Free-Form Sketching of Self-Occluding Objects

Frederic Cordier
Korea Advanced Institute of Science and Technology

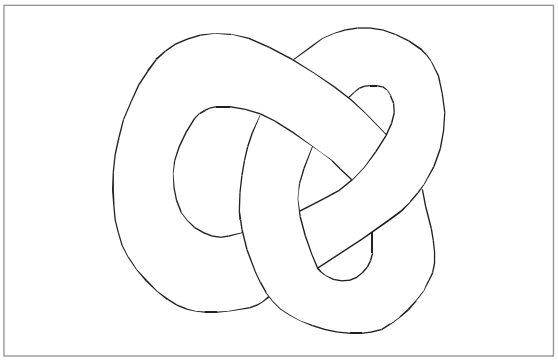
Hyewon Seo
Chungnam National University

When 3D objects occlude each other or self-occlude, their drawings typically consist of a set of contours that might partially overlap or self-overlap. The authors' method infers the hidden parts of contours and creates a smooth 3D shape matching those contours by solving a set of optimization problems.

Sketching interfaces to 3D object modeling facilitate 3D object reconstruction from a 2D drawing provided by a designer. Igarashi presented the Teddy silhouette-based sketching system, which has a simple, intuitive interface.[1] Follow-up research has mainly focused on the representation issue for the resulting 3D objects, as given in recent work: variational implicit surfaces[2] and other forms of implicit surfaces.[3]

In this article, we address a different issue, the extension of the modeling domain. In particular, we consider the modeling of self-occluding objects (or multiple objects possibly occluding each other), as Figure 1 shows. The creation of such objects with an existing tool such as Teddy[1] is rather awkward: the 2D closed curves would result in their corresponding 3D pieces that should be combined by detecting the hidden portions. We propose an integrated approach to this problem.

Our approach first extracts the 2D skeleton on the sketching plane, given a set of contours self-intersecting or intersecting each other. Then, we derive the 3D skeleton from its 2D counterpart. Finally, we construct the 3D objects(s). Our main contribution is the second step, which derives the 3D skeleton by computing the depth at each (self-)intersecting point on the 2D skeleton, while guaranteeing the intersection-free condition and C1-continuity at the corresponding points in the 3D space. We formulate the 3D skeleton construction as a sequence of constrained optimization problems to iteratively refine the shape of the skeleton off the sketching plane.



1 Drawing of a torus knot.

2D contour analysis

The input data for 3D reconstruction are a set of hand-drawn strokes that represent the visible parts of the model's 2D silhouette (see Figure 2a). The contour completion algorithm reconstructs the entire silhouette by finding the occluded parts of the silhouette curves as shown in Figure 2b. This algorithm involves two steps:

1. finding a set of spline segments that correspond to the occluded part of the silhouette, and
2. checking the validity of the topology of the reconstructed silhouette curves composed of the hand-drawn strokes and the spline segments.

The algorithm first enumerates the sets of spline segment, corresponding to all the possible ways of connecting free endpoints in a one-to-one manner. If there are more than two free endpoints, the completion problem might have several solutions. In this case, the algorithm sorts the sets of spline segments according to a cost function that reflects the splines' curvature and length. The computation time of the cost function is negligible; our algorithm can process a large number of free endpoints without significant increase in computation time.

The algorithm then examines the topology of each set in increasing order of their cost value; the aim is to find the set that solves the completion problem and that has the least distorted and shortest spline segments. A set's topology is valid if the corresponding silhouette curves are physically realizable in the 3D space. To achieve this, we implement the method proposed by Williams.[4,5] With this approach, we write an integer linear program based on the topological characteristics of the contour set, such as the numbers of segments and segment junctions. The topological validity is guaranteed when the integer linear program has a solution. In addition to ascertaining the validity of the topology, the method computes each segment's depth index. This value, which denotes the number of surfaces lying between the segment and the camera viewpoint, defines the occlusion order (that is, which segment is coming behind which other segment) of the spline segments (see Figure 2c).

Published by the IEEE Computer Society

William's method also computes the orientation of each silhouette curve, which indicates whether the object is on the left or right side of the curve. The Williams' papers[4,5] provide a detailed description of the completion algorithm.

Reconstructing the 3D skeleton

Our algorithm is suited for reconstructing 3D objects. From the graph labeling algorithm discussed in the "2D contour analysis" section, we have computed a set of closed curves. Each curve is segmented into a number of noncrossing curve segments at their (self-)intersection points. Each curve segment is labeled with its depth index—that is, the number of surfaces that lie between the viewpoint and the curve itself. Our goal here is to recover the coordinates of the nonintersecting 3D objects in a way that their corresponding 2D silhouette matches the input drawing. In the remainder of this article, we assume that the sketching plane is $z = 0$ (see Figure 3a). Also, we assume the orthogonal projection of the 3D objects onto the sketching plane ($z = 0$).

To simplify the problem, we first compute the z-coordinates of the 2D silhouette's skeleton rather than of the silhouette itself (see Figure 3b). The z-coordinates of the silhouette can then be easily inferred from the z-coordinates of its skeleton (see Figure 3c). Finally, we generate the 3D volume around the skeleton in a way that its silhouette matches the given 2D contours (see Figure 3d).

Extraction of the skeletons

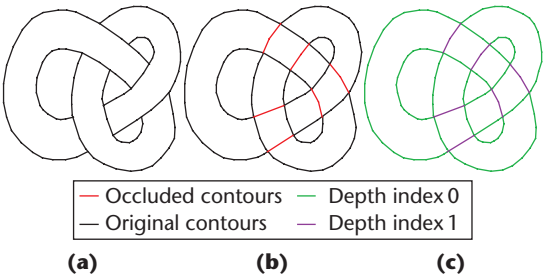
Similarly to Teddy, our program uses the skeleton computed with the chordal axis transformation of the silhouette polygon. The chordal axis is a curve that connects the center of the internal edges of the Delaunay-triangulated silhouette polygons (see Figure 3b). Igarashi et al. use the chordal axis to construct the 3D volume by inflating the region surrounded by the silhouette, using the chordal axis as the inflation's center.[1] In our case, however, we can't use constrained Delaunay triangulation because the 2D silhouette curves might contain (self-)intersections. In this article, we have implemented the method described by Cordier and Cheong, which is essentially a modified version of the plane-sweep algorithm.[6]

Criteria for estimating reconstructed object quality

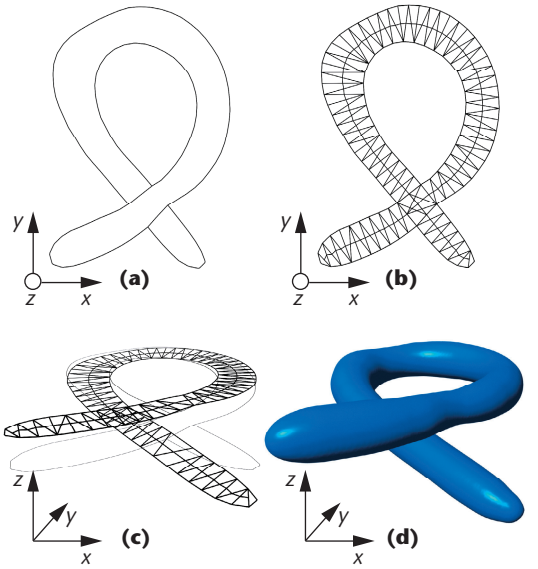
An infinite number of 3D objects exist that do not intersect and whose silhouette matches the input curves. We have defined three criteria to evaluate the shape quality of the 3D objects (see Figure 4):

- smoothness of their skeleton; we consider straight objects as having a more natural shape than twisted ones, and hence we evaluate them as better shapes;
- orientation; and
- distance of the reconstructed objects with respect to the sketching plane.

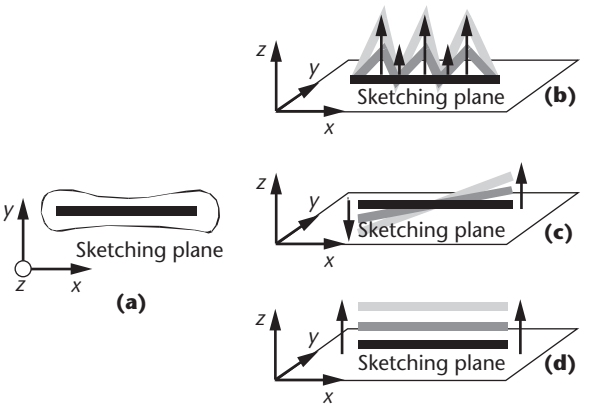
We make the two last criteria to place the reconstructed objects as close as possible to the sketching plane.



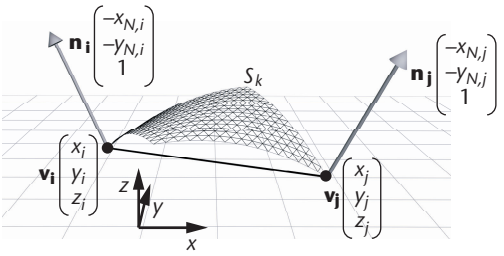
2 Building the sketch's topology: (a) initial sketch, (b) possible solution for occluded contours, and (c) depth index of each segment of the silhouette curve.



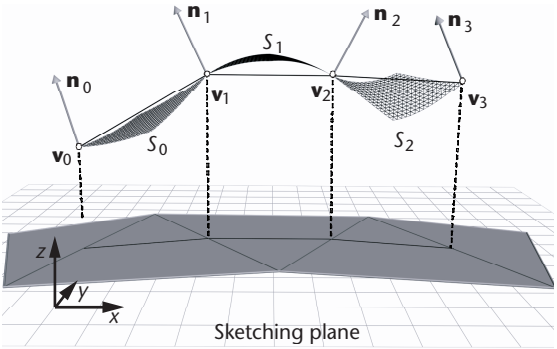
3 Overview of the reconstruction process: (a) the input curve is originally located on the sketching plane ($z = 0$); (b) the occluded segments are reconstructed and the skeleton of the silhouette curve is extracted using the constrained Delaunay triangulation; (c) the skeleton vertices are moved perpendicularly to the sketching plane (that is, parallel to the z-axis) such that the reconstructed shape (d) does not self-intersect.



4 The three criteria to estimate the quality of (a) the reconstructed shape: (b) smoothness of its skeleton, (c) its orientation, and (d) distance with respect to the sketching plane.



5 **A skeleton edge connecting $\mathbf{v_i}$ to $\mathbf{v_j}$ is associated with a quadratic surface S_k passing through $\mathbf{v_i}$ and $\mathbf{v_j}$ and whose normal vectors are $\mathbf{n}_i$ at $\mathbf{v_i}$ and $\mathbf{N}_j$ at $\mathbf{v_j}$.**



6 **The piecewise polynomial surface associated with the skeleton: a polynomial surface S_k is associated with each pair of connected vertices $\mathbf{v_i}$ and $\mathbf{v}_{i+1}$; two polynomial surfaces S_k and S_{k+1} are C1-continuous at their common vertex $\mathbf{v_i}$.**

Representing the skeleton

As we assign z-coordinates to the skeleton, we evaluate the skeleton curve's smoothness, which we can measure by its curvature and torsion energies. Intuitively speaking, the curvature energy measures the degree to which the curve is bent, and the torsion is a measure for the curve's nonplanarity. Unfortunately, these two energy terms are nonlinear[7] and can complicate the problem of optimization. Our solution is to represent the skeleton curves with a piecewise quadratic surface; by using the approximation of the quadratic surface's bending energy,[7] the energy terms become linear.

As Figure 5 shows, each edge $\mathbf{v_i}$, $\mathbf{v_j}$ is associated with a quadratic surface of the form $S_k(x, y) = a_k.x^2 + b_k.x + c_k.y^2 + d_k.y + e_k.x.y + f_k$. This surface passes through the two vertices $\mathbf{v_i}$ and $\mathbf{v_j}$, and is perpendicular to $\mathbf{n_i}$ and $\mathbf{n_j}$ at the vertices $\mathbf{v_i}$ and $\mathbf{v_j}$ respectively. For each vertex $\mathbf{v_i}$, the variables (x_i, y_i) are the position coordinates on the sketching plane; the z-coordinates (z_i) and normal coordinates $(x_{N,i}, y_{N,i})$ are the unknown variables of our reconstruction problem. Roughly speaking, the z-coordinates and normal coordinates are related, respectively, to the bending and torsion deformations of the skeleton curve. Two quadratic surfaces connected to the same vertex $\mathbf{v_i}$ are C1-continuous at the point (x_i, y_i)—that is, they have the same z-coordinate z_i and normal $\mathbf{n_i}$ vector at that point.

By choosing the piecewise quadratic surface representation (see Figure 6), we obtain the curvature energy as a convex quadratic function of its coefficients, assuming the slight bending within the surface. The minimization of such a function can be solved linearly. In addition, the curvature energy of the piecewise quadratic surface provides an estimation of both the torsion and bending energies of the skeleton curve.

A C1-continuity constraint is required to have smooth deformation at the quadratic surfaces' junction, as the skeleton bends. Intuitively speaking, the C1-continuity propagates the bending and the torsion deformation along the skeleton curves. As a vertex moves away from the sketching plane, the two surfaces adjacent to this vertex will bend smoothly with the C1-continuity constraint. Computing the curvature energy of these surfaces provides us with a simple way to estimate the skeleton curve's smoothness.

Formulating the shape reconstruction as a least squares problem

To compute the 2D skeleton's position in the 3D space, we find the z-location z_i and the normal coordinates $(x_{N,i}, y_{N,i})$ at the skeleton vertices $\mathbf{v_i} = (x_i, y_i)$ that are the solution of a least squares problem defined by an objective function and a set of constraints. The objective function's purpose is to estimate the quality of the reconstructed objects, the inequality constraints prevent the objects whose silhouettes overlap in 2D from intersecting each other, and the equality constraints maintain the C1-continuity between adjacent surfaces. Thus, we solve for an optimization problem of the following form:

$$\min_{\mathbf{X}} \left\| \boldsymbol{U}_{total}\mathbf{X} \right\| \text{subject to} \begin{cases} \boldsymbol{M}_{\text{C1}} \cdot \mathbf{X} = 0 \\ \boldsymbol{M}_{coll}\mathbf{X} \geq \mathbf{d_z} \end{cases} \quad (1)$$

where $\mathbf{X} = (z_0, x_{N,0}, y_{n,0}, \ldots z_{m-1}, x_{N,m-1}, y_{n,m-1})^T$, with m being the number of skeleton vertices. $\boldsymbol{U}_{total}$ denotes the matrix of the objective function for evaluating the quality of the reconstructed models; $\boldsymbol{M}_{\text{C1}}$ is the matrix of equality constraints for the C1-continuity, and $\boldsymbol{M}_{coll}$ is the matrix of the inequality constraints for preventing self-intersections.

Linear inequality constraints for the collisions. Since we don't want the reconstructed shapes intersecting in 3D, special care must be taken to keep some minimal distance between shapes whose 2D projected images on the sketching plane overlap. We do so by placing a set of inequality constraints on the z-coordinates of skeleton vertices. First, we identify all pairs of skeleton vertices for which the constraint must be defined. Our approach makes use of the paneling construction by Williams.[4] The paneling consists of creating a set of 2D panels (or regions) on the sketching plane. Each panel is the projected image on the sketching plane of a region of the 3D shape, and is delimited by a set of connected segments of the close curves obtained from the contour completion step, as discussed previously. Figure 7 shows an example of such paneling construction.

The paneling construction of a set of shapes whose projected images overlap on the sketching plane makes a stack of panels, each panel being assigned a depth

index—that is, the number of panels that lie between the viewpoint and the panel itself. In Figure 7, for example, the largest stack is the one composed of the panels (e_0, e_1, e_2) whose depth indices are 0, 1, and 2. With such panels, it's simple to find all skeleton vertice pairs to place the collision constraint: we first compute for every panel the list of skeleton vertices whose corresponding Delaunay edge is connected to the panel. Given a stack of panels, we then find for each panel, the immediate above panel in the stack. As these two panels overlap, we write a collision constraint for all skeleton vertice pairs that belong to the two different panels. We consider the collision constraints only on the skeleton vertices, and not on the skeleton edges. This approximation is acceptable only if the skeleton edges' length is small enough so that the distance to the skeleton is approximated as the distance to one of its vertices.

Second, we compute the minimum distance for each constraint and write them in matrix form. Given a constraint l between two vertices $\mathbf{v_i} = (x_i, y_i, z_i)$ and $\mathbf{v_j} = (x_j, y_j, z_j)$ on the skeleton as shown in Figure 8, the minimum distance between these two is

$$\left(r_i + r_j + d_{MinSurf}\right)^2 = l_l^2 + d_{z,l}^2 \qquad (2)$$

where r_i and r_j are respectively the radius of the shape at the vertices $\mathbf{v_i}$ and $\mathbf{v_j}$, l_l the distance along the sketching plane between $\mathbf{v_i}$ and $\mathbf{v_j}$, and $d_{MinSurf}$ the minimum distance between the boundaries of the reconstructed shapes; the user provides the last value. From Equation 2, we then compute the minimum distance along the z-axis for vertices $\mathbf{v_i}$ and $\mathbf{v_j}$:

$$d_{z,l} = \sqrt{\left(r_i + r_j + d_{MinSurf}\right)^2 - l_i^2}$$

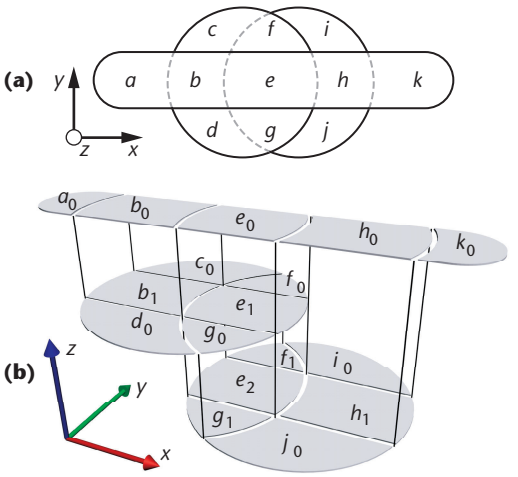
Finally, we define the linear inequality constraint for the two vertices $z_i - z_j \geq d_{z,l}$. The order of occlusion between the two overlapping skeletons determines the sign of $d_{z,l}$. The inequality constraints for all q collisions are written in the matrix form: $\boldsymbol{M}_{coll}\mathbf{X} \geq \mathbf{d_z}$. $\boldsymbol{M}_{coll}$ is a matrix $[a_{3.i,l}]$ of dimension $3m$ by q, such that $a_{3.i,l} = 1$ and $a_{3.i,l} = -1$ if a collision constraint l exists between the vertices $\mathbf{v_i}$ and $\mathbf{v_j}$ and 0 otherwise. The column vector $\mathbf{d_z}$ is defined by $(d_{z,0}, \ldots, d_{z,q-1})^T$.

In some cases, however, the linear inequalities system does not have any solution. Such cases happen when the reconstructed shapes are so tightly interlaced that nonpenetrating surfaces cannot be reconstructed.

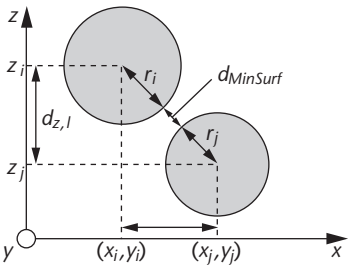
C1-continuity. The C1-continuity constraint between a quadratic surface $S_k(x,y) = a_k x^2 + b_k x + c_k y^2 + d_k y + e_k x.y + f_k$ and its adjacent quadratic surfaces at the vertices $\mathbf{v_i} = (x_i, y_i, z_i)$ and $\mathbf{v_j} = (x_j, y_j, z_j)$, see Figure 9, constitutes the following linear system:

$$S_k\left(x_i, y_i\right) = z_i, \frac{\partial S_k\left(x_i, y_i\right)}{\partial x} = x_{N,i}, \frac{\partial S_k\left(x_i, y_i\right)}{\partial y} = y_{N,i}$$

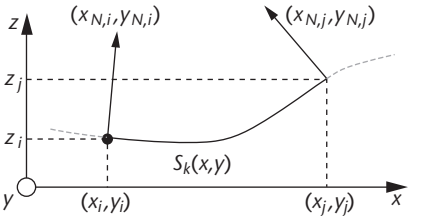
$$S_k\left(x_j, y_j\right) = z_j, \frac{\partial S_k\left(x_j, y_j\right)}{\partial x} = x_{N,j}, \frac{\partial S_k\left(x_j, y_j\right)}{\partial y} = y_{N,j}$$



7 Paneling construction (b) corresponding to the drawing (a).



8 The collision constraint *l* between vertices $\mathbf{v_i}(x_i, y_i, z_i)$ and $\mathbf{v_j}(x_j, y_j, z_j)$.



9 The C1-continuity constraints for $S_k(x, y)$ at the vertices (x_i, y_i) and (x_j, y_j).

The matrix form of this linear system is

$$\boldsymbol{M}_{\boldsymbol{C}\rightarrow\boldsymbol{X},\boldsymbol{k}} \cdot \mathbf{C_k} = \mathbf{X_{i,j}} \qquad (3)$$

where

$$\boldsymbol{M}_{\boldsymbol{C}\rightarrow\boldsymbol{X},\boldsymbol{k}} = \begin{bmatrix} x_i^2 & x_i & y_i^2 & y_i & x_i y_i & 1 \\ 2x_i & 1 & 0 & 0 & y_i & 0 \\ 0 & 0 & 2y_i & 1 & x_i & 0 \\ x_j^2 & x_j & y_j^2 & y_j & x_j y_j & 1 \\ 2x_j & 1 & 0 & 0 & y_j & 0 \\ 0 & 0 & 2y_j & 1 & x_j & 0 \end{bmatrix}$$

and $\mathbf{X_{i,j}} = [z_i x_{N,i} y_{N,i} z_j x_{N,j} y_{N,j}]^T$ and $\mathbf{C_k} = [a_k b_k c_k d_k e_k f_k]^T$.

The linear system is rank deficient, and it can be easily shown that the last row $\mathbf{R}_5$ of the matrix $\boldsymbol{M}_{\boldsymbol{C}\rightarrow\boldsymbol{X},\boldsymbol{k}}$ is a

linear combination of the other five rows, $\mathbf{R}_0, \ldots, \mathbf{R}_4$, as in $\mathbf{R}_5 = m_{k,0}\mathbf{R}_0 + m_{k,1}\mathbf{R}_1 + m_{k,2}\mathbf{R}_2 + m_{k,3}\mathbf{R}_3 + m_{k,4}\mathbf{R}_4$, where

$$m_{k,0} = \frac{2}{y_i - y_j},\ m_{k,1} = -\frac{(x_i - x_j)}{y_i - y_j},\ m_{k,2} = -1,$$
$$m_{k,3} = -\frac{2}{y_i - y_j},\ \text{and}\ m_{k,4} = -\frac{(x_i - x_j)}{y_i - y_j}$$

As $\mathbf{X_{i,j}}$ is linearly related to $\boldsymbol{M_{C\to X,k}}$ as shown in Equation 3, it follows that the similar linear dependency exists among the elements of $\mathbf{X_{i,j}}$: $m_{k,0}z_i + m_{k,1}x_{N,i} + m_{k,2}y_{N,i} + m_{k,3}z_j + m_{k,4}x_{N,j} - y_{N,j} = 0$. Combining this linear constraint for every surface, we write: $\boldsymbol{M}_{C1} \cdot \mathbf{X} = 0$ where

$$\boldsymbol{M}_{C1} = \begin{bmatrix} \mathbf{m}_{0,0} & 0 & 0 & \mathbf{m}_{0,1} & 0 & 0 & \cdots & 0 \\ 0 & \mathbf{m}_{1,0} & 0 & 0 & \mathbf{m}_{1,1} & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \ddots & \cdots & 0 \\ 0 & \ddots & \ddots & \mathbf{m}_{j,0} & 0 & 0 & \mathbf{m}_{j,1} & \vdots \\ \vdots & \cdots & 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & \mathbf{m}_{n,0} & 0 & 0 & \mathbf{m}_{n,1} \end{bmatrix}$$

with $\mathbf{m}_{k,0} = (m_{k,0}, m_{k,1}, m_{k,2})$ and $\mathbf{m}_{k,1} = (m_{k,3}, m_{k,4}, -1)$.

Curvature energy. A common approximation of the curvature energy of a thin plate $s(x, y)$ under slight bending[7] is

$$E_{\text{curv}} = \frac{D}{2}\iint_s \left(\left(\frac{\partial^2 s}{\partial x^2} + \frac{\partial^2 s}{\partial y^2} \right) - 2(1-\nu)\left(\frac{\partial^2 s}{\partial x^2}\frac{\partial^2 s}{\partial y^2} - \left(\frac{\partial^2 s}{\partial x \partial y} \right)^2 \right) \right) dxdy$$

The constant D is the Young's modulus (elasticity coefficient of the plate) and ν the Poisson's ratio. In practice, the Poisson's ratio varies from 0 to 1/2. In this article, we use D = 2 and v = 1/2, and the approximated curvature energy of a quadratic polynomial surface $S_k(x, y)$ associated with an edge $(\mathbf{v_i}, \mathbf{v_j})$, $\mathbf{v_i} = (x_i, y_i)$, $\mathbf{v_j} = (x_j, y_j)$, is

$$E_{curv,k} = l_k^2\left(4a_k^2 + 4c_k^2 + e_k^2 + 4a_k c_k\right) \quad (4)$$

where

$$l_k^2 = \left(x_j - x_i\right)^2 + \left(y_j - y_i\right)^2$$

Equation 4 can be rewritten in the matrix form

$$E_{curv,k} = l_k^2\begin{pmatrix} a_k & c_k & e_k \end{pmatrix}\boldsymbol{H}\begin{pmatrix} a_k & c_k & e_k \end{pmatrix}^T$$

where

$$\boldsymbol{H} = \begin{bmatrix} 4 & 2 & 0 \\ 2 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Since $E_{curv,k}$ has a convex quadratic polynomial form, the coefficients a_k, c_k, and e_k, minimizing this objective can be computed by solving the least squares problem:[8]

$$\min_{a_k, c_k, e_k} \left\| \boldsymbol{U} \cdot (a_k c_k e_k)^T \right\| \quad (5)$$

where the matrix

$$\boldsymbol{U} = \begin{bmatrix} 2 & 1 & 0 \\ 0 & \sqrt{3} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

is computed by the Cholesky factorization of H. Clearly, working with the approximated curvature energy is less expensive than working with that of exact form, which involves nonlinear terms.[7]

Now that we have found the curvature-energy objective function using the quadratic coefficients $\mathbf{C_k}$, we rewrite it as a function of skeleton variables $\mathbf{X_{i,j}}$ so that the curvature energy term is seamlessly integrated into the global optimization problem as in Equation 1. $\mathbf{C_k}$ and $\mathbf{X_{i,j}}$ are linearly related from Equation 3; unfortunately, the matrix $\boldsymbol{M_{C\to X,k}}$ is rank deficient and cannot be directly inverted. We therefore compute a matrix $\boldsymbol{M_{X\to C,k}}$ such that $\mathbf{C_k} = \boldsymbol{M_{X\to C,k}}\mathbf{X_{i,j}}$ is a solution to the curvature minimizing problem of Equation 5 and a solution to the linear system as in Equation 3. The "Computing the Matrix $\mathbf{M_{X\to C,k}}$" sidebar describes the computation of this matrix. The objective function of curvature energy as a function of $\mathbf{X_{i,j}}$ is given by

$$\min_{\mathbf{X_{i,j}}} \left\| \boldsymbol{U}'\boldsymbol{M}_{\boldsymbol{X\to C,k}}\mathbf{X_{i,j}} \right\| \quad (6)$$

where

$$\boldsymbol{U}' = \begin{bmatrix} 2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Combining the objective function in Equation 6 for every surface, we write:

$$\min_{\mathbf{X}} \left\| \boldsymbol{U}_{\mathbf{curv}}\mathbf{X} \right\| \quad (7)$$

where $\mathbf{U_{curv}}$ is filled with $l_k\boldsymbol{U}'\boldsymbol{M_{X\to C,k}}$ for $k = 0, \ldots n-1$, n being the number of edges.

Orientation of the skeleton. The second criterion to evaluate the quality of the reconstructed shape is the minimization of the orientation angle of its skeleton with respect to the sketching plane ($z = 0$).

Computing the Matrix $M_{X\to C,k}$

We compute the matrix $M_{X\to C,k}$ such that $\mathbf{C_k} = M_{X\to C,k}\,\mathbf{X_{i,j}}$ is the solution to the linear equality-constrained least squares problem:

$$\min_{\mathbf{C_k}} \left\| U' \mathbf{C_k} \right\| \tag{A}$$

subject to

$$M' \mathbf{C_k} = \mathbf{X_{i,j}} \tag{B}$$

where

$$U' = \begin{bmatrix} 2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \sqrt{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

and

$$M' = \begin{bmatrix} x_i^2 & x_i & y_i^2 & y_i & x_i y_i & 1 \\ 2x_i & 1 & 0 & 0 & y_i & 0 \\ 0 & 0 & 2y_i & 1 & x_i & 0 \\ x_j^2 & x_j & y_j^2 & y_j & x_j y_j & 1 \\ 2x_j & 1 & 0 & 0 & y_j & 0 \end{bmatrix}$$

Here, M' contains five of the six rows of $M_{C\to X,k}$, and is not rank deficient. One way to solve least squares problems with equality constraints is to use the method of direct elimination.[1] We introduce first the partitioning:

$$\begin{bmatrix} M' \\ U' \end{bmatrix} = \underbrace{\begin{bmatrix} M_1' & M_2' \\ U_1' & U_2' \end{bmatrix}}_{5\quad 1} \begin{matrix} \}5 \\ \}3 \end{matrix} \text{ and } \mathbf{C_k} = \begin{bmatrix} \mathbf{C_{k,1}} \\ \mathbf{C_{k,2}} \end{bmatrix} \begin{matrix} \}5 \\ \}1 \end{matrix}$$

$\mathbf{C_{k,1}}$ can be formulated as a function of $\mathbf{C_{k,2}}$ using the equality constraint Equation B:

$$\mathbf{C_{k,1}} = M_1'^{-1}\left(\mathbf{X_{i,j}} - M_2'\,\mathbf{C_{k,2}}\right) \tag{C}$$

Therefore

$$\mathbf{C_k} = \begin{bmatrix} \mathbf{C_{k,1}} \\ \mathbf{C_{k,2}} \end{bmatrix} = \begin{bmatrix} M_1'^{-1}\left(\mathbf{X}_{i,j} - M_2'\,\mathbf{C_{k,2}}\right) \\ \mathbf{C_{k,2}} \end{bmatrix}$$

Substituting this expression for $\mathbf{C_k}$ in Equation A gives

$$\min_{\mathbf{C_k}} \left\| U' \cdot \mathbf{C_k} \right\| = \min_{\mathbf{C_{k,2}}} \left\| \tilde{U}_2' \cdot \mathbf{C_{k,2}} + U_1'\, M_1'^{-1} \mathbf{X_{i,j}} \right\|$$

where $\tilde{U}_2' = U_2' - U_1'\, M_1'^{-1} U_2'$. The solution of the previous least squares problem is given by

$$\mathbf{C_{k,2}} = -\left(\tilde{U}_2'^{T} \tilde{U}_2'\right)^{-1} \tilde{U}_2'^{T} U_1'\, M_1'^{-1} \mathbf{X_{i,j}}$$

Substituting this expression of $\mathbf{C_{k,2}}$ in Equation C gives

$$\mathbf{C_{k,1}} = M_1'^{-1}\left(I + M_2' \left(\tilde{U}_2'^{T} \tilde{U}_2'\right)^{-1} \tilde{U}_2'^{T} U_1'\, M_1'^{-1} \right) \mathbf{X_{i,j}}$$

Finally, we can write the matrix $M_{X\to C,k}$ with which we can compute $\mathbf{C}_k$ as a function of $\mathbf{X}_{i,j}$:

$$\mathbf{C_k} = \begin{bmatrix} M_1'^{-1}\left(I + M_2' \left(\tilde{U}_2'^{T} \tilde{U}_2'\right)^{-1} \tilde{U}_2'^{T} U_1'\, M_1'^{-1} \right) \\ -\left(\tilde{U}_2'^{T} \tilde{U}_2'\right)^{-1} \tilde{U}_2'^{T} U_1'\, M_1'^{-1} \end{bmatrix} \cdot \mathbf{X_{i,j}} = M_{\mathbf{x}\to \mathbf{C},k}$$

The matrix $M_{X\to C,k}$ is used in the objective function of the curvature energy (see Equation 7 in the main text).

Reference

1. C. Lawson and R. Hanson, *Solving Least Squares Problems*, Prentice Hall, 1974.

The normal vector (see Figure 10) to a surface defined by the function $S_k(x, y)$ is:

$$\mathbf{n} = \left(-\frac{\partial S_k(x,y)}{\partial x}, -\frac{\partial S_k(x,y)}{\partial y}, 1 \right)$$

Previously, we defined the variables $(x_{N,i}, y_{N,i})$ of the vertex $\mathbf{v_i}$ by

$$x_{N,i} = \frac{\partial S_k(x,y)}{\partial x} \text{ and } y_{N,i} = \frac{\partial S_k(x,y)}{\partial y}$$

$S_k(x, y)$ being a quadratic surface connected to the vertex $\mathbf{v_i}$. Therefore, the objective function to minimize the skeletons' orientation angle

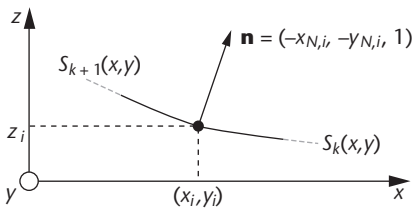
$$\sum_{i=0}^{m-1} \left({x_{N,i}}^2 + {y_{N,i}}^2 \right)$$

This objective function is written in the matrix form

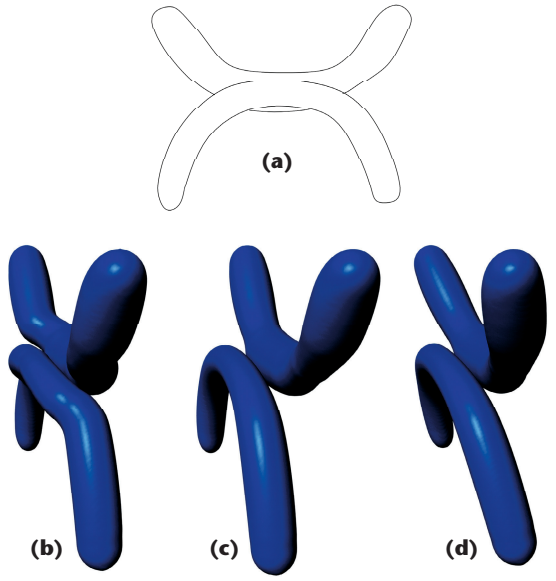
$$\min_{\mathbf{X}} \left\| \boldsymbol{U}_{\boldsymbol{orient}} \mathbf{X} \right\| \tag{8}$$

where

$$\boldsymbol{U}_{\boldsymbol{orient}} = \begin{bmatrix} 0\ 1\ 0 & & & & 0 \\ 0\ 0\ 1 & & & & \\ & 0\ 1\ 0 & & & \\ & 0\ 0\ 1 & & & \\ & & \ddots & & \\ & & & & 0\ 1\ 0 \\ 0 & & & & 0\ 0\ 1 \end{bmatrix}$$



10 The normal vector at the vertex $\mathbf{v_i}$.



11 The results of the sketch reconstruction (a) using low weight value for the (b) curvature, (c) distance, and (d) orientation objective functions.

Distance to the sketching plane. The purpose of the last objective function is to minimize the distance of the reconstructed shapes to the sketching plane. For each vertex $v_j(x_j, y_j)$ of the skeleton, the objective function is z_i^2; the objective function for all the vertices is written

$$\sum_{i=0}^{m-1}\left(z_i^2\right)$$

The matrix form of the objective function is

$$\min_{\mathbf{X}}\left\|\boldsymbol{U}_{\boldsymbol{dist}}\cdot\mathbf{X}\right\| \tag{9}$$

where

$$\boldsymbol{U}_{\boldsymbol{dist}}=\begin{bmatrix}1\ 0\ 0 & & & 0\\ & 1\ 0\ 0 & & \\ & & \ddots\ddots\ddots & \\ 0 & & & 1\ 0\ 0\end{bmatrix}$$

Solving the optimization problem

The overall objective is determined by the weighted combination of the three objectives, as in Equations 8, 9, and 10:

$$\min_{\mathbf{X}}\left\|\boldsymbol{U}_{\boldsymbol{total}}\mathbf{X}\right\| \text{ with } \boldsymbol{U}_{\boldsymbol{total}}=\begin{bmatrix}w_{curv}\cdot\boldsymbol{U}_{\boldsymbol{curv}}\\ w_{orient}\cdot\boldsymbol{U}_{\boldsymbol{orient}}\\ w_{dist}\cdot\boldsymbol{U}_{\boldsymbol{dist}}\end{bmatrix}$$

Typically, users can modulate the influence of each initial objective (curvature, orientation, and distance with respect to the sketching plane) by choosing these weight values (w_{curv}, w_{orient}, w_{dist}) as Figure 11 shows. Weight values can be defined either for all the reconstructed shapes or for each shape individually.

In this article, we limit the range of these weights so that $\boldsymbol{U}_{\boldsymbol{total}}$ does not become ill-conditioned. Otherwise, the minimization might fail because of large numerical errors. In practice, we found that the ratio of the largest to the smallest weights should be no greater than 10^3. Figure 11 shows the reconstructed models with different weight values. We found that most aesthetically pleasing results are obtained with high weight value for the curvature objective function as shown in Figure 11.

We compute the solution of Equation 1's least squares problem by using the least squares with linear inequality and equality constraints package;[8] we have rewritten this package in a way that its linear algebra routines are replaced by those of the Intel Math Kernel Library.

Finding the least squares solution is computationally the most expensive process in the shape-reconstruction algorithm. Still, the computation maintains arguably fast speed; most of the models shown in Figure 12 have been reconstructed in less than 5 seconds, once the input drawing has been provided.

Generating the 3D mesh volume from the skeleton

So far, we have focused on the optimization problem to locate the extracted skeleton from a 2D drawing in 3D space. The solution to the optimization problem is a skeleton that is both consistent with the silhouette in the sketching plane and collision-free. It also minimizes the three objective functions—that is, curvature, orientation, and distance to the sketching plane. We now describe how we compute the 3D mesh volume surrounding the given skeletons.

Briefly, the 3D shape can be obtained by inflating the region surrounded by the silhouette, using the skeleton as the center of inflation. A large number of methods exist that accomplish this task; most of them are based on implicit surfaces.[2]

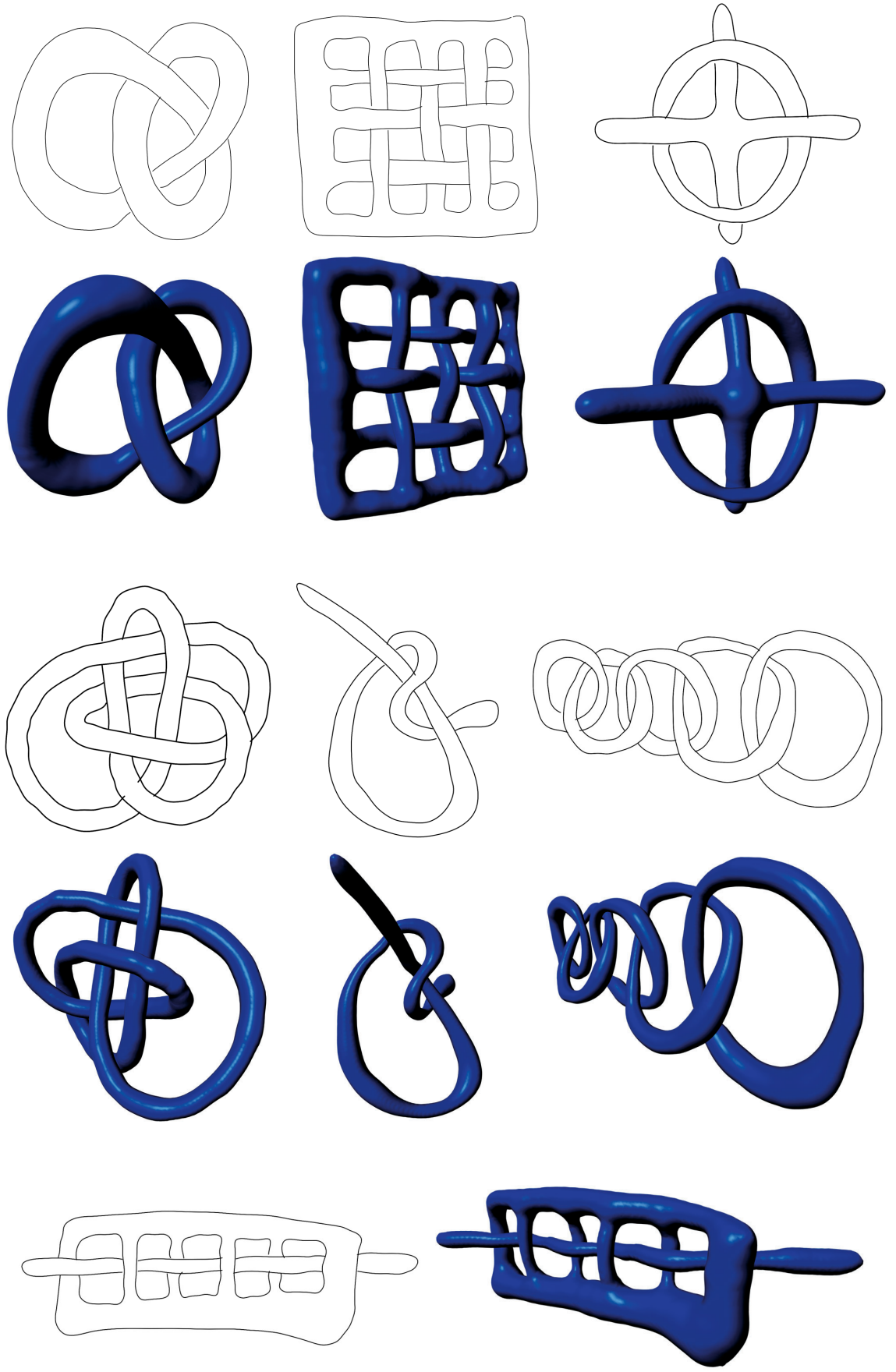
In this article, we have adopted the surface modeling method described by Alexe et al. because of its implementation simplicity.[3] In that method, a surface is generated by the blend of spherical implicit surfaces whose centers are located along the skeletons. The scalar value of the implicit surfaces at some point p is calculated as follows:

$$f_{total}(p)=\sum_{i=1}^{i=n}c_i f_i\left(r_i\right)$$

where $f_{total}(p)$ is the value of the field at point p, f_i is the field function of the ith spherical implicit surface, r_i is the distance from p to the center of the ith implicit surface, and c_i is the influence coefficient of the ith implicit surface. The parameters c_i are calculated using the nonnegative least squares optimization[8] such that the resulting implicit surface matches the contours.

Results and limitations

We implemented both the 2D drawing tool and the model reconstruction software as an AutoDesk Maya plug-in, a commercially available 3D computer graphics package. Users draw the strokes on the front orthogonal view and the results are shown on the perspective view.



12 Sketch examples.

Previous Work

A drawing of self-occluding objects cannot be processed in a stroke-based manner, because it typically consists of many unclosed curve segments. Thus, we cannot process each newly drawn curve segment individually, but the reasoning about the drawing can be commenced only when these segments are collectively taken. On the other hand, there are many technical issues common to our approach and previous works on sketching interfaces. Both require analyzing 2D curves provided by the user, and reconstructing corresponding 3D objects from the curves.

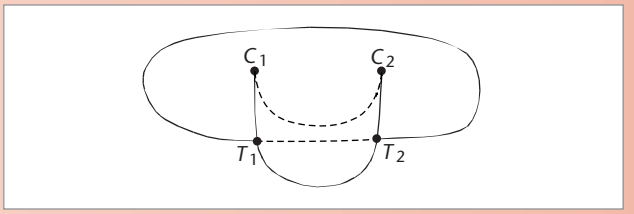
Here, we review previous works on sketching interfaces for 3D graphical modeling, which are clustered according to the class of shapes they model. The most common approach to 3D modeling with a sketching interface is to require its user to draw the visible and hidden contours of the rectilinear object to be modeled.[1] Based on the geometric correlation hypothesis, such a reconstruction technique is particularly suitable for the design of CAD-like geometric objects. However, the hypothesis these researchers use allow modeling of rectilinear objects only.

Some other sketching tools use a purely gesture-based interface. For instance, Sketch, proposed by Zeleznik et al identifies gestures from the input strokes and interprets them according to a set of predetermined rules.[2] Those rules define the way the user-supplied gestural symbols are mapped to the creation of primitive objects, or to applying operations on existing objects.

Other researchers have presented sketching interfaces for free-form modeling.[3-5] In their systems, users create a shape by drawing its 2D silhouette; the 3D mesh is generated by inflating the region surrounded by the silhouette, making wide areas fat and narrow areas thin. The created model can then be modified interactively with a set of tools such as cutting, extruding, bending, or drawing on the mesh.

Cohen et al. propose a sketching interface for 3D curve modeling—the user can model a nonplanar curve by drawing it from a single viewpoint and its shadow on the floor plane.[6] Other researchers have worked on sketching interfaces for modifying existing 3D shapes. In the system described by Nealen et al., a 3D shape is deformed by fitting its silhouette to a curve given by the user.[7]

Karpenko and Hughes have published, almost simultaneously to us, a paper describing a similar system—the modeling of free-form objects with (self-)occlusions.[8] While our method only processes 2D contours connected with T-junctions, Karpenko's method can handle a wider range of contour drawings with cusps (see Figure A). Another difference between the two approaches concerns the computation of the 3D position of the reconstructed objects. We reformulate the 3D reconstruction problem as a linear optimization problem, whereas Karpenko and Hughes use a mass-spring system. Unlike linear optimization problems, mass-spring systems have several disadvantages such as slow convergence, numerical instability, and the need to fine-tune the system parameters by trial and error. Finally, our approach has another advantage: it ensures that the reconstructed surfaces don't intersect each other, whereas Karpenko's approach doesn't have any mechanism to prevent these self-intersections.



A Contour drawing with *T*-junctions (T_1 and T_2) where two contours cross, and cusps (C_1 and C_2) where contours reverse direction.

References

1. H. Lipson and M. Shpitalni, "Correlation-Based Reconstruction of a 3D Object from a Single Freehand Sketch," *Proc. 2002 AAAI Spring Symp. Sketch Understanding*, AAAI Press, 2002, pp. 99-104.
2. R.C. Zeleznik, K.P. Herndon, and J.F. Hughes, "Sketch: An Interface for Sketching 3D Scenes," *Proc. ACM Siggraph*, Addison-Wesley, 1996, pp. 163-170.
3. T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: A Sketching Interface for 3D Freeform Design," *Proc. Siggraph*, 1999, ACM Press, pp. 409-416.
4. O. Karpenko, J.F. Hughes, and R. Raskar, "Free-Form Sketching with Variational Implicit Surfaces," *Computer Graphics Forum*, vol. 21, no. 3, 2002, pp. 585-594.
5. I.A. Alexe, V. Gaildrat, L. Barthe, "Interactive Modelling from Sketches Using Spherical Implicit Functions," *Proc. Afrigraph*, ACM Press, 2004, pp. 25-34.
6. J. Cohen et al., "An Interface for Sketching 3D Curves," *ACM I3DG Symp. Interactive 3D Graphics*, ACM Press, 1999, pp. 17-21.
7. A. Nealen et al., "A Sketch-Based Interface for Detail-Preserving Mesh Editing," *ACM Trans. Graphics*, vol. 24, no. 3, 2005, pp. 1142-1147.
8. O.A. Karpenko and J.F. Hughes, "SmoothSketch: 3D Free-Form Shapes from Complex Sketches," *ACM Trans. Graphics*, vol. 25, no. 3, 2006, pp. 589-598.

We implemented the optimization algorithm with the Intel Math Kernel Library. This numerical package is used as a library in Maya and includes efficient methods to solve linear systems involved in the least squares problem.

We have tested our method on different drawings using tablet input and evaluated the quality of the reconstructed models. The results are illustrated in Figure 12. Other examples and a demonstration video of our reconstruction algorithm are available at http://vml.kaist.ac.kr/projects.html. All these examples have been calculated with a high weight value for the curvature objective function. Given an appropriate input drawing, a 3D mesh is generated in 5 to 10 seconds. Results are comparable in quality to those obtained from Karpenko, Hughes, and Raskar;[2] however, users can now draw objects that might occlude each other or be self-occluding.

Our algorithm can only reconstruct 3D objects with a circular cross-section. Because of this limitation, most of the objects created with our tool have a tubular shape.

One way to tackle this limitation would be to extend our system to handle interactive shape modification similarly to Teddy.[1] For instance, users could apply a set of modifications such as cut, extrude, or modify the cross-section, on previously created models.

Conclusion

We have presented a method for reconstructing 3D objects from a 2D drawing, which allows modeling of objects with self-occluded parts. The power of our approach is best illustrated with the knot example shown in Figure 12. To the best of our knowledge, modeling of this class of objects is not possible with other previously developed silhouette-based modeling tools.

Another contribution of this work is the formulation of the optimization problem to compute the depth position of the reconstructed objects. Our mathematical model is simple and its implementation only requires computing the elements of the matrices of the objective functions and constraints. In spite of its simplicity, the algorithm can handle a wide range of cases such as the sketching of groups of flat objects occluding each other, or self-occluding objects that are curved in the z-direction. Our modeler does not place any limitation on the number of objects and can handle (self-)occlusions. In addition, all input drawings are processed in a uniform manner. ■

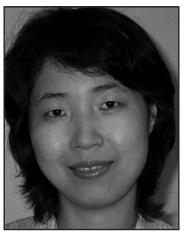
Acknowledgments

We thank Sung-Yong Shin, Young-Sang Cho, and Otfried Cheong (Korea Advanced Institute of Science and Technology) for their invaluable advice and useful comments. This work was supported by the Korea Research Foundation Grant funded by the Korean Government (KRF-2006-531-D00033), and author Cordier was supported by the Graduate School of Culture Technology (Ministry of Culture and Tourism of Korea).

References

1. T. Igarashi, S. Matsuoka, and H. Tanaka, "Teddy: A Sketching Interface for 3D Freeform Design," *Proc. Siggraph*, 1999, ACM Press, pp. 409-416.
2. O. Karpenko, J.F. Hughes, and R. Raskar, "Free-Form Sketching with Variational Implicit Surfaces," *Computer Graphics Forum*, vol. 21, no. 3, 2002, pp. 585-594.
3. I.A. Alexe, V. Gaildrat, and L. Barthe, "Interactive Modelling from Sketches Using Spherical Implicit Functions," *Proc. Afrigraph*, ACM Press, 2004, pp. 25-34.
4. L.R. Williams, *Perceptual Completion of Occluded Surfaces*, doctoral dissertation, Dept. Computer Science, Univ. of Massachusetts at Amherst, 1994.
5. L.R. Williams, "Topological Reconstruction of a Smooth Manifold-Solid from Its Occluding Contour," *Int'l J. Computer Vision*, vol. 23, no. 1, 1997, pp. 93-108.
6. F. Cordier and O. Cheong, "Constrained Delaunay Triangulation of Self-Intersecting Polygons," tech. report, Computer Science Dept., KAIST, 2005.
7. W. Wesselink, *Variational Modeling of Curves and Surfaces*, doctoral dissertation, Dept. Computing Science, Univ. of Technology, Eindhoven, 1996.
8. C. Lawson and R. Hanson, *Solving Least Squares Problems*, Prentice Hall, 1974.

Frederic Cordier *is a visiting professor at the Graduate School of Culture Technology at KAIST. His research interests include 3D modeling and texturing, human–computer interaction and physics-based simulation. Cordier has a PhD in computer science from the University of Geneva, Switzerland.*



Hyewon Seo *is an assistant professor and supervisor of the Computer Graphics Laboratory in the Department of Computer Science and Engineering at the Chungnam National University, Korea. Her research interests include imaging, visual simulation, human–computer interaction, and VR. Seo has graduate degrees in computer science from the University of Geneva and KAIST. Contact her at hseo@cnu.ac.kr.*