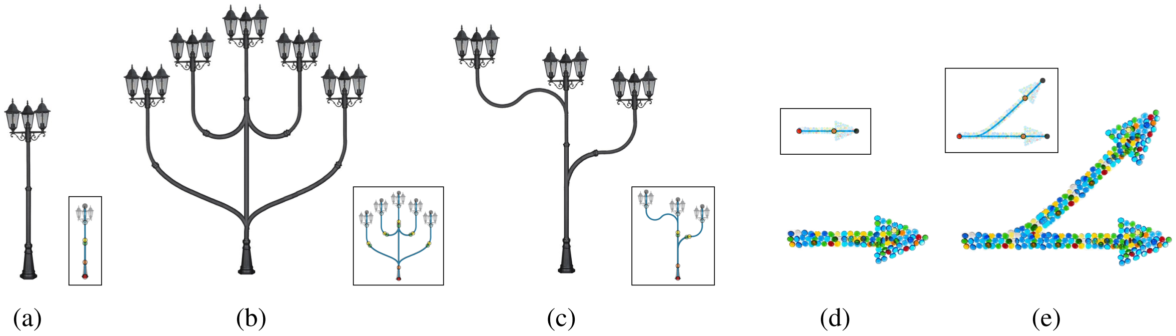
Stroke-based Editing of Object Structure

Sheng-Jie Luo[1] Chin-Yu Lin[1] I-Chao Shen[2] Bing-Yu Chen[1]

[1]National Taiwan University [2]CITI, Academia Sinica



(a) (b) (c) (d) (e)

Figure 1: *We present a new structure editing technique for synthesis of new objects according to user provided strokes. Given an input object in an image and user provided strokes indicating the skeletal structure of the object (a)(d), our method can synthesize the object with different skeletal structures based on corresponding user provided strokes (b)(c)(e).*

1 Introduction

Recent years have witnessed significant developments of structural image manipulation techniques. Some of these techniques are based on *image warping*, which stretches image content to satisfy editing constraints. Other techniques are based on *texture synthesis*, which synthesizes repeated texture to fill the target image domain so that the result resembles the source image. However, it is challenging for both categories of approaches to generate satisfying results when desired object's skeletal structure is far from the original object as shown in Figure 1. In this extended abstract, we present a new structure editing technique that is based on user provided strokes as structure constraints. Users specify a set of strokes indicating the structure of the input object in an image as well as a set of corresponding strokes in the output domain. Then, our method synthesizes the object texture along these strokes according to the direction and position of the corresponding strokes.

2 Method

Our method takes an image (I_s) of an object as input, a user first specifies a set of source strokes ($\mathbf{S}_s$) on input image representing the skeletal structure of the object. Then the user specifies corresponding target strokes ($\mathbf{S}_t$) on the output domain. Our method synthesizes a target image (I_t) based on the position and direction of $\mathbf{S}_s$ and $\mathbf{S}_t$.

In our system, a stroke is represented as a set of neighboring points which contain the information of location and local orientation of the stroke. User can also specify a number of control point pairs on the corresponding source strokes and target strokes for indicating desired input texture location. By default, the end points of each stroke are taken as control points. Our object editing method consists of three steps: *fetching source texture*, *local regions composition*, and *regions*.

Fetching source texture. First, we focus on one pair of corresponding source and target strokes S_s, S_t. The two strokes contain a set of one-to-one corresponding control points $c_i \in S_t$ and $c'_i \in S_s$. Our main idea is to sample a number of sampling points on the target stroke. Then, for a sample point $p \in S_t$ locates between two neighboring control points c_i and c_{i+1}, we find a corresponding sample point $p' \in S_s$ between two corresponding control points of c_i and c_{i+1} on the source stroke. Then the texture around p' is copied to p so that its direction is aligned to the direction of target stroke on p.

Graph cut composition. After copying the source textures from the input image to the target domain, many overlapping regions between individual textures are produced. In the step, we adopt graph cut approach to label which texture each pixel should come from for each overlapping region, so that these textures are seamlessly composited. Specifically, let $\Omega_{p,q}$ be an overlapping region of textures T_p and T_q. The composite energy function is defined as the sum of *data term* and *smoothness term* over all pixels in the overlapping region $E_{p,q} = \sum_{x\in\Omega_{p,q}} E^d_{p,q}(x, \lambda(x)) + \sum_{x\in\Omega_{p,q}} \sum_{y\in N(x)} E^s_{p,q}(x, y, \lambda(x), \lambda(y))$, where $\lambda(x)$ is the binary label of the pixel x. The data term is

$$E^d_{p,q}(x, \lambda(x)) = \begin{cases} 10000 & \text{if } x \in \partial\Omega_{p,q} \cup \partial T_p \wedge \lambda(x) = T_p \\ & \text{or } x \in \partial\Omega_{p,q} \cup \partial T_q \wedge \lambda(x) = T_q \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $\partial\Omega$ is the boundary of a region. The smoothness term is

$$E^s_{p,q}(x, y, \lambda(x), \lambda(y)) = \begin{cases} \min(d(x), d(y))/G(x, y) & \text{if } \lambda(x) \neq \lambda(y) \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where $d(x) = (\mathbf{C}_{T_p}(x) - \mathbf{C}_{T_q}(x))^4$ is a function that penalizes the color difference between the pixels on the two textures T_p and T_q, and $G(x, y) = \delta_{T_p}(x) + \delta_{T_q}(x) + \delta_{T_p}(y) + \delta_{T_q}(y)$ calculates the edge strength between the pixel x and y, $\delta(\cdot)$ refers to the color gradient magnitude on a pixel.

Global texture sampling. Our method synthsizes the textures along the target stroke. Because we only have the correspondences of the control points, we have to establish the correspondences for the rest sample points between the source and target strokes, so that the sum of a all compositing energies are minimized: $E = \sum_{\{p,q\}\in S_t} E_{p,q}$, where p and q are neighboring sample points. We adopt an dynamic programming approach to minimize the energy for all the sample points on a stroke.

3 Conclusion

We proposed a technique to synthesize object with user provided skeletal structures from an input objects and user provided strokes. The technique successfully handles both deformation and branching of the object.