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

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.

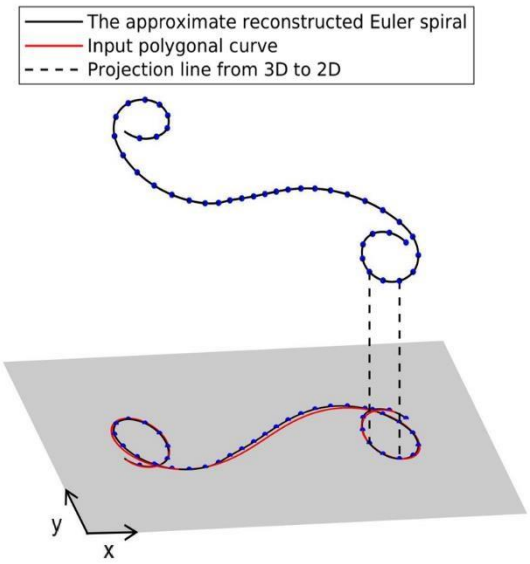

Figure 1: Starting from a planar polygonal curve (red curve) as an input we generate an approximation piecewise 3D Euler spiral, whose orthogonal projection on the plane z = 0 is close to the input polygonal curve.

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.

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

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.

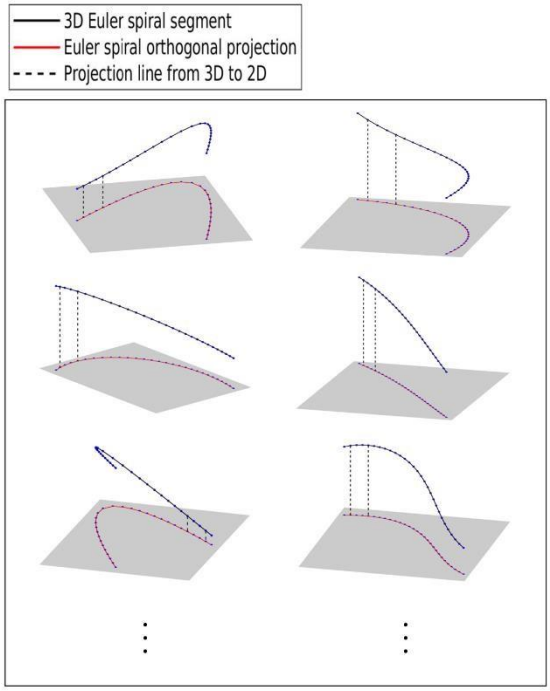

Figure 2: Visualization of our dataset. Segments of 3D Euler spirals with their planar polygonal projection.

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:

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

where L is the arc length, $s \in [0, L]$, $\kappa_s = \frac{\partial \kappa}{\partial s}$, and $\kappa_\tau = \frac{\partial \tau}{\partial s}$. 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 $0 \leq s \leq L$:

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

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.

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 $[\frac{\pi}{4}, \pi]$. 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*.

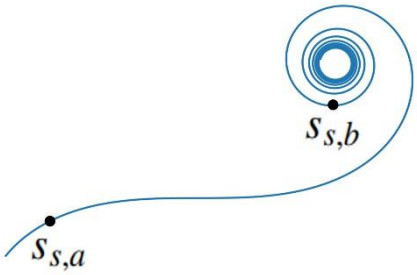

Figure 3: The starting point of all dataset segments is between $s_s \in [s_{s,a}, s_{s,b}]$.

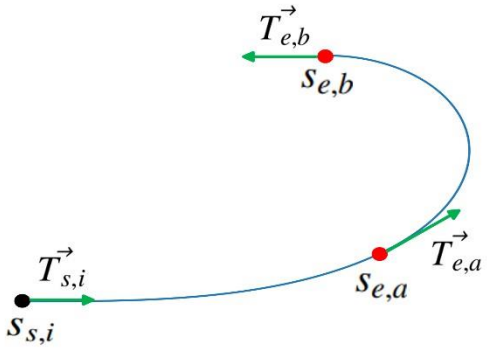

Figure 4: The ending point parameter of segment i that have a starting point s_s is between $[s_e, a, s_e, b]$.

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:

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

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

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: $S = \{p_1, p_2, \dots, p_i, \dots, p_n\}$, 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.

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.

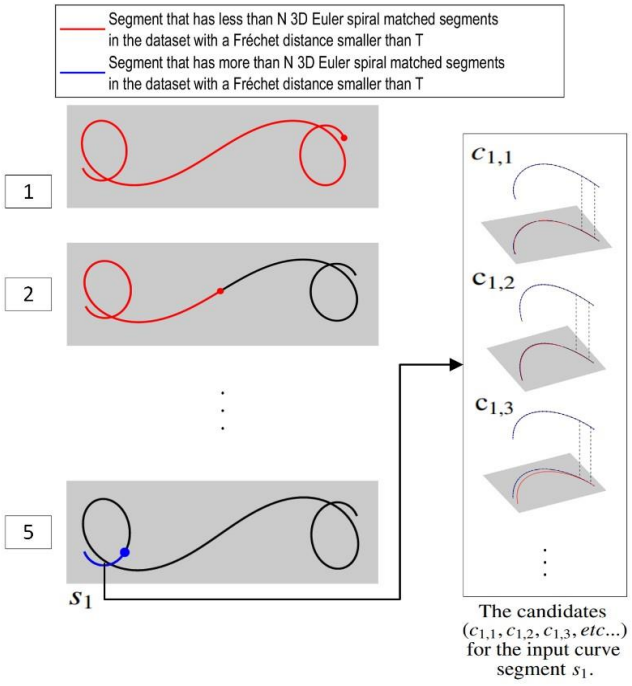

Figure 5: The process to find the first split segment of the input curve. In (1), the segmentation algorithm starts with the entire curve. In (2), the segmentation is done recursively until the number of matching candidates is larger or equal to N. In (5), the s the segmentation is finished for the first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

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.

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 $c_{m,f} \in C_{sm}$.

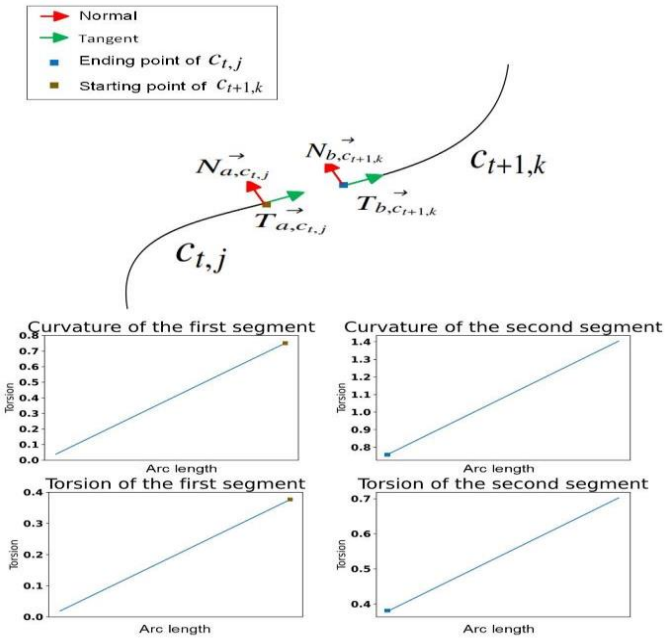

Figure 6: Visualization of the parameters that we consider while assembling two 3D Euler spiral curve segments.

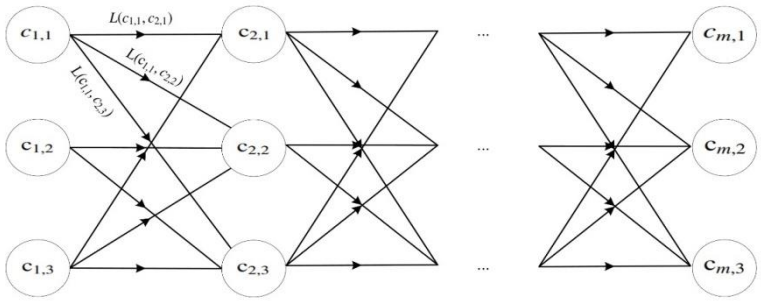

Figure 7: The architecture of the Dijkstra algorithm with $N = 3$, the nodes represent the segment candidates and the edges represent the loss function between the candidates.

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.

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.

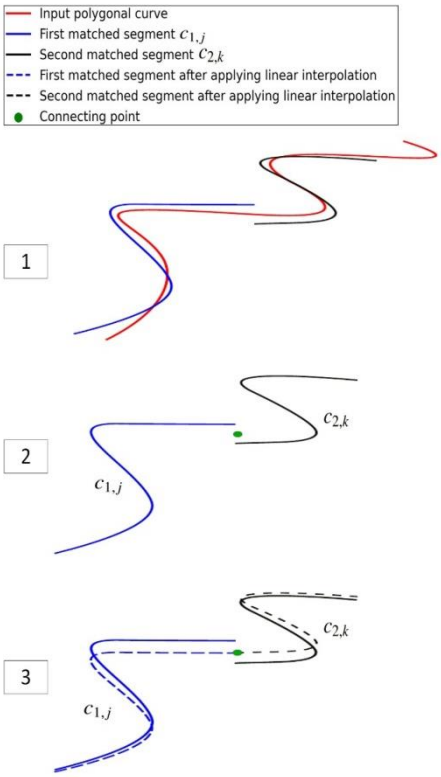

Figure 8: The initial position of the orthogonal projection of the first two matched curves that will be used for the reconstruction of the piecewise Euler spiral is shown in (1) and (2) where there's some gap between them. In (3), after applying linear interpolation the two matched segments intersect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

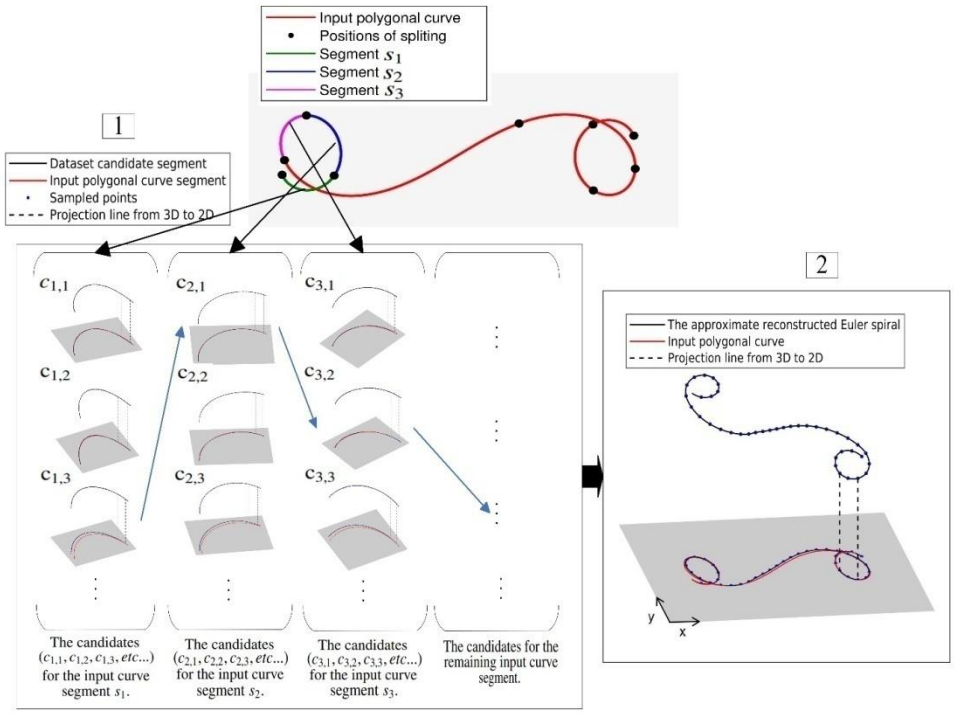

Figure 9: An overview of our approach. (1) Starting from a planar polygonal curve as an input, we split it into segments Then for each segment, we get its closest matched Euler spiral segments (candidates) from the dataset. (2) We select the set of candidates (one per each segment) that can link with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

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

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.

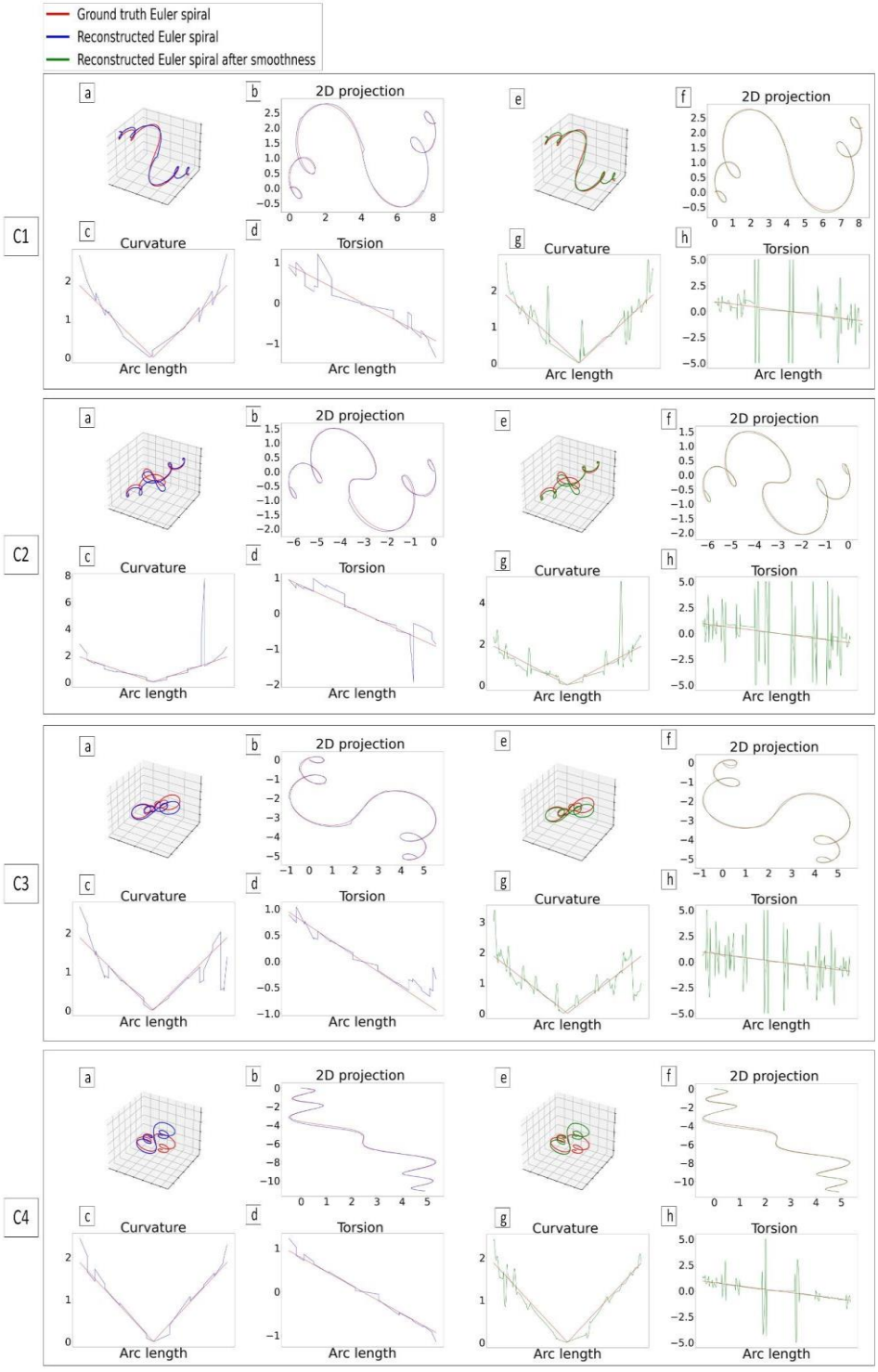

Figure 10: Comparison of our reconstructed Euler spirals before and after applying smoothing with the ground truth 3D Euler spirals (Euler spirals that include an inflection point).

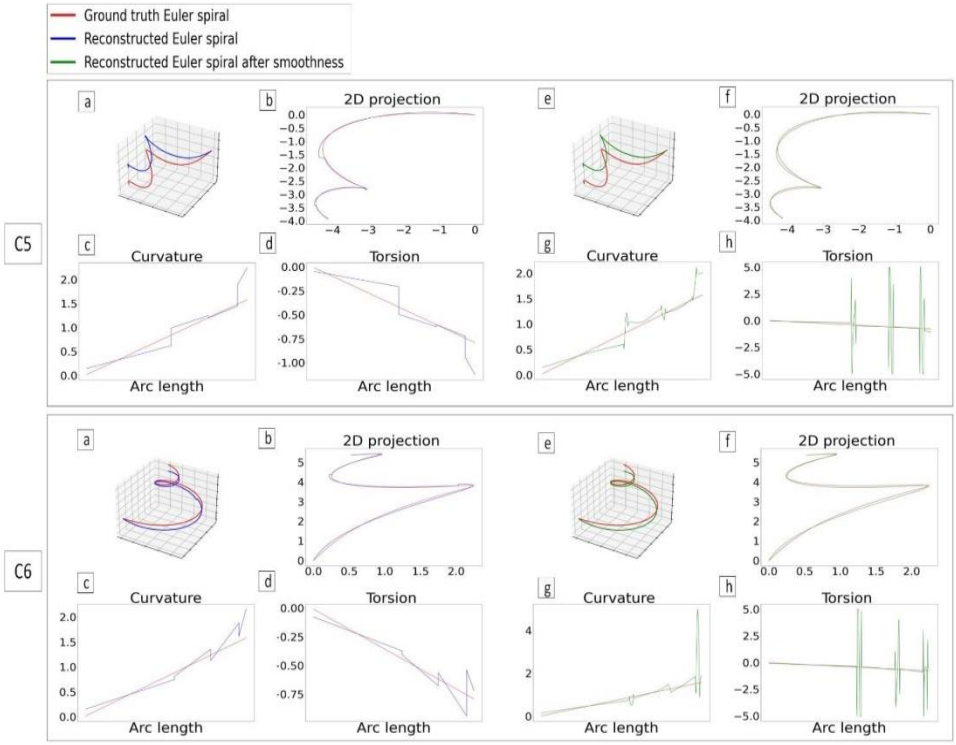

Figure 11: Comparison of our reconstructed Euler spirals before and after applying smoothing with the ground truth 3D Euler spirals (Euler spirals with only positive curvature and torsion).

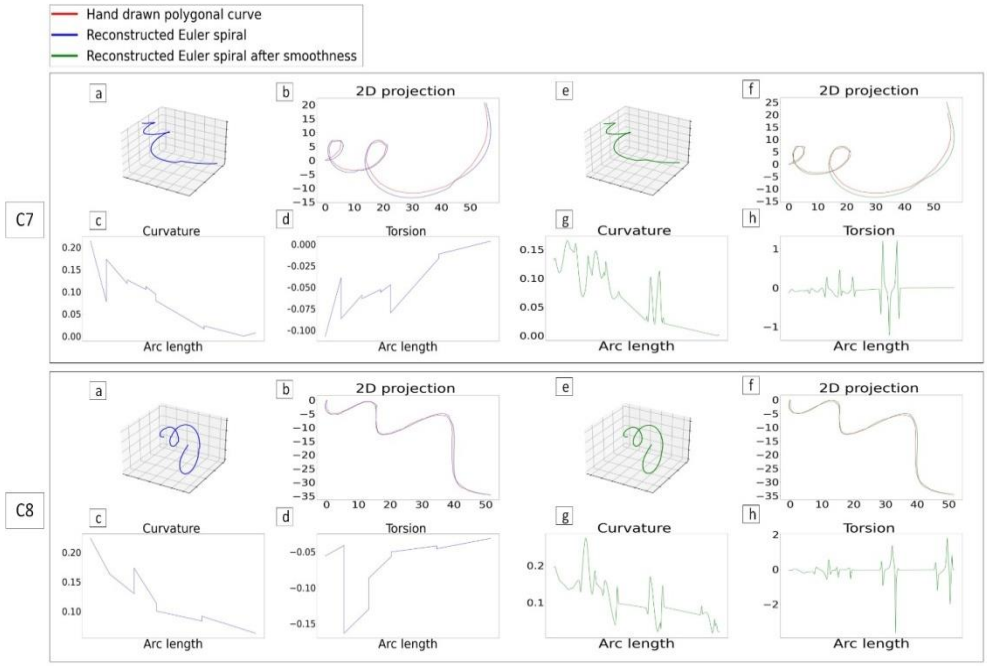

Figure 12: Reconstruction of approximate piecewise 3D Euler spirals from hand-drawn polygonal curves.

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.

[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.