Pacific Graphics (2025)
S.-S. Lin, T. Schneider, and E. Zhang (Editors)

3D Curve Development with Crossing and Twisting from 2D Drawings

Aurick Daniel Franciskus Setiadi[1], Jeng Wen Joshua Lean[1], Hao-Che Kao[1], and Shih-Hsuan Hung[1]

[1]National Tsing Hua University, Taiwan

Abstract

Designing 3D curves with specified crossings and twistings often requires tedious view adjustments. We present a 3D curve development from 2D drawing with controlled crossings and twistings. We introduce a two-strand 2D diagram that lets users sketch with explicit crossing and twisting assignments. The system extracts feature points from the 2D diagram and uses them as 3D control points. It assigns the heights and over/under relationships of the control points via an optimization and then generates twisted 3D curves using B-splines. An interactive interface links the 2D diagram to the evolving 3D curves, enabling real-time iteration. We validate our method on diverse sketches, compare it with traditional 3D curve construction, and demonstrate its utility for elastic wire art via physics-based animation.

CCS Concepts

• ***Computing methodologies*** → ***Shape modeling; Graphics systems and interfaces;***

1. Introduction

2D curves with crossing points are common in ornamentation and design. They and their 3D counterparts are widely used in engineering and artistic applications. Twisting a 3D curve can guide textures around it as a tube or direct objects along it as a path. Conventional spline tools make it difficult to achieve specific crossings or twistings, since users must frequently change the camera view and drag control points. Recently, VR tools such as Google's Tilt Brush allow users to draw 3D curves directly in a virtual environment. However, these tools require additional equipment and make twisting a 3D curve difficult.

To address this, we propose a 3D curve development that automatically generates 3D shapes from 2D drawings. Users can freely sketch 2D curves on their screens and manipulate their crossings and twistings easily on our two-strand diagram representation. We then extract feature points such as crossing points, twisting points, curvature extrema. From these points, we construct 3D candidate curves that pass over or under one another by solving an optimization problem constrained by the desired crossing locations. Finally, we assign a local frame at each point by integrating user-specified twisting angles.

We provide an interactive interface that lets users draw curves and edit crossings and twists directly in 2D while viewing the resulting 3D form in real time. We demonstrate our system on various artistic curves (Figure 1(a)) and on physics-based animations of elastic wires (Figure 1(b)). We also conduct a user study to validate the efficiency of our method.

2. Methodology

Given a 2D hand drawing, our system first converts the sketching to a set of cubic Bézier curves with interpolating cubic B-spline [Kno99] (Figure 2 (b)). It then represents the splines with the features as a two-strand diagram for users to design the crossing and twisting along the curves (Figure 2 (c)). For instance, users can decide curve segments over or under others and twist the curve in different directions. In the generation stage, our system extracts the feature points on the 2D curves and samples 3D points on those feature points (Figure 2 (d)). Our system then constructs the 3D curves by arranging the crossing points with the desired crossing as highlighted in the zoom-in box of Figure 2 (e), and integrating the twisting angles from the diagram as illustrated by the arrows.

2.1. 2D Curve Analysis

To better construct 3D curves, we analyze the 2D curves made of cubic Bézier curves and extract the feature points as the control points of the 3D curves. We compute the *inflections* where the winding direction of the spline changes and *cusps* where there is a sharp corner on the spline with the canonical form [SD89]. We also extract the extreme curvature points and intersection points (the crossing points) on the curves. Next, we construct the 3D curves with the feature points.

2.2. 3D Curve Construction

We construct 3D curves with interpolating B-splines [Kno99] that pass through the feature points captured in the input 2D curves. For

DOI: 10.2312/pg.20251306

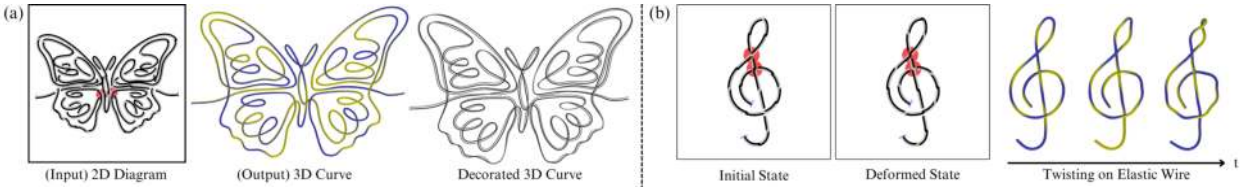

Figure 1: *We (a) show the decorated curves with textures or surrounding line patterns designed by our 3D curve development, and (b) demonstrate a physically-based animation driven by the 2D diagrams.*

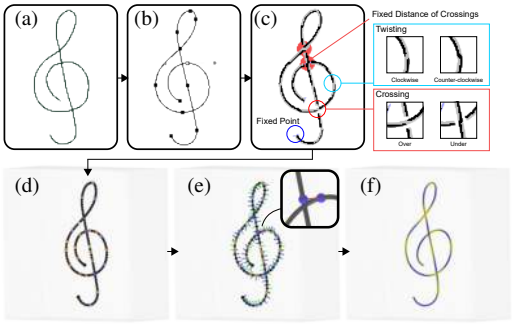

Figure 2: *This figure illustrates the pipeline of developing a 3D curve from 2D drawings.*

the crossings, we determine the heights of the feature points, $\mathbf{z}$, via an optimization. For each curve, users specify a target curve height ζ and may fix selected points $z_i \in \mathbf{Z}$ to prescribed heights $\bar{z}_i$. We collect crossing-point pairs from the 2D diagram, (z_i, z_j), and label them as over $\mathbf{O}$, under $\mathbf{U}$, or distance-fixed $\mathbf{D}$. Users can also set the separation distance $d_{i,j}$. To obtain smooth 3D curves, we introduce a Laplacian term [SCOL*04] and solve the following problem:

$$\begin{aligned} \mathbf{z}^\star = \arg\min_{\mathbf{z}} & \sum_{i=1}^{n} \left(z_i - \sum_{j \in \mathcal{N}_i} \frac{w_{ij}}{W_i} z_j \right)^2 + \lambda \sum_{i=1}^{n} (z_i - \zeta)^2 \\ \text{s.t.} \quad & z_i - z_j \geq d_{i,j} && \text{for } (z_i, z_j) \in \mathbf{O}, \\ & z_i - z_j \leq -d_{i,j} && \text{for } (z_i, z_j) \in \mathbf{U}, \\ & |z_i - z_j| = d_{i,j} && \text{for } (z_i, z_j) \in \mathbf{D}, \\ & z_i = \bar{z}_i && \text{for } z_i \in \mathbf{Z}. \end{aligned} \tag{1}$$

We set $\lambda = 3$ and compute the Laplacian term using the connected neighbors $\mathcal{N}_i$ of each feature point with w_{ij} as their distance in the 2D curve and $W_i = \sum_{j \in \mathcal{N}_i} w_{ij}$. We solve the equation by turning it into a least squares problem with a linear constraint and using the quadratic programming to obtain the results. Finally, we interpolate the 3D points with interpolating B-splines.

For the twistings, we apply the linear interpolation between the twist points. The two-strand diagram lets us estimate the desired twisting by counting strand crossings. Users mark clockwise $(+)$ or counter-clockwise $(-)$ twist points on the two-strand diagram; between identical signs we impose a π rotation, whereas opposite signs impose two opposing $\pi/2$ rotations, preventing extra twists except one additional point on odd-twist loops. The local frames of the 3D curves are then built by extending the Frenet frames of the input 2D curves to 3D, rotating normals to follow new tangents, then applying the integrated twist angle.

3. Results and Evaluation

Our system enables decoration of 3D curves generated from 2D drawings and supports physics-based animations. For the computational performance, 3D curve construction average 0.13 seconds from 20 curves with total 1,068 points. We conduct a user study to evaluate the efficiency of 3D curve development with crossings and twistings. We invite 10 participants and ask them to create 3D curves in both Blender and our system. A design is considered valid if its Fréchet distance to target curves is at most 0.5. On average, the participants spend 18 and 10 minutes on Blender and our system, respectively. We also ask the participants to rate our two-strand diagram and curve editing on a 1 to 5 scale. The clarity of crossings and twistings in our two-strand diagrams received scores of 4.2 and 4.3, respectively, while the ease of editing crossings and twistings received scores of 4.2 and 3.9.

4. Conclusions

We present a 3D curve development from 2D drawing with controlled crossings and twists. Users begin by freely sketching 2D curves easily; our system then automatically lifts them into 3D space, resolving crossing points via an optimization. Our approach handles a wide variety of input shapes and supports the design of elastic wires. Beyond curve design, we envision several extensions. For instance, by sweeping or lofting along these optimized 3D paths, one can generate developable surfaces for wire-art fabrication.

5. ACKNOWLEDGMENTS

We are grateful to the anonymous reviewers. The work was supported in part by the Ministry of Science and Technology of Taiwan (113-2222-E-007-006 and 114-2221-E-007-115).

References

[Kno99] KNOTT G. D.: *Interpolating cubic splines*, vol. 18. Springer Science & Business Media, 1999. 1

[SCOL*04] SORKINE O., COHEN-OR D., LIPMAN Y., ALEXA M., RÖSSL C., SEIDEL H.-P.: Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (2004), pp. 175–184. 2

[SD89] STONE M. C., DEROSE T. D.: A geometric characterization of parametric cubic curves. *ACM Transactions on Graphics (TOG) 8*, 3 (1989), 147–163. 1