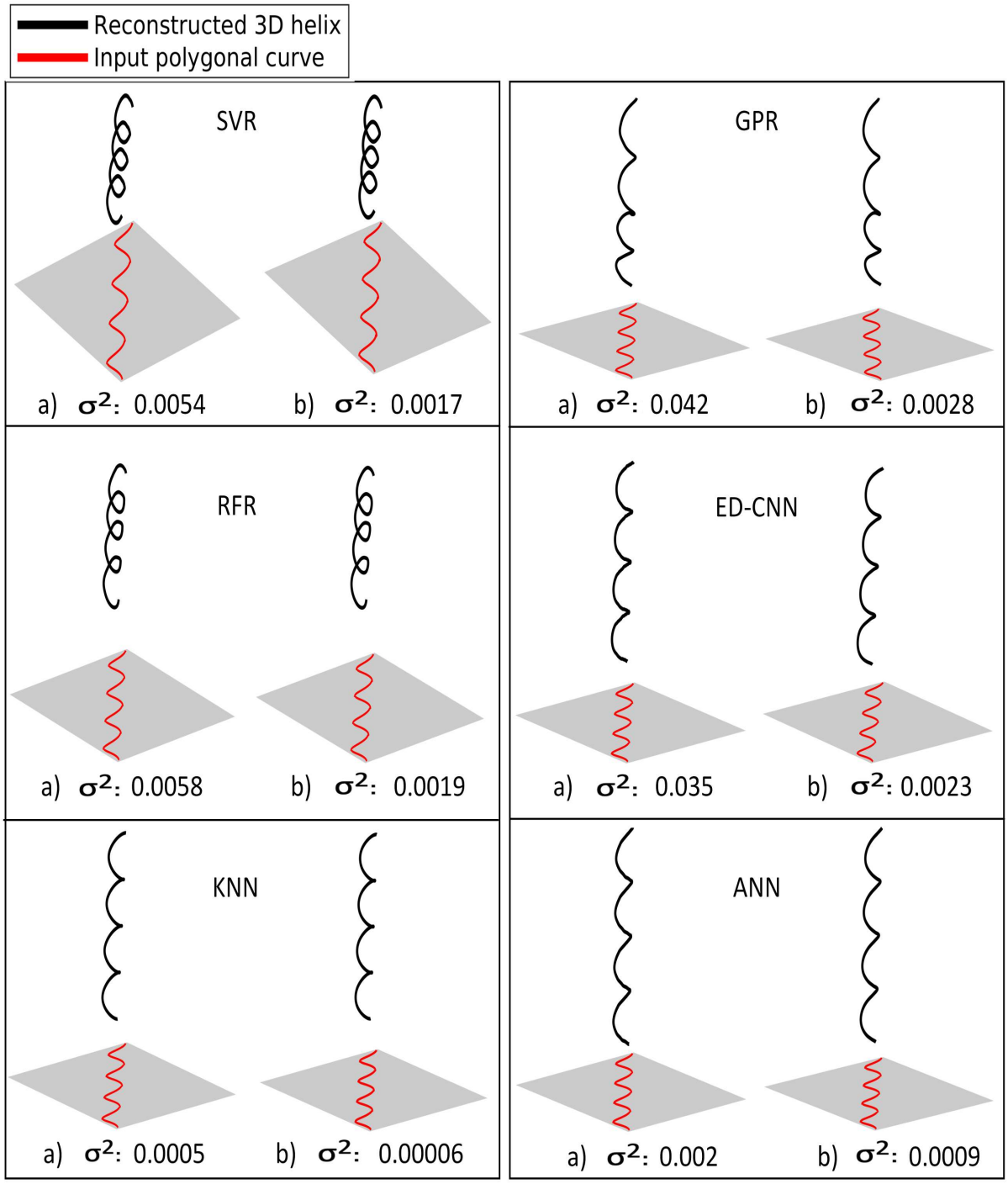

Figure 3.19: The reconstructed 3D circular helix using the orthogonal projection of synthetic circular helix before and after applying the post-processing smoothing algorithm in (a) and (b) respectively. σ^2 is the variance of curvature of each reconstructed curve.

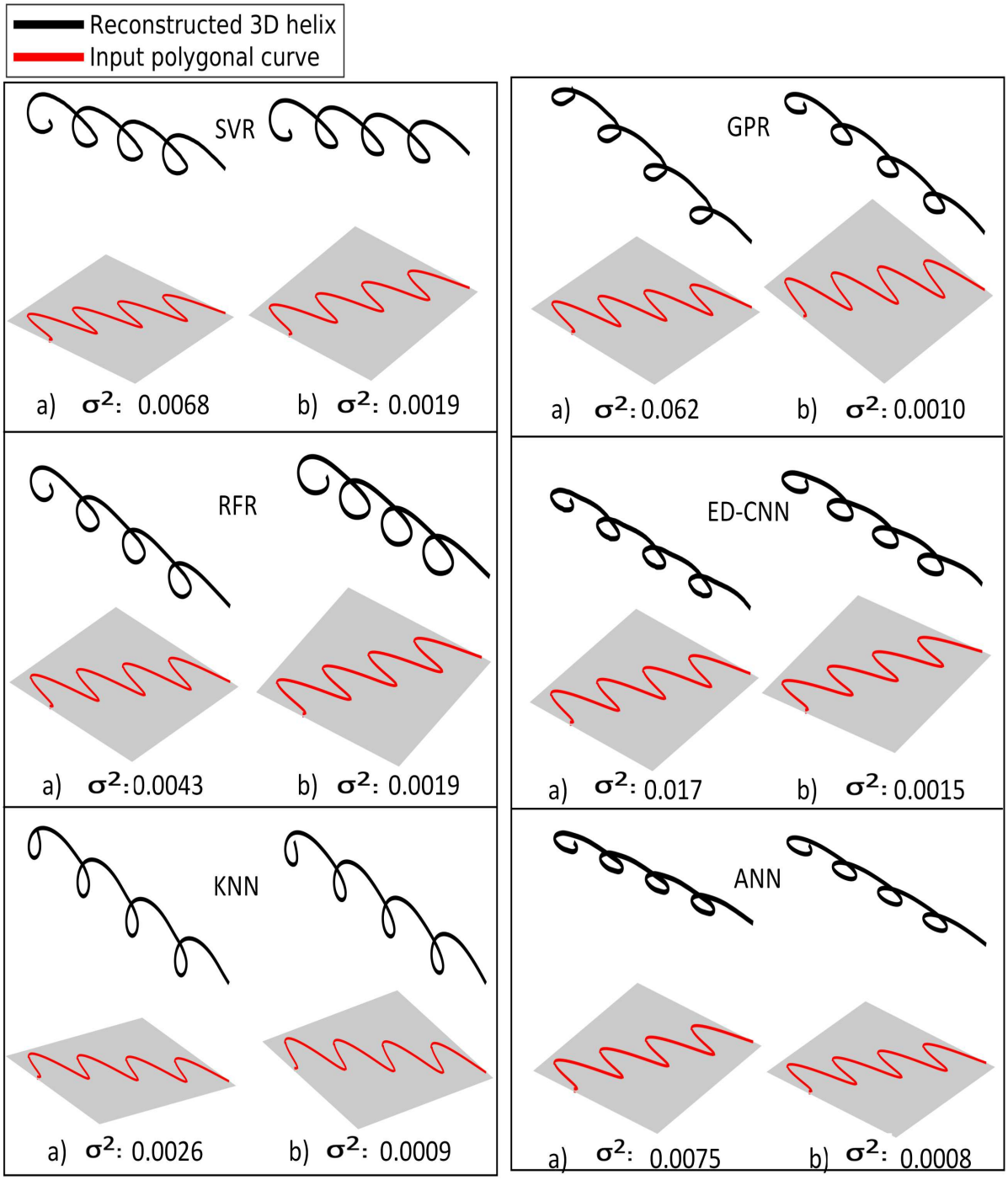

Figure 3.20: The reconstructed 3D circular helix using the orthogonal projection of synthetic circular helix before and after applying the post-processing smoothing algorithm in (a) and (b) respectively. σ^2 is the variance of curvature of each reconstructed curve.

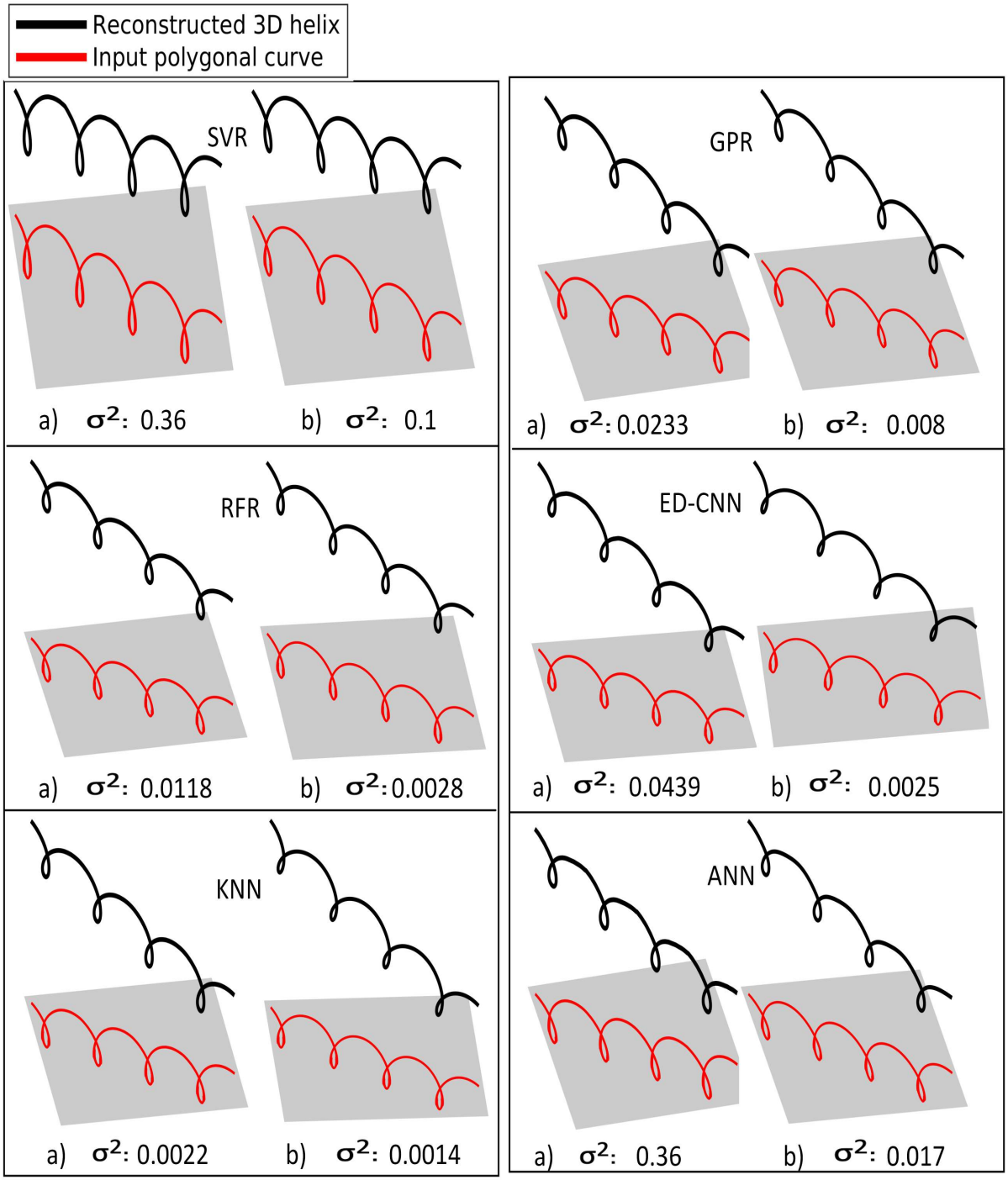

Figure 3.21: The reconstructed 3D circular helix using the orthogonal projection of synthetic circular helix before and after applying the post-processing smoothing algorithm in (a) and (b) respectively. σ^2 is the variance of curvature of each reconstructed curve.

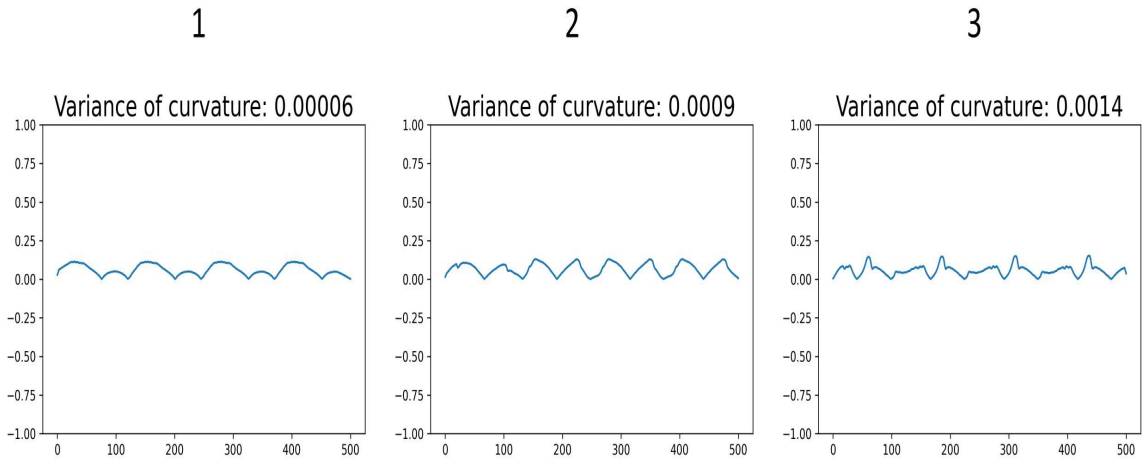

Figure 3.22: (1), (2), and (3) show the curvature of the reconstructed 3D circular helices using KNN-R algorithm shown in Figure 3.19, Figure 3.20, and Figure 3.21 respectively.

3.6.2 Results using hand-drawn input data

To evaluate our approach against real-life input data, we test it using hand-drawn curves. Figure 3.23, and Figure 3.24 shows the reconstructed curves generated using SVR, RFR, KNN-R, GPR, ED-CNN, and ANN models before and after applying the post-processing smoothing algorithm in (a) and (b) respectively. The results demonstrate that all ML algorithms provided visually accepted results except SVR in Figure 3.24. Once again, KNN-R and ANN consistently outperformed other models in terms of the error parameter $E_{\text{cost}(P_{3D})} = \sigma^2(\kappa_{(P_{3D})})$. It is important to note that the viewpoint of the reconstructed curve may differ among the models in both figures, as we selected the viewpoint that offered the best visualization of the reconstructed curve for each model. Figure 3.25 shows the curvature of the reconstructed curves using KNN-R algorithm shown in Figure 3.23, and Figure 3.24 respectively.

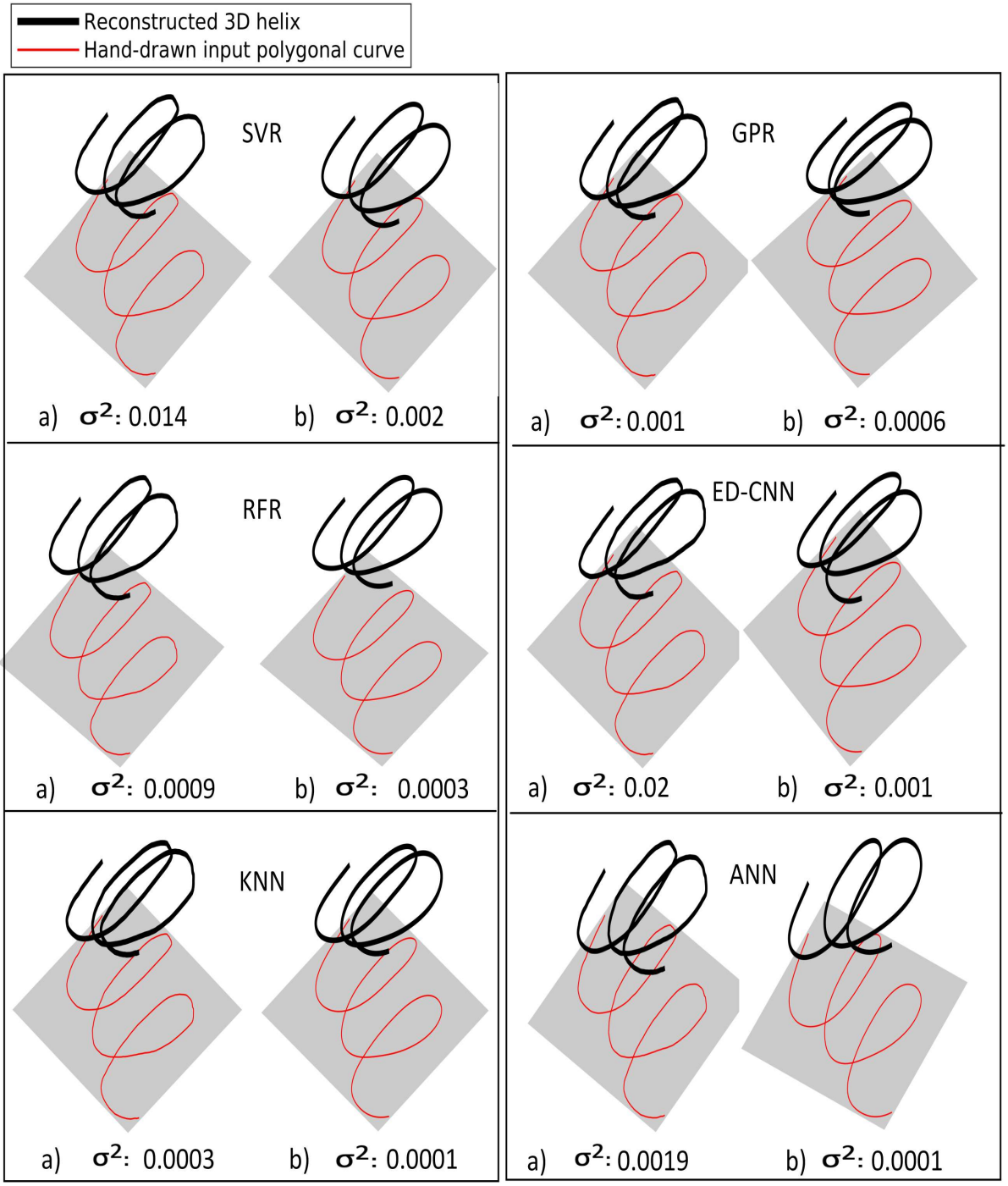

Figure 3.23: The reconstructed 3D circular helix using hand-drawn curve before and after applying the post-processing smoothing algorithm in (a) and (b) respectively. σ^2 is the variance of curvature of each reconstructed curve.

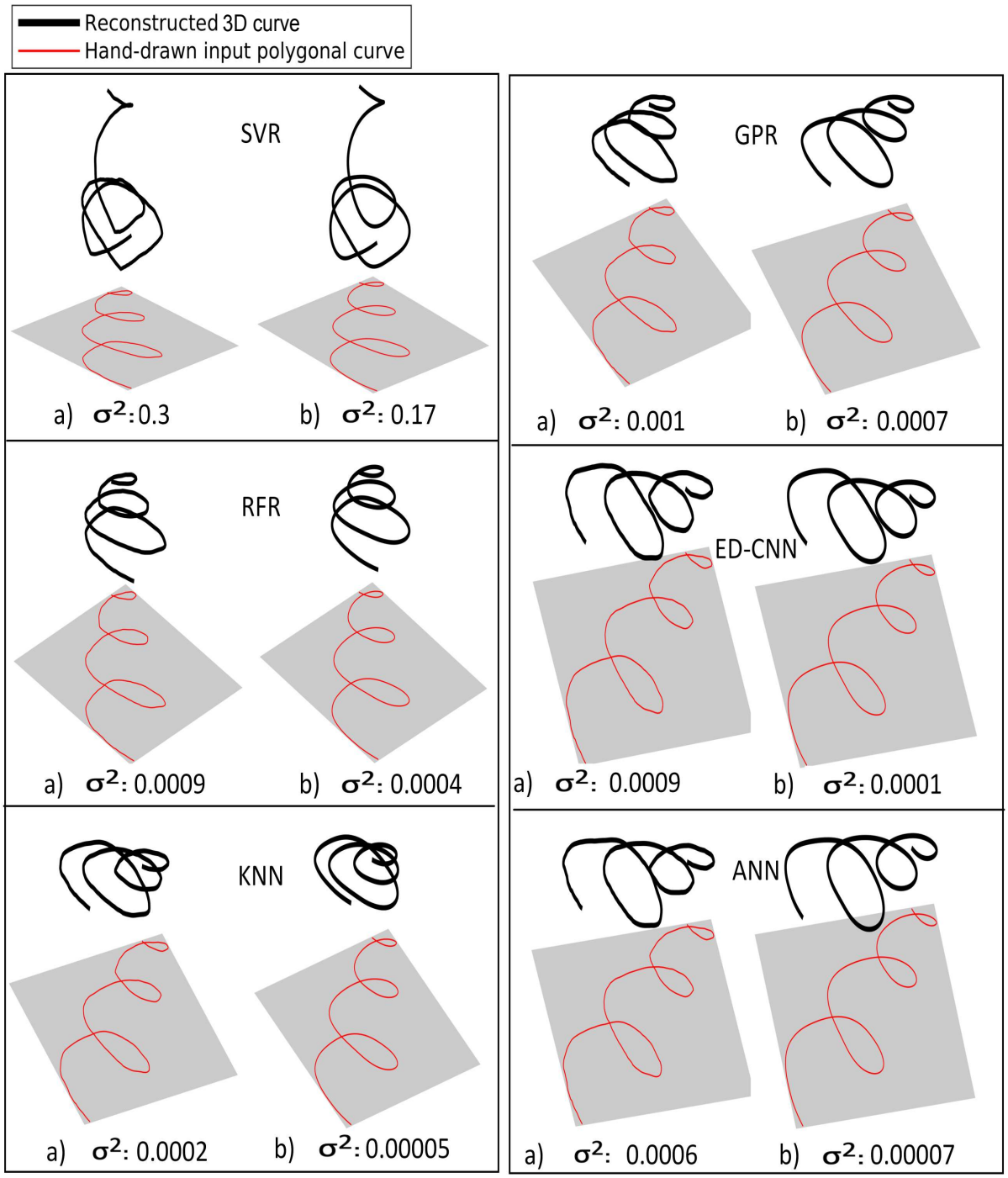

Figure 3.24: The reconstructed 3D curve using hand-drawn curve before and after applying the post-processing smoothing algorithm in (a) and (b) respectively. σ^2 is the variance of curvature of each reconstructed curve.

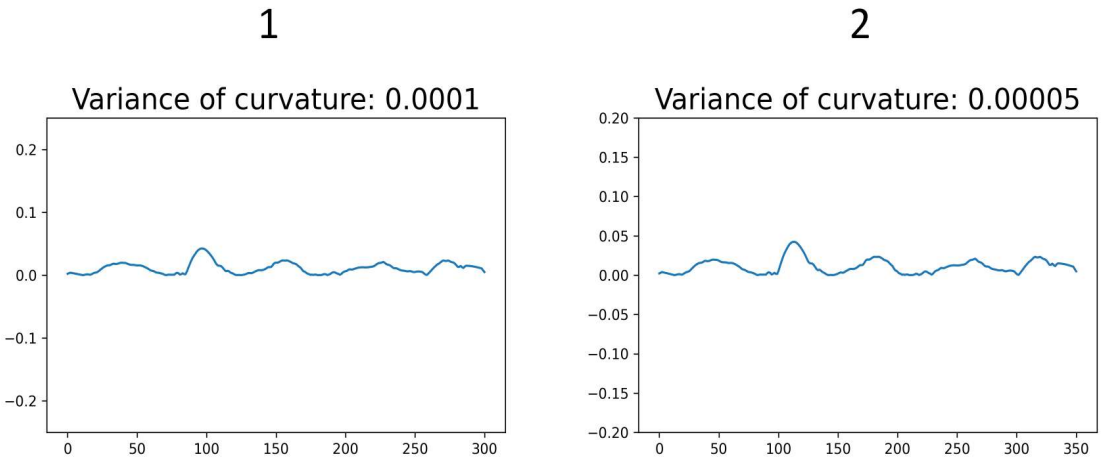

Figure 3.25: (1), and (2) show the curvature of the reconstructed 3D curves using KNN-R algorithm shown in Figure 3.23, and Figure 3.24 respectively.

3.7 Limitations

While our approach demonstrates promising results, it does have some limitations that should be considered:

Firstly, our methods were able to generate approximate and not exact circular helix. The curvature and torsion of our reconstructed curves were not strictly constant with arc length. Therefore previous work such as (Cherin et al. 2014, Cordier et al. 2016) produced a better 3D reconstruction of Euler spiral. However, our primary contribution lies in the evaluation of ML models for the 3D reconstruction of circular helices.

Additionally, our reconstruction method may encounter difficulties when dealing with specific types of curves. Straight-line curves and curves with sharp corners pose challenges to the accuracy of our models, as they deviate significantly from the true helical shape that the models are trained on. Therefore, the performance of our approach may be compromised when applied to such curves.

Another limitation lies in the segmentation of the input curve. Currently,

we divide the curve into segments and reconstruct each segment independently before assembling the reconstructed curves together to form the approximate reconstructed 3D circular helix. This segmentation strategy can introduce discontinuities in the form of C^1 and C^2 discontinuities between the reconstructed curves. Consequently, the smoothness and continuity of the overall reconstructed circular helix may be affected, particularly at the connecting points of the segments. One possible solution is to reconstruct multiple curves for each input segment. Therefore, in the end, we form the reconstructed 3D circular helix as the assembling of reconstructed curves, one per input curve. The selection of these curves will be based on their continuity with the previous and next curves.

3.8 Conclusion

This chapter presents an extensive study on the application of ML algorithms for the 3D reconstruction of circular helices from planar polygonal curves. Various ML algorithms were described and trained specifically for circular helix reconstruction through careful algorithm selection and hyperparameter tuning. The proposed approach was evaluated using polygonal curves representing the orthogonal projection of true 3D circular helices as well as hand-drawn curves. The results demonstrate that most ML algorithms achieved visually and systematically acceptable reconstructions, with a notable performance by algorithms like KNN-R and ANN, while SVR exhibited a weaker performance.

In conclusion, with the recent success of ML algorithms in many domains, it was quite interesting to test such algorithms for the 3D reconstruction of circular helices. The results show the capability of these algorithms to produce nice approximately reconstructed 3D circular helices.

Chapter 4

3D reconstruction of curves

4.1 Introduction

In the previous chapter, ML models, particularly ANN and KNN-R, demonstrated their effectiveness in reconstructing an approximate 3D helix from planar polygonal curves. However, the simplicity of the helix curve, with its constant curvature and torsion along its arc-length, played a crucial role in the success of ML models for helix reconstruction. Nevertheless, when confronted with more complex curve families like Euler spiral curves, which exhibit linearly evolving curvature and torsion, or free-form curves, these models could not provide good reconstruction. Another limitation of ML models is that they can only produce one reconstruction for each segment of the input curves. This limitation prevents us from imposing curvature and torsion continuity constraints during the assembly process of the 3D reconstructed pieces. It is important to note that a polygonal curve in 2D can correspond to multiple curves in the 3D space.

Motivated by the aforementioned limitations, we explore alternative methods to tackle the challenges posed by more complex families of curves. In this chapter, we present a novel approach for reconstructing 3D Euler spirals using a curve-matching technique in a piecewise manner. Additionally, we introduce a more generalized method capable of reconstructing various curve types. This method is based on the idea of fitting a set of ellipses to the input curve, which enables us to determine the osculating circles and, subsequently, the tangent at each point of

the curve, facilitating the 3D reconstruction process.

4.2 3D reconstruction of Euler spiral curves

This section presents our published work (Ali Fakih 2023), where we introduced a novel approach for reconstructing 3D Euler spiral curves using a piecewise curve-matching technique. Given a 2D input curve that we have to perform its 3D reconstruction. We first divide it into smaller pieces. For each piece, we search for the closest matches from a dataset we created that contains pieces of Euler spirals. Then, we connect the matched pieces and apply a smoothing algorithm to generate the reconstructed piecewise 3D Euler spiral curve. For additional results of our reconstruction in this section, please refer to Appendix A.

International Journal of Computer Science, Engineering and Information Technology (IJCSEIT),
Vol.13, No.5, October 2023

PIECEWISE RECONSTRUCTION OF 3D EULER SPIRALS FROM PLANAR POLYGONAL CURVES

Ali Fakih, Frederic Cordier and Yvan Maillot

IRIMAS, EA 7499, Université de Haute Alsace, Mulhouse, France

ABSTRACT

In this article, we propose a method for reconstructing approximate piecewise 3D Euler spirals from planar polygonal curves. The method computes the 3D coordinates of approximate Euler spiral such that its orthogonal projection onto the 2D plane is the closest possible to the input curve. To achieve this, a dataset is created, comprising Euler spiral segments and their orthogonal projections. Given an input curve, it is sampled and split into segments. Each segment is matched with the closest Euler spiral segments from the dataset, forming a pool of candidates. The optimal set of connected Euler spiral segments is then selected to reconstruct the approximate piecewise 3D Euler spiral. The selection prioritizes smoothness continuity at connecting points while minimizing the distance between the orthogonal projection and the input curve. We evaluate our method against synthetic 3D Euler spirals by applying our reconstruction to the orthogonal projection of the synthetic Euler spirals.

KEYWORDS

Euler spiral, 3D reconstruction, piecewise reconstruction.

1. INTRODUCTION

3D Euler spirals are aesthetically pleasing curves whose curvature and torsion evolve linearly with arc length [1]–[3]. They possess desirable properties, such as invariance to similarity transformations (translation, rotation, and scaling), symmetry, extensibility, and smoothness [4].
In this work, we focus on the 3D reconstruction of Euler spirals from planar polygonal curves which can be used in different applications since there are many domains in which the 3D Euler spirals are useful like aerospace vehicles (by creating smooth transitions between different flight regimes), turbomachinery (by creating smooth, continuously curved flow paths), medical devices (to design medical devices that need smooth and curved paths through the body like catheters and endoscopes).

Despite the presence of all the mentioned properties, to the best of our knowledge, there is no previous work in the literature that aims to reconstruct the 3D Euler spirals from planar polygonal curves. The goal of this paper is to fill this gap by proposing an algorithm that reconstructs piecewise approximate 3D Euler spirals from planar polygonal curves. The algorithm takes a polygonal curve in the plane $z = 0$ as input and produces an approximate piecewise 3D Euler spiral whose orthogonal projection on the plane $z = 0$ is close to the input polygonal curve, as illustrated in Figure *1*.

This paper is organized as follows. Section 2 presents the related work to our approach. The overview of our approach is introduced in 3. In Section 4, we describe how we create our dataset. Section 5 presents the algorithm we use to split the input polygonal curve and search the dataset for the closest matched 3D Euler spiral segments. In Section 6, we explain the algorithm to select

Dhinaharan Nagamalai (Eds): CGASP, VCOI, BLKCA - 2023
pp. 01-19, 2023. IJCSEIT– 2023 DOI : 10.5121/ijcseit.2023.13501

International Journal of Computer Science, Engineering and Information Technology (IJCSEIT),
Vol.13, No.5, October 2023

and assemble the segments from the dataset to form the piecewise reconstructed 3D Euler spiral. The results of our method are presented in Section 7. Finally, in Section 8, we conclude this work and suggest some future research directions.

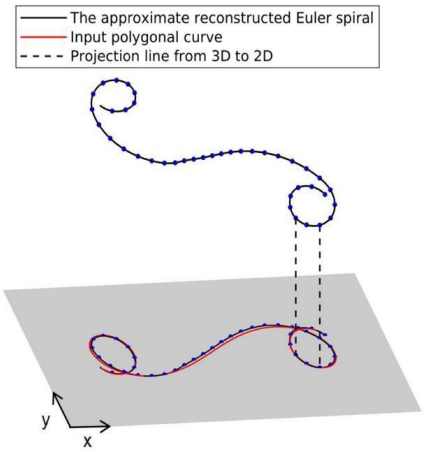

Figure 1: Starting from a planar polygonal curve (red curve) as an input we generate an approximation piecewise 3D Euler spiral, whose orthogonal projection on the plane z = 0 is close to the input polygonal curve.

2. RELATED WORK

As mentioned before, there is no previous work in the literature that focuses on reconstructing 3D Euler spirals from planar polygonal curves. However, there are three categories of works that are relevant to ours. First, we review the previous work on sketch-based modeling. Next, we discuss the reconstruction of 3D circular helices from their planar orthogonal projection; circular helices are special case of 3D Euler spiral. Finally, we also review the work done on the 2D and 3D Euler spiral and its applications.

2.1. Sketch-based modeling

The idea behind the sketch-based modeling is to start from a drawn shape in the (x,y) plane which is composed of lines, and then try to reconstruct the 3D shape whose projection onto the (x,y) plane matches the input sketch. Some of the first reconstruction methods were done by solving an optimization whose unknown variables are the third coordinates of the input sketch references [5], [6] but the main drawback was that these methods only deal with rectilinear shapes.

With the rise of machine learning algorithms, several researchers have worked on solving the sketch-based modeling problem using deep learning [7]–[11] but these algorithms are tailored to specific shapes, such as faces, cars, and chairs, making them limited in their applicability. We refer the reader to the state-of-the-art paper [12] for more details about sketch-based modeling using deep learning.

International Journal of Computer Science, Engineering and Information Technology (IJCSEIT),
Vol.13, No.5, October 2023

2.2. The reconstruction of 3D circular helices from their planar orthogonal projection

Circular helices are a special case of the 3D Euler spiral, sharing the property that their curvature and torsion evolve linearly. However, the Euler spiral's curvature and torsion can vary along its arc length, while the curvature and torsion of the circular helix are constant. There are multiple previous works done on the reconstruction of 3D helices from their orthogonal projection [13]–[15] and they achieved good results by applying optimization algorithms but the limitation of these works is that they are also problem specific and only work for the helix shape.

2.3. 2D and 3D Euler spirals

The 2D Euler spiral, also known as the Clothoid or Cornu spiral, is a curve whose curvature evolves linearly with arc length. It was independently discovered by several researchers including Bernoulli, Euler, and Talbot [16]. Many researchers have used 2D Euler spirals in computer-aided design. For example, in [17] they used two spirals to connect successive control polygon points in the form of parabola-like. The work most closely related to ours is that of [18]–[20]. In their work, they also reconstructed Clothoid splines in a piecewise manner from polygonal curves and achieved good results. However, their method is limited to 2D Clothoids and cannot be used for 3D reconstruction, as opposed to our method which produces a piecewise 3D Euler spiral that fits a planar polygonal curve.

The 3D Euler spiral is the curve whose curvature and torsion evolve linearly with arc length [4]. Several works have been done on the generation of 3D Euler spirals. In [21], the authors aimed to generate a 3D Euler spiral, starting by refining a polygon, such that the polygon satisfies the linearity evolution of the curvature along the arc but they ignored the torsion. In [22], the authors defined the closed form parametrization of 3D Euler spirals by modeling the problem as a linear time-variant system and studied its stability with Lyapunov techniques [23]. Their most significant achievement was defining the closed form of the 3D Euler spiral in terms of the standard Fresnel integrals that satisfy the property of both curvature and torsion evolving linearly with arc length. In [4], the authors extend the Euler spirals from 2D to 3D by solving an optimization problem and proving many of their properties, including their invariance to similarity transformation (translation, rotation, and scaling), symmetry, extensibility, smoothness, and roundness. Furthermore, in this article, they used the 3D Euler spiral for the archaeological reconstruction such as the completion of the shape of some broken ancient sculptures and objects. However, the proposed algorithm in this paper only works in 3D space, thus it cannot be used to create 3D Euler spirals from planar polygonal curves.

3. OVERVIEW

The purpose of our method is to reconstruct approximate piecewise 3D Euler spirals from planar polygonal curves. The input of our method is a 2D polygonal curve in the (x, y) plane and the output is a piecewise 3D Euler spiral such that its orthogonal projection onto the (x, y) plane is close to the input polygonal curve.

The common approach for curve matching is by using the equation that computes the coordinates of the points along the curve through the arc length parameterization. However, this form does not exist for the 3D Euler spiral. Alternatively, point coordinates can be computed through optimization, but this would require a large computation time.Instead of applying the typical curve matching algorithms, we create a dataset that contains short segments of 3D Euler spirals with their planar polygonal projection. To enable comprehensive coverage of the diverse shapes

International Journal of Computer Science, Engineering and Information Technology (IJCSEIT),
Vol.13, No.5, October 2023

that may be encountered in the input polygonal curve, we make the dataset have segments of various lengths and we apply different rotations them in addition to performing a uniform sampling and scaling to enhance the representation of the segment and to remove the scaling factor from our dataset.

In that direction, the reconstruction starts by splitting the input 2D polygonal curve Sinto segments $\{s_1,\ s_2,\ \dots\ \}$; the splitting is based on the dichotomy approach [24] to find a number of matched 3D Euler spiral segments from the dataset for each segment of the input curve. $C_{si}=\{c_{i,1},\ c_{i,2},\ \dots\ \}$ represents the set of candidate matched segments for segment s_i. At this point for each segment of the input curve, we have a pool of possible matched Euler spiral segments (candidates) from the dataset. We implement a new algorithm that selects the optimal connection of candidates (one candidate from each candidate pool) using the Dijkstra algorithm [25]. The selection of any two candidates that will link with each other is based on multiple criteria such as their curvature continuity, torsion continuity, tangent continuity, normal continuity, and how far their polygonal projection is from the input curve. The result of this step is a piecewise Euler spiral represented in the form of a 3D polygonal curve. A smoothing is then applied on this polygonal curve to improve the C^0 and C^1-continuity in order to obtain a reconstructed curve that is more eye-pleasing.

4. DATASET

We create a dataset composed of 1,400,832 short 3D Euler spiral segments, with each segment represented by a polygonal curve composed of 30 points in the (x, y, z) plane. The coordinates of the 30 points together with the properties of the Euler spiral segment such as its curvature, torsion, tangent, and normal values are stored in a text file. The overall size of the dataset is around 1.2 GB. We decide to represent each segment with 30 points. A larger number of points would provide a better representation of the segments, but would also result in an excessively large dataset. On the other hand, a smaller number of points would cause the loss of important details and features. We determine that representing our segments with 30 points provides the best balance between a good curve representation and a manageable dataset size.

The orthogonal projection of each segment onto the (x, y) plane is obtained by removing the z coordinate from all points that form the segment. This orthogonal projection will be used to find the matched segments to the input 2D polygonal curve. The size of the dataset is quite large since it contains 3D Euler spiral segments of varying starting points, lengths, and orientations. This is to ensure that the dataset covers all possible cases for the reconstruction of any arbitrary curve.

Additionally, we apply uniform sampling to all dataset segments to ensure consistent representation and standardize the comparison between them and the input curve. Finally, we apply uniform scaling to all segments to remove the scale factor from our dataset and facilitate dealing with input curves of any scale. Figure *2* shows some segments of the dataset.

International Journal of Computer Science, Engineering and Information Technology (IJCSEIT),
Vol.13, No.5, October 2023

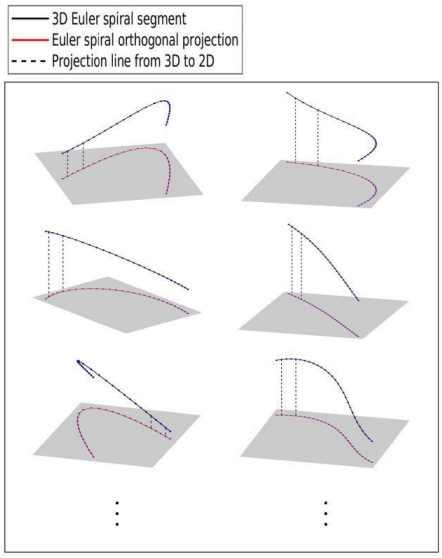

Figure 2: Visualization of our dataset. Segments of 3D Euler spirals with their planar polygonal projection.

4.1. Dataset creation

In order to generate our dataset which is composed of short segments of 3D Euler spirals, we follow the approach introduced in [4] where they define the 3D Euler spiral as the curve that should minimize the sum of the square variation of its curvature κ and torsion τ, therefore minimizing the following integral:

$$(\kappa,\tau) = \int_0^L [\kappa_s^2(s) + \tau_s^2(s)]\, Sds, \quad (1)$$

where $L$is the arc length, $s\in[0,L]$, $\kappa_s = {}_{\partial\kappa}\_\_{}_{\partial s}$,and $\kappa_\tau = {}^{\partial\tau}{}_{\partial s}\_\_$. Minimizing Equation 1 using the Euler-Lagrange equation leads to a curve whose curvature and torsion evolve linearly. Thus, for some constants $\kappa_0, \tau_0, \gamma, \delta\in R$, and for $0 \le s \le L$:

$$\kappa(s) = \kappa_0 + \gamma s,$$
$$\tau(s) = \tau_0 + \delta s, \quad (2)$$

In order to ensure that our dataset adequately covers the diverse range of input polygonal curves, we make the starting point parameter of the dataset segment to have different values along the Euler spiral. To achieve this, we uniformly sample the starting point parameters along the curve in the interval $s_s\in[s_{s,a},\ s_{s,b}]$ as illustrated in Figure *3* where $s_{s,i}$ is the starting point parameter of the segment i of the dataset. By doing so, we ensure that our dataset includes segments whose curvature changes from negative to a positive value, under the assumption that the curvature is negative at $s_{s,a}$ and positive at $s_{s,b}$. We select $s_{s,}$ as the upper limit starting point parameter since as depicted in Figure *3*, the curve exhibits a repeating pattern and is getting similar to a circular shape beyond that point.

International Journal of Computer Science, Engineering and Information Technology (IJCSEIT),
Vol.13, No.5, October 2023

Concerning the length of the segments, we choose to make the segments short because long segments would require a larger dataset to cover all possible input curves. We achieve this by limiting the tangent angle difference between the starting point parameter $\vec{T}_{s}$, and the ending point parameter $\vec{T}_{e,i}$ to a value within the range of $[\pi_{-4}, \pi]$. This ensures that the ending point parameter $s_{e,i}$ of the segment i that have a starting point parameter $s_{s,i}$ will be $s_{e,i} \in [s_{e,a}, s_{e,b}]$ as illustrated in Figure 4.

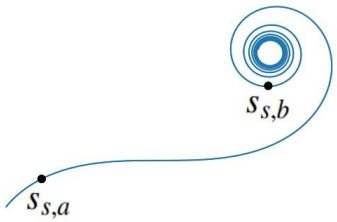

Figure 3: The starting point of all dataset segments is between $s_s \in [s_{s,a}, s_{s,b}]$.

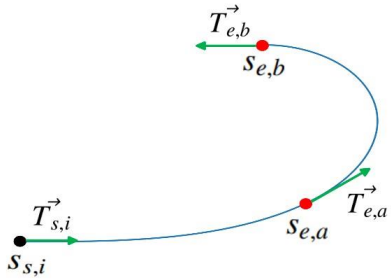

Figure 4: The ending point parameter of segment i that have a starting point s_s is between $[s_e, a, s_e, b]$.

To represent all possible projections of the 3D Euler spiral segments, we apply a rotation along x, y, and z axis for each segment of the dataset. Let M_i, R_x, $R_{y,i}$, and $R_{z,i}$ be respectively the matrix that represents the point's coordinates of the segment i of the dataset, the rotation matrices along the x, y, and z-axis:

$$M_i = \begin{pmatrix} x_{i,1} & y_{i,1} & z_{i,1} \\ x_{i,2} & y_{i,2} & z_{i,2} \\ \vdots & \vdots & \vdots \\ x_{i,30} & y_{i,30} & z_{i,30} \end{pmatrix}, \quad R_{x,i}(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{pmatrix}$$

$$R_{y,i}(\beta) = \begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{pmatrix}, \quad R_{z,i}(\lambda) = \begin{pmatrix} \cos(\lambda) & -\sin(\lambda) & 0 \\ \sin(\lambda) & \cos(\lambda) & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

International Journal of Computer Science, Engineering and Information Technology (IJCSEIT),
Vol.13, No.5, October 2023

where α, β, and $\lambda$are respectively the rotation angle along x, y, and z axis. These rotation angles are uniformly sampled in the range $[0,2\pi[$ in order to cover all possible rotation angles. The rotated 3D Euler spiral segment c_i is defined as:

$$ci = MiRx,i(\alpha)Ry,i(\beta)Rz,i(\lambda).$$

4.2. Uniform scaling and sampling

Our dataset is composed of 3D Euler spiral segments. After splitting the input polygonal curve into segments, for each input segment, we look for its closest matched Euler spiral segments in the dataset. Since the input curve is in 2D, the matched segments should be in 2D as well. To achieve this, we orthogonally project our dataset segments onto the (x,y) plane.

To establish a systematic search approach, we take advantage of the 3D Euler spiral's property of invariance to similarity transformations (translation, rotation, and scaling) proved in [4]. We apply a uniform scaling for all segments in our dataset, making them all have the same length while preserving their original aspect ratio. This removes the scaling factor from our dataset and allows us to deal with any segment of the input polygonal curve, regardless of its length. Therefore, when we split the input polygonal curve into segments, we first scale each segment to the same length as dataset segments, then we search for its closest matched Euler spiral segments in the dataset.

The search for the closest matched segments for each input curve segment is based on the 2D Fréchet distance [26] between the input segment and the orthogonal projection of all our dataset segments. To do this, we require both input segments and the dataset segments to have the same number of points, uniformly sampled at regular intervals along the arc length. This ensures that the distance between any two consecutive points is equal in both, enabling a fair comparison. We use the Fréchet distance as a metric to compare curves because it takes into account both the spatial location and ordering of points along the curves, which makes it more suitable than other distance metrics.

To efficiently store and search for the closest matching 3D Euler spiral segments in the dataset, we store the dataset in a K-d tree [27] (k-dimensional tree). K-d tree is a data structure that is used for efficient multidimensional search operations, particularly for finding the nearest neighbors of a given point in a space of any number of dimensions. It works by recursively dividing the search space into smaller regions called hyperrectangles, which are represented by nodes in the tree. In our case, each node in the tree represents a 3D Euler spiral segment from the dataset. By using a K-d tree to store our dataset, we can perform fast and accurate nearest neighbor searches to find the closest matching segments to a given input curve segment.

5. CURVE SEGMENTATION

The input of our method is a planar polygonal curve as: $S = \{p_1, p_2, \ldots, p_i, \ldots, p_n\}$, where p_i is the coordinates of the point of index i in the curve and n the number of points of that curve. To perform the approximate piecewise 3D Euler spiral reconstruction from the input curve, we first split it into segments, with each segment starting from the point that follows the last point of the previous segment. We sample and scale each segment uniformly, similarly to how we processed the dataset segments. Next, we search for the closest matching Euler spiral segments from the dataset for each input curve segment.

International Journal of Computer Science, Engineering and Information Technology (IJCSEIT),
Vol.13, No.5, October 2023

During the segmentation, our purpose is to decompose the curve into the smallest possible number of segments; this is to avoid unnecessary computation time due to handling more segments than needed. At the end of the segmentation, each segment of the input curve should have at least N matched segments from the dataset such that the Fréchet distance between them and the input segment is less than a threshold T, where N and T are user-defined parameters. Additionally, we search for the set of the closest N matched segments (candidates) for each input segment and not only the closest segment since in many cases the planar orthogonal projection of two curves could be similar while their shape and orientation are quite different in 3D space.

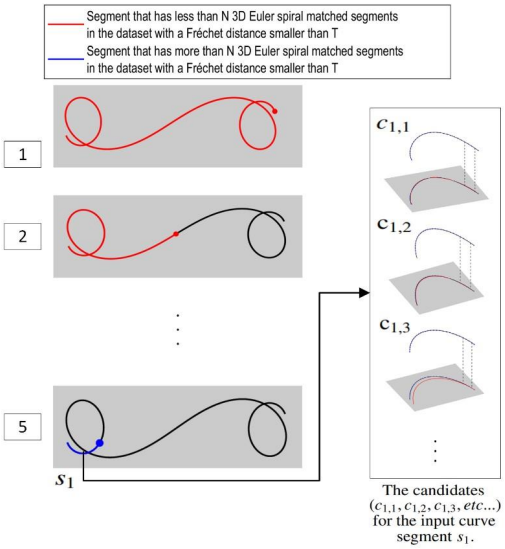

Figure 5: The process to find the first split segment of the input curve. In (1), the segmentation algorithm starts with the entire curve. In (2), the segmentation is done recursively until the number of matching candidates is larger or equal to N. In (5), the s the segmentation is finished for the first segment S_1.

The segmentation begins by taking the entire input curve as the first input segment. This segment is uniformly scaled and sampled in the same way as for the dataset segments, then we search for its closest matched Euler spiral segments in the dataset stored in the K-d tree. If the number of matched segments whose Fréchet distance to the input segment is less than a threshold T exceeds a certain number N then the segmentation process is done. Otherwise, we recursively split the segment into half (inspired by the dichotomy approach) and take its first half to repeat the matching process until satisfying the previous condition. Figure *5* illustrates the recursive segmentation to find the first segment of the input curve. We repeat the same process for the remaining part of the input curve until it is fully covered.

Noting that we opted not to use a dynamic programming approach for segmentation, as it is more computationally expensive. Instead, our approach recursively splits the input curve in half, motivated by the fact that our dataset segments have various ending point positions, as shown in Figure *4*. Furthermore, for each segment of the input curve, we consider multiple matched segments from the dataset, rather than just the closest match, to ensure that our segmentation approach is both fast and effective.

International Journal of Computer Science, Engineering and Information Technology (IJCSEIT),
Vol.13, No.5, October 2023

In the pseudocode below the function $spli(\ )$ defines the boundaries of the split segment by splitting the input polygonal curve S from p_{v1} to p_{v2}. p_{v1} and p_{v2} are respectively the points of vertex indices v_1 and v_2 of the input polygonal curve S with $v_1 < v_2$. The function $ScalingAndSamplin(\ )$ applies uniform scaling and sampling in the same way we did for our dataset. This means that the sampled input segments will have the same number of points as the dataset segments. The function $GetMatchingCandidates(\ )$ takes a given segment with a t as input, then returns the closest matched Euler spiral segments from the dataset whose Fréchet distance to the segment is smaller than T. Finally, the function $AddItemsToLis(\ )$, adds the closest matched segments to a list which will be the result of this algorithm. This algorithm stops when the starting point index v_1 exceeds n indicating that the entire input polygonal curve has been processed.

Algorithm 1: $Segmentation\ a\lg orithm$

Input: S ▷ The input polygonal curve
T ▷ Frechet distance threshold
N ▷ The minimum number of matched segments
n ▷ The number of points of S
v_1 1 ▷ The starting point of the split segment
$v_2 = n$ ▷ The ending point of the split segment $Results$ = [] ▷ The result of the algorithm

Output: $Results$

Function: $Segmentation(S, T, N, n, v_1, v_2, Results)$:

if $v_1 \geq n$ **then**
return Results
end
$segment = spli(S, v_1, v_2)$
$segment = ScalingAndSampling(segment)$
$matchCand = GetMatchingCandidate(segment, T)$ **if** $sizeOf(matchCand) \geq N$ **then**
$AddItemsToList(Results, matchCand)$
$v_1 = v_2 + 1$
$v_2 = n$
else
$v_2 = v_2/2$
end
return $Segmentation(S, T, N, n, v_1, v_2, Results)$

6. CONNECTING THE SEGMENTS

To build the piecewise reconstructed 3D Euler spiral, we need to first find the optimal assembling of the candidates for each split segment. Once the optimal connecting is found, a post-processing step is required to reduce the discontinuity of the reconstructed curve at the connection points.

International Journal of Computer Science, Engineering and Information Technology (IJCSEIT),
Vol.13, No.5, October 2023

6.1. Finding the optimal connecting of the candidates

The splitting and matching algorithm applied to the input polygonal curve S results in its partition into m segments $\{s_1, s_2, \dots, s_m\}$, each has Nmatched candidate segments obtained from the dataset. Specifically, $C_{si}=\{c_{i,1}, c_{i,2}, \dots, c_{i,N}\}$ represents the set of N candidate for the segment s_i, where $1 \leq i \leq m$. Each one of these candidates is a 3D Euler spiral curve segment that has its own properties, such as its curvature, torsion, tangent, and normal values.

For each segment s_i we have N candidates. The final approximate Euler spiral reconstructed curve R_S is obtained by selecting one candidate per input segment s_i and connecting these selected candidates. Specifically, R_S is represented as the set of selected candidates:

$$R_S=\{c_{i,1}, c_{i,2}, \dots, c_{i,N}\},$$

where $1 \leq t \leq m$, $c_{1,j} \in C_{s1}$, $c_{2,k} \in C_{s2}$, $c_{t,l} \in C_{st}$, and $c_{m,f} \in C_{sm}$.

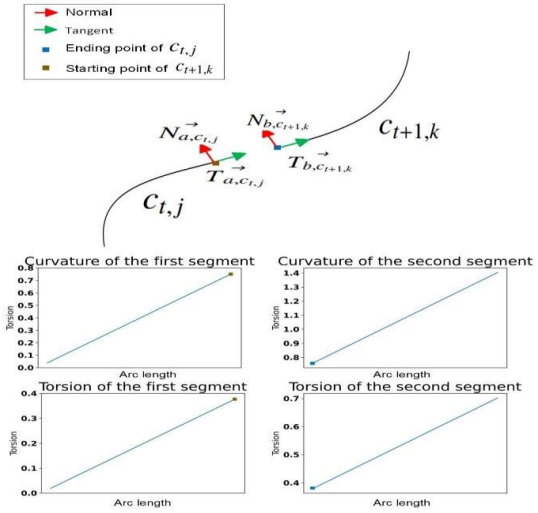

Figure 6: Visualization of the parameters that we consider while assembling two 3D Euler spiral curve segments.

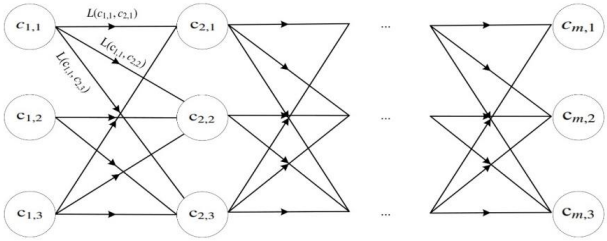

Figure 7: The architecture of the Dijkstra algorithm with $N = 3$, the nodes represent the segment candidates and the edges represent the loss function between the candidates.

International Journal of Computer Science, Engineering and Information Technology (IJCSEIT),
Vol.13, No.5, October 2023

In order to find the optimal connecting between the candidates of the input segments, the selection of any two candidates $(c_{t,j},\ c_{t+1,k})$ that are going to follow each other in the 3D reconstructed curve will be based on the degree of the smoothness continuity at their connecting points as shown in Figure *6*. By degree of smoothness continuity, we mean how smoothly the two candidates (segments of 3D Euler spiral curve) link with each other. Based on the Equation 2, to measure the degree of smoothness, we define a loss function that calculates the squared difference between the curvature $\kappa_{a,ct,j}$, the curvature coefficient $\gamma_{a,ct,j}$, the torsion $\tau_{a,ct,j}$, the torsion coefficient $\delta_{a,ct,j}$, the tangent $\vec{T}_{a,c_{t,j}}$, and the normal $\vec{N}_{a,c_{t,j}}$ at the ending point of $c_{t,j}$ and the curvature
$\kappa_{b,ct+1,k}$, the curvature coefficient $\gamma_{b,ct+1,k}$, the torsion $\tau_{b,ct+1,k}$, the torsion coefficient $\delta_{b,ct+1,k}$, the tangent $\vec{T}_{b,c_{t+1,k}}$, and the normal $\vec{N}_{b,c_{t+1,k}}$ at the starting point of $c_{t+1,k}$:

$$\begin{aligned} L(c_{t,j}, c_{t+1,k}) = w_\kappa \left((\kappa_{a,c_{t,j}} - \kappa_{b,c_{t+1,k}})^2 + (\gamma_{a,c_{t,j}} - \gamma_{b,c_{t+1,k}})^2\right) \\ + w_\tau \left((\tau_{a,c_{t,j}} - \tau_{b,c_{t+1,k}})^2 + (\delta_{a,c_{t,j}} - \delta_{b,c_{t+1,k}})^2\right) \\ + w_{\vec{T}} ||\vec{T_{a,c_{t,j}}} - \vec{T_{b,c_{t+1,k}}}||^2 + w_{\vec{N}} ||\vec{N_{a,c_{t,j}}} - \vec{N_{b,c_{t+1,k}}}||^2, \end{aligned} \quad (3)$$

where $||.\ ||\hat{}2$ being the vector length. w_κ, w_τ, $w_{\vec{T}}$, and $w_{\vec{N}}$ are user-specified weight values to give more importance to one the properties (curvature, torsion, tangent) over the others. In our implementation, they are all equal to 1.0.

Let $C_E(R_S)$ be the cost function of the 3D Euler spiral reconstruction for the connecting of the segment candidates in R_S. Specifically, $C_E(R_S)$ is calculated as the sum of the loss functions [Equation 3] between each pair of consecutive candidates in this selected connecting of candidates. To obtain the optimal 3D Euler spiral reconstruction, it is necessary to identify the connecting set of segment candidates that results in the minimum cost, among all possible sets.
We utilize the Dijkstra algorithm [25] to identify the optimal connecting set of candidates that will form the approximate piecewise reconstructed 3D Euler spiral. In this algorithm, we consider R_S as a path from the source node which can be any candidate $c_{1,}$ of the first input segment s_1 to the destination node which can be any candidate $c_{m,f}$ of the last input segment s_m. The candidates represent the nodes and the loss function $(c_{t,j},\ c_{t+1,k})$between each pair of consecutive candidates represent the edges as shown in Figure *7*. Our target is to search for the optimal path between any candidates of $_{s1}$ to any candidates of C_{sm} which will ultimately determine the set of candidates R_S that form the piece-wise reconstructed Euler spiral while minimizing $C_{ES}(R_S)$.
To achieve that, we apply Dijkstra's algorithm by assigning all candidates of $_{s1}$ a cost equal to zero and assigning infinite values to the remaining candidates. Since all candidates of C_{s1} have a cost equal to zero, we can begin applying the algorithm from any of these candidates. The algorithm stops when we add a candidate of C_{sm} to the visited list of Dijkstra's algorithm, indicating that we found the shortest path from a candidate of C_{s1} to a candidate of C_{sm}. Specifically, we stop after adding the first candidate of C_{sm} to the visited list of Dijkstra's algorithm, as all other candidates of C_{sm} are in the unvisited list of Dijkstra's algorithm which means that their cost is higher.

International Journal of Computer Science, Engineering and Information Technology (IJCSEIT),
Vol.13, No.5, October 2023

6.2. Post processing

The connecting of the optimal set of candidates based on the position of their respective split segments in the input curve exhibits a gap between them as shown for the first two matched segments in Figure *8* (1), and Figure *8* (2). To address this issue, we apply a smoothing technique to the reconstructed piecewise Euler spiral. First, each segment of the piecewise Euler spiral is sampled into a polygonal curve. Next, we apply a uniform scaling to these polygonal curves in a way to make their extremity intersect at the same location followed by resampling the entire reconstructed curve by interpolating the x, y, and z coordinates based on the cumulative Euclidean distance between successive points to ensure C^0-continuity as shown in Figure *8* (3). Regarding C^1-continuity it was already nearly preserved because the loss function \ref{eq:3}, which ensured that each of two consecutive Euler spiral segments must have a near continuous tangent at the end of the first segment and the beginning of the second segment. On the other hand, G^2 and G^3continuity (curvature and torsion continuity) are slightly degraded since they both rely on the second derivative which makes them very sensitive to small changes in the curve segment.

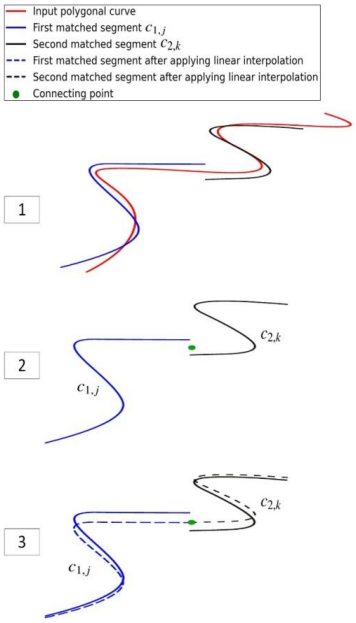

Figure 8: The initial position of the orthogonal projection of the first two matched curves that will be used for the reconstruction of the piecewise Euler spiral is shown in (1) and (2) where there's some gap between them. In (3), after applying linear interpolation the two matched segments intersect at the same point.

An overview of the entire approach starting from the segmentation and matching step to the segments connection and finally, the curve smoothing to form the approximate reconstructed 3D Euler spiral is illustrated in Figure *9*.

International Journal of Computer Science, Engineering and Information Technology (IJCSEIT),
Vol.13, No.5, October 2023

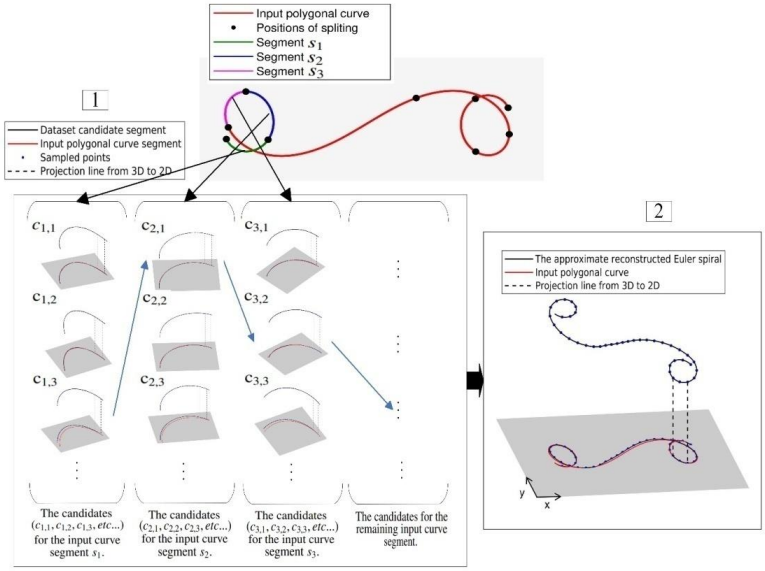

Figure 9: An overview of our approach. (1) Starting from a planar polygonal curve as an input, we split it into segments Then for each segment, we get its closest matched Euler spiral segments (candidates) from the dataset. (2) We select the set of candidates (one per each segment) that can link with each other in the smoothest way. Then we apply a curve smoothing to form the reconstructed curve.

7. RESULTS

Our algorithms are implemented in Python. The dataset is comprised of 1,400,832 3D Euler spiral segments, each of which is composed of 30 3D points in addition to its properties such as its curvature, torsion, tangent, and normal values stored in a text file. The overall size of the dataset is 1.2 GB. During the splitting and matching algorithm, we set the variables N and T to 350 and 0.2, respectively, where N represents the minimum number of the closest matched dataset segments for each input polygonal curve segment. To be considered a match, each of these closest dataset segments must have a Fréchet distance with the input polygonal curve segment smaller than the threshold T. We chose $N = 350$ to ensure a sufficiently large number of potentially matched candidates from the dataset for each input polygonal curve segment. This is necessary because two curves may appear close in 2D while their shape and orientation are quite different in 3D. We set the threshold $T = 0.2$ to ensure that the selected segments are a close match to the input polygonal curve segment while minimizing the Fréchet distance between them. Note that all the Euler spiral segments in the dataset have their length equal to 1.0.

7.1. Experimental results

To evaluate the performance of our reconstruction algorithm, we did 3 experiments. In the first one, we generate 100 ground truth Euler spirals, all of which include an inflection point where the curvature and torsion change sign. In the second experiment, we generate 100 segments of Euler spirals, all of which have only a positive curvature and torsion. We use the orthogonal projection of these 3D Euler spirals as an input to our method in these two experiments. Then we compared

International Journal of Computer Science, Engineering and Information Technology (IJCSEIT),
Vol.13, No.5, October 2023

the reconstructed Euler spirals with the ground truth Euler spirals. In the third experiment, we reconstruct an approximate piecewise 3D Euler spiral from a hand-drawn polygonal curve.

To evaluate the similarity between the smoothed reconstructed 3D Euler spirals and the ground truth spirals, we use a uniform sampling and scaling method in order to standardize the comparison criteria between all reconstructed curves. Specifically, we first uniformly sample and scale the ground truth Euler spirals to make them have the same length of 10.0. We also apply uniform sampling and scaling to their corresponding reconstructed curve in order to make them have the same number of points and the same scaling ratio.

Next, we calculate the distance between the 2D orthogonal projections of each pair of corresponding points on the smoothed reconstructed 3D Euler spirals and the ground truth Euler spirals. We also calculate the difference between the curvatures and torsions of each pair of their corresponding points. These torsion and curvature differences are calculated on the 3D curves. We estimate the similarity between them using three metrics:

1. The average Euclidean distance between their polygonal projections d_p.
2. The average difference between their curvatures d_c.
3. The average difference between their torsions d_t.

The results of the first two experiments are presented in the Figure *10*, and Figure *11*, which show four reconstructions out of 100 for the first experiment and two out of 100 for the second experiment. More reconstruction results are presented in the this link: https://drive.google.com/file/d/1HS_tgVVPSnv7YfAJrZdxiKDmlkywUTeq/view?usp=sharing.Each reconstruction includes a comparison between the ground truth Euler spiral and the approximate reconstructed Euler spiral before applying the post-processing smoothing in {(a),(b),(v),(d)}, and after applying the post-processing smoothing in {(e),(f),(g),(h)}. Wherein (a) and (e), we show both curves in 3D. In (b) and (f), we show their orthogonal projection. In (c) and (g), we show the absolute value of their curvature. Finally, in (d) and (h), we show their torsion. The similarity metrics d_p, d_c, and d_t of the results shown in Figure *10*, and Figure *11* are presented in Table *1*. Note that we obtained these values after applying uniform sampling and scaling on all reconstructed curves to standardize the comparison criteria between them.

The results of the third experiment are presented in Figure *12*. The Figure shows two 3D Euler spiral reconstructions from hand-drawn curves before applying the post-processing smoothness in {(a),(b),(v),(d)} and after applying it in {(e),(f),(g),(h)}. Wherein (a) and (e), we show the reconstructed curve in 3D. In (b) and (f), we show the orthogonal projection of the reconstructed Euler spiral and the hand-drawn polygonal curve. In (c) and (g), we show the absolute value of curvature of the reconstructed Euler spiral. Finally, in (d) and (h), we show its torsion.

We have to mention that in sub-figures (c) and (d) of Figure *10*, Figure *11*, and Figure *12* we show the concatenation of the curvature and torsion of the matched Euler spiral pieces, rather than the entire reconstructed curve. Therefore, we do not observe any significant peaks on the graphs. Conversely, sub-figures (g) and (h) demonstrate the curvature and torsion of the entire smoothed reconstructed curve.

Regarding the results, it is shown in Figure *10*, Figure *11*, and Figure *12* that our reconstruction algorithm produces visually eye-pleasing curve and a quite similar result to the ground truth Euler spirals. Our reconstruction provides 3D curves with C^0 and C^1-continuity. Even G^2 and G^3continuity can be considered acceptable since they are mainly continuous except for some peaks in the curvature and torsion occurring mainly at the joining points of the segments. This is

International Journal of Computer Science, Engineering and Information Technology (IJCSEIT),
Vol.13, No.5, October 2023

because curvature and torsion rely on the second derivative, which makes them highly sensitive to small changes in the curve that may be difficult to discern visually.

Table 1: The similarity metrics (average 2D distance d_p, average curvature difference d_c and average torsion difference d_t) between the smoothed reconstructed Euler spirals of [Figure 10, Figure 11] and the ground truth Euler spirals.

Curve name	d_p	d_c	d_t
C1	0.04	0.34	0.88
C2	0.06	0.37	1.11
C3	0.13	0.43	1.01
C4	0.13	0.21	0.46
C5	0.52	0.09	0.23
C6	0.11	0.11	0.25

7.2. Theoretical and experimental time complexity

We use the K-d tree to search for the matched segments from the dataset for each input segment. The complexity of the searching time inside the K-d tree is (log(Z)), where $Z$is the number of nodes in the tree, in our case, it is 1,400,832.

We used Dijkstra's algorithm to search for the optimal set of connected segments to form the reconstructed piecewise Euler spiral. The time complexity of the Dijkstra algorithm is ((V + E) log(V)), where V is the number of nodes and E is the number of edges in the graph. In our case, the number of nodes is $V = N^2 \times (m - 1)$ and the number of edges is $E = N \times m$, where m represents the number of segments that we split the input polygonal curve into, while Nrepresents the number of matched segment candidates from the dataset for each input segment.

To conduct our experiments, we utilized a computer with an Intel Core i7-10700K processor, 32 GB of RAM, and an NVIDIA Quadro RTX 4000 graphics card. The machine was running Ubuntu 18.04.6 LTS as the operating system. The time required for each reconstruction varies depending on the number of segments used to split the input curve. On average, the reconstruction process takes around 50 seconds, with approximately 40 seconds spent on splitting the curve and searching for matched curves in the dataset, and 10 seconds spent on finding the best connecting of segments using Dijkstra's algorithm. The time required to apply the smoothing process is negligible.

International Journal of Computer Science, Engineering and Information Technology (IJCSEIT),
Vol.13, No.5, October 2023

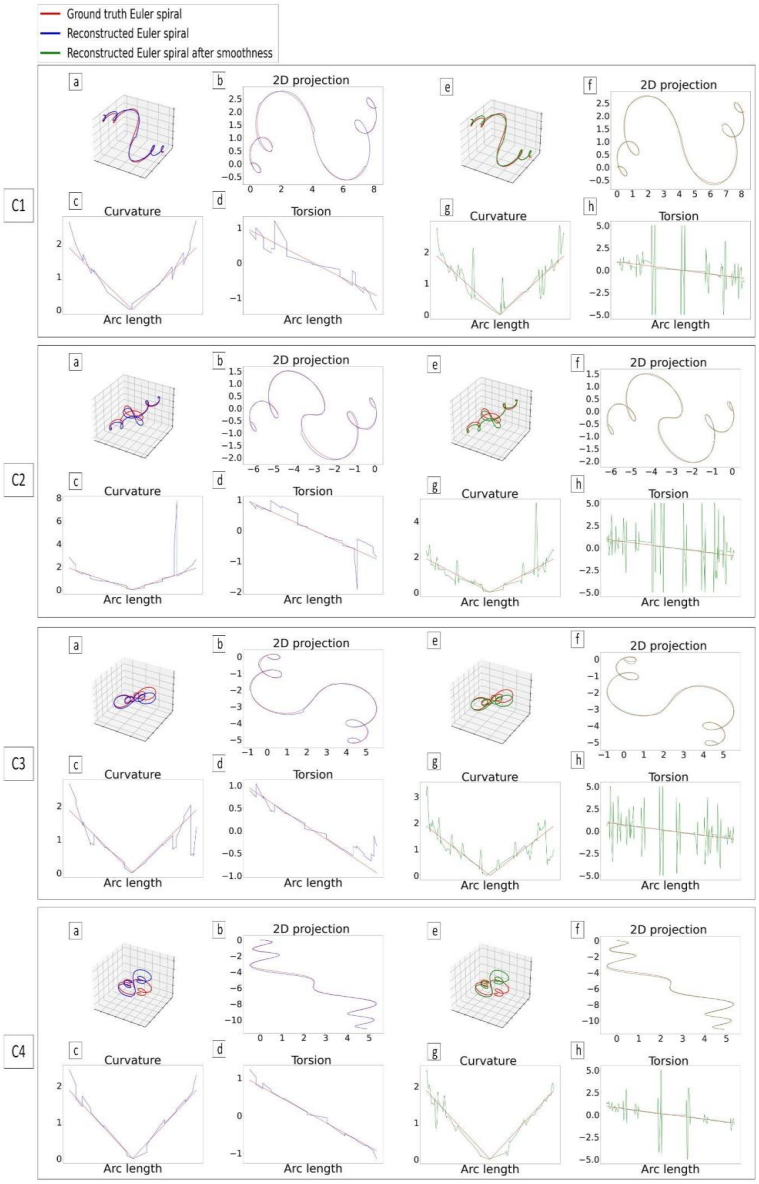

Figure 10: Comparison of our reconstructed Euler spirals before and after applying smoothing with the ground truth 3D Euler spirals (Euler spirals that include an inflection point).

International Journal of Computer Science, Engineering and Information Technology (IJCSEIT),
Vol.13, No.5, October 2023

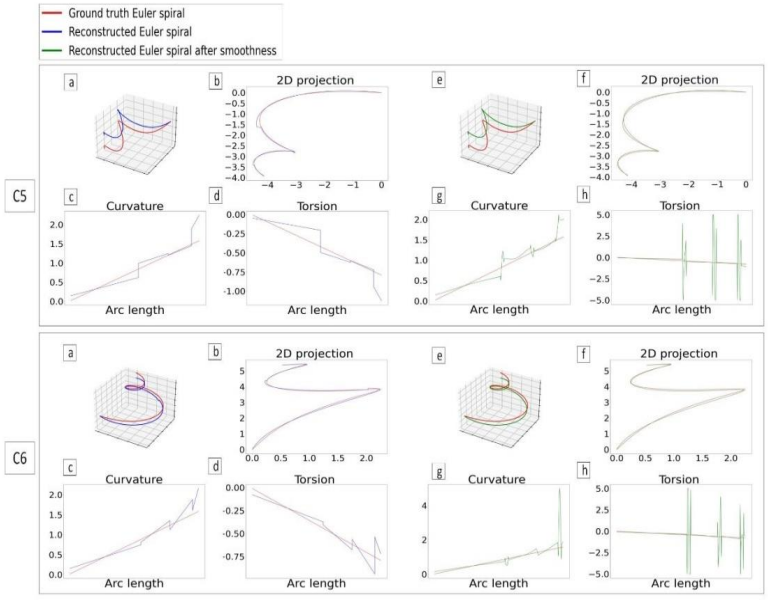

Figure 11: Comparison of our reconstructed Euler spirals before and after applying smoothing with the ground truth 3D Euler spirals (Euler spirals with only positive curvature and torsion).

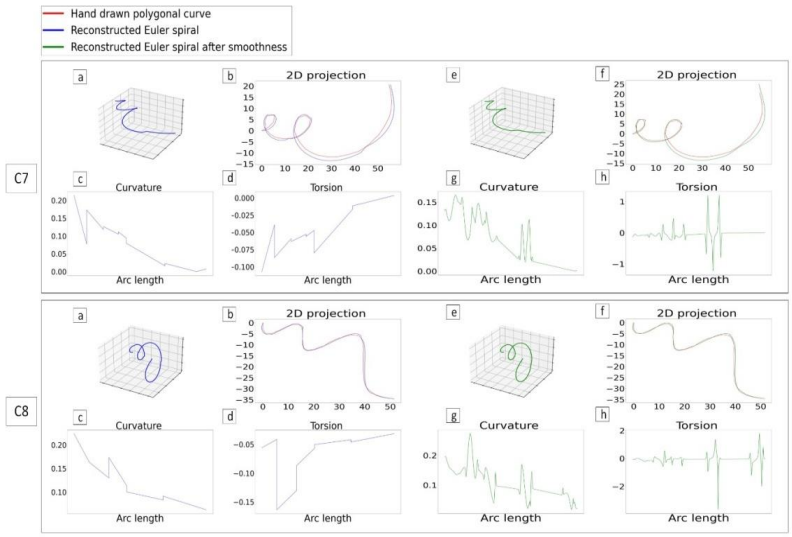

Figure 12: Reconstruction of approximate piecewise 3D Euler spirals from hand-drawn polygonal curves.

International Journal of Computer Science, Engineering and Information Technology (IJCSEIT),
Vol.13, No.5, October 2023

8. CONCLUSION

In this paper, we have proposed a novel method for approximating a piecewise 3D Euler spiral that accurately fits a planar polygonal curve. The effectiveness of our method has been demonstrated with a large number of input polygonal curves. Our reconstruction curves exhibited to have C^0 and C^1 continuity which makes it possible to be used for 3D modeling and for generating motions and trajectories. Future work could be to test our method against real-life applications.

REFERENCES

[1] B. B. Kimia, I. Frankel, and A.-M. Popescu, "Euler spiral for shape completion," *International Journal of Computer Vision 54*, vol. 1, pp. 159–182, 2003.
[2] D. E. Knuth, "Mathematical typography," *Bulletin AMS 1*, vol. 2, pp. 337–372, 1979.
[3] S. Ullman, "Filling-in the gaps: The shape of subjective contours and a model for their generation," *Biol Cybern*, vol. 25, pp. 1–6, 1976.
[4] A. T. Gur Harary, "3D Euler spirals for 3D curve completion," *Computational Geometry*, vol. 45, pp. 115–126, 2012.
[5] E. Brown and P. S. P. Wang, "Three-dimensional object recovery from twodimensional images: a new approach," in *Intelligent robots and computer vision xv: Algorithms, techniques, active vision, and materials handling*, 1996, pp. 138– 147.
[6] K. Shoji, K. Kato, and F. Toyama, "3-D interpretation of single line drawings based on entropy minimization principle," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 90–95, 2001.
[7] N. Nozawa, H. Shum, E. Ho, and S. Morishima, "Single Sketch Image based 3D Car Shape Reconstruction with Deep Learning and Lazy Learning," in *Proceedings of the 2020 International Conference on 3D Vision*, 2020, pp. 898– 907.
[8] L. Yang, J. Wu, J. Huo, Y.-K. Lai, and Y. Gao, "Learning 3D face reconstruction from a single sketch," *Graph Models*, vol. 115, pp. 101–102, 2021.
[9] F. Wang *et al.*, "Deep 3D Shape Reconstruction from Single-View Sketch Image," *The 8th International Conference on Digital Home*, pp. 184–189, 2020.
[10] Y. Shen, C. Zhang, H. Fu, K. Zhou, and Y. Zheng, "DeepSketchHair: Deep Sketch-based 3D Hair Modeling," *IEEE Trans Vis Comput Graph*, vol. 27, pp. 3250–3263, 2021.
[11] H. Yang, Y. Tian, C. Yang, Z. Wang, L. Wang, and H. Li, "Sequential learning for sketch-based 3D model retrieval," *Multimed Syst*, pp. 1–18, 2022.
[12] P. Xu, T. M. Hospedales, Q. Yin, Y.-Z. Song, T. Xiang, and L. Wang, "Deep learning for free-hand sketch: A survey," *IEEE Trans Pattern Anal Mach Intell*, vol. 45, no. 1, pp. 285–312, 2022.
[13] F. Cordier, M. Melkemi, and H. Seo, "Reconstruction of helices from their orthogonal projection," *Comput Aided Geom Des*, vol. 46, pp. 1–15, 2016.
[14] N. Cherin, F. Cordier, and M. Melkemi, "Modeling piecewise helix curves from 2D sketches," *Computer-Aided Design*, vol. 46, pp. 258–262, 2014.
[15] X. Marchal, F. Bertails, and F. Hétroy, "Reconstruction de trochoides à partir de courbes 2D," *Technical report*, 2009.
[16] R. Levien, "The Euler spiral: a mathematical history," *Tech. Rep. UCB/EECS2008-111, EECS Department, University of California*, 2008.
[17] D. J. Walton and D. S. Meek, "A controlled clothoid spline," *Comput Graph*, vol. 29, pp. 353–363, 2005.
[18] I. Baran, J. Lehtinen, and J. Popović, "Sketching clothoid splines using shortest paths," in *Computer Graphics Forum*, 2010, pp. 655–664.
[19] J. McCrae and K. Singh, "Sketching piecewise clothoid curves," *Comput Graph*, vol. 33, no. 4, pp. 452–461, 2009.
[20] J. McCrae and K. Singh, "Neatening sketched strokes using piecewise french curves," in *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*, 2011, pp. 141–148.
[21] L. Guiqing, L. Xianmin, and L. Hua, "3D discrete clothoid splines," in *International Conference on Computer Graphics*, 2001, pp. 321–324.

International Journal of Computer Science, Engineering and Information Technology (IJCSEIT),
Vol.13, No.5, October 2023

[22] M. Frego, "Closed form parametrisation of 3D clothoids by arclength with both linear varying curvature and torsion," *Appl Math Comput*, vol. 421, p. 126907, 2022.

[23] A. M. Lyapunov, "The general problem of the stability of motion," *Int J Control*, vol. 55, no. 3, pp. 531–534, 1992.

[24] J. M. Lien, "A dichotomy algorithm for computing smooth paths," *Commun ACM*, vol. 24, no. 11, pp. 682–690, 1981.

[25] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer Math (Heidelb)*, vol. 1, no. 1, pp. 269–271, 1959.

[26] M. Fréchet, "Sur quelques points du calcul fonctionnel," *Rendiconti del Circolo Matematico di Palermo*, vol. 22, no. 1, pp. 1–74, 1906, doi: 10.1007/bf03013491.

[27] J. L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *Association for Computing Machinery*, vol. 18, pp. 509–517, 1975.

4.3 3D reconstruction of free-form curves

In the previous section, we described a method for reconstructing 3D curves; however, it had limitations as it was confined to Euler spiral curves. In this section, we aim to develop a method capable of reconstructing free-form curves while minimizing curvature variation along the curve.

The field of computer graphics has made significant advancements in areas such as rendering and animation. However, a persistent challenge remains in accurately capturing user intentions for 3D modeling in a simple and intuitive manner. Interacting with users becomes complicated due to the limitations of two-dimensional interfaces, which do not properly accommodate our natural perception and thinking in a three-dimensional environment. One particularly challenging aspect is drawing 3D curves within a 2D input device. These curves are crucial in various applications such as defining the trajectories of moving objects or modeling of 3D objects. Several works have been proposed to address this problem, such as changing perspectives or requiring the artist to draw the shadow of the curve. However, most of these methods pose some constraints to the artists, leading to reduced efficiency. This section intends to address this problem, in particular, the modeling of 3D curve from a single drawing. Unlike most of the existing methods, our approach does not require any other input or user interaction than the planar curve of the drawing.

The main idea of the method is driven by the following observation. Let C_{3D} be a 3D polygonal curve and $O_{3D,i}$ the osculating circle at the midpoint of a segment $e_{3D,i}$ of this curve. The osculating circle is the circle that has the same tangent and the same curvature as at the midpoint of $e_{3D,i}$. Let C_{2D} , $e_{2D,i}$ and $O_{2D,i}$ be the orthogonal projection on the (x,y) plane of C_{3D}, $e_{3D,i}$ and $O_{3D,i}$ respectively. $O_{2D,i}$ is an ellipse whose tangent and curvature are equal to those of the curve C_{2D} at the midpoint of the segment $e_{2D,i}$ (see Figure 4.1). Now, we are only provided the planar curve C_{2D} and our goal is to reconstruct the C_{3D} curve. Given a segment $e_{2D,i}$ of C_{2D}, we compute a set of ellipses whose curvature and tangent match those at the midpoint of $e_{2D,i}$. If we assume that this set of ellipses is large enough, one of these ellipses must be the projection of an osculating circle being very close to the actual osculating circle of the curve C_{3D} to reconstruct. We repeat this

process of computing the set of osculating circle candidates for each segment of C_{3D}. Then, the problem comes down to selecting the best osculating circle for each segment such that the variation of curvature is minimized along the curve. This is a simple graph problem that is solved using Dijkstra's algorithm. These osculating circles are then used to determine the tangent for each segment and to compute the 3D coordinates of the points of C_{3D}.

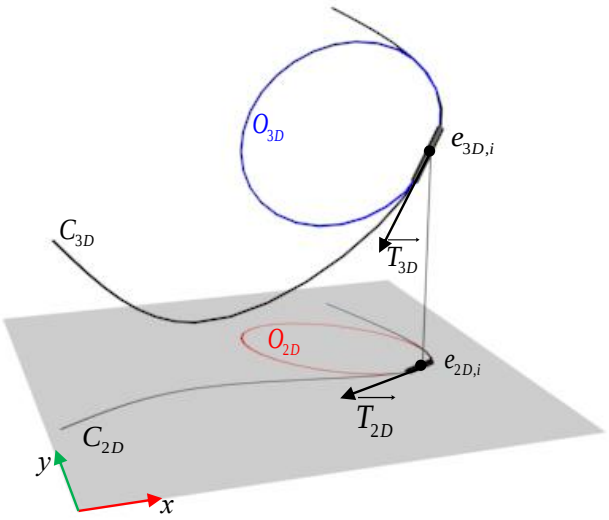

Figure 4.1: The curve C_{3D} with its osculating circle O_{3D} at the midpoint of the edge$e_{3D,i}$. The curve C_{2D} which is the orthogonal projection of C_{3D} and the ellipse O_{2D} which is the orthogonal projection of the osculating circle $O_{3D,i}$.

The contribution of this work is twofold. First, we propose the idea of fitting ellipses to the input planar curve; these ellipses are then used to determine the osculating circles and in turn, the orientation of the tangent at the midpoint of each segment of the 3D curve to reconstruct. The second contribution is a method to solve the optimization problem which aims at finding the optimal osculating circles such that the variation of the curvature along the 3D reconstructed curve is minimized. The usual approach would require solving a large nonlinear optimization using the gradient-descent method. Instead, the solution space is discretized to form a graph, each node of this graph being a candidate for the osculating

circles at the midpoint of each segment of the curve. The edges of the graph correspond to the adjacency relationship between the segments. The solution of this optimization problem is then solved using Dijkstra's algorithm.

The remainder of this section is organized as follows: First, we provide the problem definition of our approach. Then, we delve into the details of our algorithm for fitting ellipses to a planar polygonal curve. Subsequently, we explain the process of computing the osculating circles from the ellipses. Afterward, we describe the final step of the algorithm, which involves computing the coordinates of the 3D curve using the previously computed tangent of the osculating circles. Following that, we present the results obtained from our approach. Finally, we conclude this section with a summary of our findings.

4.3.1 Problem definition

Our method takes as input a planar curve C_{2D} in the (x, y) plane and generates a 3D curve C_{3D} whose orthogonal projection onto the (x,y) plane matches the input C_{2D} curve and such that the change of curvature is minimized along this curve C_{3D}. The input curve C_{2D} is a polygonal curve composed of p points denoted $v_{2D,1}, v_{2D,2}...v_{2D,p}$. The set of $p-1$ segments is $e_{2D,1}, e_{2D,2}...e_{2D,p-1}$ with $e_{2D,i}$ being the segment connecting $v_{2D,i}$ to $v_{2D,i+1}$. This curve is assumed to be at least G^1 continuous; it should not contain any sharp corners and the curvature can be accurately computed for every segment. The reconstructed 3D curve C_{3D} is composed of p points $v_{3D,1}, v_{3D,2}...v_{3D,p}$, each of these points corresponding to a point in the input curve C_{2D}. The segments are denoted $e_{3D,1}, e_{3D,2}...e_{3D,p-1}$.

The reconstruction is done iteratively, starting from the first segment $e_{2D,1}$ of the input curve C_{2D}. The driving idea is to fit a set of ellipses to each segment of the input curve, i. e. ellipses whose curvature and tangent match the curvature and tangent at the midpoint of the segment $e_{2D,i}$ (Figure 4.2(b)). These fitted ellipses are then used to compute the set of candidates for the osculating circle at each segment of the curve C_{3D} (Figure 4.2(c)). Next, we formulate an optimization problem that aims at finding the best osculating circle for each segment such that the change of curvature is minimized along the 3D reconstructed curve. This is done by building a graph whose nodes are the candidates for the osculating circles

and by applying Dijkstra's algorithm. The last step is to compute the translation along the z-axis for each segment (Figure 4.2(e)) using the tangent of the osculating circles previously computed (Figure 4.2(d)).

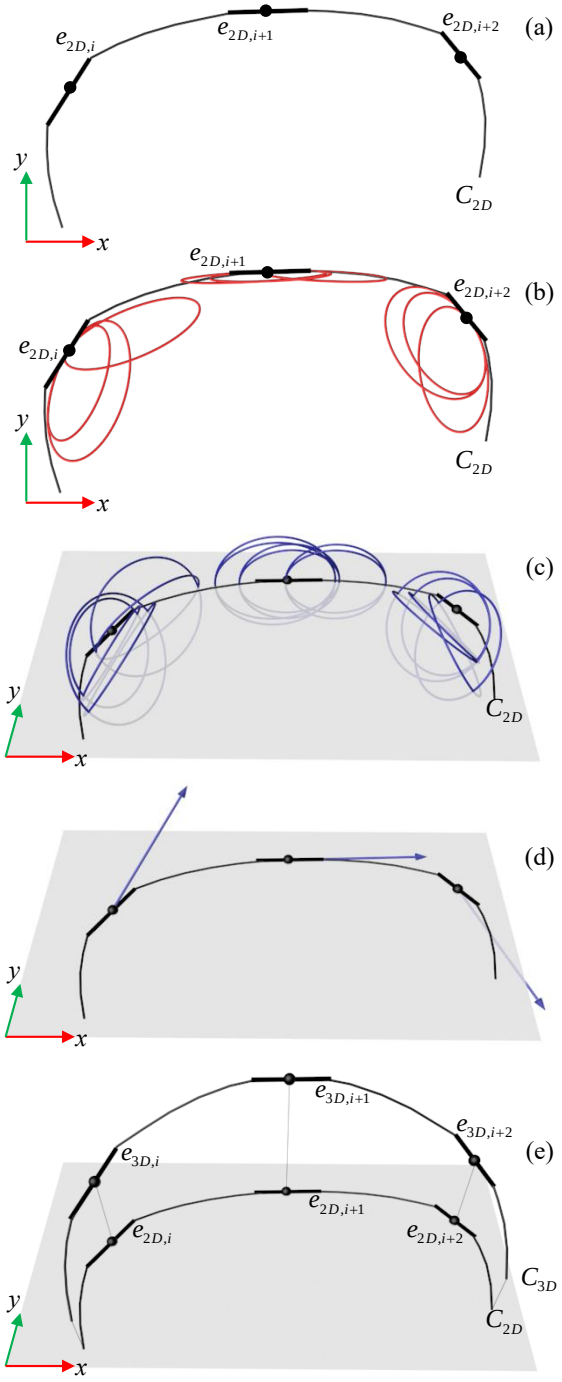

Figure 4.2: Overview of the method. The input curve C_{2D} in with 3 adjacent segments $e_{2D,i}$, $e_{2D,i+1}$ and $e_{2D,i+2}$ (a). For the sake of clarity, only a subset of the segments is drawn. Ellipses are first fitted to the midpoint of each segment along the curve such that they have the same curvature and tangent as the corresponding midpoint (b). Candidates for the osculating circles at each midpoint are generated (c). These osculating circles are then used to estimate the tangent of the curve C_{3D} to reconstruct (d). Finally, the curve C_{3D} is reconstructed by lifting the points such as to satisfy the estimated tangent.

4.3.2 Ellipse fitting

This section describes the method to fit a set of ellipses to the midpoint of a segment $e_{2D,i}$ of the input planar curve C_{2D}. The parametric equation of the ellipse is given as follows:

$$\begin{aligned} x(t) &= a\cos t, \\ y(t) &= b\sin t. \end{aligned} \tag{4.1}$$

We assume that $a < b$. These parameters a and b are the half-length of the minor and major axes respectively. The parameter t has a value in the interval $0, 2\pi[$. The tangent vector $\vec{T(t)}$ for the ellipse of Equation 4.1 is given as follows:

$$\begin{aligned} \vec{T(t)} &= \begin{pmatrix} \frac{x'(t)}{\sqrt{(x'(t))^2+(y'(t))^2}} \\ \frac{y'(t)}{\sqrt{(x'(t))^2+(y'(t))^2}} \end{pmatrix} \\ &= \begin{pmatrix} \frac{-asin(t)}{\sqrt{(asin(t))^2+(bcos(t))^2}} \\ \frac{bcos(t)}{\sqrt{(asin(t))^2+(bcos(t))^2}} \end{pmatrix}. \end{aligned} \tag{4.2}$$

We also provide the equation of the curvature $k(t)$ of the ellipse:

$$\begin{aligned} k(t) &= \frac{||x'(t)y''(t) - y'(t)x''(t)||}{||(x'(t))^2 + (y'(t))^2||^{\frac{3}{2}}} \\ &= \frac{ab}{\left(a^2sin^2(t) + b^2cos^2(t)\right)^{\frac{3}{2}}}. \end{aligned} \tag{4.3}$$

Let $k_{2D,i}$ and $\vec{T_{2D,i}}$ be respectively the curvature and tangent of at the midpoint of the segment $e_{2D,i}$ of the curve C_{2D}. These curvature and tangent are computed using the method proposed by Lewiner et al. (Lewiner et al. 2005). We aim at finding the values of t, a, and b such that $k_{2D,i} = k(t)$. As one may observe, there is an infinite number of ellipses with different values for a, b, and t and that satisfies the equation $k_{2D,i} = k(t)$ (see Figure 4.3).

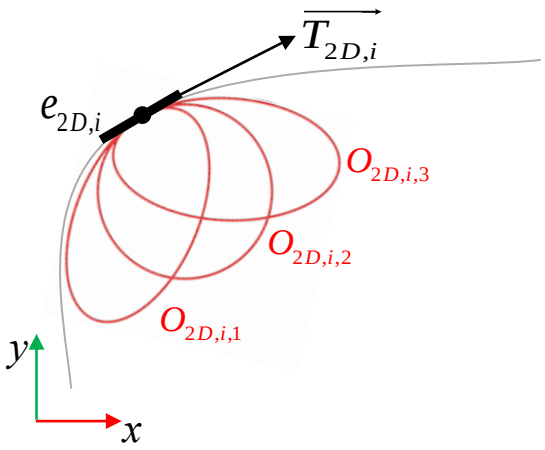

Figure 4.3: Given the midpoint of a segment $e_{2D,i}$ alongC_{2D} with its tangent $\vec{T_{2D,i}}$ and curvature $k_{2D,i}$, we compute the set of ellipses $S_{O,2D,i} = \{O_{2D,i,1}, O_{2D,i,2}, O_{2D,i,3}\}$ with the specified curvature $k_{2D,i}$.

Our method to determine this set of ellipses works as follows. We compute a sampling of a and b in the interval $[r_{min}, r_{max}]$ denoted $S_a = \{a_1, a_2, \ldots, a_m\}$ and $S_b = \{b_1, b_2, \ldots, b_m\}$ respectively; m is the number of samples for a and b. In our implementation, m is equal to 30. The user-defined parameters $r_{min}$and r_{max} are respectively the smallest and largest radii of the osculating circles of the 3D curve to reconstruct. The set of ellipses is found by setting a to one of the values of Sa and b to one of the values of Sb. The value of t is found by solving the equation $k\,(t) = k_{2D,i}$, which is rewritten as follows:

$$sin(t) = \sqrt{\frac{(ab)^{\frac{2}{3}} - k_{v_{2D,i}}^{\frac{2}{3}} b^2}{k_{v_{2D,i}}^{\frac{2}{3}} \left(a^2 - b^2\right)}} \tag{4.4}$$

This equation has a solution if $\frac{ab}{b^2} \leq k_{2D,i} \leq \frac{ab}{a^2}$ with $a < b$. The inverse sine function gives two solutions for t. This set of ellipses that have been computed for a specific midpoint of the segment $e_{2D,i}$ along the input planar curve C_{2D} is denoted $S_{O,2D,i} = \{O_{2D,i,1} \ldots O_{2D,i,j} \ldots O_{2D,i,n}\}$ with n being the number of ellipses and $O_{2D,i,j}$ being an ellipse whose parameters are $a \in S_a$ and $b \in S_b$.

Note that our fitting algorithm is different from computing the osculating ellipse (Williamson 1912). The osculating ellipse has a 4th-order contact at the midpoint of the segment $e_{2D,i}$ of the curve C_{2D}; this implies that the second derivatives of the curvature are equal (Rutter 2000). In our case, the ellipses have only 2nd-order contact since we only require the curvatures and the tangent of the curves to be equal. As the 2nd-order contact is less restrictive, a large number of ellipses will be taken into account for the next step to compute the osculating circles. If we compute the osculating ellipse and the part of the curve has some noise, it could happen that this ellipse is not a good approximation of the osculating circle, making the reconstruction of the 3D curve very difficult.

4.3.3 Finding the osculating circles

The next step is to compute the osculating circles for each segment $s_{3D,i}$ of the curve C_{3D} using the set of ellipses $S_{O,2D,i} = \{O_{2D,i,1} \dots O_{2D,i,j} \dots O_{2D,i,n}\}$ that has been computed for each segment $s_{2D,i}$ of the curve C_{2D}. Each of these ellipses is the orthogonal projection of a circle with a specific radius and orientation with respect to the projection plane. Let $S_{O,3D,i} = \{O_{3D,i,1} \dots O_{3D,i,j} \dots O_{3D,i,n}\}$ be the set of circles whose orthogonal projection are the ellipses of the set $S_{O,2D,i}$; these are the candidates for the osculating circles at the midpoint of the segment $s_{3D,i}$. We now aim at selecting one of them such that the change of curvature is minimized along the C_{3D} curve.

Minimizing the change of the curvature implies that two neighboring segments $s_{3D,i}$ and $s_{3D,i+1}$ should have their osculating circle with similar radius, center, orientation, and center. This means that we should select one circle in each set $S_{O,3D,i}$ and $S_{O,3D,i+1}$ such that the difference of the radius, orientation, and center is the smallest possible. This is a combinatorial problem that we solve using Dijkstra's algorithm.

Construction of the graph

We first build a graph whose nodes are the candidates for the osculating circles. The set of nodes is defined as the union of all the sets $S_{E,3D,i}$ for the midpoint of each segment $e_{3D,i}$ of the curve C_{3D} : $S_{O,3D,1} \cup S_{O,3D,2} \dots \cup S_{O,3D,m} =$

$\{O_{3D,1,1} \dots O_{3D,i,j} \dots O_{3D,p,n}\}$. The edges of this graph correspond to the neighboring relationship between adjacent segments along the curve C_{3D}. An edge is put between two nodes $O_{3D,i,k}$ and $O_{3D,j,l}$ if the two corresponding segments $e_{3D,i}$ and $e_{3D,j}$ are adjacent along the curve C_{3D}. The construction of the graph is shown in Figure 4.4.

A weight is associated with each edge to determine how similar the two osculating circles are. Let $O_{3D,i,j}$ and $O_{3D,i+1,k}$ be the osculating circles of two neighboring segments $e_{3D,i}$ and $e_{3D,i+1}$ respectively. Let $r_{3D,i,j}$, $c_{3D,i,j}$, $\vec{N_{3D,i,j}}$ and $\vec{T_{3D,i,j}}$ be the radius, center, binormal, and tangent of the osculating circle $O_{3D,i,j}$ and $r_{3D,i+1,j}$, $c_{3D,i+1,j}$, $\vec{N_{3D,i+1,j}}$ and $\vec{T_{3D,i,j+1}}$ the radius, center, binormal and tangent vectors of the osculating circle $O_{3D,i+1,j}$. Let $\alpha_{i,j,i+1,k}$ be the angle between $\vec{N_{3D,i,j}}$ and $\vec{N_{3D,i+1,j}}$ and $\beta_{i,j,i+1,k}$ the angle between $\vec{T_{3D,i,j}}$ and $\vec{T_{3D,i+1,j}}$. The weight is calculated with a cost function $W_O(\,)$ which takes into account the difference of radii and center as well as the misalignment of the binormal and tangent of the osculating circles:

$$W_O\left(r_{3D,i+1,j}, r_{3D,i,j}, c_{3D,i+1,j}, c_{3D,i,j}, \alpha_{i,j,i+1,k}\right) = \delta\left((r_{3D,i+1,j} - r_{3D,i,j})^2 + |c_{3D,i+1,j} - c_{3D,i,j}|^2\right) + (1-\delta)\left(\alpha_{i,j,i+1,k}^2 + \beta_{i,j,i+1,k}^2\right) \tag{4.5}$$

with δ being a user-defined parameter to put more weight on the radii and center difference or the binormal and tangent misalignment. Figure 4.5 shows an example of the cost function between two osculating circles.

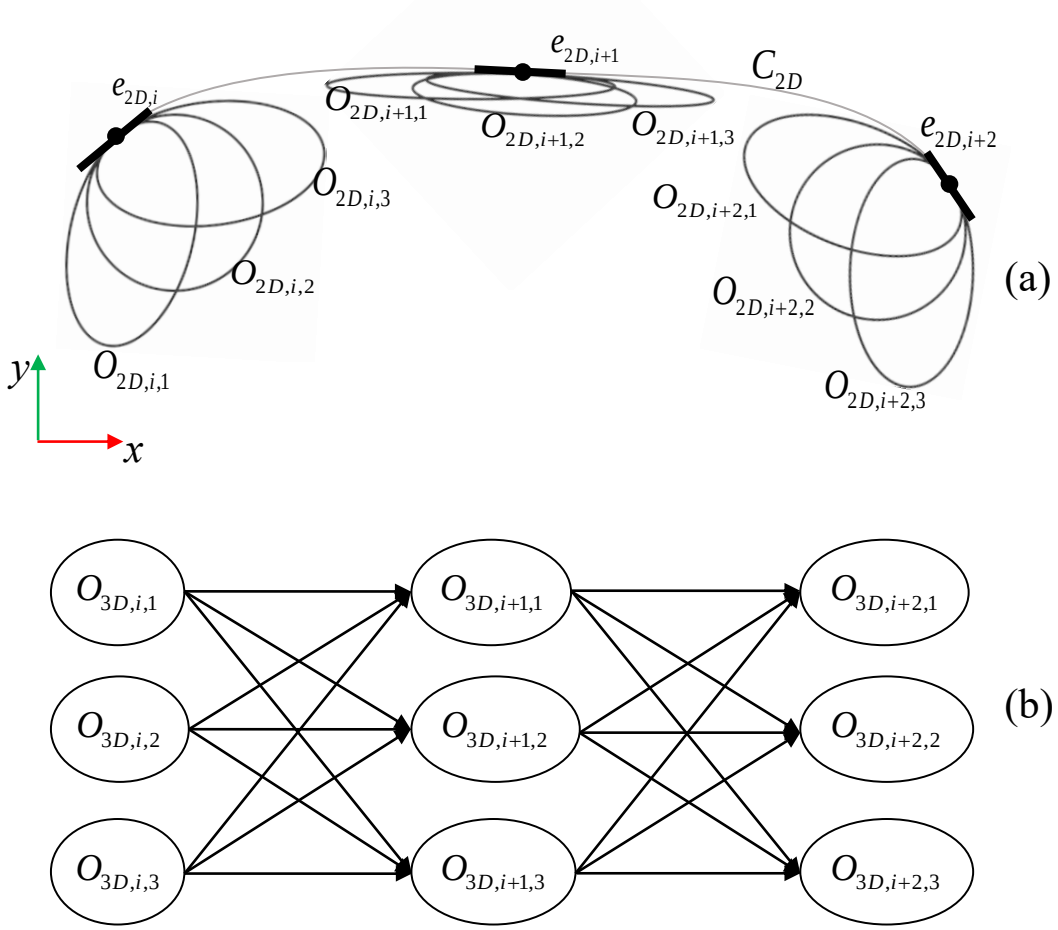

Figure 4.4: The graph to compute the osculating circles. For the sake of clarity, the curve C_{2D} is composed of 3 segments only and a subset of the ellipses is shown.

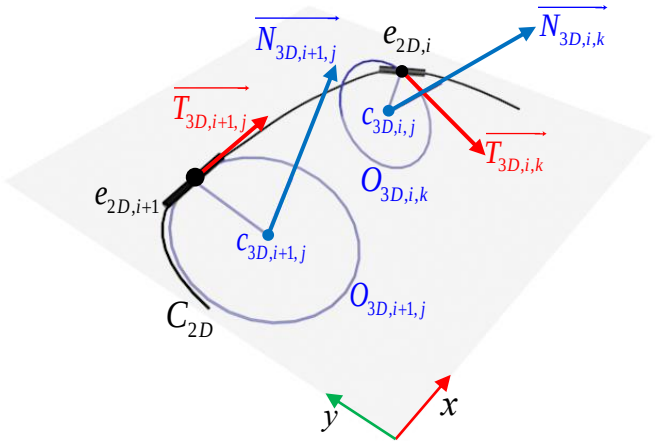

Figure 4.5: The cost function between two osculating circles $O_{3D,i,k}$ and $O_{3D,j,l}$ corresponding to the midpoint of the segments $e_{3D,i}$ and $e_{3D,i+1}$ respectively. $e_{3D,i}$ and $e_{3D,i+1}$ are adjacent along the curve.

Using the Dijkstra's algorithm to compute the osculating circles

We use Dijkstra's algorithm to compute the optimal osculating circle for every segment $e_{3D,i}$ of the C_{3D} curve. For a given source node in a weighted graph, this algorithm finds the shortest path between that node and every other node. We have implemented a modified version of this algorithm as follows. The source is the set of osculating circle candidates $S_{O,3D,1} = \{O_{3D,1,1} \dots O_{3D,1,j} \dots O_{3D,1,n}\}$ of the first segment of the curve $e_{3D,1}$. The target is the set of osculating circle candidates $S_{O,3D,p-1} = \{O_{3D,p-1,1} \dots O_{3D,p-1,j} \dots O_{3D,p-1,n}\}$ of the last segment $s_{3D,p-1}$. Initially, all the nodes are in the unvisited node set, except those belonging to the source set $S_{O,3D,1}$. The nodes of this source set are given the distance 0. The algorithm works iteratively. It finds the unvisited node v with the minimal distance value; this distance value is computed using the weight of the associated edge. This node is then removed from the unvisited node set. The algorithm stops when one of the nodes of the target set $S_{O,3D,p-1}$ has been visited.

The result of Dijkstra's algorithm is an approximation of the osculating circle for each vertex of the curve C_{3D}. These osculating circles provides an estimation of the tangent $\vec{T}_{3D,i}$ at the midpoint of each segment $e_{3D,i}$ of the curve C_{3D} (see Figure 4.6).

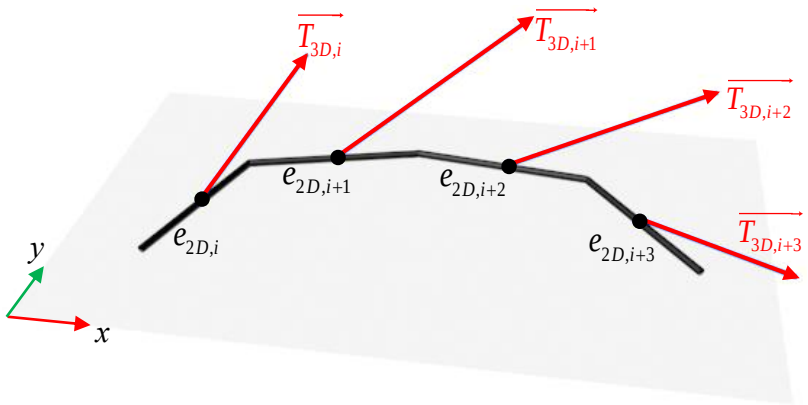

Figure 4.6: Estimation of the tangents resulting from the Dijkstra's algorithm.

4.3.4 Reconstruction of the 3D curve using the estimated tangent at the segments

The last step of the 3D reconstruction is to compute the z-coordinates of all the points $v_{3D,i}$ of the curve C_{3D}. The x and y coordinates are the same as the corresponding point $v_{2D,i}$ in the curve C_{2D} since the curve C_{2D} is the orthogonal projection of C_{3D} onto the (x,y) plane.

The z-coordinates are computed iteratively starting from the first endpoint $v_{2D,1}$. Its z-coordinate is set to 0.0. The z-coordinates of a point $v_{2D,i+1}$ are computed by adding a value Δz_{i+1} to the z-coordinate of the previous point $v_{2D,i}$. This value Δz_{i+1} is defined using the estimated tangent $\vec{T_{3D,i}}$ at the midpoint of the segment $e_{2D,i}$ as follows:

$$\Delta z_{i+1} = \frac{||v_{2D,i+1} - v_{2D,i}||}{\sqrt{(x_{T3D,i})^2 + (y_{T3D,i})^2}} z_{T3D,i} \tag{4.6}$$

with $x_{T3D,i}$, $y_{T3D,i}$ and $z_{T3D,i}$ being the x, y and z coordinates of $\vec{T_{3D,i}}$ respectively (see Figure 4.7).

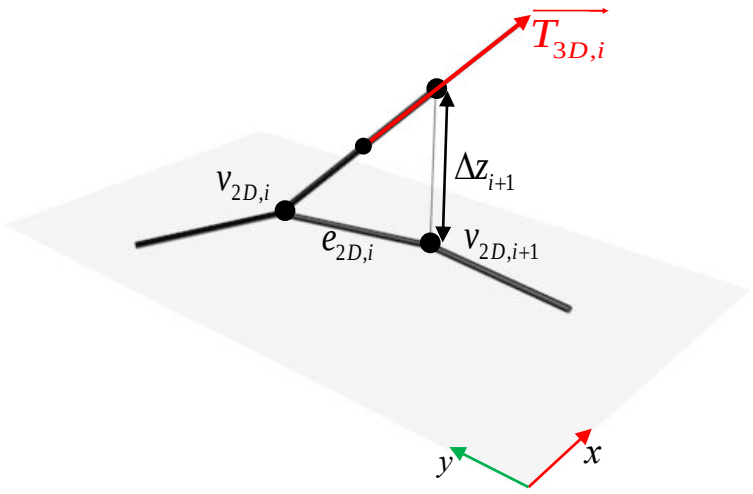

Figure 4.7: Computation of the z-coordinate of the point $v_{2D,i+1}$ using the estimated tangent $\vec{T_{3D,i}}$.

4.3.5 Results

Our method has been implemented in Python and has been tested with a variety of curves. The results are show in Figure 4.9 , Figure 4.10 and Figure 4.11. For each reconstruction, we provide the input curve, three views of the reconstructed 3D curve, and the graphs for the curvature and torsion of the reconstructed curve. These curvature and torsion values are calculated with respect to the arclength. The computation time ranges from 10 to 60 seconds depending on the number of points of the input curve. The number of points of all the curves ranges from 30 to 200. To the best of our knowledge, there is no previous work that performs the 3D reconstruction from free-form curves, hence we can not compare our results with any previous work.

Reconstruction of circular helices

Circular helices are 3D curves whose curvature and torsion are constant. Figure 4.9(a) and Figure 4.9(b) show the reconstruction from a 2D curve which is the orthogonal projection of a circular helix with different orientations. The projected curve has been uniformly resampled before the reconstruction. The curvature and torsion of the circular helix are $4.65 \cdot 10^{-2}$ and $1.17 \cdot 10^{-2}$ respectively. As one may observe, the curvature of the reconstructed curve is very close to that of the circular helix. The value of torsion is less accurate. This is because the torsion of the curve is not taken into account during the reconstruction process.

Reconstruction of spirals and hand-drawn curves

The reconstruction has been done with other curves. These curves are hand-drawn except the curve in Figure 4.9(c) which is a 2D Euler spiral and the curve in Figure 4.11(c) which is an Archimedean spiral. Similarly to the reconstruction of circular helices, our algorithm is able to generate a 3D curve whose variation of curvature is small along the curve. This is particularly true for the curves shown in Figure 4.10(a) and Figure 4.11(a).

Limitations

The user is required to provide the lower and upper bounds of the circular helix parameters. The reconstruction fails in case these values are not properly set. One solution could be to compute a first reconstruction with large upper and lower bounds. These values can then be refined iteratively with trials and errors. The reconstruction does not work for some specific curves. The first case concerns curves with G^1 discontinuity, that is, curves with sharp corners (see Figure 4.8(a)). This is because our method requires the curvature values to be defined at every segment; this is needed for the ellipse fitting (see subsection 4.3.2). The second case concerns connected segments whose curvature is null; this happens when the curve is composed of several points that are colinear. This set of consecutive colinear segments is actually the projection of a straight line in 3D; our method is not able to reconstruct straight lines in 3D since osculating circles are not defined for straight lines. On the other hand, our method handles a single segment whose curvature is null; these segments are usually located at the inflection point of the curve (see Figure 4.9(b)).

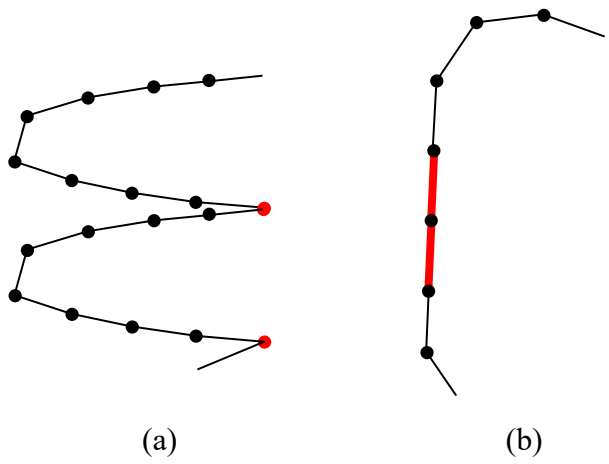

Figure 4.8: Curves that cannot be processed by the reconstruction algorithm: curves with a sharp corner shown (red dots in (a)) and curves with 2 or more connected segments whose curvature is null (red segments in (b)).

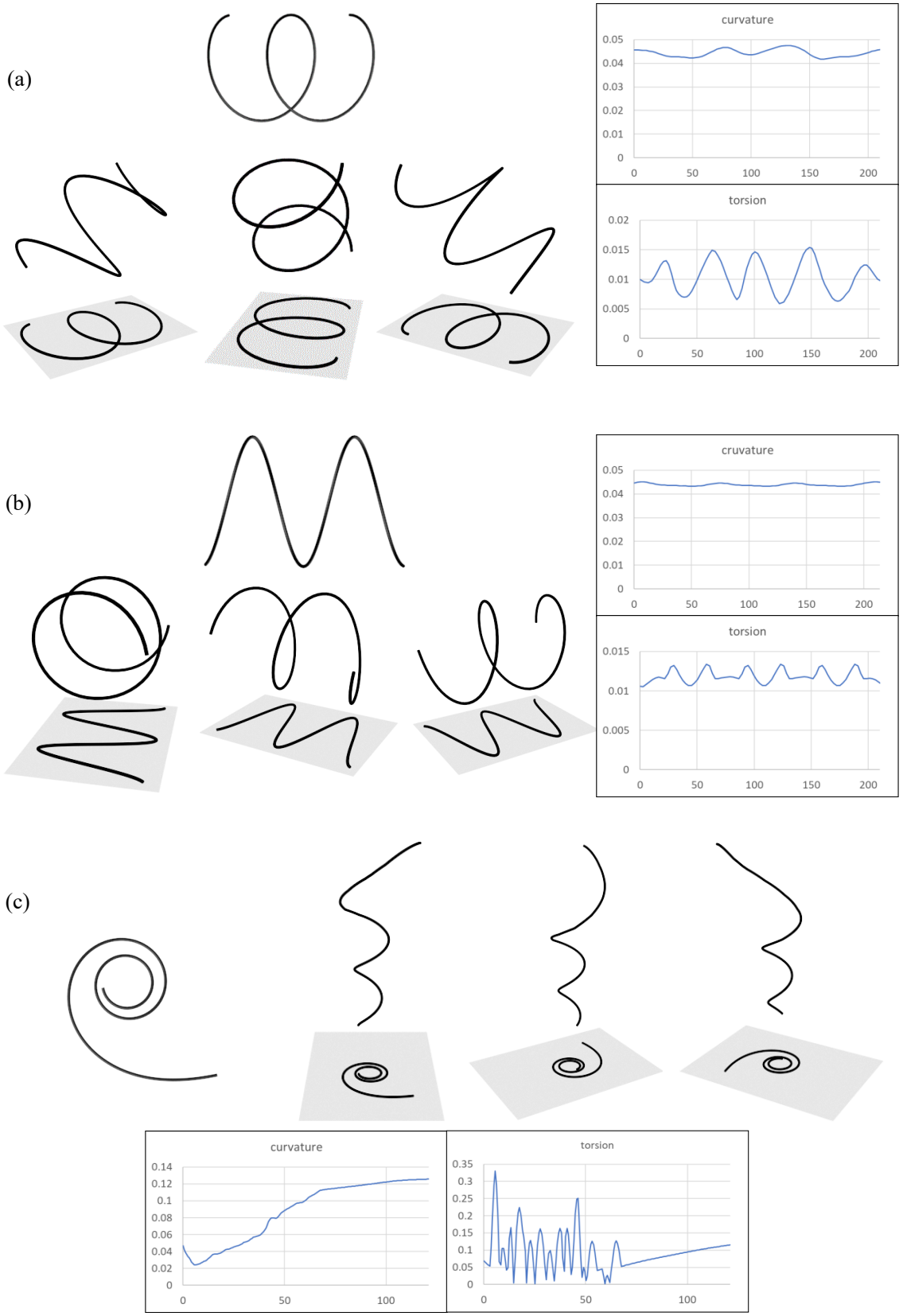

Figure 4.9: Reconstruction using the orthogonal projection of a circular helix with different orientations (a) and (b) and using the 2D Euler spiral (c).

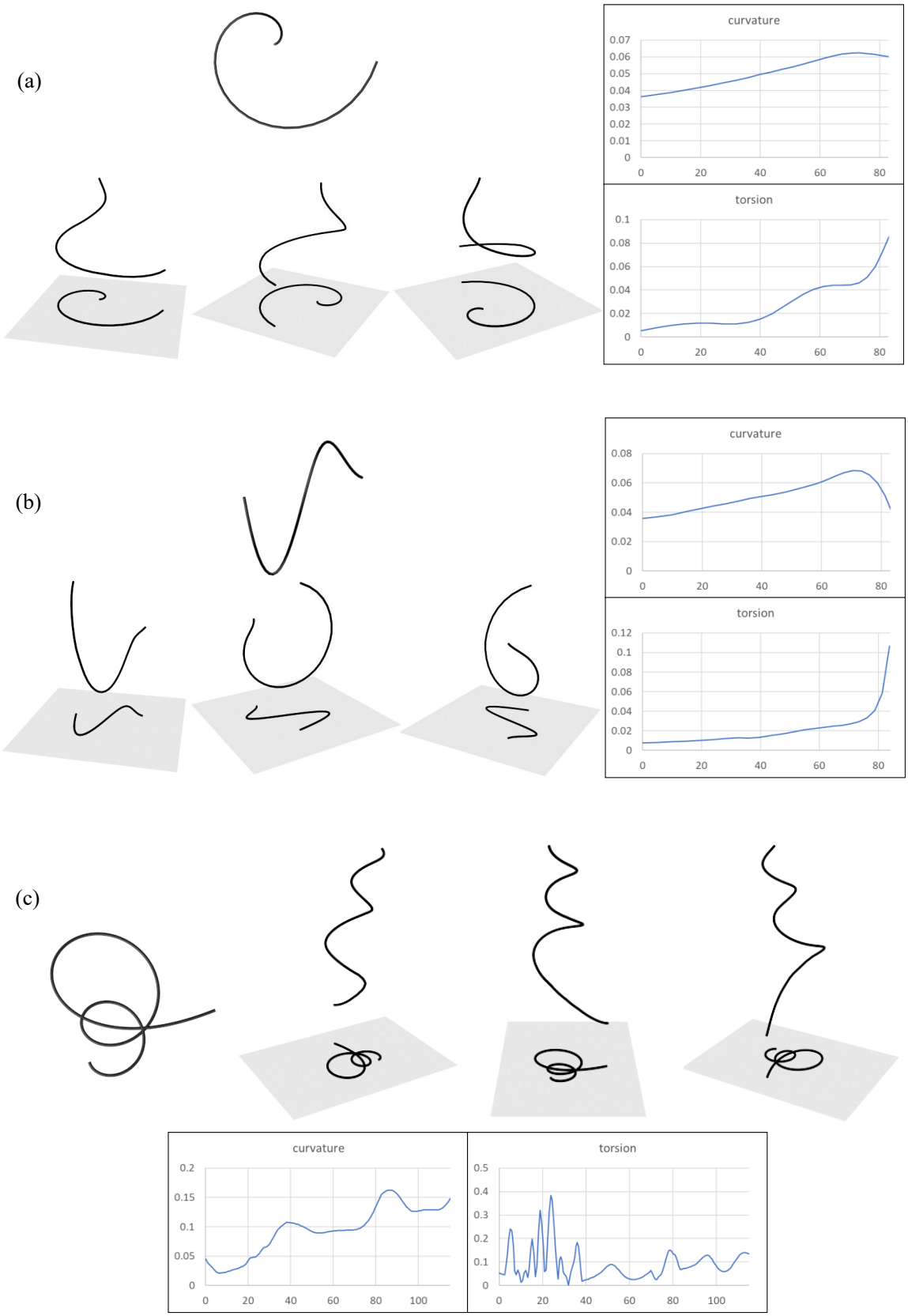

Figure 4.10: Reconstruction using a 2D Euler spiral (a) and using hand-drawn curves (b) and (c).

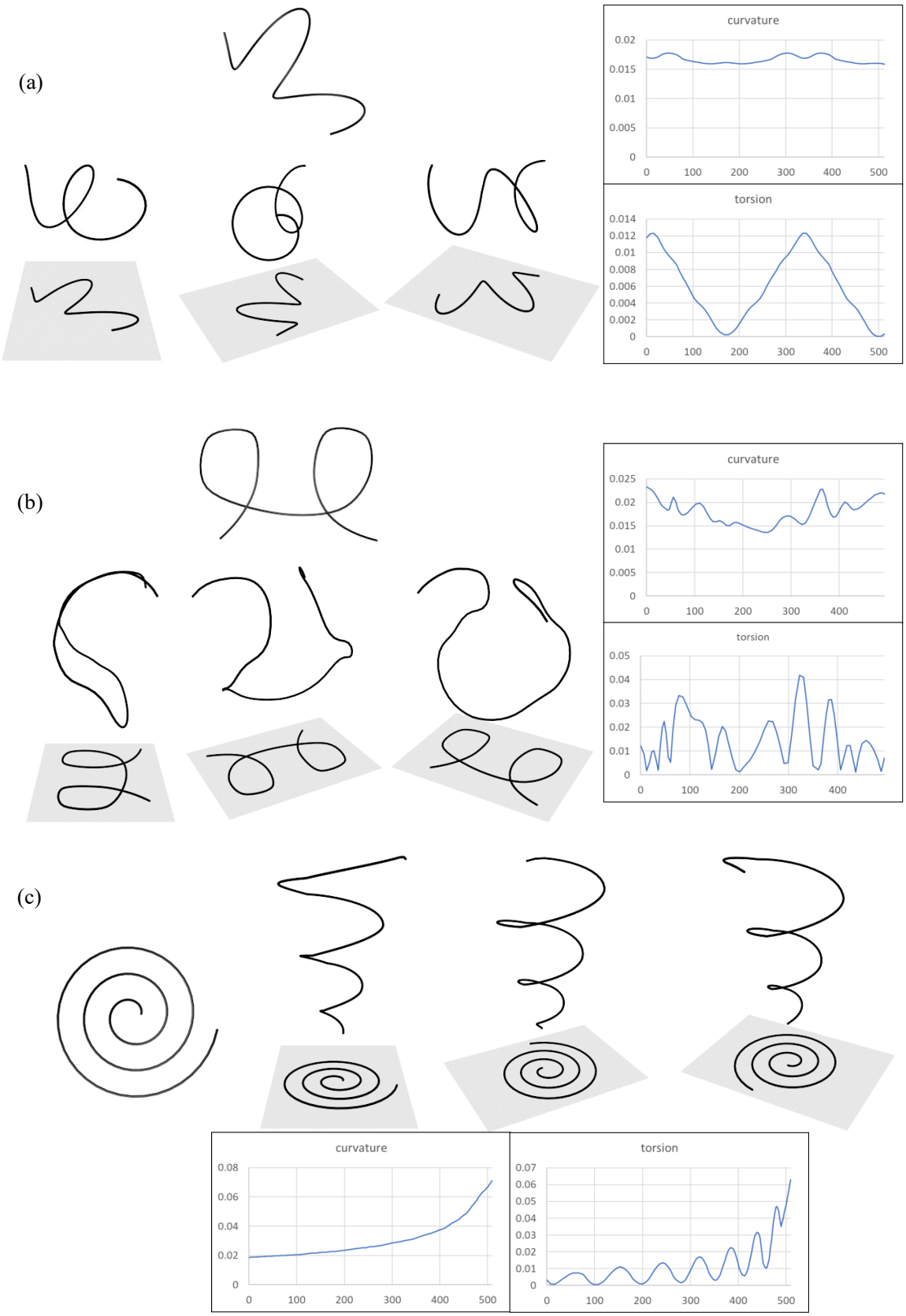

Figure 4.11: Reconstruction using hand-drawn curves (a) and (b) and from an Archimedean spiral (c).

4.4 Conclusion

In this chapter, we introduced two different approaches for the 3D reconstruction of curves. The first one performs the 3D reconstruction of an input curve in the form of the orthogonal projection of Euler spiral. Doing so, we achieved an eye-pleasing reconstruction, by using a curve-matching technique, applied in a piecewise manner. The second performs the 3D reconstruction of an input curve in many forms such as spirals, circular helices, etc. It also achieved good results by minimizing the variation of the reconstructed curve. This was accomplished by fitting ellipses to the input curve, enabling us to determine the osculating circles and tangents at each curve point to reconstruct in 3D.

Chapter 5

Conclusion and perspectives

This thesis began with the objective of exploring methods for the 3D reconstruction of curves from a single-view 2D drawing. In Chapter 3, we investigated the performance of various ML algorithms for reconstructing circular helices. Many ML algorithms were tested and more than one of them provides a good reconstruction. However, when confronted with more complex curves such as Euler spiral curves, the ML algorithms faced limitations. To address this, Chapter 4 introduced two distinct approaches. The first approach focused on the reconstruction of Euler spiral curves by employing a curve-matching technique in a piecewise manner. Although it yielded visually appealing results, its scope was limited to Euler spiral curves. To broaden the applicability of our reconstruction approach, we developed a novel method capable of reconstructing multiple curve types, including circular helices, spirals, and free-form curves. This approach minimized the curvature variance of the curve by fitting ellipses to the input curve, allowing us to determine the osculating circles and tangents at each curve point for 3D reconstruction.

In conclusion, this thesis has made significant strides in the field of 3D curve reconstruction. By leveraging ML algorithms and innovative approaches, we have demonstrated effective techniques for reconstructing circular helices, Euler spiral curves, and hand-drawn curve types.

There are several paths to future work. One possible improvement would be to take into account the variation of the torsion in addition to the curvature. Another future work would be to extend the method to handle the reconstruction of

close curves. Furthermore, our contributions serve as a solid foundation for future studies and practical applications across diverse fields such as computer graphics, computer vision, and virtual reality. In particular, there is potential to apply our methods in real-world scenarios, such as modeling the intricate trajectories of moving objects, which is crucial in fields like robotics and animation. Moreover, the techniques developed could find innovative applications in the domain of wire art reconstruction.

Appendices

Appendix A

Additional results for the 3D reconstruction of Euler spirals

This supplementary material presents additional results of our experiments in Chapter 4 (First section: 3D reconstruction of Euler spiral curves). The first five figures illustrate examples of the results obtained in the first experiment, in which we generated 100 ground truth Euler spirals, all of which contained an inflection point where the curvature and torsion change sign. We used the orthogonal projection of these spirals as input for our reconstruction algorithm. The last five figures show some of the results obtained in the second experiment, in which we generated 100 segments of Euler spirals with only positive curvature and torsion. We also used the orthogonal projection of these segments as input for our reconstruction algorithm.

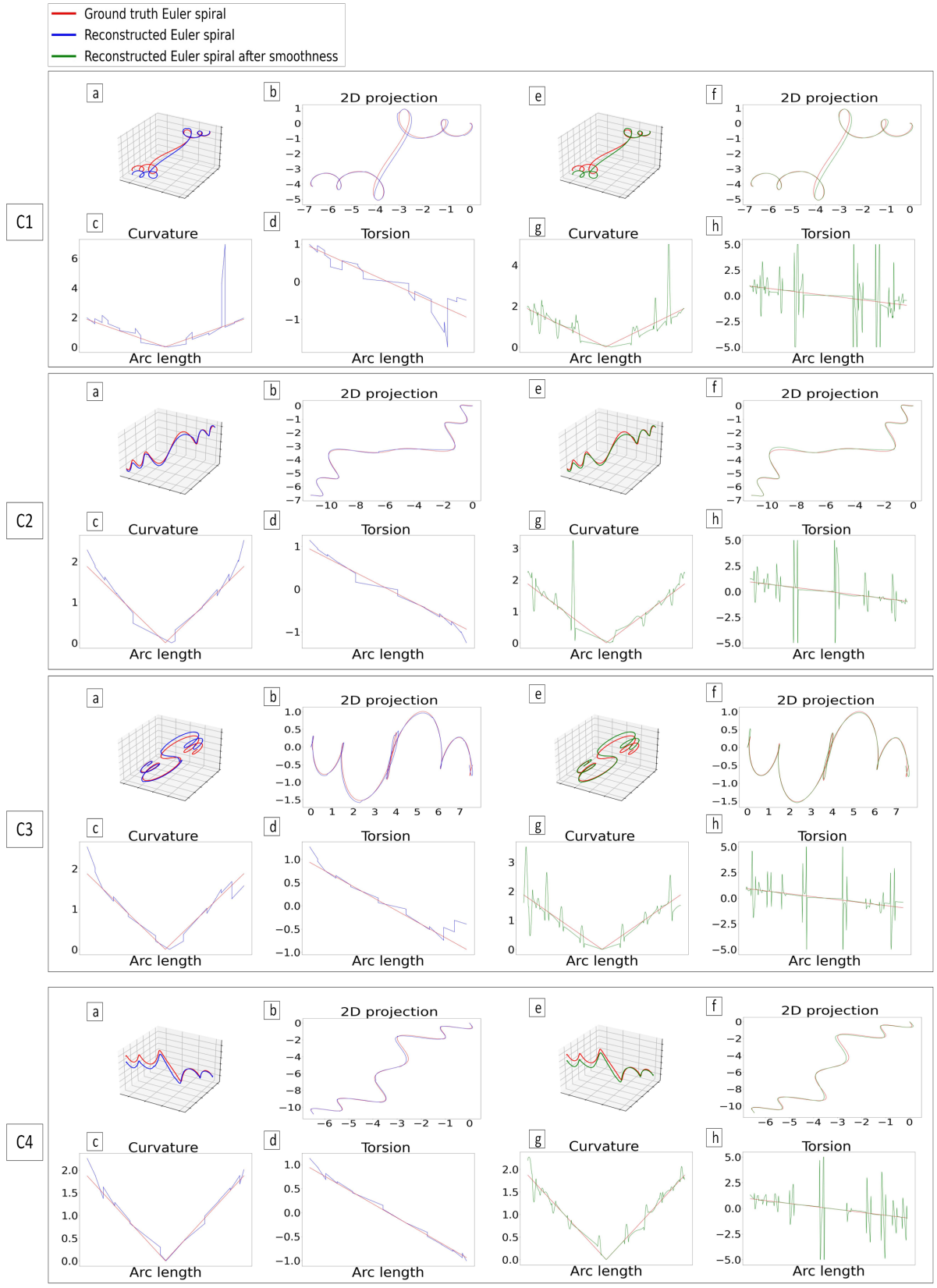

Figure A.1: Comparison of our reconstructed Euler spirals before and after applying smoothing with the ground truth 3D Euler spirals (Euler spirals that include an inflection point).

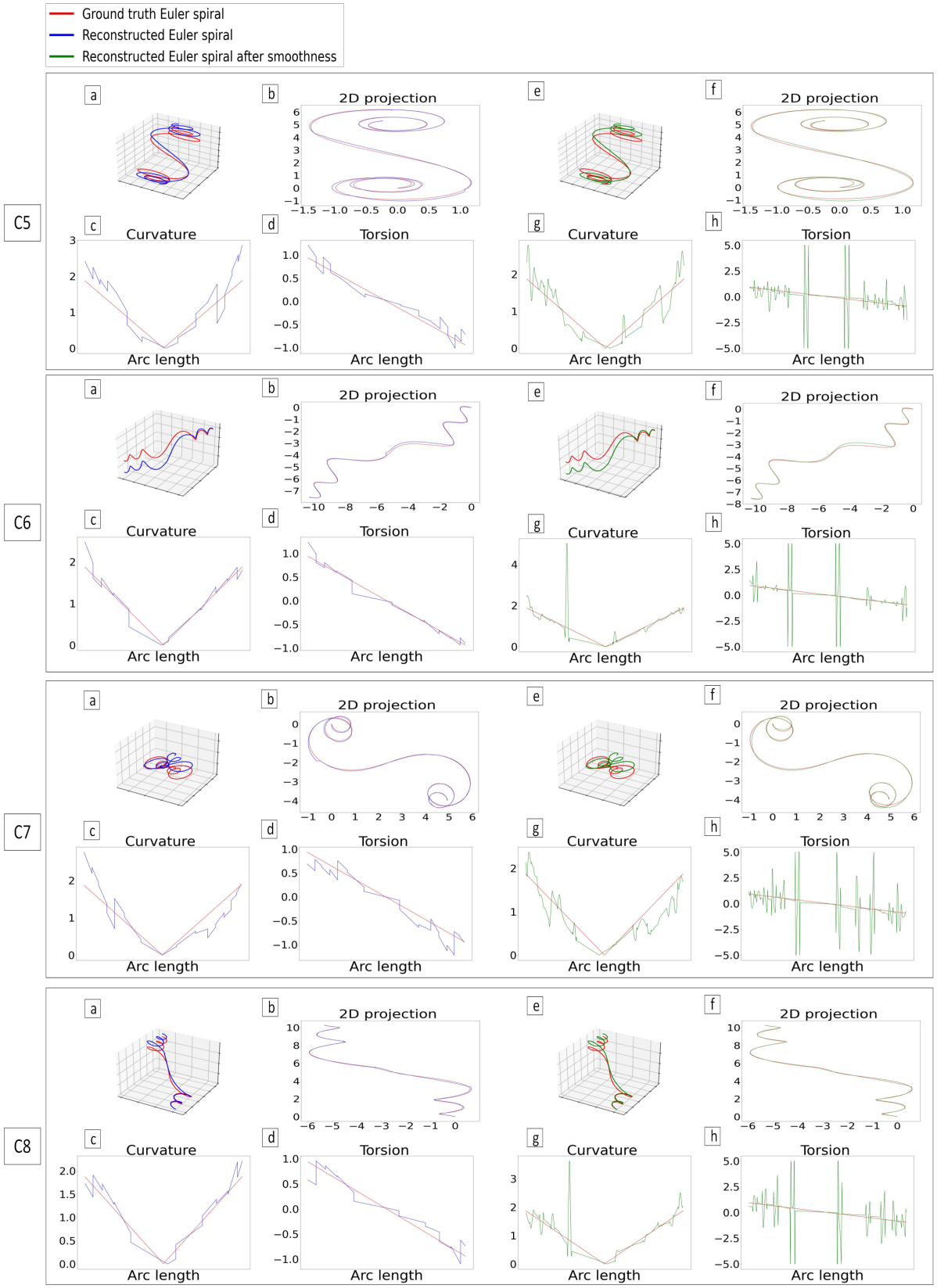

Figure A.2: Comparison of our reconstructed Euler spirals before and after applying smoothing with the ground truth 3D Euler spirals (Euler spirals that include an inflection point).

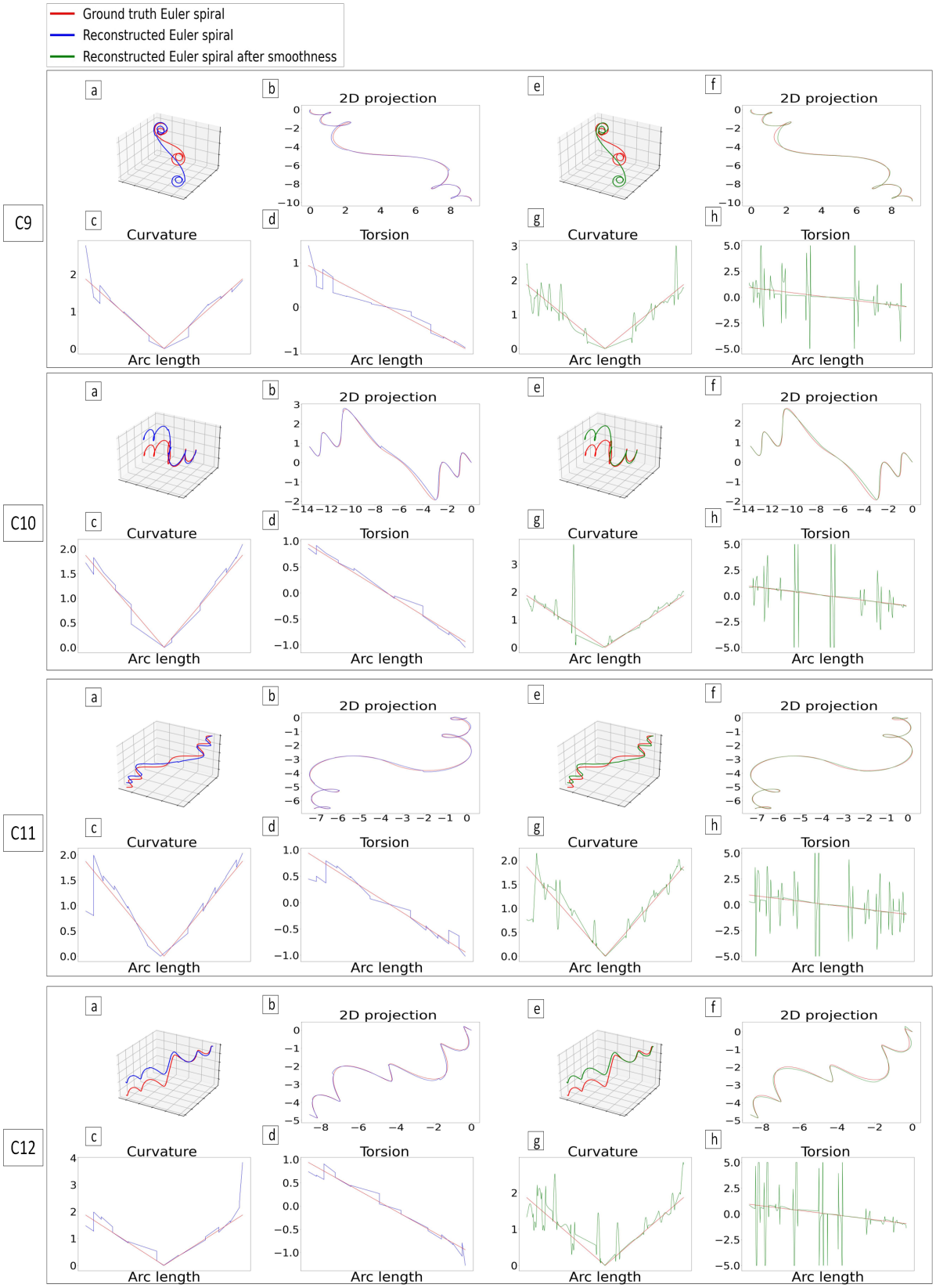

Figure A.3: Comparison of our reconstructed Euler spirals before and after applying smoothing with the ground truth 3D Euler spirals (Euler spirals that include an inflection point).

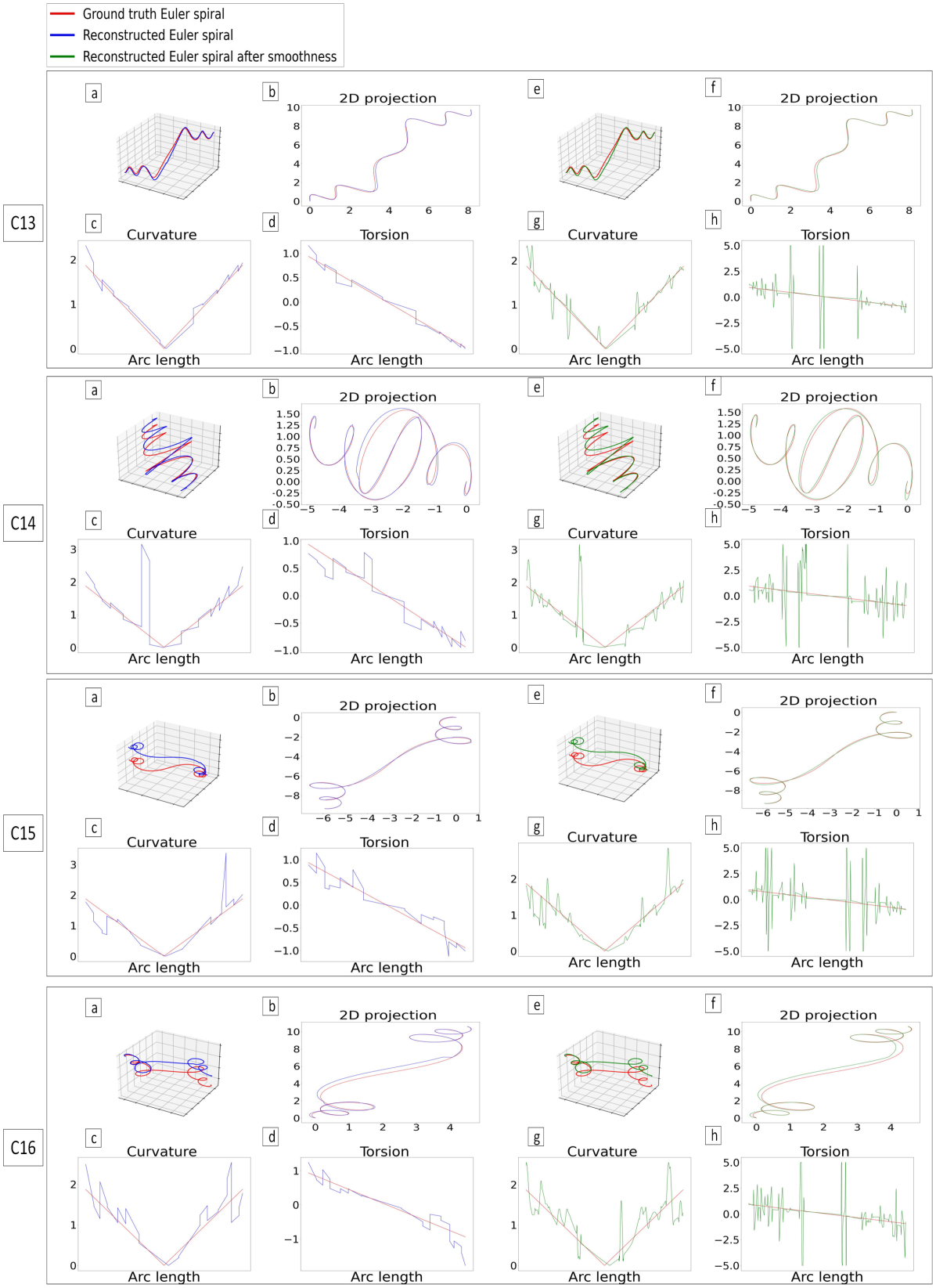

Figure A.4: Comparison of our reconstructed Euler spirals before and after applying smoothing with the ground truth 3D Euler spirals (Euler spirals that include an inflection point).

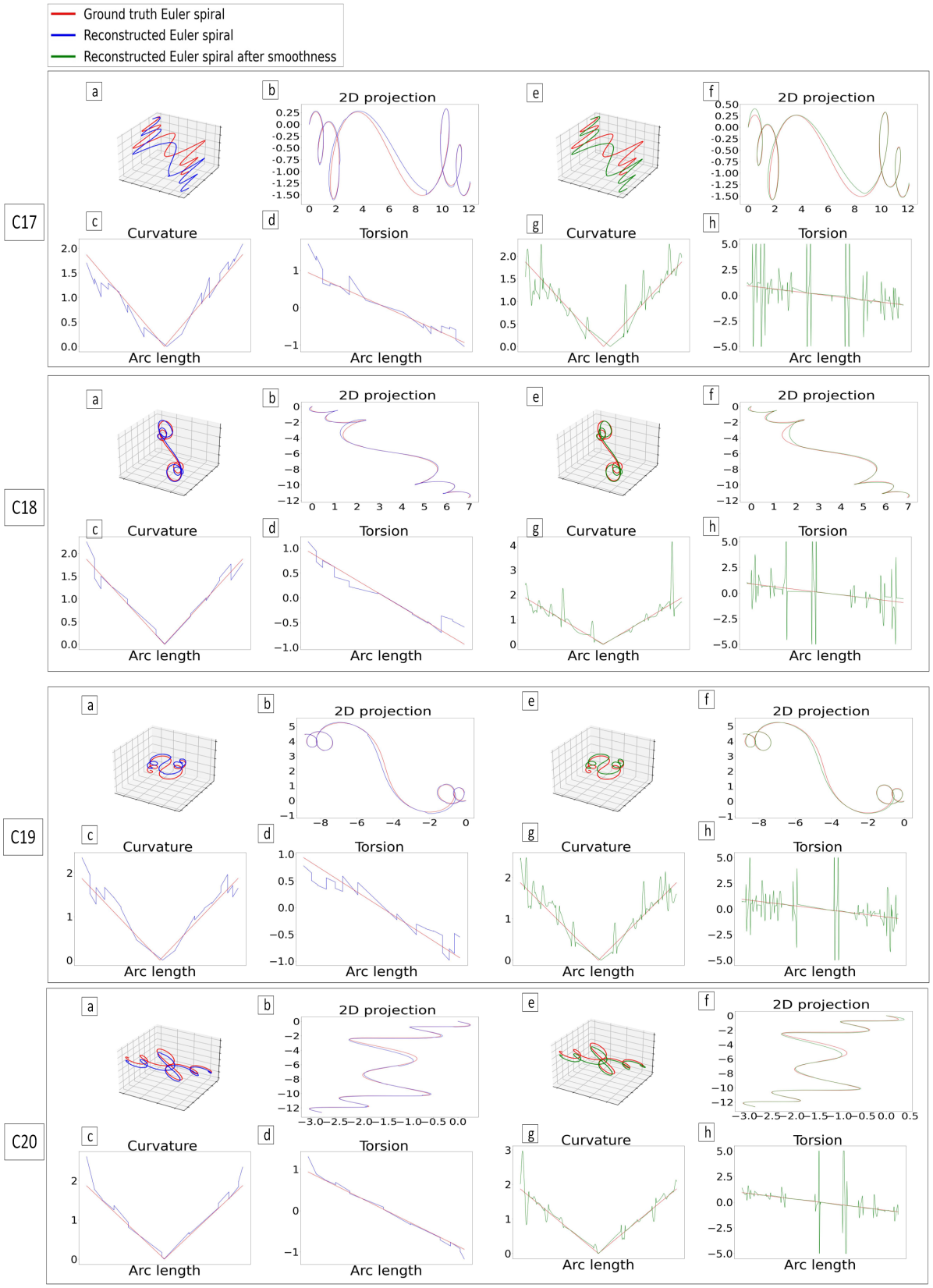

Figure A.5: Comparison of our reconstructed Euler spirals before and after applying smoothing with the ground truth 3D Euler spirals (Euler spirals that include an inflection point).

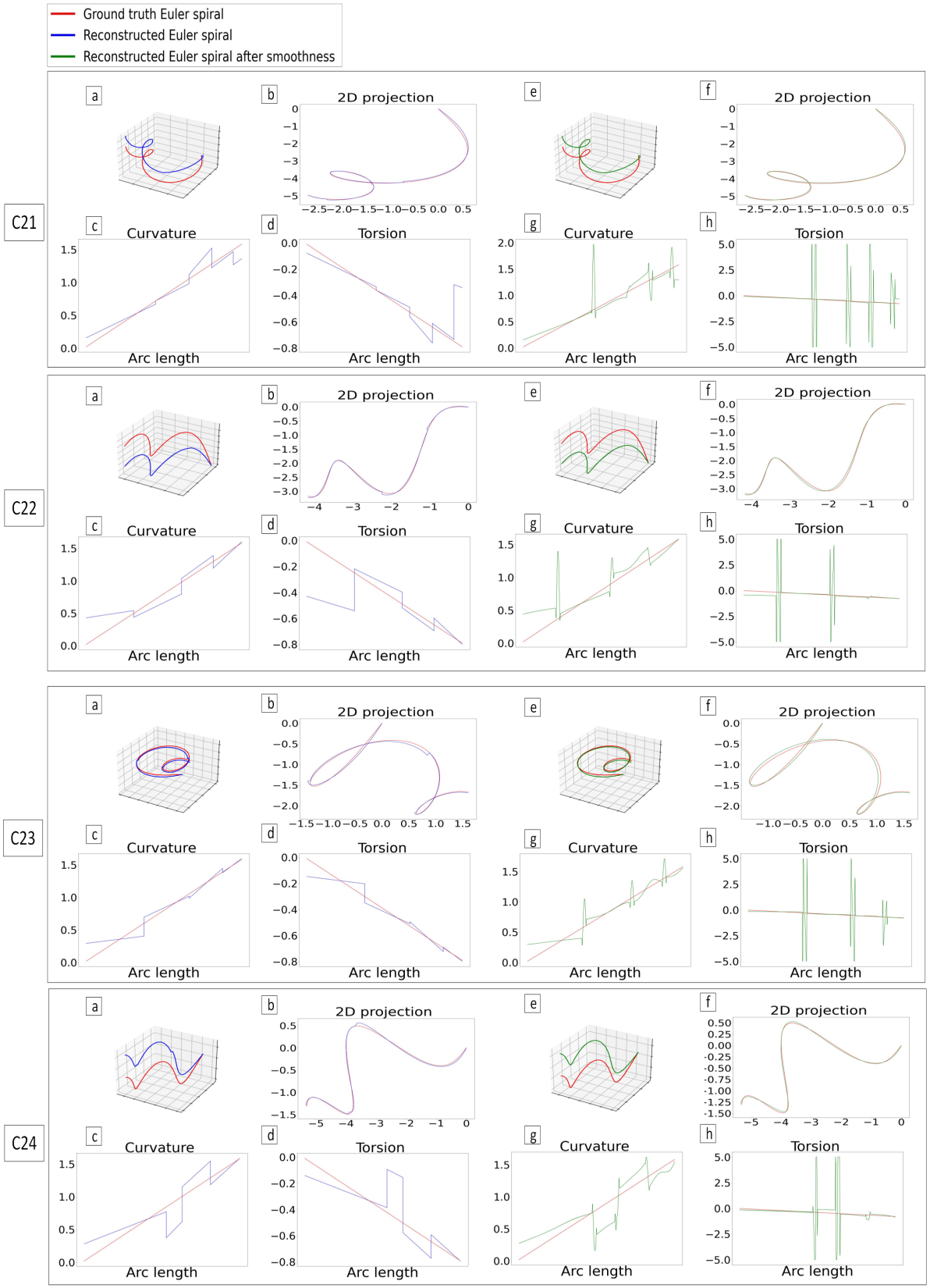

Figure A.6: Comparison of our reconstructed Euler spirals before and after applying smoothing with the ground truth 3D Euler spirals (Euler spirals with only positive curvature and torsion).

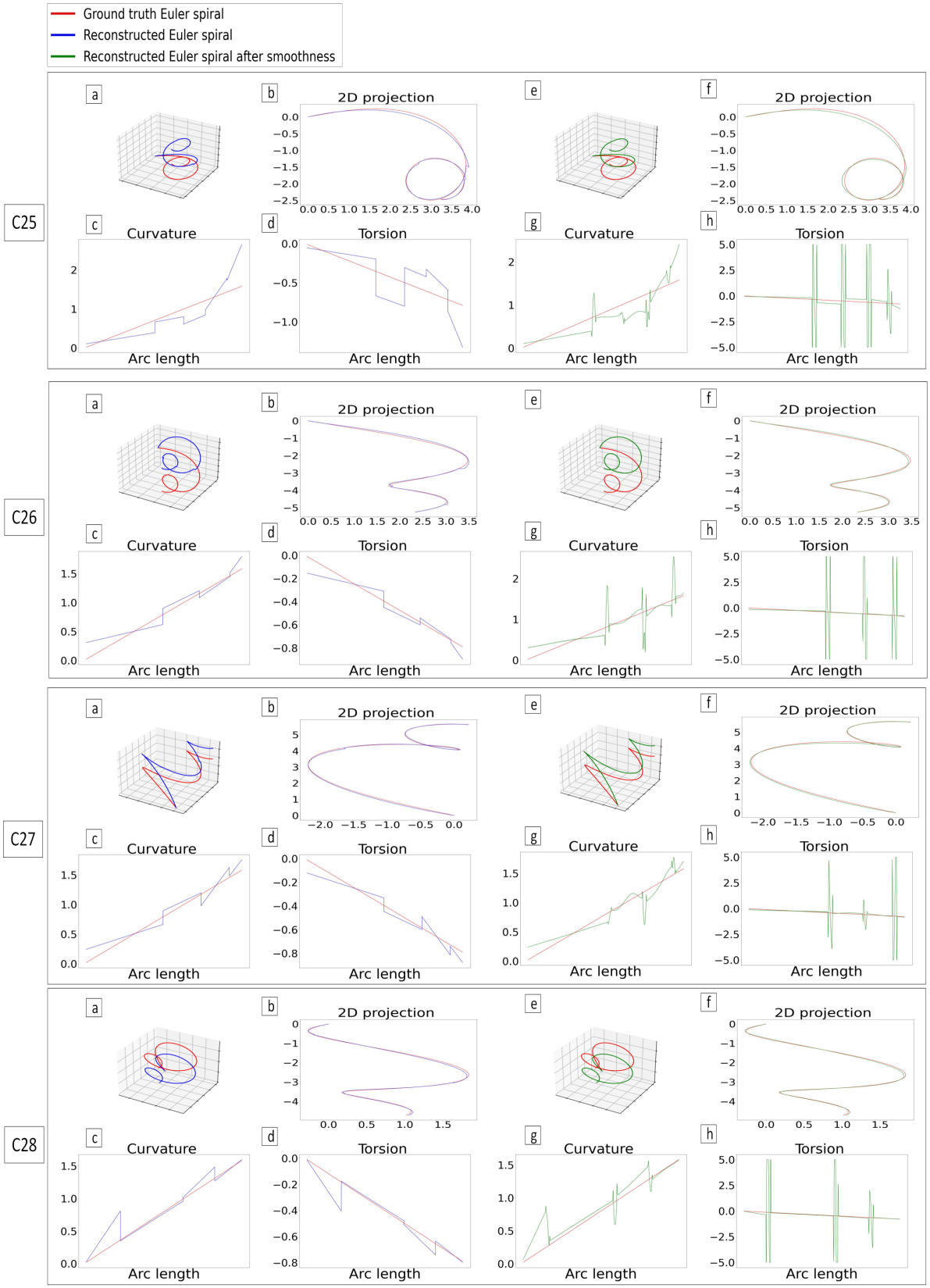

Figure A.7: Comparison of our reconstructed Euler spirals before and after applying smoothing with the ground truth 3D Euler spirals (Euler spirals with only positive curvature and torsion).

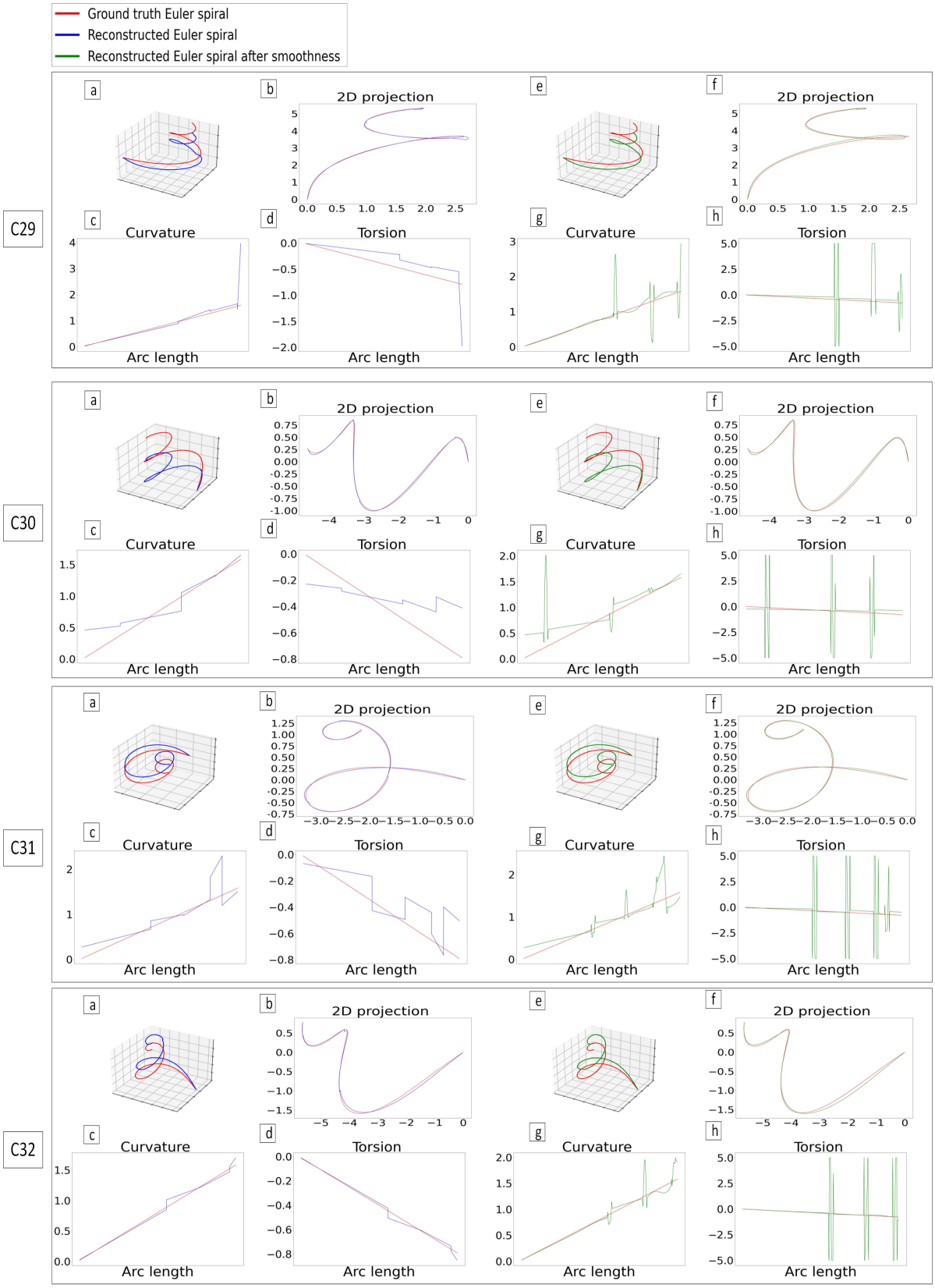

Figure A.8: Comparison of our reconstructed Euler spirals before and after applying smoothing with the ground truth 3D Euler spirals (Euler spirals with only positive curvature and torsion).

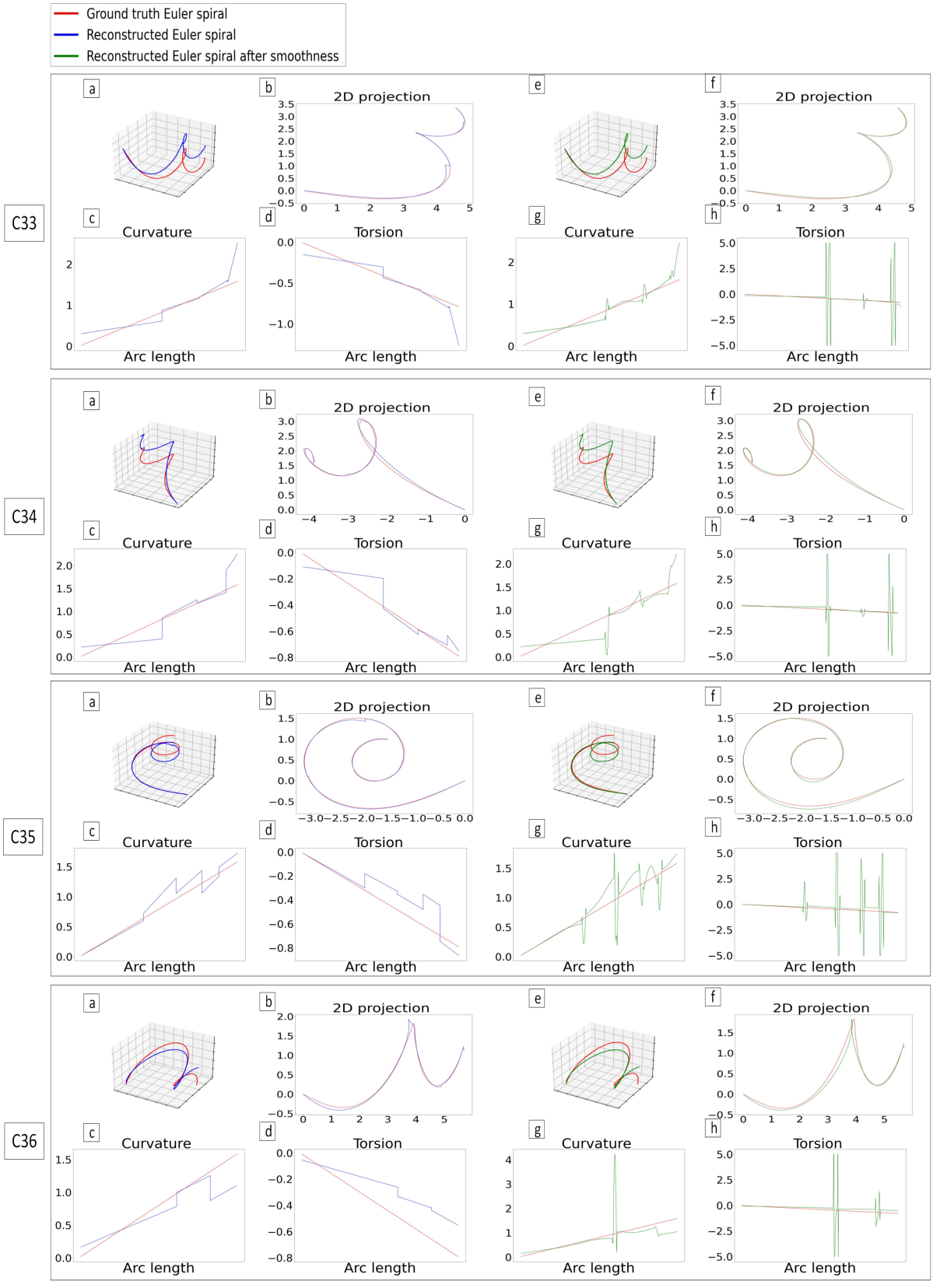

Figure A.9: Comparison of our reconstructed Euler spirals before and after applying smoothing with the ground truth 3D Euler spirals (Euler spirals with only positive curvature and torsion).

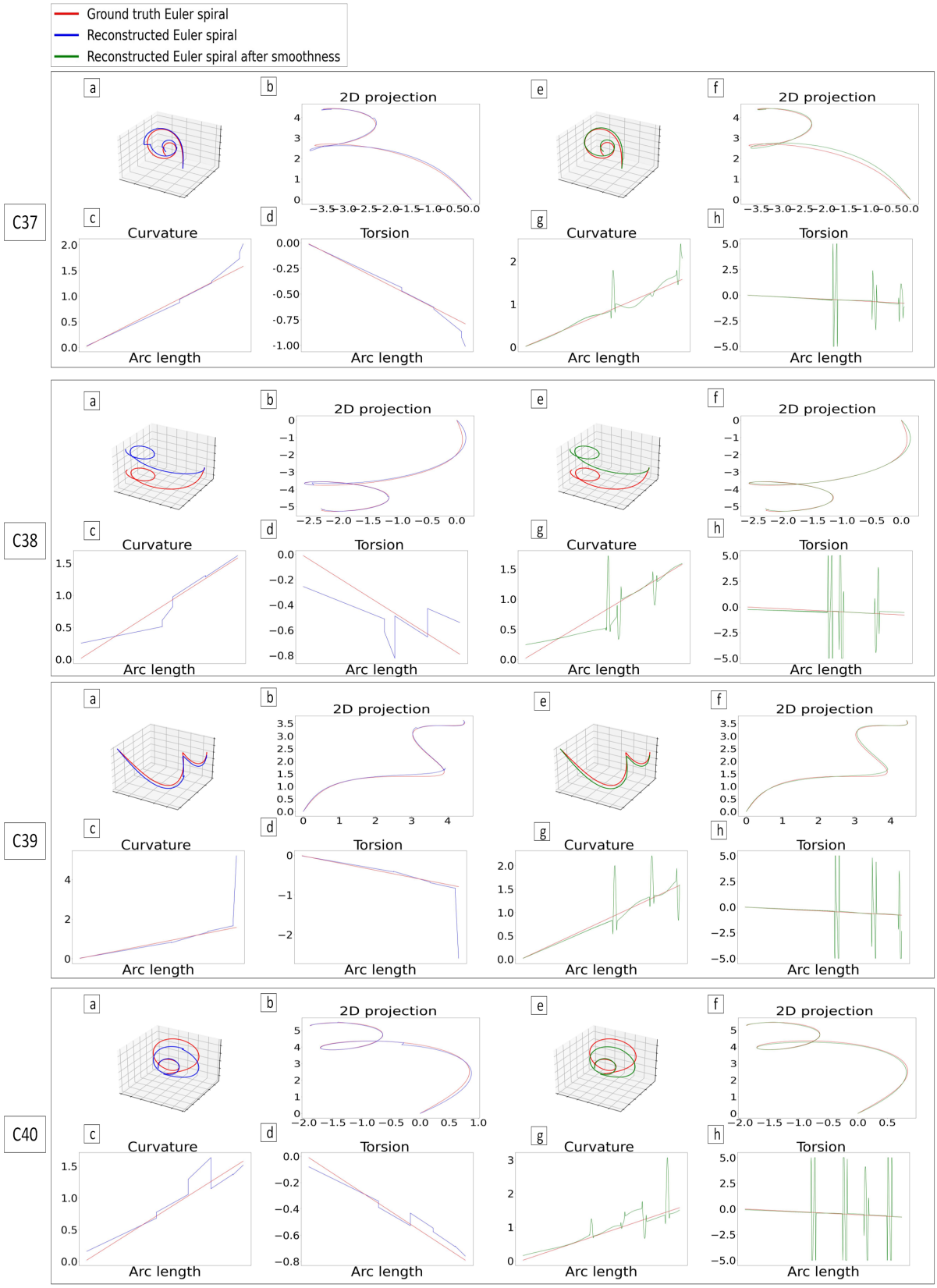

Figure A.10: Comparison of our reconstructed Euler spirals before and after applying smoothing with the ground truth 3D Euler spirals (Euler spirals with only positive curvature and torsion).

Bibliography

Ali Fakih, Frederic Cordier, Y. M. (2023), 'Piecewise reconstruction of 3d euler spirals from planar polygonal curves', *International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)* **13**(5), 01–19.
URL: *10.5121/ijcseit.2023.13501*

Alldieck, T., Magnor, M., Bhatnagar, B. L., Theobalt, C. & Pons-Moll, G. (2019), Learning to reconstruct people in clothing from a single rgb camera, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 1175–1186.

Aloisi, V., Carmignato, S., Schlecht, J. & Ferley, E. (2016), Investigation on metrological performances in ct helical scanning for dimensional quality control, *in* '6th Conference on Industrial Computed Tomography'.

Arora, R. & Singh, K. (2021), 'Mid-air drawing of curves on 3d surfaces in virtual reality', *ACM Transactions on Graphics (TOG* **40**(3), 1–17.

Bae, S.-H., Balakrishnan, R. & Singh, K. (2008), 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', pp. 151–160.

Baran, I., Lehtinen, J. & Popović, J. (2010), Sketching clothoid splines using shortest paths, *in* 'Computer Graphics Forum', Vol. 29, Wiley Online Library, pp. 655–664.

Bertails, F., Audoly, B., Cani, M.-P., Querleux, B., Leroy, F. & Lévêque, J.-

L. (2006), 'Super-helices for predicting the dynamics of natural hair', *ACM Transactions on Graphics (TOG)* **25**(3), 1180–1187.

Bhatnagar, B. L., Tiwari, G., Theobalt, C. & Pons-Moll, G. (2019), Multi-garment net: Learning to dress 3d people from images, *in* 'Proceedings of the IEEE/CVF international conference on computer vision', pp. 5420–5430.

Bing, P., Liu, W. & Zhang, Z. (2021), 'Deepcednet: an efficient deep convolutional encoder-decoder networks for ecg signal enhancement', *IEEE Access* **9**, 56699–56708.

Breiman, L. (2001), 'Random forests', *Machine learning* **45**, 5–32.

Brown, E. & Wang, P. S. (1996), Three-dimensional object recovery from two-dimensional images: a new approach, *in* 'Intelligent robots and computer vision xv: Algorithms, techniques, active vision, and materials handling', Vol. 2904, SPIE, pp. 138–147.

Cherin, N., Cordier, F. & Melkemi, M. (2014), 'Modeling piecewise helix curves from 2d sketches', *Computer-Aided Design* **46**, 258–262.

Choy, C. B., Xu, D., Gwak, J., Chen, K. & Savarese, S. (2016), 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction, *in* 'Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14', Springer, pp. 628–644.

Cohen, J. M., Markosian, L., Zeleznik, R. C., Hughes, J. F. & Barzel, R. (1999), An interface for sketching 3d curves, *in* 'Proceedings of the 1999 symposium on Interactive 3D graphics', pp. 17–21.

Cordier, F., Melkemi, M. & Seo, H. (2016), 'Reconstruction of helices from their orthogonal projection', *Computer Aided Geometric Design* **46**, 1–15.

Cordier, F., Seo, H., Melkemi, M. & Sapidis, N. S. (2013), 'Inferring mirror symmetric 3d shapes from sketches', *Comput. Aided Des* **45**(2), 301–311.

Dibra, E., Jain, H., Öztireli, C., Ziegler, R. & Gross, M. (2016), Hs-nets: Estimating human body shape from silhouettes with convolutional neural networks, *in* '2016 fourth international conference on 3D vision (3DV)', IEEE, pp. 108–117.

Dibra, E., Jain, H., Oztireli, C., Ziegler, R. & Gross, M. (2017), Human shape from silhouettes using generative hks descriptors and cross-modal neural networks, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 4826–4836.

Dickerson, R. E. (1983), 'The dna helix and how it is read', *Scientific American* **249**(6), 94–111.

Dijkstra, E. W. (1959), 'A note on two problems in connexion with graphs', *Numerische mathematik* **1**(1), 269–271.

Do, P. N. B. & Nguyen, Q. C. (2019), A review of stereo-photogrammetry method for 3-d reconstruction in computer vision, *in* '2019 19th International Symposium on Communications and Information Technologies (ISCIT)', IEEE, pp. 138–143.

Draper, N. R. & Smith, H. (1998), *Applied regression analysis*, Vol. 326, John Wiley & Sons.

Farsangi, S., Naiel, M. A., Lamm, M. & Fieguth, P. (2020), 'Rectification based single-shot structured light for accurate and dense 3d reconstruction', *Journal of Computational Vision and Imaging Systems* **6**(1), 1–3.

Fei Mai, Y. (2010), '3d curves reconstruction from multiple images', *DICTA* p. 462–467.

Feng, M., Gilani, S. Z., Wang, Y. & Mian, A. (2018), 3d face reconstruction from light field images: A model-free approach, *in* 'Proceedings of the European conference on computer vision (ECCV)', pp. 501–518.

Forterre, Y. & Dumais, J. (2011), 'Generating helices in nature', *science* **333**(6050), 1715–1716.

Frego, M. (2022), 'Closed form parametrisation of 3d clothoids by arclength with both linear varying curvature and torsion', *Applied Mathematics and Computation* **421**, 126907.

Friedman, J. H. (2001), 'Greedy function approximation: a gradient boosting machine', *Annals of statistics* pp. 1189–1232.

Gecer, B., Ploumpis, S., Kotsia, I. & Zafeiriou, S. (2021), 'Fast-ganfit: Generative adversarial network for high fidelity 3d face reconstruction', *IEEE transactions on pattern analysis and machine intelligence* **44**(9), 4879–4893.

Goriely, A. & Tabor, M. (1998), 'Spontaneous helix hand reversal and tendril perversion in climbing plants', *Physical Review Letters* **80**(7), 1564.

Gryaditskaya, Y., Hähnlein, F., Liu, C., Sheffer, A. & Bousseau, A. (2020), 'Lifting freehand concept sketches into 3d', *ACM Transactions on Graphics (TOG)* **39**(6), 1–16.

Guiqing, L., Xianmin, L. & Hua, L. (2001), 3d discrete clothoid splines, *in* 'International Conference on Computer Graphics', p. 321–324.

Gur Harary, A. T. (2012), '3d euler spirals for 3d curve completion', *Computational Geometry* **45**, 115–126.

Ham, H., Wesley, J. & Hendra, H. (2019), 'Computer vision based 3d reconstruction: A review', *International Journal of Electrical and Computer Engineering* **9**(4), 2394.

Han, Z., Ma, B., Liu, Y.-S. & Zwicker, M. (2020), 'Reconstructing 3d shapes from multiple sketches using direct shape optimization', *IEEE Transactions on Image Processing* **29**, 8721–8734.

Hartley, R. & Zisserman, A. (2003), *Multiple view geometry in computer vision*, Cambridge university press.

He, G., Sheng, C., He, H., Zhou, R., Yuan, D., Ning, X. & Ning, F. (2020), 'Mathematical and geometrical modeling of braided ropes bent over a sheave', *Journal of Engineered Fibers and Fabrics* **15**, 1558925020939726.

Hähnlein, F., Gryaditskaya, Y. & Sheffer, Alla, A. B. (2022), Symmetry-driven 3d reconstruction from concept sketches, *in* 'SIGGRAPH (Conference Paper Track', Vol. 19, p. 1–19.

Iarussi, E., Bommes, D. & Bousseau, A. (2015), 'Bendfields: Regularized curvature fields from rough concept sketches', *ACM Transactions on Graphics (TOG)* **34**(3), 1–16.

Kawagoe, Y. & Murai, N. (1996), 'A novel basic region/helix-loop-helix protein binds to a g-box motif cacgtg of the bean seed storage protein β-phaseolin gene', *Plant Science* **116**(1), 47–57.

Ke, J., Wu, Z.-y., Liu, Y.-s., Xiang, Z. & Hu, X.-d. (2020), 'Design method, performance investigation and manufacturing process of composite helical springs: A review', *Composite Structures* **252**, 112747.

Kim, Yongkwan, S.-H. B. (2016), 'Sketchingwithhands: 3d sketching handheld products with first-person hand posture', *UIST* p. 797–808.

Kimia, B. B., Frankel, I. & Popescu, A.-M. (2003), 'Euler spiral for shape completion', *International Journal of Computer Vision 54* **1**, 159–182.

Knuth, D. E. (1979), 'Mathematical typography', *Bulletin AMS 1* **2**, 337–372.

Knyaz, V. A., Kniaz, V. V. & Remondino, F. (2018), Image-to-voxel model translation with conditional adversarial networks, *in* 'Proceedings of the European Conference on Computer Vision (ECCV) Workshops', pp. 0–0.

Krs, V., Yumer, E., Carr, N., Benes, B. & Měch, R. (2017), 'Skippy: Single view 3d curve interactive modeling', *ACM Transactions on Graphics (TOG* **36**(4), 1–12.

Kwan, Kin Chung, H. F. (2019), 'Mobi3dsketch: 3d sketching in mobile ar', *CHI* p. 176.

LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998), 'Gradient-based learning applied to document recognition', *Proceedings of the IEEE* **86**(11), 2278–2324.

Levien, R. (2008), 'The euler spiral: a mathematical history', *Tech. Rep. UCB/EECS-2008-111, EECS Department, University of California* .

Lewiner, T., Gomes Jr, J. D., Lopes, H. & Craizer, M. (2005), 'Curvature and torsion estimators based on parametric curve fitting', *Computers & Graphics* **29**(5), 641–655.

Li, C.-L., Zaheer, M., Zhang, Y., Poczos, B. & Salakhutdinov, R. (2018), 'Point cloud gan', *arXiv preprint arXiv:1810.05795* .

Li, W. (2021), 'Pen2vr: A smart pen tool interface for wire art design in vr', *STAG* p. 119–127.

Lien, J. M. (1981), 'A dichotomy algorithm for computing smooth paths', *Communications of the ACM* **24**(11), 682–690.

Lin, J., Yuan, Y., Shao, T. & Zhou, K. (2020), Towards high-fidelity 3d face reconstruction from in-the-wild images using graph convolutional networks, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 5891–5900.

Lyapunov, A. M. (1992), 'The general problem of the stability of motion', *International journal of control* **55**(3), 531–534.

McClelland, J. L., Rumelhart, D. E., Group, P. R. et al. (1987), *Parallel Distributed Processing, Volume 2: Explorations in the Microstructure of Cognition: Psychological and Biological Models*, Vol. 2, MIT press.

McCrae, J. & Singh, K. (2009), 'Sketching piecewise clothoid curves', *Computers & Graphics* **33**(4), 452–461.

McCrae, J. & Singh, K. (2011), Neatening sketched strokes using piecewise french curves, *in* 'Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling', pp. 141–148.

McIlroy, D., Zhang, D., Kranov, Y. & Norton, M. G. (2001), 'Nanosprings', *Applied Physics Letters* **79**(10), 1540–1542.

Nozawa, N., Shum, H., Ho, E. & Morishima, S. (2020), Single sketch image based 3d car shape reconstruction with deep learning and lazy learning, *in* 'Proceedings of the 2020 International Conference on 3D Vision', IEEE, pp. 898–907.

Nozawa, N., Shum, H. P., Feng, Q., Ho, E. S. & Morishima, S. (2022), '3d car shape reconstruction from a contour sketch using gan and lazy learning', *The Visual Computer* pp. 1–14.

Piuze, E., Kry, P. G. & Siddiqi, K. (2011), Generalized helicoids for modeling hair geometry, *in* 'Computer Graphics Forum', Vol. 30, Wiley Online Library, pp. 247–256.

Rasmussen, C. E., Williams, C. K. et al. (2006), *Gaussian processes for machine learning*, Vol. 1, Springer.

Rutter, J. (2000), *Geometry of Curves*, CRC Press.

Saggiorato, G., Alvarez, L., Jikeli, J. F., Kaupp, U. B., Gompper, G. & Elgeti, J. (2017), 'Human sperm steer with second harmonics of the flagellar beat', *Nature communications* **8**(1), 1415.

Savitzky, A. & Golay, M. J. E. (1964), 'Smoothing and differentiation of data by simplified least squares procedures', *Analytical Chemistry* **36**, 1627–1639.

Shen, Y., Zhang, C., Fu, H., Zhou, K. & Zheng, Y. (2021), 'Deepsketchhair: Deep sketch-based 3d hair modeling', *IEEE Transactions on Visualization and Computer Graphics* **27**, 3250–3263.

Shoji, K., Kato, K. & Toyama, F. (2001), '3-d interpretation of single line drawings based on entropy minimization principle', *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **2**, 90–95.

Smith, E. J. & Meger, D. (2017), Improved adversarial systems for 3d object generation and reconstruction, *in* 'Conference on Robot Learning', PMLR, pp. 87–96.

Smola, A. J. & Schölkopf, B. (2004), 'A tutorial on support vector regression', *Statistics and computing* **14**, 199–222.

Sun, X., Wu, J., Zhang, X., Zhang, Z., Zhang, C., Xue, T., Tenenbaum, J. B. & Freeman, W. T. (2018), Pix3d: Dataset and methods for single-image 3d shape modeling, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 2974–2983.

Tachella, J., Altmann, Y., Mellado, N., McCarthy, A., Tobin, R., Buller, G. S., Tourneret, J.-Y. & McLaughlin, S. (2019), 'Real-time 3d reconstruction from single-photon lidar data using plug-and-play point cloud denoisers', *Nature communications* **10**(1), 4984.

Tachella, J., Altmann, Y., Ren, X., McCarthy, A., Buller, G. S., Mclaughlin, S. & Tourneret, J.-Y. (2019), 'Bayesian 3d reconstruction of complex scenes from single-photon lidar data', *SIAM Journal on Imaging Sciences* **12**(1), 521–550.

Tian, X., Liu, R., Wang, Z. & Ma, J. (2022), 'High quality 3d reconstruction based on fusion of polarization imaging and binocular stereo vision', *Information Fusion* **77**, 19–28.

Ullman, S. (1976), 'Filling-in the gaps: The shape of subjective contours and a model for their generation', *Biological Cybernetics* **25**, 1–6.

Walton, D. J. & Meek, D. S. (2005), 'A controlled clothoid spline', *Computers Graphics* **29**, 353–363.

Wang, F., Yang, Y., Zhao, B., Jiang, J., Zhou, T., Jiang, D. & Cai, T. (2020), 'Deep 3d shape reconstruction from single-view sketch image', *The 8th International Conference on Digital Home* pp. 184–189.

Williamson, B. (1912), 'An elementary treatise on the differential calculus: containing the theory of plane curves, with numerous examples, longmans'.

Wittung, P., Nielsen, P. E., Buchardt, O., Egholm, M. & Norde´ n, B. (1994), 'Dna-like double helix formed by peptide nucleic acid', *Nature* **368**(6471), 561–563.

Wood, E., Baltrušaitis, T., Hewitt, C., Johnson, M., Shen, J., Milosavljević, N., Wilde, D., Garbin, S., Sharp, T., Stojiljković, I. et al. (2022), 3d face reconstruction with dense landmarks, *in* 'Computer Vision–ECCV 2022: 17th European

Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XIII', Springer, pp. 160–177.

Wu, J., Zhang, C., Xue, T., Freeman, B. & Tenenbaum, J. (2016), 'Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling', *Advances in neural information processing systems* **29**.

Wu, J., Zhang, C., Zhang, X., Zhang, Z., Freeman, W. T. & Tenenbaum, J. B. (2018), Learning shape priors for single-view 3d completion and reconstruction, *in* 'Proceedings of the European Conference on Computer Vision (ECCV)', pp. 646–662.

Xie, H., Yao, H., Sun, X., Zhou, S. & Zhang, S. (2019), Pix2vox: Context-aware 3d reconstruction from single and multi-view images, *in* 'Proceedings of the IEEE/CVF international conference on computer vision', pp. 2690–2698.

Xu, B., Chang, W., Sheffer, A., Bousseau, A., McCrae, J. & Singh, K. (2014), 'True2form: 3d curve networks from 2d sketches via selective regularization', *ACM Transactions on Graphics* **33**(4).

Xu, P., Hospedales, T. M., Yin, Q., Song, Y.-Z., Xiang, T. & Wang, L. (2022), 'Deep learning for free-hand sketch: A survey', *IEEE transactions on pattern analysis and machine intelligence* **45**(1), 285–312.

Yang, B., Rosa, S., Markham, A., Trigoni, N. & Wen, H. (2018), 'Dense 3d object reconstruction from a single depth view', *IEEE transactions on pattern analysis and machine intelligence* **41**(12), 2820–2834.

Yang, G., Cui, Y., Belongie, S. & Hariharan, B. (2018), Learning single-view 3d reconstruction with limited pose supervision, *in* 'Proceedings of the European Conference on Computer Vision (ECCV)', pp. 86–101.

Yang, H., Tian, Y., Yang, C., Wang, Z., Wang, L. & Li, H. (2022), 'Sequential learning for sketch-based 3d model retrieval', *Multimedia Systems* pp. 1–18.

Yang, L., Wu, J., Huo, J., Lai, Y.-K. & Gao, Y. (2021), 'Learning 3d face reconstruction from a single sketch', *Graphical Models* **115**, 101–102.

Ye, H., Kwan, K. & Fu, H. (2021), '3d curve creation on and around physical objects with mobile ar', *IEEE transactions on visualization and computer graphics* **28**(8), 2809–2821.

Yu, E., Arora, R., Stanko, T., Bærentzen, J. A. & Singh, Karan, A. B. (2021), 'Cassie: Curve and surface sketching in immersive environments', *CHI* **190**, 1–190.

Yu, X., DiVerdi, S., Sharma, A. & I, Yotam, G. (2021), 'Scaffoldsketch: Accurate industrial design drawing in vr', *UIST* p. 372–384.

Yue, Y.-T., Zhang, X., Yang, Y.-L., Ren, G. & Choi, Yi-King, W. W. (2017), 'Wiredraw: 3d wire sculpturing guided with mixed reality', *CHI* p. 3693–3704.

Zhang, B. & Wonka, P. (2022), Training data generating networks: Shape reconstruction via bi-level optimization, *in* 'International Conference on Learning Representations'.

Zhang, J., Chen, X., Cai, Z., Pan, L., Zhao, H., Yi, S., Yeo, C. K., Dai, B. & Loy, C. C. (2021), Unsupervised 3d shape completion through gan inversion, *in* 'Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition', pp. 1768–1777.