Sketch-Based Tree Modeling Using Markov Random Field

Xuejin Chen[1], Boris Neubert[2], Ying-Qing Xu[3], Oliver Deussen[2], and Sing Bing Kang[4]

[1]University of Science and Technology of China
[2]University of Konstanz
[3]Microsoft Research Asia
[4]Microsoft Research

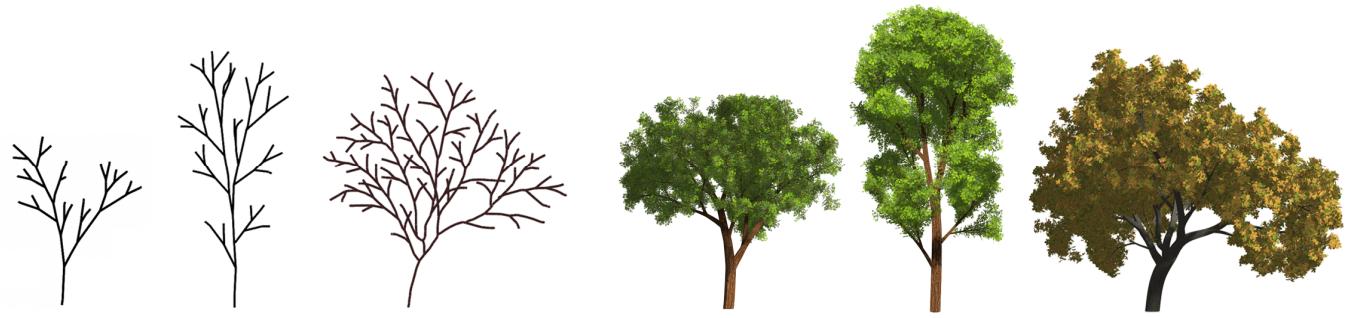

Figure 1: Examples of 3D tree models generated from freehand sketches. From left to right: three sketches, and corresponding tree models.

Abstract

In this paper, we describe a new system for converting a user's freehand sketch of a tree into a full 3D model that is both complex and realistic-looking. Our system does this by probabilistic optimization based on parameters obtained from a database of tree models. The best matching model is selected by comparing its 2D projections with the sketch. Branch interaction is modeled by a Markov random field, subject to the constraint of 3D projection to sketch. Our system then uses the notion of self-similarity to add new branches before finally populating all branches with leaves of the user's choice. We show a variety of natural-looking tree models generated from freehand sketches with only a few strokes.

Keywords: Sketching, tree modeling, geometric modeling, Markov random field.

1 Introduction

Achieving realism is one of the major goals of computer graphics, and many approaches ranging from physics-based modeling to image-based rendering have been proposed. Unfortunately, creating new content for realistic rendering remains tedious and time consuming. The problem is exacerbated when the content is being *designed*. 3D modeling systems are cumbersome to use and therefore ill-suited to the early stage of design (unless the design is already well-formulated).

Designers therefore continue to favor freehand sketching for conceptual design. Sketching appeals as an artistic medium because of its low overhead in representing, exploring, and communicating geometric ideas. Indeed, such speculative representations are fundamentally different in spirit and purpose from the definitive media that designers use to present designs. What is needed is a seamless way to move from conceptual design drawings to presentation rendering; this is our focus. This paper extends sketch-based modeling to complex and realistic-looking 3D tree models and thus addresses one of the still existing gaps in this area.

We introduce a new, easy-to-use, and flexible method for creating tree models from freehand sketching. It allows the user to very quickly generate a variety of unique 3D models of trees. Given the 2D sketch, our system searches for a 3D interpretation whose projection matches the sketch *and* is natural-looking. We formulate the problem within a graphical modeling framework, using a database of trees as priors. Inference is done in two steps. First, the initial shape is obtained by bottom-up local optimization at each branch segment from tree root to the rest of the drawn branches. Second, this shape is then refined to avoid interpenetration between branches.

Once the 3D branches have been recovered, the model is augmented with more branches using self-similarity as a guiding principle. Subtrees are randomly selected and appropriately scaled and oriented before being attached to end branches. Finally, the user can select the leaves to be automatically added to the branches based on botanical rules. This completes the 3D tree model. Examples of tree models generated from sketches can be seen in Figure 1.

2 Related Work

Rule-Based Plant Modeling. Techniques for computer-generated plants were introduced as early as 1966 by Ulam [1966]. They are soon followed by the introduction of L-systems as a formalism for simulating the development of multicellular organs in terms of division, growth, and death of individual cells [Lindenmayer 1968]. Prusinkiewicz et al. [1996] derived a systematic description for using L-systems to model plants. Other

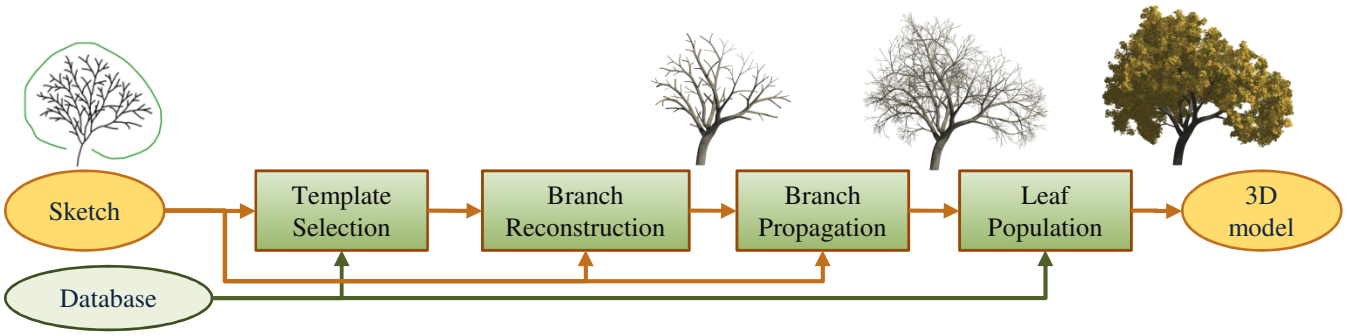

Figure 2: Overview of our sketch-based tree modeling system.

approaches have also been proposed, ranging from parameterized algorithms [Oppenheimer 1986; de Reffye et al. 1988; Holton 1994; Weber and Penn 1995] to combined approaches such as the xfrog system [Lintermann and Deussen 1999]. However, all these rule-based systems require the user to manually finetune a number of parameters in order to create the desired model.

Rule-based systems are difficult for the novice user to operate because they require not only specialized knowledge on biomechanics and biology for effective parameter specification. The user must also understand how the rules are applied or even formulated evenly. In a number of such systems, the global shape of trees is difficult to control—slight changes in the local rules may result in significant changes in the global shape.

The xfrog system and subsequent graphical L-system editors [Prusinkiewicz et al. 2001] allow the user to manipulate complex parameters graphically. Despite the increased ease of use, such systems still require the user to specify the less intuitive function plots, curves, and surface parameters that govern appearance (which are separate from the model shape).

In comparison, our sketching system generates a 3D tree model by having the user merely sketch the desired tree shape. The user does not need to understand what L-systems are or know what parameters need to be manipulated. Our system solves the inverse procedural modeling problem by inferring the generating parameters and 3D tree shape from the drawn sketch. While L-systems require parameters to be predefined and manipulated, our sketching system provides the user a highly intuitive way to produce the desired 3D tree models.

Sketch-based Tree Modeling. Sketch-based systems were developed to provide a more intuitive way of generating plant models. For example, Okabe et al.'s system [2005] reconstructs the 3D branching pattern from 2D drawn sketches in different views by maximizing distances between branches. They use additional gesture-based editing functions to add, delete, or cut branches. Moreover, example-based editing is supported to generate branches or leaves using some existing tree models.

Their system, however, requires the user to draw many branches to describe detailed structures. Because their system does not support automatic propagation of branches, a complex tree would require extensive user interaction. In comparison, our system allows the user to generate 3D models by merely drawing a few strokes in single view and optionally defining the overall shape of the tree by loosely sketching the contour of the crown.

Ijiri et al.'s system [2006] is based on L-systems. The user draws a single free-form stroke to control the growth of a tree. The change in the shape of the stroke is used as a graphical metaphor for modifying the L-system parameters. However, this system supports only two simple production rules, and the user is not allowed to control the overall shape of the tree. This severely limits the expressive power of the system. In contrast, our system is capable of generating complex, natural-looking tree models from a limited number of strokes to describe the tree shape.

Image-Based Plant Modeling. Rather than requiring the user to manually specify the plant model, there are approaches that instead use images to help generate 3D models. For example, Shlyakter et al. [2001] use an L-system-based growth mechanism that is constrained by the visual hull produced from photographs. A purely image-based modeling approach is described by Reche-Martinez et al. [2004]. Here, a set of carefully registered photographs is used to determine the volumetric shape and opacity of a given tree. The data is stored as a huge set of volume tiles and therefore requires a significant amount of memory. Furthermore, its generic volumetric representation makes it hard to edit.

Neubert et al. [2007] proposed a method to produce 3D tree models from several photographs based on limited user interaction. Their system is a combination of image-based and sketch-based modeling. From loosely registered input images, a voxel volume is achieved with density values which are used to estimate an initial set of particles. A 3D flow simulation is performed to trace the particles downward to the tree basis. Finally, the twigs and branches are formed according to the particles. The user is required to draw some branches if there is no information about branching in the images. Density information is critical to simulate the particle. The user has to draw the foliage density if image information is missing. The final reconstructed branches does not have the exactly same shape with the drawn branches.

By comparison, our system only requires the user to draw the branches with several strokes in *single view* and generates the same branch shape through a sketch. With the approach in [Neubert et al. 2007], it is difficult to simulate specific patterns of smaller branches using the particle flow mechanism. For the trees with very steep branching angles, it is hard to create the model using flow simulation because the particles merge when getting too close. This limits the number of achievable plant forms.

Tan et al. [2007] proposed another method to reconstruct the 3D model from multiple images. The user can also manually edit the branch structures. However, the system requires a significant amount of preprocessing work such as segmentation of branches, leaves, and background. User intervention may be required to correct errors in the 3D branch extraction or seed branch growth. However, mismatches between the construction of hidden branches and

observed leaves tend to produce floating 3D leaves.

Image-based approaches have the best potential for producing realistic-looking plants, since they rely on images of real plants. At the same time, they can produce only models of preexisting plants. Designing new plants would be a major issue without manual editing. Our sketching system is a good compromise that allows a small amount of intuitive input to produce new and realistic-looking trees.

3 Overview of System

The components of our tree sketching system are shown in Figure 2. The user needs to simply provide only a few strokes of branches, and optionally the crown of the tree. The database contains typical tree exemplars and their associated global parameters. Based on the shape of the sketch, the system first selects the closest tree exemplar ("template"); the template's global parameters are subsequently used as a prior for constructing the 3D geometry.

We assume that the sketch is drawn under orthographic projection. This allows the problem of constructing the 3D geometry of the sketch to be reduced to estimating the depths of branch segment endpoints. A reasonable shape of a tree can be reconstructed from its projection because trees have characteristic shapes, which provide powerful priors for 3D reconstruction. Such priors allow humans to perceive tree shapes from sketches. The location of each branch depends on the location of adjacent branches but the overall shape is dictated by global tree parameters. The local interconnectivity of information and imposition of global priors makes the reconstruction problem a natural fit for the use of Markov random field (an undirected graphical model). There is a direct mapping of branches to graph nodes, local interconnectivity of branches to interaction between nodes, and global tree parameters to data terms at nodes.

As such, we formulate the problem as Markov random field, with each branch segment as a node and its depth as a variable. More specifically, we are dealing with a Markov tree. (For a detailed description, please refer to textbooks such as [Bishop 2006].) In addition, we impose rules governing the tree shape as spatial relationships between neighboring nodes. These relationships are made explicit by introducing additional nodes, producing what is called a *factor graph* (a bipartite graph with two kinds of nodes).

Solving the factor graph produces the 3D shape of the drawn branches. The system then propagates branches using the principle of self-similarity: it randomly selects replication blocks, scales, and reorients them, and then attaches them to open branches. If drawn, the crown constrains the overall shape of the tree during branch propagation. If the crown is not drawn, the branches are propagated by a fixed number of generations. To complete the tree model, the user can either select a leaf template from the tree database, or use the default leaf associated with the preselected template. The system populates the tree based on botanical rules. While this is only an approximation of natural diversity, the variety of trees shown in this paper demonstrates the visual modeling power of our system.

In the next section, we first describe the tree data structure before detailing our 3D reconstruction technique.

4 Tree Data Structure

The tree is composed of a set of branch segments. Each branch segment b has its local coordinate system, the position of end point $\mathbf{p}$ and the segment vector $\mathbf{v}$. With each branch segment as a node, we build a Markov random field of the whole tree. Each segment is the parent of all its outgoing branch segments. Since there is one and only one path between each pair of nodes, the Markov random field is converted to a Markov tree. For example, from the sketch (Figure 4 (a)), the Markov tree is Figure 4 (b).

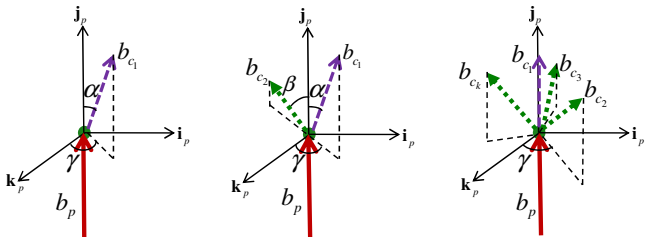

Figure 3: Illustration of tree parameters described in Section 4.

The relation between parent-child branch segments is represented by the geometry transformation. In the local coordinate system of the parent b_p, the scaled rotation from the segment vector $\mathbf{v}_p^p$ of b_p to its child segment vector $\mathbf{v}_c^p$ is defined by a scale s and two rotation angles ϕ and φ around the x and y axis of the parent coordinate system respectively. Then we have $\mathbf{v}_c^p = sR_y(\varphi)R_x(\phi)\mathbf{v}_p^p$.

The branching shapes are typically spatially variant for the whole tree. To account for this, we use Gaussian distribution to characterize the probability distribution of the tree parameters. The scale s of each pair of parent-child segments has the identical distribution, that is $s \sim \mathcal{N}(\mu_s, \sigma_s)$. However, for the rotation angles, the distribution is not identical.

For a parent segment b_p having K child segments $\{b_{c_1}, \ldots, b_{c_k}\}$, the apical segment b_{c_1} is distinguished. Assume the divergence angles and the orientation adjustment angles of all the apical segments have identical distribution, that is $\mathcal{N}(\mu_\alpha, \sigma_\alpha)$ and $\mathcal{N}(\mu_\gamma, \sigma_\gamma)$ respectively. The other child segments are lateral. The divergence angle of all the lateral segments have identical distribution, that is $\mathcal{N}(\mu_\beta, \sigma_\beta)$. The distribution of orientation adjustment angle depends on its label. Based on the different orientation adjustment angle, the child segments distribute evenly around their parent segment. As shown in figure 3,

1. If $K = 1$, $\phi_{c_1} \sim \mathcal{N}(\mu_\alpha, \sigma_\alpha)$ and $\varphi_{c_1} \sim \mathcal{N}(\mu_\gamma, \sigma_\gamma)$. The distribution parameters of first child segment is $\theta_1 = \{\mu_\alpha, \sigma_\alpha, \mu_\gamma, \sigma_\gamma\}$.

2. If $K = 2$, the rotation angles of the two child segments are $\phi_{c_1} \sim \mathcal{N}(\mu_\alpha, \sigma_\alpha), \varphi_{c_1} \sim \mathcal{N}(\mu_\gamma, \sigma_\gamma)$ and $\phi_{c_2} \sim \mathcal{N}(\mu_\beta, \sigma_\beta), \varphi_{c_2} \sim \mathcal{N}(\mu_\gamma + \pi, \sigma_\gamma)$ respectively. The two sets of distribution parameters are $\theta_1 = \{\mu_\alpha, \sigma_\alpha, \mu_\gamma, \sigma_\gamma\}$ and $\theta_2 = \{\mu_\beta, \sigma_\beta, \mu_\gamma + \pi, \sigma_\gamma\}$.

3. If $K > 2$, the rotation angles of apical segment is $\phi_{c_1} \sim \mathcal{N}(\mu_\alpha, \sigma_\alpha), \varphi_{c_1} \sim \mathcal{N}(\mu_\gamma, \sigma_\gamma)$. For the other child (lateral) segments, their rotation angles are $\phi_{c_i} \sim \mathcal{N}(\mu_\beta, \sigma_\beta), \varphi_{c_i} \sim \mathcal{N}(\mu_\gamma + \frac{2\pi(i-1)}{K-1}, \sigma_\gamma)$, $i = 2, \ldots, K$. The corresponding parameter for each child segment is: $\theta_1 = \{\mu_\alpha, \sigma_\alpha, \mu_\gamma, \sigma_\gamma\}$ and $\theta_i = \{\mu_\beta, \sigma_\beta, \mu_\gamma + \frac{2\pi(i-1)}{K-1}, \sigma_\gamma\}$, $i = 2, \ldots, K$.

Let $\Theta = \{\mu_s, \sigma_s, \mu_\alpha, \sigma_\alpha, \mu_\beta, \sigma_\beta, \mu_\gamma, \sigma_\gamma\}$ denote the global tree parameters, we can deduce the Gaussian distribution parameters θ_i of the rotation angles of each child segment b_{c_i} according to its label i. We now describe how we reconstruct the 3D branches within our graphical modeling framework.

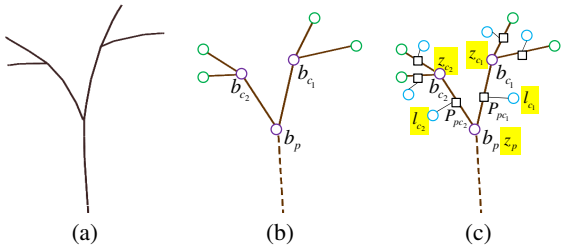

Figure 4: Mapping sketch to factor graph. (a) The sketched tree, (b) corresponding Markov tree, (c) final factor graph. Each node in the graph represents a branch segment in the sketch.

5 Branch Reconstruction

The strokes in the sketch are first split into a set of inter-connected branch segments. Since we assume orthographic projection, given the sketch (Figure 4(a)), we need to only extract the depths associated with the branch segment ends.

5.1 Factor Graph Construction

Each branch segment b_i is represented by the position of its end point $\mathbf{p}_i$ and its direction $\mathbf{v}_i$. Given a parent segment b_p and its child segment b_c, $\mathbf{v}_c = \mathbf{p}_c - \mathbf{p}_p$. The transformation of these two vectors are modeled by a scale s and rotation matrix R_{pc} such that $\mathbf{v}_c = sR_{pc}\mathbf{v}_p$. Note, however, that the tree parameters Θ specify the rotation R_c^p between parent segment and child segment in the parent's local coordinate system while the observed (x,y) and unknown z are defined in the global coordinate system. We need to convert the coordinate systems from local to global:

$$\mathbf{v}_c = sR_{pc}\mathbf{v}_p = sR_p^0 R_c^p R_0^p \mathbf{v}_p, \tag{1}$$

where $R_{pc} = R_p^0 R_c^p R_0^p$, R_0^p is the rotation matrix from the global coordinate system to the local coordinate system of the parent segment, R_p^0 is the inverse.

The rotation angles ϕ and φ of each pair of parent-child branch segments can be deduced from the rotation matrix R_c^p. Our target is to look for the optimal depth z_i of each branch so that the transformation parameters have the largest probability according to the Gaussian distribution. As indicated in Section 4, for a parent segment with K child branches, there are K sets of parameters $\theta_i, i = 1,\ldots,K$ associated with the rotation angles. To differentiate between the type of child branches, we introduce another variable, the label for the child segment, $l_i = \{1,2,...,K\}$ to get the distribution from Θ. Given Θ, the joint probability of the two branch vectors $\mathbf{v}_i$, $\mathbf{v}_{p_i}$ and label l_i of the child segment is

$$P(\mathbf{v}_i, \mathbf{v}_{p_i}, l_i|\Theta) \quad \propto \quad P(s_i|\Theta)P(\phi_i|\theta_{l_i})P(\varphi_i|\theta_{l_i}). \tag{2}$$

From the 2D strokes drawn by the user, 2D coordinates of the branch nodes $X = \{x_1,\ldots,x_N\}$ and $Y = \{y_1,\ldots,y_N\}$ are observed for the N branch segments in the tree. We look for the optimal value for the unknown depths $Z = \{z_1,\ldots,z_N\}$ of the segment vector and labels $L = \{l_1,\ldots,l_N\}$ for all the branch segments to maximize the posterior given the sketch and global parameters Θ:

$$P(Z,L|X,Y,\Theta) \propto \prod_{i=1}^{N} P(\mathbf{v}_i, \mathbf{v}_{p_i}, l_i|\Theta), \tag{3}$$

assuming independence between child segments (for simplicity).

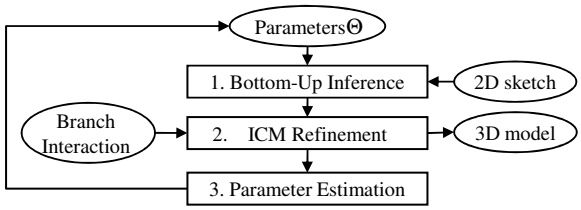

Figure 5: The framework of Markov tree inference.

The posteriors in (3) are made explicit by introducing the factor nodes in the factor graph. The child segment label l_{c_i} between each pair of parent-child segments is introduced as another variable node to the factor graph. The square node P_{pc_i} between between b_{c_i} and b_p represents the conditional joint probability defined in (2). By multiplying all the factor nodes in the factor graph, we arrive at the posterior $P(Z,L|X,Y,\Theta)$ in (3) for the whole tree.

5.2 Markov Tree Inference

By inferring the unknown variables of the branch segments, the 2D sketch is mapped to the 3D branches by relying on the tree parameters Θ of the template selected from the database (Section 7). The default behavior of our system is to jointly estimate the depths of the branch segments *and* Θ, using the template values as initialization. As a result, in optimizing the tree shape based on the sketch, the final tree parameters may drift from those of the template. The user can choose to override this default behavior and insist that Θ be preserved in the optimization. The characteristics of the final reconstructed tree would be the same as those of the template, but at a cost of sub-optimality of fit to the sketch.

Overriding the default behavior makes the optimization simpler, because Θ is unchanged throughout the optimization. We describe this case first (Section 5.2.1). Branch interaction (ensuring good spatial distribution and avoiding interpenetration) is accounted for in the optimization. We then describe the default system behavior in Section 5.3, where both tree shape and Θ are optimized in an EM-like (expectation-maximization) fashion (Figure 5).

5.2.1 Inferring Branches with Fixed Parameters

In Formula (2), the calculation of rotation angles between child and parent segment vectors depend on rotation matrix $R_{p_i}^0$ linking local to global coordinate systems, as shown in Formula (1). Unfortunately, there is no closed-form solution for the objective function defined in Formula (3).

We approximate global inference in Formula (3) with a two-step approach. The first step is bottom-up inference which computes a local solution at each generation of parent-child branches, starting from root and ending with the terminal segments in the sketch. The second step refines the first step's results using the Iterated Conditional Mode (ICM) (see [Bishop 2006]).

Bottom-up Inference To handle the rotation chain in the tree, we use *ancestral sampling* [Bishop 2006] to get the best sample of the unknown variables z_i, l_i of each branch b_i generation by generation, starting from the root segment. Generally, the root branch segment b_r's local coordinate system is consistent with the global coordinate system $R_r^0 = I$ (identity 3×3 matrix) and $z_r = 0$. Then, we estimate the hidden variables of its descendant branch segments generation by generation. The rotation between the local and global coordinate systems of a child segment is propagated as $R_c^0 = R_p^0 R_c^p$.

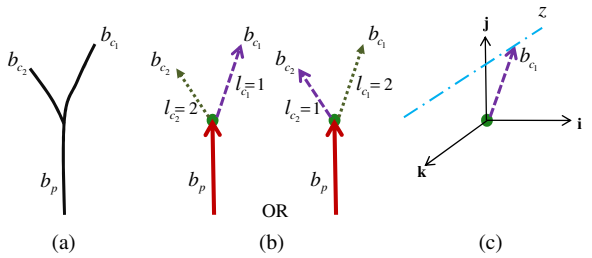

Figure 6: Illustration of optimization for one generation: (a) Sketch, (b) two hypotheses of labels for the child segments, and (c) given the hypothesis, search depth z (here, for b_{c_1}).

Suppose a parent branch segment b_p has segment vector $\mathbf{v}_p$ and local-to-global rotation R_p^0. Suppose, also, that b_p has K child branches $b_{c_i}, i = 1,\ldots,K$. For this generation, while fixing $\mathbf{v}_p$, we look for the optimal values of z_{c_i}, l_{c_i}, $i = 1,\ldots,K$ associated with the child branches to maximize the posterior $\prod_{i=1}^{K} P(\mathbf{v}_{c_i},\mathbf{v}_p,l_{c_i}|\Theta)$. The label l_{c_i} of all the child segments of b_p must be unique. We test all combinations of the labels and choose the best solution at each generation. By optimizing child branches from root to terminal nodes, we can approximate the posterior in Formula (3).

Take Figure 6 as a simple illustration of the local optimization. Here, the parent segment b_p has two child segments b_{c_1} and b_{c_2}, whose labels l_{c_1}, l_{c_2} and depths z_{c_1}, z_{c_2} are unknown (Figure 6(a)). There are two possible combinations of labels of b_{c_1} and b_{c_2} (Figure 6(b)). Under each hypothesis of segment labels, the corresponding parameter prior θ_1 and θ_2 for b_{c_1} and b_{c_2} are extracted from Θ as described in Section 4. For each child segment, we search for the optimal z_{c_i} to maximize the posterior $P(\mathbf{v}_{c_i},\mathbf{v}_p,l_{c_i}|\Theta)$ given the hypothesized branch label l_{c_i} (Figure 6(c) shows the case for child segment b_{c_1}). Finally, the solution under the label hypothesis that maximizes $\prod_{i=1}^{2} P(\mathbf{v}_{c_i},\mathbf{v}_p,l_{c_i}|\Theta)$ is chosen as the solution.

By optimizing the unknown variables at each node from the root of the terminal nodes generation by generation, we now have an initial 3D shape of the drawn branches. The next step is local refinement to account for branch interactions (competition between branches and avoiding interpenetration).

ICM Refinement Since the process of bottom-up inference produces local solutions at every generation, we follow up with Iterated Conditional Mode (ICM) [Bishop 2006] to refine the result. In order to avoid the complexity caused by the mapping between different branch coordinate systems, the depth z_i of each vertex is directly refined to make the divergence angle and scale of each pair of child and parent segment as consistent with the corresponding global parameters as possible.

More specifically, for a pair of parent-child segment b_i and b_{p_i}, the distributions of scale $s_i = |\mathbf{v}_i|/|\mathbf{v}_{p_i}|$ and divergence angle $\phi_i = \cos^{-1}\left(\frac{\mathbf{v}_i\cdot\mathbf{v}_{p_i}}{|\mathbf{v}_i||\mathbf{v}_{p_i}|}\right)$ are respectively defined as $s_i \sim \mathcal{N}(\mu_s,\sigma_s)$ and $\phi_i \sim \mathcal{N}(\mu_\alpha,\sigma_\alpha)$ if $l_i = 1$, otherwise $\phi_i \sim \mathcal{N}(\mu_\beta,\sigma_\beta)$ (see Section 4). The probability considering these two items is defined as $P_{sa}(\mathbf{v}_i,\mathbf{v}_{p_i}|l_i,\Theta) = \prod_{i=1}^{N} P(s_i|\Theta)P(\phi_i,l_i|\Theta)$.

During the growth of a tree, each branch competes with others to get as much space as possible. As a result, the tree branches typically distribute uniformly within the tree volume. We model this competition between branches as probability field to affect the inference. The 3D space constrained within the crown is discretized to voxels (in our experiments, we use a grid of $N_v = 25\times25\times25$), with each voxel's branch density being d_i at its center $\mathbf{p}_{v_i}$. The probability of a segment growing to the point $\mathbf{p}_i$ is $P_d(\mathbf{p}_i) = \prod_{i=1}^{N_v}\exp\{\frac{-k_e d_i}{1+\|\mathbf{p}_i-\mathbf{p}_{v_i}\|^2}\}$, where k_e is the *branch interaction factor*. Its default value is 5. By encouraging the branch to grow to the position where the density of the branches is low, the branches attempt to get as much free space around them as possible.

At each step of ICM, we update one node by $z^{(t+1)} - z^{(t)} = \frac{\partial P_{ICM}}{\partial z_i}|_{\{\mathbf{v}_j\},j=1,\ldots,N,j\neq i}$ while fixing the other nodes to increase $P_{ICM} = \prod_{i=1}^{N} P_{sa}(\mathbf{v}_i,\mathbf{v}_{p_i}|l_i,\Theta)P_d(\mathbf{p}_i)$. The refinement terminates when the changes in the nodes $\sum_{i=1}^{N}|z_i^{(t+1)} - z_i^{(t)}| < \varepsilon(\varepsilon = 0.01)$.

Interpenetration Between Branches. During the inference of the depths of branches, branch interpenetration is avoided if possible. We check for interpenetration right after the bottom-up inference, and, if necessary, move the affected branches away from each other along the z-axis. The ICM refinement step is then applied to make the branching shape more consistent. (An example showing avoidance of interpenetration through the refinement step can be seen in the video.)

5.3 Inferring both Branches and Parameters

The default behavior of our system is to jointly optimize Z, L, and the global tree parameter Θ to fit the current sketch better. To simplify the inference, we adopt an EM-like (expectation-maximization) algorithm. In the expectation step, the integral of all hidden variables (Z and L) is computed to estimate the parameters Θ. We are primarily interested in finding the optimal hidden variables rather than parameter estimation, and modified the expectation step to reflect this.

At iteration t, instead of evaluating the expectation given parameters $\Theta^{(t)}$, we look for the optimal hidden variables $Z^{(t)}$ and $L^{(t)}$ of all the branches that maximizes the posterior $P(Z^{(t)},L^{(t)}|X,Y,\Theta^{(t)})$ (*E-step*). The maximization step uses $Z^{(t)}$ and $L^{(t)}$ to estimate $\Theta^{(t)}$ by maximizing $P(\Theta|Z^{(t)},L^{(t)},X,Y)$ (*M-step*):

$$P(\Theta|Z^{(t)},L^{(t)},X,Y) = \prod_{i=1}^{N} P(\Theta|s_i,\alpha_i,\beta_i,\gamma_i). \quad (4)$$

At the *E-step*, $Z^{(t)}$ and $L^{(t)}$ are computed using fixed parameters $\Theta^{(t)}$ in the same way as described in Section 5.2.1. At the *M-step*, the four parameters of the child branches are mapped to the Gaussian distribution defined by Θ, that is $(s_i,\alpha_i,\beta_i,\gamma_i) \sim \{\mathcal{N}(\mu_s,\sigma_s),\mathcal{N}(\mu_\alpha,\sigma_\alpha),\mathcal{N}(\mu_\beta,\sigma_\beta),\mathcal{N}(\mu_\gamma,\sigma_\gamma)\}$. The tree parameters are updated ($\Theta^{(t)} \rightarrow \Theta^{(t+1)}$) as the average and variance of the branch rotation angles and scale at iteration t.

In our experiments, we require fewer than 10 iterations to generate good results. This allows our system to be interactive. The results in our paper show that the reconstructed 3D tree model is plausible even though it is a local solution. After depths of all the branch segments have been recovered, we use cubic Bezier to interpolate the shape of the branches and generate meshes [Neubert et al. 2007].

6 Branch Propagation and Leaf Population

The sketch drawn by the user is typically sparse. The resulting 3D model associated with the sketch alone is unlikely to look compelling due to the lack of complexity. One option would be the user draws a complicated tree structure, which would be manual-intensive. While this option is supported by our system, there is a

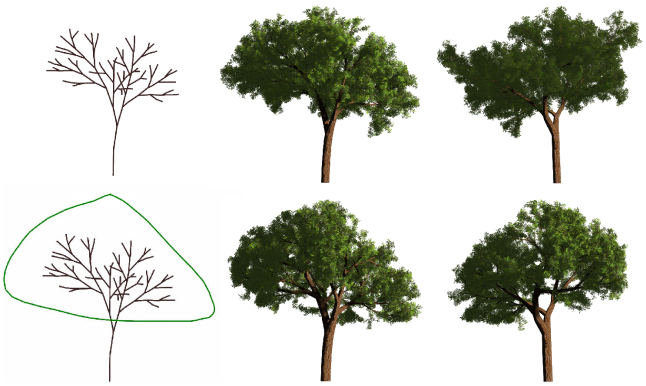

Figure 7: Effect of crown. Top row: without crown, bottom row: with crown. From left to right: sketch, two views of the complete tree model.

better way: our system automatically adds complexity to the model based on what has been drawn.

This automatic feature is based on the principle of self-similarity which is widely used in plant modeling [Shlyakhter et al. 2001; Tan et al. 2007]. Our system randomly picks a replication block (a parent segment and its child segments) from the model, picks an open branch segment, scales, and orients the replication block, and attaches the transformed replication block to the open branch.

The growth of branches is limited by the crown—more specifically, its surface of local revolution (see [Okabe et al. 2005] for details). If the new propagated branches reaches this surface, they are scaled back to touch the surface, and the propagation terminates here. Note that if the crown is not drawn by the user, the system propagates the branches by a fixed number of generations. The default is 5 generation, which produces good results. The user can specify a different number if desired. Figure 7 shows branch propagation results with and without crown. By drawing the crown, the user exerts more control over the shape of the tree.

The branch growing procedure is similar to the hidden branch construction algorithm described in [Tan et al. 2007], but with two important differences. First, they compute the 3D hull from multiple photos, while ours is generated from a 2D curve. Second, in their system, the branch growth is achieved by *randomly orienting* branches before adding them to the tree. In our system, the new branches inherit the branch angle parameters associated with the replication blocks, thus fixing their orientations with respect to the open branches. To make the tree appear more natural, our system also introduces new lateral branches along disproportionately long branches caused by inaccurate sketch.

The leaves and twigs (small branches) attach to large branches in regular and predictable ways. Instead of generating leaf geometry from a sketch, our system by default uses the leaf model associated with the chosen template (Section 7). The user can override the default leaf model by selecting a different one from any other tree exemplar in the database. The leaves are replicated and placed on branches based on botanical laws governing leaf arrangement. For example, the leaves on the same twig can be directly opposite or alternate with a deviation angle.

We now describe our tree database, which is used to supply an appropriate set of tree parameters Θ as prior to the graphical modeling framework.

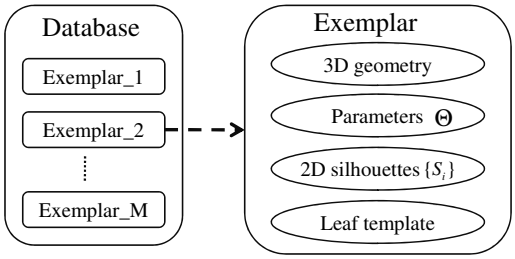

Figure 8: Structure of tree database.

7 Database of Tree Templates

The chances of producing natural-looking tree models are enhanced if geometry reconstruction is guided by pregenerated natural-looking ones. Such models can be obtained through image-based modeling [Reche-Martinez et al. 2004; Tan et al. 2007] or through careful modeling [Lintermann and Deussen 1999; Neubert et al. 2007]. Our tree modeling technique is agnostic to the method through which the tree models in the database are generated; our technique adapts to whatever tree parameters are supplied. In our implementation, we use tree models that were generated using the method described in [Neubert et al. 2007] as templates. There are 20 different tree exemplars in our database; they were chosen to represent a reasonably wide variety of trees.

Our system automatically looks for an exemplar from the database that best matches the sketch as the template to use for 3D reconstruction. The selection is done by comparing the crown and branching shapes of the sketch with the 2D silhouettes and projected branching shapes (respectively) of exemplars in the database.

The user may draw the crown as a single curve or series of curves. Our system then computes the convex hull of both the crown and branch strokes except the root branch. This convex hull is used to match the 2D silhouettes of exemplar in the template selection process.

We use the footprint descriptor [Lamdan et al. 1988] as curve features for matching. It is a simple but robust method to measure the similarity of curves. First, we normalize the crown curve so that $\max(w,h) = 1$, where w,h are the width and height of the crown's bounding rectangle. Let the center of the normalized curve be C. We then compute the radial distance distribution from C to the curve, yielding a K_f dimensional vector $\mathbf{f}$ (for K_f regularly sampled directions). $\mathbf{f}$ is the footprint descriptor. Each tree exemplar is associated with a set of 2D silhouettes under different views. We compute the footprint descriptor $\mathbf{f}_E$ for each silhouette in the same manner. The similarity of the crown shape between the sketch S and the exemplar E under a view v is defined as $L_c(S,E|v) = -\|\mathbf{f}_S - \mathbf{f}_{E|v}\|$.

To measure the similarity between the branching shapes of the sketch and exemplar, we calculate the average 2D length ratio and angle between each pair of parent-child segments of tree. For a parent segment in the 2D sketch, it is not clear if each child segment is apical or lateral. Our system assumes the child segment with the smallest 2D angle with the parent segment is apical while the rest are lateral. Assuming the length ratio s_{2D}, apical angle α_{2D}, and lateral angles β_{2D} of the 2D branches are Gaussian distributed, we estimate their mean and standard deviation: $x_{2D} \sim \mathcal{N}(\mu_{x,2D}, \sigma_{x,2D})$, where $x = s, \alpha, \beta$. For each exemplar in the database, its 3D geometry is projected to 2D under different views. For each 2D projection, the Gaussian distributions of length ratio $s_{e,2D}$, apical angle $\alpha_{e,2D}$, and lateral angle $\beta_{e,2D}$ between each pair of parent-child segments are similarly estimated: $x_{e,2D} \sim \mathcal{N}(\mu_{x,e,2D}, \sigma_{x,e,2D})$, $x = s, \alpha, \beta$.

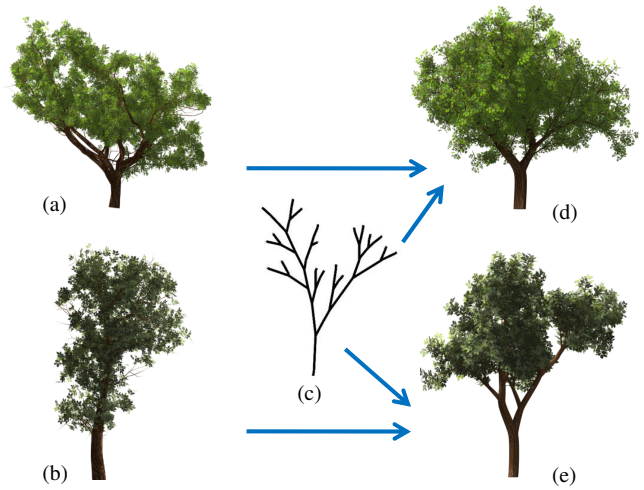

Figure 9: Different results generated from the same sketch, but with different manually assigned exemplars. (a)(b) are two different exemplars used to guide the inference; (c) is the input sketch; (d)(e) are corresponding output tree models.

We define similarity of the branching shapes between the sketch and the exemplar under view v as

$$L_b(S,E|v) = -\sum_{x=s,\alpha,\beta} \frac{(\mu_{x,2D} - \mu_{x,e,2D|v})^2}{\sigma_{x,2D}^2 + \sigma_{x,e,2D|v}^2}.$$

If the user draws the crown, the similarity between the sketch and exemplar (from a given view v) is the sum of similarities in crown and branching shapes: $L(S,E|v) = L_c(S,E|v) + L_b(S,E|v)$. If the crown is not drawn, the similarity is set as $L(S,E|v) = L_b(S,E|v)$. The exemplar is selected for which similarity between one of its views and the sketch (given by $L(S,E|v)$) is maximized. This exemplar (with parameters Θ) will be the default template for shape inference. The user may choose to override the default by simply clicking a thumbnail to select an exemplar.

Figure 9 shows the results generated from the same sketch but with different exemplars. This illustrates that our system is capable of generating models that can look radically different, depending on which tree exemplar gets chosen.

Note that the selected exemplar serves to provide a good initial point of the tree parameters; the final parameters inherit tree characteristics from *both* the selected exemplar and the drawn sketch. Therefore, perceptually different 3D tree models can be generated from different sketches based on the same exemplar. From the 20 exemplars in our current database, a good variety of tree models can be generated (as shown throughout the paper).

The tree exemplars serve to add realism to the output, but are not absolutely necessary. It is possible to just have preset tree parameters which are then used for all sketches. However, the resulting tree shapes may not appear as compelling. As with most data-driven recovery systems, the more tree exemplars that are available (with a wider variation of shapes), the better our results are expected to be.

8 Results and Discussions

Figure 10 shows a variety of results generated from sketches of different trees. The complexity of the sketch ranges from a small number of drawn strokes to a large number, with and without the crown.

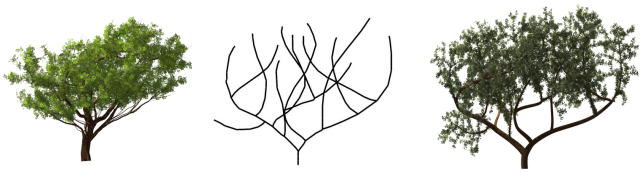

Figure 11: Another comparison between the exemplar and resulting tree model. From left to right: Exemplar automatically selected by our system, input sketch, and output tree model.

Figure 12: Failure example. From left to right: sketch, two views of the reconstructed branches.

As shown in (a), a complex and reasonably realistic-looking tree can be generated from just 8 strokes. Of course, the user can better control the specific shape of the tree by drawing more branches and the crown, as can be seen at (b). Figure 10(c), (d) and (e) show three other tree models generated from sketches with varying numbers of strokes, with and without the crown.

Generally, the user can create visually compelling tree models at interactive rates using our system (for a 2.67 GHz PC). The reconstruction of simply sketched strokes is accomplished interactively. After propagating for a few seconds, realistic tree models can be produced, as shown in Figure 10 (a) and (b). The user has the option of generating more complicated tree models with more branches, at the expense of longer propagation time. For example, the propagation of Figure 10 (e) took about 50 seconds to produce 4,291 branch segments.

Our system is able to generate complex and realistic-looking trees having distinct branching structure from freehand sketch. The results are as good as what the database can provide. The user is also allowed to creatively generate new trees with irregular shape (Figure 11). There are significant differences between the exemplar and the output. However, it is not easy for our system to model the tree with curvy branches such as liana or calyx canthus. Figure 12 shows such an example. The user intended to draw a tree with a curvy main branch growing along one direction. However, our optimization objective was constructed to make the branches consistent throughout the whole tree and to make them spread out. As a result, the generated 3D model does not look very natural (as seen from a side view).

In our current implementation, our system can generate trees with a particular type of branching structure. While many tree species can easily be approximated by a self-similarity, others such as palm trees or spruces are hard to create. It is possible to modify our system to accommodate spatially-varying parameter sets or branching structures, but this is a topic for future work. The cost is a more complicated interface which presents the user with more (potentially domain-specific) options.

There are several other possible extensions to our system. It currently has no editing options—it would certainly be useful for the user to be able to interactively rotate the completed tree to edit as desirable. Editing operations include branch removal or addition and depth changes (and have the system automatically update the model). Furthermore, in our current implementation, the propaga-

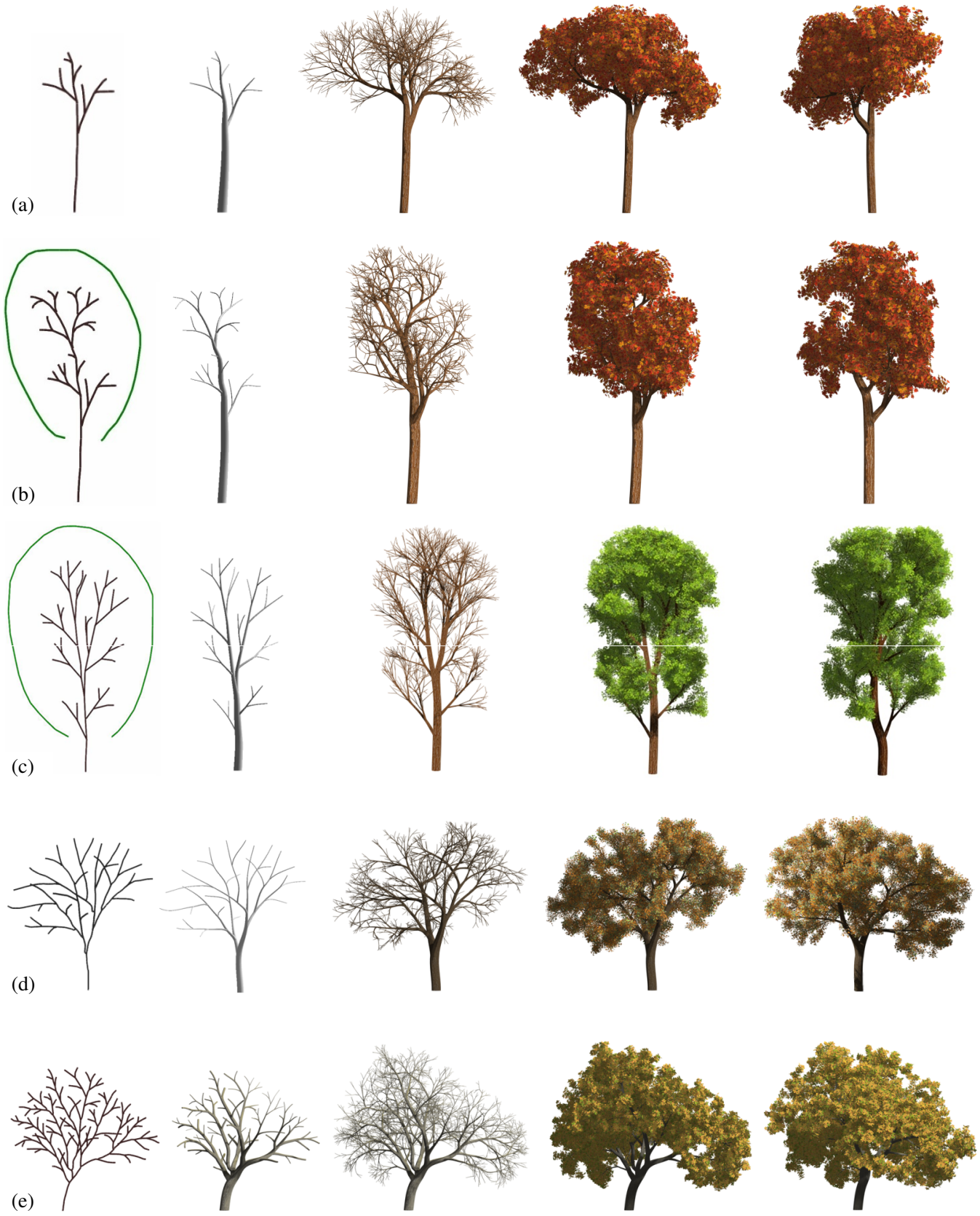

Figure 10: A variety of results generated from sketches of different trees. From left to right: sketch, after branch reconstruction and propagation, and two views of the complete tree model.

tion of thousands of branches could not be finished in interactive rate. In the future, the implementation would be optimized to get interactive propagation.

The current version of our system does not provide the option for specifying environmental effects to influence the shape of the tree model. It may, in certain cases, be desirable to be able to specify obstacles (e.g., buildings) for the tree to "grow" around. Another interesting case is to generate models of several trees in close proximity to each other, with each influencing the development of others.

9 Concluding Remarks

We have presented a new, easy-to-use system that is capable of generating realistic-looking 3D models from freehand sketches. The key to this system is the graphical modeling framework that takes a tree template as prior. By modeling the branch interaction with Markov tree, the 3D shape can be inferred from the drawn sketches. Our system makes use of a database of pregenerated realistic-looking tree models, which contributes to the high quality of the output. Once the depths for the sketch have been extracted, our system replicates branches to add complexity to the tree model, followed by leaf population based on botanical rules. In this paper, we demonstrated a wide range of tree models that can be generated from sketches of varying complexity.

Acknowledgement

We would like to thank the reviewers for their valuable feedback. We are grateful to Fang Wen and Lin Liang at Microsoft Research Asia for initial discussions of the project.

References

BISHOP, C. 2006. *Pattern Recognition and Machine Learning*. Springer.

DE REFFYE, P., EDELIN, C., FRANCON, J., JAEGER, M., AND PUECH, C. 1988. Plant models faithful to botanical structure and development. In *Computer Graphics (SIGGRAPH '88 Proc.)*, J. Dill, Ed., vol. 22, ACM SIGGRAPH, 151–158.

HOLTON, M. 1994. Strands, gravity and botanical tree imagery. *Computer Graphics Forum 13*, 1, 57–67.

IJIRI, T., OWADA, S., AND IGARASHI, T. 2006. The sketch L-system: Global control of tree modeling using free-form strokes. In *Smart Graphics*, 138–146.

JIRASEK, C., PRUSINKIEWICZ, P., AND MOULIA, B. 2000. Integrating biomechanics into developmental plant models expressed using L-systems. In *Proceedings of the 3rd Plant Biomechanics Conference*, 615–624.

LAMDAN, Y., SCHWARTZ, J. T., AND WOLFSON, H. J. 1988. Object recognition by affine invariant matching. In *CVPR*, 335–344.

LINDENMAYER, A. 1968. Mathematical models for cellular interaction in development,parts I and II. *Journal of Theoretical Biology 18*, 280–315.

LINTERMANN, B., AND DEUSSEN, O. 1999. Interactive modeling of plants. *IEEE Computer Graphics and Applications 19*, 1, 56–65.

MĚCH, R., AND PRUSINKIEWICZ, P. 1996. Visual models of plants interacting with their environment. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 397–410.

NEUBERT, B., FRANKEN, T., AND DEUSSEN, O. 2007. Approximate image-based tree-modeling using particle flows. *ACM Trans. Graph. (SIGGRAPH) 26*, 3, 88.

OKABE, M., OWADA, S., AND IGARASHI, T. 2005. Interactive design of botanical trees using freehand sketches and example-based editing. In *Computer Graphics Forum*, Eurographics 2005, vol. 24.

OPPENHEIMER, P. 1986. Real time design and animation of fractal plants and trees. In *Computer Graphics (SIGGRAPH 86 Conf. Proc.)*, vol. 20, 55–64.

PRUSINKIEWICZ, P., HAMMEL, M., HANAN, J., AND MECH, R. 1996. L-systems: from the theory to visual models of plants. In *Machine Graphics and Vision*, 365–392.

PRUSINKIEWICZ, P., MÜNDERMANN, L., KARWOWSKI, R., AND LANE, B. 2001. The use of positional information in the modeling of plants. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 289–300.

RECHE-MARTINEZ, A., MARTIN, I., AND DRETTAKIS, G. 2004. Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Trans. Graph. 23*, 3, 720–727.

SHLYAKHTER, I., ROZENOER, M., DORSEY, J., AND TELLER, S. 2001. Reconstructing 3d tree models from instrumented photographs. *IEEE Computer Graphics and Applications*, 53–61.

TAN, P., ZENG, G., WANG, J., KANG, S. B., AND QUAN, L. 2007. Image-based tree modeling. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, ACM, New York, NY, USA, 87.

ULAM, S. 1966. Pattern of growth of figures: mathematical aspects. In *Module, Proportion, Symmetry, Rhythm*, G. Keps, Ed. Braziller, New York, 64–74.

WEBER, J., AND PENN, J. 1995. Creation and rendering of realistic trees. In *SIGGRAPH 95 Conf. Proc.*, 119–128.