CASSIE: Curve and Surface Sketching in Immersive Environments

Supplemental Materials

Emilie Yu
Inria, Université Côte d'Azur
Sophia-Antopolis, France
emilie.yu@inria.fr

Rahul Arora
University of Toronto
Toronto, Ontario, Canada
arorar@dgp.toronto.edu

Tibor Stanko
Inria, Université Côte d'Azur
Sophia-Antopolis, France
tibor.stanko@inria.fr

J. Andreas Bærentzen
Technical University of Denmark
Kongens Lyngby, Denmark
janba@dtu.dk

Karan Singh
University of Toronto
Toronto, Ontario, Canada
karan@dgp.toronto.edu

Adrien Bousseau
Inria, Université Côte d'Azur
Sophia-Antopolis, France
adrien.bousseau@inria.fr

This document provides additional implementation details of our VR sketching system, and additional results from the user study.

1 MATHEMATICAL FORMULATION OF CURVE BEAUTIFICATION

Intersection constraint. To constrain the poly-Bézier curve $B(t)_{t\in[0,1]}$ to intersect a point p_{target}, we first compute the closest point on the curve from p_{target} :

$$p^* = B(t^*); \quad \text{where } t^* = \arg\min_t \|B(t) - p_{\text{target}}\|. \tag{1}$$

Then, if we are close to an existing control point P_0^κ, such that: $\|P_0^\kappa - p^*\| < \delta/2$, we constrain that control point. Otherwise, we split the input curve at t^* using de Casteljau's algorithm. This yields a new control point on the curve $P_0^\kappa = p^*$. In both cases, we express the hard constraint c as:

$$c = P_0^\kappa - p_{\text{target}} = 0. \tag{2}$$

We rely on the fidelity energy and our greedy constraint selection strategy to reject constraints that would deform the input too much, and to select which constraint to apply at a control point if there are more than one.

Closed curve constraint. We express the constraint that enables the creation of closed loops the same way as an intersection constraint between the endpoints of the curve: P_0^0 the first control point and P_3^{K-1} the last (K being the total number of cubic Bézier in the poly-Bézier curve):

$$c = P_0^0 - P_3^{K-1} = 0. \tag{3}$$

G^1 continuity constraint. We want the poly-Bézier curve to have G^1 continuity even after the beautification process. That is, we want to satisfy some control point equality and tangent alignment between successive Bézier segments:

$$g = P_3^{k-1} - P_0^k = 0\,, \quad \text{for } k \in [1, K-1], \text{ and} \tag{4}$$

$$g = \frac{(P_3^{k-1} - P_2^{k-1})}{\|P_3^{k-1} - P_2^{k-1}\|} - \frac{(P_1^k - P_0^k)}{\|P_1^k - P_0^k\|} = 0\,, \quad \text{for } k \in [1, K-1]. \tag{5}$$

For the C^0 constraint (Equation 4), we simply remove all control points variables P_0^k, for $k \in [1, K-1]$ from the optimization.

For the G^1 constraint (Equation 5), we linearize this constraint by approximating the norms of the control polygon edges by the initial norms: $\|P_i^k - P_{i-1}^k\| \approx \|\bar{P}_i^k - \bar{P}_{i-1}^k\|$. Making this constraint linear enables us to solve the optimization efficiently and gives reasonable results in our use case.

Tangent alignment constraint. We encourage the tangent at a control point on the curve P_0^k to align with a target direction T_{target} using the soft constraint:

$$E_{\text{tangent}} = \|(P_1^k - P_0^k) \times T_{\text{target}}\|^2. \tag{6}$$

Planarity constraint. We encourage all control points to lie in a plane with normal vector $\vec{n}$:

$$E_{\text{planar}} = \sum_{i\in\{0,1,2\},\ k\in[0,K-1]} \|(P_{i+1}^k - P_i^k) \cdot \vec{n}\|^2. \tag{7}$$

Fidelity energy. Our formulation of the fidelity energy extends the *Projection accuracy* energy described by Xu et al. [4], to compare 3D curves, instead of 2D curves.

We want to minimize both variation in absolute position of the control points P_i^k from the input positions $\bar{P}_i^k$ and variation in the slope of Bézier polygon edges $e_{ik} = P_{i+1}^k - P_i^k$ (see Fig. 1). This leads us to define the *fidelity energy* as

$$E_{\text{fidelity}} = \frac{1}{|P_i^k|\delta^2} \sum_{i,k} \|P_i^k - \bar{P}_i^k\|^2 + \frac{1}{|e_{ik}|} \sum_{i,k} \frac{\|(P_{i+1}^k - P_i^k) - (\bar{P}_{i+1}^k - \bar{P}_i^k)\|^2}{\|\bar{P}_{i+1}^k - \bar{P}_i^k\|^2}. \tag{8}$$

We normalize each term by, respectively, the total number of control points $|P_i^k| = 3K + 1$ and the total number of Bézier control polygon edges $|e_{ik}| = 3K$, K being the number of cubic Bézier. As our Bézier curves are all arbitrarily subdivided, depending on both input stroke curvature variations and the number of intersection constraints applied, we normalize these terms so that we can later

CHI '21, May 8–13, 2021, Yokohama, Japan

ACM ISBN 978-1-4503-8096-6/21/05…$15.00
https://doi.org/10.1145/3411764.3445158

compare fidelity energy between different candidate results. To control the scale of the energy, we normalize the first term by the proximity threshold δ (with δ from Sec. 4.4 in the main document) and the second term by the initial lengths of the control polygon edges $\bar{P}^k_{i+1} - \bar{P}^k_i$. This also accounts for the uneven lengths of the Bézier control polygon edges, allowing greater deviation for longer edges, as in Xu et al. [4].

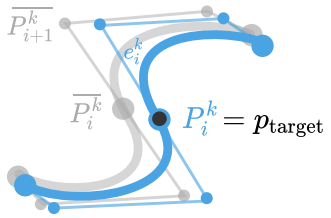

Figure 1: Input stroke (gray) and optimized stroke (blue) to match the intersection constraint p_{target}.

2 SOLVING THE OPTIMIZATION PROBLEM

The complete optimization problem that we solve, for the control points position variables $\{P^k_i\}, i \in [0,3]\ ,\ k \in [0, K-1]$, the m_h hard constraints and the m_s soft constraints is:

$$\min_{\{P^k_i\}} E_{\text{fidelity}}(P^k_i) + \sum_{j \in \{0,\ldots,m_s-1\}} E_j(P^k_i) \qquad (9)$$
$$st.\ \ c_l(P^k_i) = 0\ ,\ \text{for } l \in \{0,\ldots,m_h-1\}.$$

We define P as a column vector with $3(3K+1)$ lines, corresponding to the control points position variables, with one line per control point coordinate.

For each hard constraint $l \in \{0,\ldots,m_h-1\}$, we have n linear equations, such that we can write the constraint as: $c_l = C_l P = 0$, C_l a $n \times 3(3K+1)$ matrix.

The objective function has a linear gradient, and all hard constraints are linear so we can find the solution to this optimization problem by solving a linear system (Equation 10), using the Lagrange multipliers method [2]:

$$\begin{bmatrix} A & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} P \\ \Lambda \end{bmatrix} = \begin{bmatrix} B \\ 0 \end{bmatrix}. \qquad (10)$$

We build A and B such that $AP - B = \frac{\partial}{\partial P}(E_{\text{fidelity}} + \sum E_j)$. The matrix C corresponds to the vertically stacked C_l blocks of the hard constraints. Λ is the vector of Lagrange multipliers.

For the linear solve, we use an LU factorization implemented by the MathNET Numerics library [3]. On Intel machines, we use the Math Kernel Library [1] as our BLAS backend to speed up the linear algebra computations.

3 BEAUTIFICATION OF STRAIGHT LINES

We employ a simpler algorithm to beautify straight strokes.

3.1 Selecting the constraints

Since a line is uniquely defined by two points, we cannot apply more than two intersection constraints along a line segment. We use a simple heuristic to select the best two constraints, based on the assumption that users would not draw a stroke longer than it needs to be. Following this assumption, we select the two constraints that are closest to the stroke endpoints. In addition, we use the angular threshold θ (see Sec. 4.4 in the main document) to determine whether a line segment is close enough to one of the three world axes to be constrained to align with it.

3.2 Applying the constraints

To constrain a line segment to one or two intersection constraints, we simply update one or both endpoints such that the line segment passes through the constraint(s). If the constraint is close enough to one of the endpoints (within a distance δ, with δ from Sec. 4.4 in the main document), we set the endpoint to be at the constraint position. If there are two intersection constraints and they are not near the endpoints, we project both endpoints on the line formed by the intersection constraints. Finally, if there is one intersection constraint that is not near one of the endpoints, we translate the line segment so that it passes through the constraint.

4 ADDITIONAL USER STUDY RESULTS

Fig. 2 shows the results from the six non-professional user study participants.

REFERENCES

[1] Intel Corporation. 2009. *Intel Math Kernel Library. Reference Manual.* Intel Corporation.
[2] Jorge Nocedal and Stephen Wright. 2006. *Numerical optimization.* Springer Science & Business Media, Berlin-Heidelberg, Germany.
[3] Christoph Rüegg et al. 2009. MathNET Numerics. https://numerics.mathdotnet.com/.
[4] Baoxuan Xu, William Chang, Alla Sheffer, Adrien Bousseau, James McCrae, and Karan Singh. 2014. True2Form: 3D curve networks from 2D sketches via selective regularization. *ACM Transactions on Graphics* 33, 4, Article 131 (2014), 13 pages.

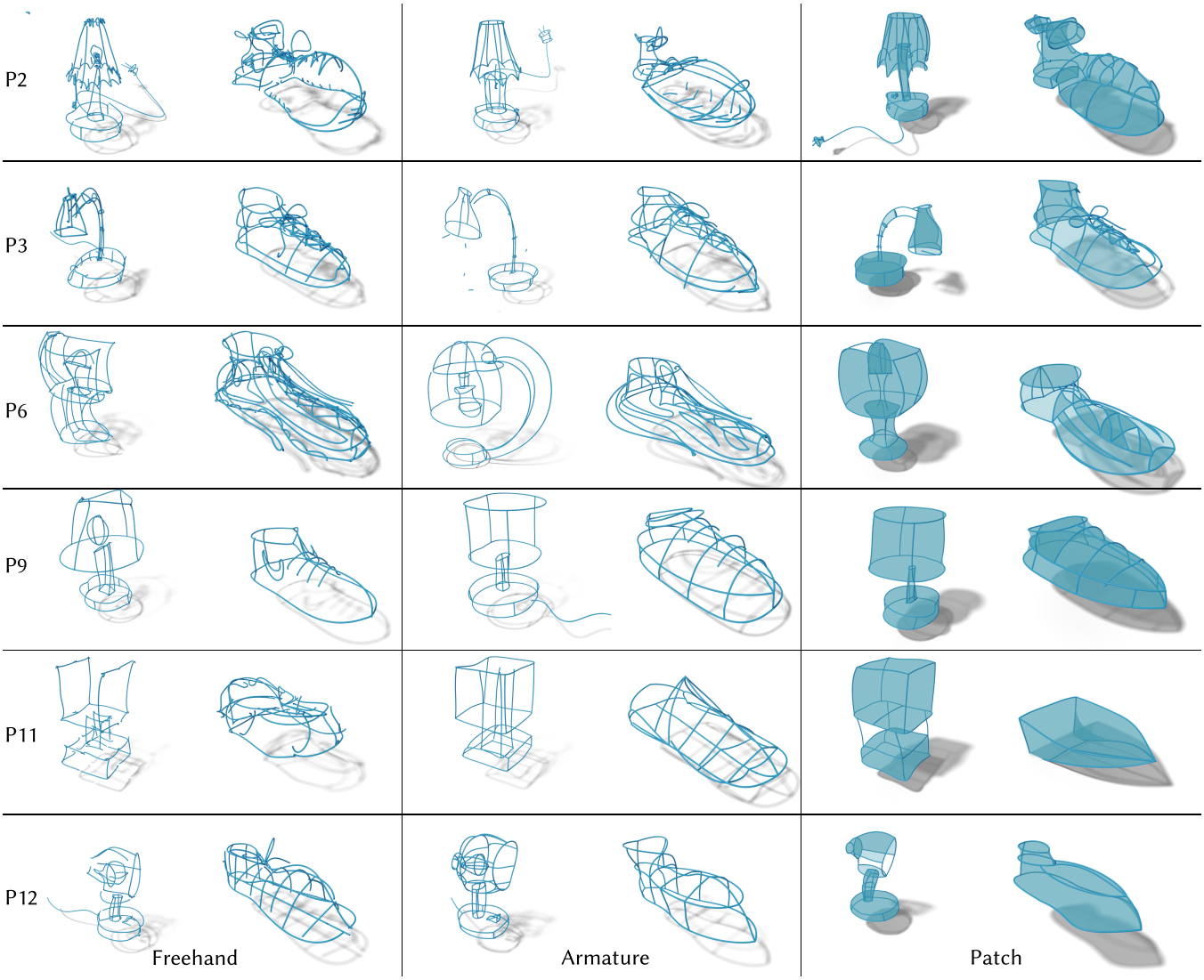

Figure 2: Designs created by the six other participants in the study using all three systems.