

Implementation details of SmoothSketch: 3D free-form shapes from complex sketches

Olga Karpenko

Brown University

John F. Hughes

In the paper we introduced SmoothSketch - a system that infers reasonable 3D free-form shapes from contour drawings containing so-called tee-junctions and cusps. Figure 1 shows the flow of operations in the system. At the high level, there are three main components: figural completion, the paneling construction, and smooth embedding. The boxes highlighted in light green show the algorithmic details covered in this sketch. Further details can be found in the supplemental material accompanying the paper.

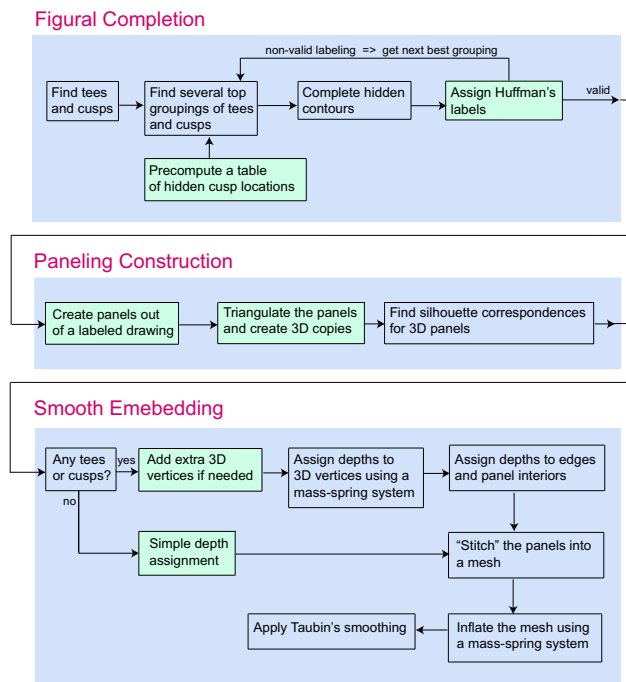


Figure 1: A diagram showing the flow of operations in the system.

Assigning Huffman's labels and checking the validity of the labeling: An orientation is assigned to each stroke in such a way that the surface always remains on the left as one traverses the contour. We assign Huffman "depth" labels to each stroke to indicate the number of surfaces in front of the stroke: visible contours all get label 0; the values of the labels change at tees and cusps according to Huffman's rules; this allows us to propagate the labeling to all curves.

Precomputing a table of hidden cusp locations: As mentioned in the paper, the problem of estimating the location of a hidden cusp (given the corresponding visible cusp and the T-junction) is an expensive search problem that involves simulating a large number of directional random walks. To keep the system interactive, we precompute the table of locations of the hidden cusps for a number of locations of tees/visible cusps.

Creating 2D panels out of a labeled drawing: Visible and hidden contours split the labeled drawing into planar regions ("panels"). At this step of the algorithm, each panel is represented as a loop of consecutive strokes. To compute those, we construct a graph

where the nodes are tees and cusps, and the edges are the strokes connecting them (with orientations assigned). The algorithm consists of the following steps: adding edges going in the opposite direction to the graph; identifying "valid" counter-clockwise edge cycles (which become "preliminary panels") and removing them from the graph; identifying "valid" clockwise edge cycles (which we call "hole cycles"); and merging the hole cycles with the panels.

Triangulating the panels; the issue of two distinct points having the same 2D location: Consider the stroke of the big panel for the kidney bean drawing (see Figure 2). There are two distinct points on the stroke corresponding to the red point in the left picture that have the same coordinates. It is important to keep them distinct because in the paneling construction stage they are identified with different edges in other panels. That means that when we convert this stroke (or, rather, set of strokes) to a 2D triangular mesh, we need to handle this case separately (by default, Delaunay triangulation will remove the duplicates).

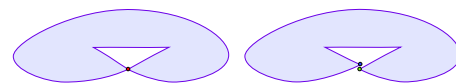


Figure 2: (left) A panel for a bean shape. (right) There are actually two distinct points corresponding to the original red 2D vertex

Inflation of objects where some strokes have no tees or cusps: The inflation algorithm described in the paper first assigns depths to the vertices that correspond to 3D locations of tees and cusps, then to the silhouette edges connecting the vertices by interpolating the depths of vertices, and finally, to the interiors of the panels. If the user input stroke does not contain any tees or cusps, then we use a different, simpler, algorithm: place panels some distance apart from each other, and connect the corresponding silhouette edges with triangular strips. After that, the steps are the same as in the first case: the mass-spring system is created and relaxed. There is also a third case (Figure 3) that needs to be handled separately: consider a drawing that consists of a simple outer stroke and a complex inner stroke corresponding to a hole/ holes. In this case, although the internal stroke contains tees and cusps, the above algorithm will not work - it is not clear how to assign depths to the 3D edges corresponding to the outer stroke. We add an extra vertex on the outer stroke, as well as update the mass-spring system used to assign depths to the vertices.

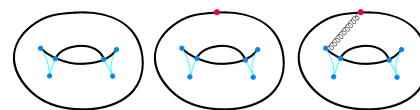


Figure 3: An extra vertex shown in red is added to the outer stroke so that the depths could be assigned to all edges.