Single Image Tree Modeling

Ping Tan[1] Tian Fang Jianxiong Xiao Peng Zhao Long Quan

[1]National University of Singapore Hong Kong University of Science and Technology

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

Figure 1: Single image tree modeling. (a) Single input image of a tree downloaded from www.flickr.com. (b) Strokes drawn by the user, only two strokes for this example. (c) The automatic synthesis of the tree branches. (d) The complete tree model rendered at the same viewpoint as the input image.

Abstract

In this paper, we introduce a simple sketching method to generate a realistic 3D tree model from a single image. The user draws at least two strokes in the tree image: the first crown stroke around the tree crown to mark up the leaf region, the second branch stroke from the tree root to mark up the main trunk, and possibly few other branch strokes for refinement. The method automatically generates a 3D tree model including branches and leaves. Branches are synthesized by a growth engine from a small library of elementary subtrees that are pre-defined or built on the fly from the recovered visible branches. The visible branches are automatically traced from the drawn branch strokes according to image statistics on the strokes. Leaves are generated from the region bounded by the first crown stroke to complete the tree. We demonstrate our method on a variety of examples.

1 Introduction

Trees are ubiquitous and hard to model in a realistic way because of the large varieties and their natural complexity in geometry. Progress has been made over the years in modeling trees. Many methods [Weber and Penn 1995; Prusinkiewicz et al. 2001; Reche-Martinez et al. 2004; Quan et al. 2006; Tan et al. 2007; Neubert et al. 2007] can achieve highly realistic tree models. Yet all these methods need significant amount of efforts to produce good results, either in the sense of tuning tree growing parameters or in the sense of image processing and 3D reconstruction.

To overcome the current difficulties, we propose a method that drastically simplifies the modeling process. We use a single image to model a tree. From this single image and few user drawn strokes, our system first generates the whole tree branching structure by non-parametric synthesis, then complete tree model with leaves. Our system is remarkably simple and generates visually convincing results. It makes the modeling of large scale vegetation affordable. It could be part of a cost-effective solution to build realistic-looking environments for movie post-production, architectural designs, games, and web applications.

1.1 Related work

The growth of trees adheres to strong botanic rules and patterns. Many previous methods are rule based. [Prusinkiewicz et al. 1994] introduced a series of methods based on the idea of the generative L-system. [Weber and Penn 1995] used geometric rules to produce realistic-looking trees. [de Reffye et al. 1988] designed rules according to botanical knowledge. These techniques generate good results, but they often require expertise for effective use. The idea behind rule based methods is that the branch and leaf arrangement follow a pattern which can be predicted with a set of rules and parameters. However, these rules and parameters are nontrivial to set.

In the past several years, image based methods become popular. These methods use images to recover 3D structure of the tree. [Reche-Martinez et al. 2004] recovered the opacity and color of each cell of a volume containing the tree. Although realistic results can be produced by this method, its volumetric representation makes editing and animation almost impossible. [Shlyakhter et al. 2001] reconstructed a visual hull according to image silhouette and fit a branch structure to the hull. [Neubert et al. 2007] used particle flow to extract branch structure from the recovered tree volume. Instead of relying on approximate geometry information such as visual hull, [Quan et al. 2006] and [Tan et al. 2007] used structure from motion algorithm to compute a set of 3D points over the tree. Then 3D points were used to generate triangle mesh model of branches and leaves. All these image based methods need multiple images as input. Typically, these images should span a large range of view angles to correctly recover 3D information. Furthermore, the tree should be first segmented out from background in all images. Although there are some well developed tools [Li et al. 2004; Rother et al. 2004], such a segmentation of multiple images is tedious and time consuming .

Instead of relying on rules or images, tree models can also be generated according to 3D scanner data or 2D user sketches. [Xu et al. 2007] produced very good tree models from a set of 3D points cap-

ACM Reference Format
Tan, P., Fang, T., Xiao, J., Zhao, P., Quan, L. 2008. Single Image Tree Modeling. *ACM Trans. Graph. 27*, 5, Article 108 (December 2008), 7 pages. DOI = 10.1145/1409060.1409061 http://doi.acm.org/10.1145/1409060.1409061.

tured by scanner. [Okabe et al. 2005] generated tree models from a branch skeleton sketched by user. However, this kind of sketching method requires many user interventions and it is often hard to sketch a realistic tree for an amateur user.

1.2 Our approach

Given a single image of a tree, we draw strokes on the image to create a tree model. We start to draw a crown stroke to mark up the leaf region in the image. Then a branch stroke is drawn from the tree root to mark the main trunk. Branches visible in the image are automatically traced out around drawn branch strokes to minimize the user intervention and achieve the highest realism. A few other branch strokes could be added to complete visible branches when necessary. The branch structure patterns encoded in visible branches are used to build up a small library of elementary subtrees to grow the entire tree. If too few visible branches exist in the image, the tree could also grow according to some predefined subtree patterns. Branches work as a hidden determinant structure of the tree. Once branches are ready, leaves can be generated easily according to the branch structure and image information. One example can be seen in Figure 1. The input single image is shown in (a), with two strokes drawn by the user as (b). Our method first grows a branch structure illustrated in (c) then complete the tree with leaves as in (d).

Our method can be regarded as a mixture of rule based method, image based method and sketch based method. Rather than applying parametric rules for branch generation, we use the local branch shapes to synthesize new branches. This is essentially a non-parametric tree growing system. Different from previous image-based methods, we design our system to work with a single input image. We do not intend to recover 3D structure directly from image. In contrast, we use the image as a guide for non-parametric tree growing, i.e. the growth should lead to a result close to the image. Different from a pure sketching [Okabe et al. 2005], we only draw a few strokes. The image statistics underlined by the strokes allows us to recover more tree structures.

2 Image Plane Sketching

Previous image-based methods [Tan et al. 2007; Reche-Martinez et al. 2004; Neubert et al. 2007] need a tree segmentation to make use of image information for modeling. However, a good tree segmentation is usually very difficult. Our method does not depend on high quality segmentation. In our system, the user draws a few strokes to mark out foliage and visible branches by taking advantage of the tree prior to facilitate segmentation.

User interface The user draw strokes on the image by moving the mouse cursor and holding a button. (Left button for the crown and right button for branches.) Similar UI is designed in [Li et al. 2004] for image segmentation. For simplicity, we always use one stroke to mark the crown. The foliage region is automatically extracted by the method described in the following paragraph according to this stroke. The user then draw strokes to mark out branches. After each branch stroke, a tracing algorithm is trigged to follow the visible branches close to the stroke. The traced visible branches are displayed over the image. If not satisfied with this result, the user has the option of adding or deleting strokes for correction. Unlike a pure sketching system [Okabe et al. 2005], we have the image information underlying the drawn strokes that allows extremely simple sketching.

Figure 1 shows an example in which we need only two strokes: the first crown stroke in red and the second branch stroke in blue.

Figure 2: The extracted foliage region via minimizing the Gibbs energy. These are the results for the cherry tree in Figure 1 and the oak tree in Figure 7. Note that our modeling needs only a coarse segmentation.

Foliage extraction Foliage is extracted from the closed region by the first crown stroke, which roughly follows the crown boundary. 'GrabCut' [Rother et al. 2004] extracts object inside a bounding rectangle by analyzing the different appearance inside and outside of the rectangle. The 'GrabCut' is less effective here as both inside and outside of the crown stroke could contain significant amount of leaf colors. For extraction, we simply compute a Gaussian mixture model (GMM) for the pixel RGB values in the region closed by the crown stroke. We employ a mixture of 10 Gaussians for large variation of colors due to the background. Then we take the four most green or red Gaussian components as leaf clusters. And the remaining six components are considered as background clusters. With these appearance models $G(I_x, \theta^{\mathcal{F}}), G(I_x, \theta^{\mathcal{B}})$ for the foreground and background, we compute a graph-cut based extraction to detect leaf pixels. Here, $G(\cdot, \theta)$ is the pdf function of GMM distribution, I_x indicates the RGB values at pixel x, $\theta^{\mathcal{F}}, \theta^{\mathcal{B}}$ are GMM parameters.

At each pixel x, we compute a $0-1$ label β_x via graph cut, where $\beta_x = 0$ represents leaf pixels and $\beta_x = 1$ represents background pixels. A Gibbs energy of the following form is defined over the enclosed region of crown stroke

$$\sum_x E_d(\beta_x, \theta^{\mathcal{F}}, \theta^{\mathcal{B}}) + \sum_{(x,y)\in\mathbf{N}} E_s(\beta_x, \beta_y),$$

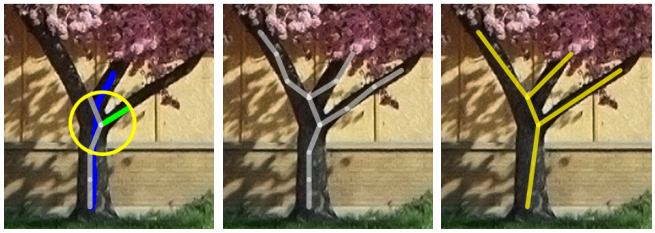
where $\mathbf{N}$ is the set of all 4-neighbor pixel pairs,
$E_d(\beta_x, \theta^{\mathcal{F}}, \theta^{\mathcal{B}}) = -\beta_x log G(I_x, \theta^{\mathcal{B}}) - (1-\beta_x) log G(I_x, \theta^{\mathcal{F}})$
is the data term, and

$$E_s(\beta_x, \beta_y) = \begin{cases} 0 & \beta_x = \beta_y \\ \lambda/|I_x - I_y| & \beta_x \neq \beta_y \end{cases}$$

is the smooth term. Graph-cut algorithm [Kolmogorov and Zabih 2002] is applied to minimize this Gibbs energy by assigning a 0 or 1 for each β_x. The constant λ indicating the strength of smoothness is set to 60 in our implementation. Before the extraction, we usually expand the enclosed region by morphology expansion 10 times to allow more freedom for the user's sketching.

Figure 2 shows the result of the foliage extraction from the input image and the stroke in Figure 1. It is should be noticed that we do not require a very accurate segmentation, which is an important advantage of our method.

Visible branch tracing To minimize the user intervention, the system automatically traces along branch strokes to detect nearby visible branches in the image. This tracing is trigged after each branch stroke is drawn. We apply a method inspired by the 'Lazy Snapping' [Li et al. 2004]. Pixels on the branch stroke are used as samples to compute an appearance model for the branch. All the other pixels are samples to compute the non-branch appearance. Again, a GMM model is used for the appearance model. Since the branch stroke could cover leaf pixels (e.g. in the first example in Figure 7), we discard the Gaussian component in the branch GMM if it is too close to some component in the foliage GMM $G(I_x, \theta^{\mathcal{F}})$. The branch appearance model is denoted as $G(I_x, \theta^{\mathcal{T}})$. The appearance model for non-branch pixels is $G(I_x, \theta^{\mathcal{N}})$. Typically, 5 Gaussian distributions are used for each model.



(a) (b) (c)

Figure 3: (a) We move a circle along the branch stroke to trace visible branches. The fork is detected and a new green branch is created automatically. (b) The branch is recovered by connecting all circle centers during tracing. (c) The initial branch is simplified to discard redundant joints.

With these two appearance models, we move a circle along the branch stroke from bottom to top. At each position, pixels on the circle are classified as branch pixel or non-branch pixel by a maximum likelihood estimation. We use a $0-1$ variable α to facilitate this classification, where 0 means branch pixel and 1 for others. For each pixel x on the circle, we compute α_x by maximizing the following likelihood

$$(1-\alpha_x)G(I_x,\theta^{\mathcal{T}})+\alpha_x G(I_x,\theta^{\mathcal{N}}).$$

Typically, multiple branch pixels will be detected on the circle. And these pixels form clusters. We discard a cluster if there are non-branch pixels along the line segment connecting the cluster center and circle center (via the maximum likelihood estimation). The circle center will move to the remaining cluster center to continue tracing. In the case of multiple clusters left on the circle, they are processed in a breadth first manner. The branch skeleton is detected by connecting all these circle centers during tracing. This skeleton is overlayed in the image. If the user is not satisfied with this result, he can add more strokes to correct it or delete wrong branches. This skeleton is then simplified by discarding redundant joints, which is not a fork and the branch direction does not change drastically ($< 30^o$) at that joint. In our implementation, the circle radius is fixed as 50 pixels for all examples (image resolution at about 1500 pixels).

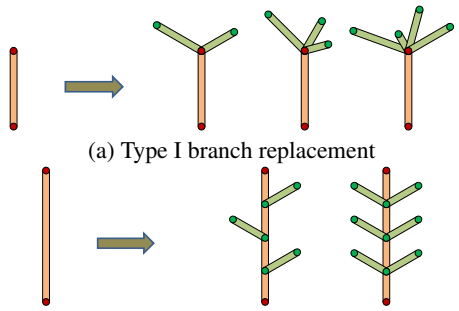
At the tree root, the branch thickness is also computed by varying the circle radius to find a largest circle whose pixels are all branch pixels. This thickness computation is unreliable at small twigs. We simply set branch radius to 75% of its parent, although better botanical rules can be used according to [Weber and Penn 1995].

As shown in Figure 3 (a), the branch segment indicated by the green line is correctly detected, although the drawn stroke does not pass through it. A branch system is retrieved by connecting circle centers in sequence as shown in (b). This initial result contains many fragment line segments, which is undesirable for the non-parametric synthesis in Section 3.1. The final branch after discarding some redundant joints is shown in (c).

3 Tree Growing

Branching structure is the determinant hidden structure of trees. Once branches are recovered, leaves can be generated along branches to complete the tree. Many previous tree modeling methods [Xu et al. 2007; Tan et al. 2007; Neubert et al. 2007] model trees this way by focusing on the modeling of branch system.

Once visible branches and foliage region are extracted from the image, we develop a tree grow engine to automatically generate the whole tree branch in 3D space by following the given image. We only seek a plausible solution that is possible with the tree priors and the inherent self-similar structural patterns of the tree. Similar engine is used in [Xu et al. 2007; Tan et al. 2007], where 3D points are used to guide the growth.



(a) Type I branch replacement

(b) Type II branch replacement

Figure 4: A branch is replaced by a library subtree. We call the red branch in the subtree as 'supporting branch' for easy reference.

3.1 Growth engine

The engine starts with the creation of a library of elementary subtrees from the visible branches. Then a non-parametric synthesis approach is used to systematically generate invisible branches to complete the branching structure.

Initialization

- **Conversion of 2D branches into 3D:** Visible branches interactively traced in Section 2 are defined in the image plane. We first convert these branches from 2D to 3D before growing. From a single image, there is no enough information to accurately reconstruct the branch position in 3D space. Here, we employ the approach proposed in [Okabe et al. 2005] to generate a 3D branch structure from the extracted visible branches. The basic idea is to greedily adjust branches' orientation so that distance among them are as large as possible. We assume an orthographic camera model to relate 3D branch position and image coordinate.
- **Creation of the library:** We then built a library of elementary subtrees. These library subtrees are built from the recovered visible branches by taking all its subtrees. If there is too few subtrees (as the first example in Figure 7), we add predefined subtrees in the library. Figure 4 shows the predefined subtrees in our implemented system. Obviously, this predefined library can be further enriched to handle larger varieties of trees. It is remarkable that we produced all of our results with only at most 8 subtrees in the current implementation.

Non-parametric synthesis Starting from the 3D visible branch and a library, we take a non-parametric approach to grow tree. The synthesis process simply iteratively replaces an existing branch by a library subtree. Figure 4 shows a single step of the non-parametric branch growth. There are two types of branch replacement in our system. In type I replacement, new branches grow at the end of its 'supporting branch' (i.e. shown as the red segment in a subtree). In type II replacement, new branches can grow along the 'supporting branch'.

The selection of the branch to be replaced and the library subtree is driven by minimizing the cost function defined in Section 3.2. Each time, the resulting synthesis is pruned by the extracted foliage silhouette. We empirically run the following three steps iteratively about 100 times for each tree.

- **Selection of a branch to be replaced:** We go through a small set of existing branches and take the one whose replacement gives the lowest cost function defined in Section 3.2.

To create this set of existing branches, we choose branches with larger radius and older generation. We sort all existing branches according to their radius. Then branches are selected sequentially from the sorted array. Each branch is selected with a probability, which is inversely proportional to its generation.

- **Selection of a replacing library subtree:** For the selected branch to be replaced, we search all the available library subtrees (at most 8) to find that one giving the lowest cost of the function defined in Section 3.2.

  Except for the subtrees generated from visible branches, the user can add predefined subtrees: type I, type II or both. A subtree is rotated around its 'supporting branch' and scaled before it is used to replace some existing branch. There are two parameters to be determined in this operation.

  - **The rotation angle** of the subtree around its 'supporting branch' is searched among 12 quantized levels of 360 degrees that gives the lowest cost.
  - **The scaling factor** of the subtree is determined such that its 'supporting branch' is of the same length as the replaced branch.

- **Branch pruning:** After replacement, the resulting branches are pruned based on the detected foliage region and the existing branches. Any branch going beyond the foliage region is removed, so are the new branches if they are too close to some existing branches.

3.2 Data-driven attractors

The growth engine is driven by the data to produce realistic result. The input image information is 2D and only weakly controls the growth in the desired tree volume. Hence, we introduce some 3D points based on heuristics to control the growth better.

Image attractors To make the result after growing similar to image, We define a set of image attractors $s_i, i = \{1, 2, \cdots N\}$ that are sampled evenly in the foliage region with a fixed interval as illustrated by the yellow points in Figure 5 (a). These attractors control the tree growth by requiring that each attractor should be close to the resulting tree T. The growth is then driven by minimizing a cost defined as $E^{2D}(T) = \sum_i dist(s_i, T)$, where $dist(s_i, T)$ is the distance of an attractor s_i to the projection of tree T onto the image. To compute $dist(s_i, T)$, we also sample a set of points $p_i, i = \{1, 2, \cdots M\}$ along the image projection of the tree T, as the blue points in Figure 5 (a). The distance function is then defined as $dist(s_i, T) = \min_j(dist(s_i, p_j))$.

Extrapolated 3D attractors The image driven growth could lead to an unbalanced tree, where only front branches are generated. This is because such an unbalanced tree can minimize the cost $E^{2D}(T)$ well without any back branches. Similar problem exists in previous sketching systems like [Okabe et al. 2005]. To alleviate this problem the tree is rotated 90 degrees around its main trunk and merged with the original one in [Okabe et al. 2005]. This method solves the problem at the cost of creating inconsistent visible branches with the image.

We introduce some extrapolated 3D points to balance the tree growth. These 3D points are generated by two steps. First, take the set of branch joints of the current tree. Then, rotate these joints 90 degree around the main trunk. Here the distance between these 3D points and the tree is to be minimized and the cost is defined as $E^{3D}(T) = \sum_i dist(d_i, T)$. $dist(d_i, T)$ is the distance between the 3D point d_i and the tree T. Again the tree T is sampled as a set of points $t_i, i = \{1, 2, \cdots, L\}$ in 3D space along the branch skeleton to compute the $dist(d_i, T) = \min_j(dist(d_i, t_j))$.

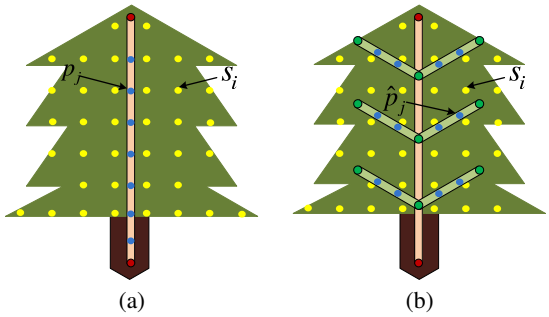


Figure 5: Yellow points in the foliage region are the image attractors. (a) Blue points are sampled over the tree to compute the distance between an attractor and the tree. (b) After branch replacement, the distance between attractors and the tree is updated according to newly created branches.

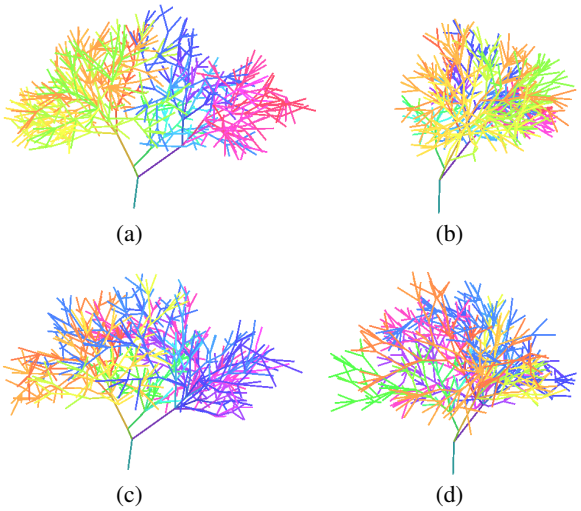


Figure 6: (a) and (b) are the branch generated by image driven growth viewed from the front and side. Although the image driven growth can guarantee the result similar to the image in front, the result looks unnatural from the side. (c) and (d) are results computed by alternating image driven growth and 3D point driven growth. Both front and side results look good.

In our current implementation, the growth is driven by alternating the image attractors and the 3D point attractors. Figure 6 illustrates the effectiveness of this alternating strategy. The pure image driven growth yields good results when viewed from the same direction as the input image as in (a), while unnatural results (b) viewed from an orthogonal viewpoint. By alternating the image driven growth and 3D point driven growth, a better result can be obtained in (c) and (d).

Speedup The growth engine involves a large amount of computation of the distances between the attractors and the tree. In each replacement iteration, we only add more branches to the tree and no existing branch is discarded. So the distance computation in previous iteration can be reused for speedup purpose. For each attractor s (no matter image point or 3D point), we record its distance to the tree T^n at the $n-$th iteration as d_s^n. At the $n+1-$th iteration, we compute the distance between s and newly created branches as $\hat{d}_s$. The distance between s and T^{n+1} is then updated as $d_s^{n+1} = min\{d_s^n, \hat{d}_s\}$. This is illustrated in Figure 5 (b).

Figure 9: Modeling woods with multiple trees. On the top is the input image. The rendering of the recovered model is at the bottom.

4 Completing the Tree

The leaves of the tree are automatically synthesized from the recovered branch structure and textured with the input image. Each leaf is represented by a flat rectangle with the size of 1/10 of the main trunk radius. Each branch generates from a range of 50 to 200 leaves proportional to it length. The arrangement of the leaves around the branch is randomized. We keep only those leaves that are projected inside the foliage region in the input image. Leaves are textured according to their projected position on the input image. The generic leaf shape, leaf size, density and arrangement of leaves along a branch are all parameterized in our current implementation. But the default values are used throughout all examples of this paper.

5 Results

We test our algorithm with several different examples to demonstrate its effectiveness. A typical example for a cherry tree downloaded from www.flickr.com is shown in Figure 1. Its foliage region is shown in Figure 2 and its branch tracing procedure is illustrated in Figure 3. The complete branching structure generated by growth engine is shown in Figure 1 (c). The complete cherry tree model is then rendered as in Figure 1 (d). For this example, we draw two strokes, and used both subtrees of visible branches and predefined subtrees of type I. The branch tracing is realtime, which provides rapid feedback to the user to decide whether additional branch strokes are needed. However, the branch growing typically takes about 20 minutes on a PC with 2.4G CPU. The population of the tree with leaves takes another 10 minutes.

More examples are shown in Figure 7. For the sycamore tree in the first row, it takes only two strokes. The tracing does not add any more branches. Its branching structure is entirely synthesized from the library of predefined subtrees of type II. For the oak tree in the second row, it takes three strokes. Part of its visible branches are traced automatically, but one is missing due to the dense foliage and is added by a second branch stroke. For the second cherry tree, it takes 3 strokes for the branches as the tracing is more challenging. The oak tree and cherry tree are both downloaded from www.flickr.com.

The simplicity of our method makes the modeling affordable for woods as well. By casually capturing the image shown in Figure 9, we are able to get all four trees models from a total of 16 strokes in the input image. The relative positions of trees in 3D is set manually. A more systematic usage of the method in urban environment is shown in Figure 8. Trees along a street can only be captured from close viewpoints due to space constraint, which makes previous image-based tree modeling methods less applicable. Here, we model each tree from a single image then align all trees manually in 3D space. In Figure 8, the trees are modeled with 2, 2, 5, 3, 3, 2, 2 strokes from left to right. All trees are grown with predefined subtrees of type I.

6 Conclusion & Future Work

We have described a simple and effective system for constructing realistic tree models from a single image. Our system was designed to minimize user interaction. The resulting system is simple and practical in that an amateur user only needs to sketch a few strokes in the single input image.

There are some directions for future works. For example, we could fill the gap between this single image based method and previous multi-views methods to improve the 3D shape while keeping the modeling simplicity. We could also improve the system in runtime efficiency to realtime. Another interesting direction for future research is to make the modeling procedure fully automatic. The described system could be upgraded to full-automatic if automatic tree detection and visible branch tracing algorithm are available.

7 Acknowledgement

We thank Marcus Lundberg, Sean Pecor and Jason Ramsay, for their permission to use their pictures in this paper. Ping Tan is supported by Singapore FRC Grant R-263-000-477-112. This work is also supported by Hong Kong RGC Grants 618908, 619107, 619006 and RGC/NSFC N-HKUST602/05.

References

DE REFFYE, P., EDELIN, C., FRANÇON, J., JAEGER, M., AND PUECH, C. 1988. Plant models faithful to botanical structure and developmentr. In *SIGGRAPH 1988*.

KOLMOGOROV, V., AND ZABIH, R. 2002. What energy functions can be minimized via graph cuts? In *Proc. of European Conf. on Computer Vision*.

LI, Y., SUN, J., TANG, C.-K., AND SHUM, H.-Y. 2004. Lazy snapping. *SIGGRAPH 2004*.

NEUBERT, B., FRANKEN, T., AND DEUSSEN, O. 2007. Approximate image-based tree-modeling using particle flows. *SIGGRAPH 2007*.

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

PRUSINKIEWICZ, P., JAMES, M., AND MĚCH, R. 1994. Synthetic topiary. In *SIGGRAPH 1994*, 351–358.

PRUSINKIEWICZ, P., MÜNDERMANN, L., KARWOWSKI, R., AND LANE, B. 2001. The use of positional information in the modeling of plants. In *SIGGRAPH 2001*.

QUAN, L., TAN, P., ZENG, G., YUAN, L., WANG, J., AND KANG, S. B. 2006. Image-based plant modeling. *SIGGRAPH 2006*, 599–604.

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

Figure 7: Examples are sorted in the increasing number of strokes: (a) The single input image. (b) The synthesized branch structure. Subtrees are shown at the corner. (c) Complete tree model with leaves rendered at the same viewpoint as the input image. (d) Rendered from a novel view point.

Figure 8: Street-side tree modeling: several input images captured along a street in the top row, and the tree models rendered at the same viewpoint on the bottom row.

RECHE-MARTINEZ, A., MARTIN, I., AND DRETTAKIS, G. 2004. Volumetric reconstruction and interactive rendering of trees from photographs. *SIGGRAPH 2004*.

ROTHER, C., KOLMOGOROV, V., AND BLAKE, A. 2004. "grabcut": interactive foreground extraction using iterated graph cuts. In *SIGGRAPH 2004*, 309–314.

SHLYAKHTER, I., ROZENOER, M., DORSEY, J., AND TELLER, S. 2001. Reconstructing 3d tree models from instrumented photographs. *IEEE Comput. Graph. Appl. 21*, 3, 53–61.

TAN, P., ZENG, G., WANG, J., KANG, S. B., AND QUAN, L. 2007. Image-based tree modeling. In *SIGGRAPH 2007*, 87.

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

XU, H., GOSSETT, N., AND CHEN, B. 2007. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Trans. Graph. 26*, 4, 19.