Image-based Plant Modeling

Long Quan Ping Tan Gang Zeng Lu Yuan Jingdong Wang Sing Bing Kang*

The Hong Kong University of Science and Technology *Microsoft Research

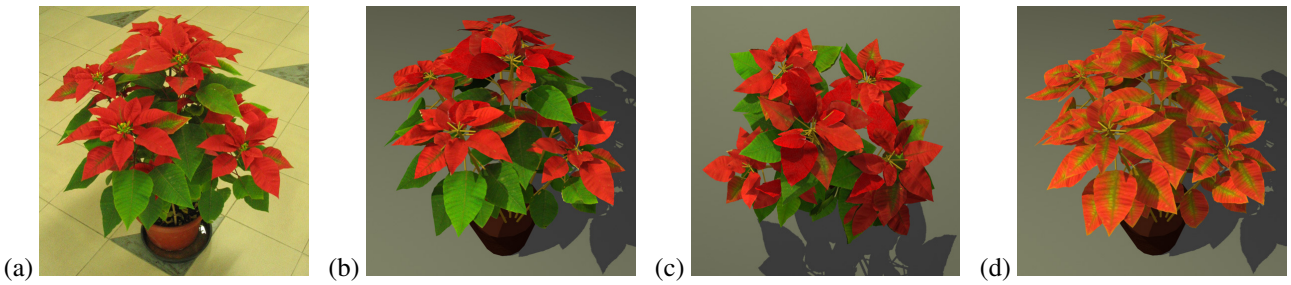

Figure 1: Image-based modeling of poinsettia plant. (a) An input image out of 35 images, (b) recovered model rendered at the same viewpoint as (a), (c) recovered model rendered at a different viewpoint, (d) recovered model with modified leaf textures.

Abstract

In this paper, we propose a semi-automatic technique for modeling plants directly from images. Our image-based approach has the distinct advantage that the resulting model inherits the realistic shape and complexity of a real plant. We designed our modeling system to be interactive, automating the process of shape recovery while relying on the user to provide simple hints on segmentation. Segmentation is performed in both image and 3D spaces, allowing the user to easily visualize its effect immediately. Using the segmented image and 3D data, the geometry of each leaf is then automatically recovered from the multiple views by fitting a deformable leaf model. Our system also allows the user to easily reconstruct branches in a similar manner. We show realistic reconstructions of a variety of plants, and demonstrate examples of plant editing.

CR Categories: I.3.5 [Computer Graphics]: Computational geometry and object modeling—Modeling packages; I.4.5 [Image Processing and computer vision]: Reconstruction.

Keywords: Tree modeling, plant modeling, image-based modeling, photography.

1 Introduction

Plants remain one of most difficult kinds of object to model due to their complex geometry and wide variation in appearance. While techniques have been proposed to synthetically generate realistic-looking plants, they either require expertise to use (e.g., [Prusinkiewicz et al. 1994]) or they are highly manual intensive. Current image-based techniques that use images of real plants have either produced models that are not easily manipulated (e.g., [Reche-Martinez et al. 2004]) or models that are just approximations (e.g., [Shlyakhter et al. 2001]).

Our approach is image-based as well, but we explicitly extract geometry and we strictly enforce geometric compatibility across the input images. Image acquisition is simple: The camera need not be calibrated, and the images can be freely taken around the plant of interest. Our modeling system is designed to take advantage of the robust structure from motion algorithm developed in computer vision community. It is also designed to allow the user to quickly recover the remaining details in the form of individual leaves and branches. Furthermore, it does not require any expertise in botany to use. We show how plants with complicated geometry can be constructed with relative ease. One of the motivations for developing an image-based approach to plant modeling is that the geometry computation from images tend to work remarkably well for textured objects [Hartley and Zisserman 2000], and the plants are often well textured.

Once the realistic geometry of a plant has been extracted, it can be used in a number of ways, for example, as part of an architectural design, in games, or even for the scientific study of plant growth. Furthermore, since geometry is available, it can be easily manipulated or edited. We show examples of plant reconstruction and editing in this paper.

2 Prior work

Many approaches have been proposed to model plants and trees, and they can be roughly classified as either rule-based or image-based.

Rule-based methods. Rule-based methods use compact rules or grammar for creating models of plants and trees. As a prime example, Prusinkiewicz et al. [1994] developed a series of approaches based on the idea of the generative L-system. Weber and Penn [1995] use a series of geometric rules to produce realistic-looking trees. De Reffye et al. [1988] also use a collection of rules, but the rules are motivated by models of plant growth. There are also a number of techniques that take into account various kinds of tree interaction with the environment (e.g., [Mech and Prusinkiewicz 1996; Prusinkiewicz et al. 2001; Van Haevre and Bekaert 2003; Noser et al. 2001; Noser and Thalmann 1994]).

While these methods are capable of synthesizing impressive-looking plants, trees, and forests, they are based on rules and parameters that are typically difficult to use for a non-expert. Plants have specific shapes that are formed naturally (inherent growth pattern),

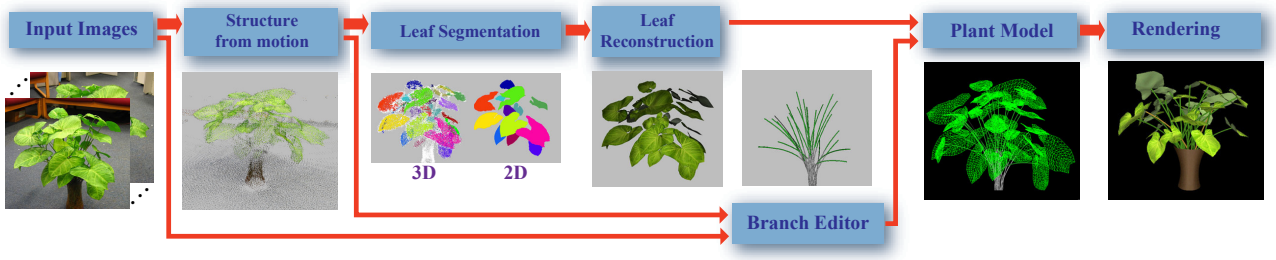

Figure 2: The overview of our image-based plant modeling approach.

caused by external biological factors (such as disease), a result of human activity (such as localized pruning), or shaped by other external factors (such as fire, flood, or nearby structures). Generating a model that very closely resembles an actual plant under a variety of real-world conditions would not be easy using such type of approach.

Image-based methods. Image-based methods directly model the plant using image samples (our proposed technique is one such method). Han et al. [2003] described a Bayesian approach to modeling tree-like objects from a single image using good priors. The prior models are ad hoc, and the type of recovered 3D model is rather limited. The approaches described in [Sakaguchi 1998; Shlyakhter et al. 2001] mainly use the visual hull of the tree computed from silhouettes to represent a rough shape of the tree. The tree volume is then used to create the tree branching structure for synthesizing leaves. Shlyakhter et al.'s system [2001] starts with an approximation of medial axis of the estimated tree volume, and ends with a simple L-system fit. Sakaguchi et al. [1998; 1999] use simple branching rules in voxel space instead of L-system for building the branching structure. All these methods generate only approximate shapes with limited realism.

More recently, Reche et al. [2004] proposed a technique for computing a volumetric representation of the tree with opacity. While their results look impressive, their approach does not recover explicit geometries of the branches and leaves. As a result, their technique is limited to visualization only, with no direct means for animation or editing.

3 Overview of Plant Modeling System

Our plant modeling system uses only images for modeling. There are three parts in our system: image acquisition and structure from motion, leaf segmentation and recovery, and interactive branch recovery. The system is summarized in Figure 2.

We use a hand-held camera to capture images of the plant at different views. We then apply a standard structure from motion technique to recover the camera parameters and a 3D point cloud.

Next, we segment the 3D data points and 2D images into individual leaves. To facilitate this process, we designed a simple interface that allows the user to specify the segmentation jointly using 3D data points and 2D images. The data to be partitioned is implemented as a 3D undirected weighted graph that gets updated on-the-fly. For a given plant to model, the user first segments out a leaf; this is used as a deformable generic model. This generic leaf model is subsequently used to fit the other segmented data to model all the other visible leaves. Our system is also designed to use the images as guides for interactive reconstruction of the branches.

The resulting model of the plant very closely resembles the appearance and complexity of the real plant. Just as important, because the output is a geometric model, it can be easily manipulated or edited.

4 Preliminary Processes

Image acquisition is simple: the user just uses a hand-held camera to capture the appearance of the plant of interest from a number of different overlapping views. The main caveat during the capture process is that appearance changes due to changes in lighting and shadows should be avoided. For all the experiments reported in this paper, we used between 30 to 45 input images taken around each plant.

Prior to any user-assisted geometry reconstruction, we extract point correspondences and ran structure from motion on them to recover camera parameters and a collection of 3D points. Standard computer vision techniques have been developed to estimate the point correspondences across the images and the camera parameters [Hartley and Zisserman 2000; Faugeras et al. 2001]. We used the approach described in [Lhuillier and Quan 2005] to compute a semi-dense cloud of reliable 3D points in space. This technique is used because it has been shown to be robust and capable of providing sufficiently dense point clouds for depicting objects. This technique is well-suited because plants tend to be highly textured.

The semi-dense feature points used in [Lhuillier and Quan 2005] are not the points of interest [Hartley and Zisserman 2000], but regularly re-sampled image points from a kind of disparity maps. One example is shown in Figure 2. We are typically able to obtain about a hundred thousand 3D points that unsurprisingly tend to cluster at textured areas. These points help by delineating the shape of the plant. Each 3D point is associated with images where it was observed; this book-keeping is useful in segmenting leaves and branches during the modeling cycle.

5 Graph-based Leaf Extraction

We next proceed to recover the geometry of the individual leaves. This is clearly a difficult problem, due to the similarity of color between different overlapping leaves. To minimize the amount of user interaction, we formulate the leaf segmentation problem as interactive graph-based optimization aided by 3D and 2D information. The graph-based technique simultaneously partitions 3D points and image pixels into discrete sets, with each set representing a leaf. We bring the user into the loop to make the segmentation process more efficient and robust.

One unique feature of our system is the joint use of 2D and 3D data to allow simple user assist, as all our images and 3D data are perfectly registered. The process is not manual intensive as we only need to have one image segmentation for each leaf. This is because the leaf reconstruction algorithm (see Section 6) needs only one image segmentation boundary per leaf. This is sub-optimal but substantially more efficient.

There are two main steps to the segmentation process: automatic segmentation by a graph partition algorithm, followed by user interaction to refine the segmentation. Our system responds to user input by immediately updating the graph and image boundaries.

5.1 Graph partition

The weighted graph $G = \{V, E\}$ is built by taking each 3D point as a node and connecting it to its K-nearest neighboring points (K=3) with edges. The K-nearest neighor is computed using 3D Euclidean distance, and each connecting edge should at least be visible at one view. The weight on each edge reflects the likelihood that the two points being connected belong to the same leaf. We define a combined distance function for a pair of points (nodes) p and q as

$$d(p,q) = (1-\alpha)\frac{d_{3D}(p,q)}{\sqrt{2}\sigma_{3D}} + \alpha\frac{d_{2D}(p,q)}{\sqrt{2}\sigma_{2D}},$$

where α is a weighting scalar set to 0.5 by default. The 3D distance $d_{3D}(p,q)$ is the 3D Euclidean distance, with σ_{3D} being its variance. The 2D distance measurement, computed over all observed views, is

$$d_{2D}(p,q) = \max_i\{\max_{u_i\in[p_i,q_i]} g_i(u_i)\}.$$

The interval $[p_i, q_i]$ specifies the line segment joining the projections of p and q on the ith image. u_i is an image point on this line segment. The function $g(\cdot)$ is the color gradient along the line segment, it is approximated using color difference between adjacent pixels. σ_{2D} is the variance of the color gradient. The weight of a graph edge is defined as $w(p,q) = e^{-d^2(p,q)}$.

The initial graph partition is obtained by thresholding the weight function $w(\cdot)$ with k set to 3 by default,

$$w(p,q) = \begin{cases} e^{-d^2(p,q)}, & \text{if } d_{3D} < k\sigma_{3D} \text{ and } d_{2D} < k\sigma_{2D}, \\ 0, & \text{otherwise.} \end{cases}$$

This produces groups that are clearly different. However, at this point, the partitioning is typically coarse, requiring further subdivision for each group. We use the normalized cut approach described in [Shi and Malik 2000]. The normalized cut computation is efficient in our case as the initial graph partition significantly reduces the size of each subgraph, and the weight matrix is sparse as it has at most $(2K+1)N$ non-zero elements for a graph of N nodes for K-nearest neighbors. This approach is effective due to the joint use of 2D and 3D information. Figure 3 shows that if only 3D distance is used ($\alpha = 0$), a collection of leaves are segmented but not individual ones. When images are used ($\alpha = 0.5$) as well, each collection is further partitioned into individual leaves using edge point information.

5.2 User interface

In general, the process of segmentation is subjective; the segments that represent different objects depend on the person's perception of what an object is. In the case for partitioning the image into leaves, while the interpretation is much clearer, the problem is nonetheless still very difficult. This is because leaves in the same plant look very similar, and boundaries between overlapping leaves are often very subtle.

We designed simple ways the user can help refine areas where automatic segmentation fails. In the interface, each current 3D group is projected into the image as a feedback mechanism. The user can modify the segmentation and obtain the image boundary segmentation in any of the following ways:

Click to confirm segmentation. The user can click on an image region to indicate that the current segmentation group is acceptable. This operation triggers 2D boundary segmentation, described in Section 5.4.

Draw to split and refine. The user can draw a rough boundary to split the group and refine the boundary. This operation triggers two actions: First, it cuts off the connecting edges crossing the marked boundary, so that points inside the boundary are relabelled, which in turn causes an update of the graph partition by splitting the group into two subgroups. The graph updating method is described in Section 5.3. Second, it triggers 2D boundary segmentation.

Click to merge. The user can click on two points to create an connecting edge to merge the two subgroups.

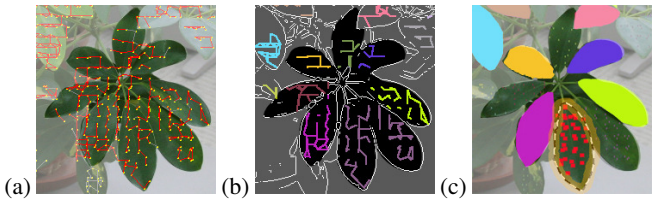

Figure 3: Benefit of jointly using 3D and 2D information. (a) The projection of visible 3D points (in yellow) and connecting edges (in red) are superimposed on an input image. Using only 3D distance resulted in coarse segmentation of the leaves. (b) The projection of segmented 3D points with only the connecting edges superimposed on the gradient image (in white). A different color is used to indicate a different group of connecting edges. Using both 3D and 2D image gradient information resulted in segmentation of leaflets. (c) Automatically generated leaflets are shown as solid-colored regions. The user drew the rough boundary region (thick orange line) to assist segmentation, which relabels the red points and triggers a graph update. The leaflet boundary is then automatically extracted (dashed curve).

5.3 Graph update

The graph update for affected groups is formulated as a two-label graph-cut problem [Boykov et al. 2001] that minimizes the following energy function:

$$E(l) = \sum_l (1-\delta(l_p,l_q))\frac{1}{d^2(p,q)+\epsilon} + \sum_l D(l_p),$$

where $\delta(l_p, l_q)$ is 1 if $l_p = l_q$, 0 if $l_p \neq l_q$, and $l_p, l_q = \{0, 1\}$. ϵ is a very small positive constant set to 0.0001. The data term $D(\cdot)$ encodes the user-confirmed labels:

$$\begin{cases} D(0) = 0, \\ D(1) = \infty, \end{cases} \text{ if } l_p = 0 \text{ and } \begin{cases} D(0) = \infty, \\ D(1) = 0, \end{cases} \text{ if } l_p = 1.$$

It is implemented as a min-cut algorithm that produces a global minimum [Boykov et al. 2001]. The complexity of the min-cut is $O(N^3)$, with N nodes and at most $5N$ edges for our graph. Since each group is usually rather small (a few thousand nodes), the update is immediate, allowing the interface to provide real-time visual feedback.

5.4 Boundary segmentation

The image segmentation for a given group of 3D points in a given image is also solved as a two-way graph-cut problem, this time using a 2D graph (not the graph for our 3D points) built with pixels as nodes. Our segmentation algorithm is similar to that of [Li et al. 2004]. However, for our algorithm, the foreground and background are automatically computed, as opposed to being supplied by the user in [Li et al. 2004].

The foreground is defined as the entire region covered by the projected 3D points in a group. The background consists of the projections of all other points not in the group currently being considered. As was done in [Li et al. 2004], we oversegment each image using

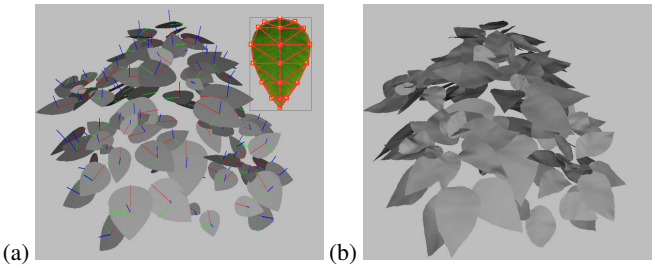

Figure 4: Leaf reconstruction for poinsettia (Figure 1). (a) Reconstructed flat leaves using 3D points. The generic leaf model is shown at top right. (b) Leaves after deforming using image boundary and closest 3D points.

the watershed algorithm in order to reduce the complexity of processing. Any reference to the image is actually a pointer to a color segment rather than to a pixel.

6 Model-based Leaf Reconstruction

Since leaves in the same plant are typically very similar, we adopt the strategy of extracting a generic leaf model from a sample leaf and using it to fit all the other leaves. This strategy turns out to be more robust as it reduces uncertainty due to noise and occlusion by constraining the shapes of leaves.

6.1 Extraction of a generic leaf model

To extract a generic leaf model, the user manually chooses an example leaf from its most fronto-parallel view, as shown in Figure 4. The texture and boundary associated with the leaf are taken to be the flat model of the leaf. The leaf model consists of three polylines: two for the leaf boundary and one for the central vein. Each polyline is represented by about 10 vertices. The leaf model is expressed in a local Euclidean coordinate frame with the $x-$axis being the major axis. The region inside the boundary is triangulated; the model is automatically subdivided to increase the accuracy of the model, depending on the density of points in the group.

6.2 Leaf reconstruction

Leaf reconstruction consists of four steps: generic flat leaf fit, 3D boundary warping, shape deformation, followed by texture assignment.

Flat leaf fit. We start by fitting the generic *flat* leaf model to the group of 3D points. This is done by computing the principal components of the data points of the group via SVD decomposition. A flat leaf is reconstructed at the local coordinate frame determined by the first two components of the 3D points. Then, the flat leaf is scaled in two directions by mapping it to the model. The recovered flat leaves are shown in Figure 4(a).

There is, however, an orientation ambiguity for the flat leaf. Note that this orientation ambiguity does not affect the whole geometry reconstruction procedure described in this section, so that the ambiguity is not critical at this point. A leaf is usually facing up and away from a branch. We use this heuristic to find the orientation of the leaf, i.e., facing up and away from the vertical axis going through the centroid of the whole plant. For a complicated branching structure, the leaf orientations may be incorrect. Once the branches have been reconstructed (Section 7), the system will automatically recompute the leaf orientation using information from the branching structure.

Leaf boundary warping. While the 3D points of a leaf are adequate for locating its pose and approximate shape, they do not completely specify the leaf boundary. This is where boundary information obtained from the images is used. Each group of 3D points are associated with 2D image segmentations at different views (if such segmentation exists). A leaf boundary will not be refined if there is no corresponding image segmentation for the leaf. On the other hand, if multiple regions (across multiple views) exist for a leaf, the largest region is used.

Once the 3D leaf plane has been recovered, we back-project the image boundary onto the leaf plane. We look for the best affine transformation that maps the flat leaf to the image boundary by minimizing the distances between the two sets of points of the boundaries on the leaf plane in space. We adapted the ICP (Iterative Closest-Point) algorithm [Besl and McKay 1992] to compute the registration of two curve boundaries. We first compute a global affine transformation using the first two components of the SVD decomposition as the initial transformation. Correspondences are established by assigning each leaf boundary point to the closest image boundary; these correspondences are used to re-estimate the affine transformation. The procedure stops when the mean distance between the two sets of points falls below a threshold.

Shape deformation. The final shape of the leaf is obtained by locally deforming the flat leaf in directions perpendicular to the plane to fit the nearest 3D points. This adds shape detail to the leaf (Figure 4(b)).

Texture reconstruction. The texture of each leaf first inherits that of the generic model. The texture from the image segmentation is subsequently used to overwrite the default texture. This is done to ensure that occluded or slightly misaligned parts are textured.

7 Branch Extraction and Reconstruction

Once the leaves have been reconstructed, the next step would be to reconstruct the branches to complete the plant model. Unfortunately, the branching structure is difficult to reconstruct automatically from images due to occlusion and segmenting difficulties similar to those for leaves. One possible approach would be to use rules or grammar (L-systems). However, it is not clear how these can be used to fit partial information closely or how the novice user can exercise *local* control to edit the 3D branches (say, in occluded areas).

Our solution is to design a data-driven editor that allows the user to easily recover the branch structure. We model each branch as a generalized cylinder, with the skeleton being a 3D spline curve. The cylindrical radius can be spatially varying. It is specified at each endpoint of each branch, and linearly interpolated between the endpoints. While the simple scheme may not follow specific botanical models, it is substantially more flexible and easier to handle—and it can be used to closely model the observed plant.

The user is presented with an interface with two areas: an area showing the current synthesized tree (optionally with 3D points and/or leaves as overlay), and the other showing the synthetic tree superimposed on an input image. The image can be switched to any other input image at any time. User operations can be performed in any one area, with changes propagated to the other area in real-time as feedback. There are four basic operations the user can perform:

Draw curve. The user can draw a curve from any point of an existing branch to create the next level branch. If the curve is drawn in 3D, the closest existing 3D points are used to trace a branch. If drawn in 2D, at each point in the curve, its 3D coordinate is assigned to be the 3D point whose projection is the closest. If the closest points having 3D information are too far away, the 3D position is inherited from its parent branch.

Move curve. The user can move a curve by clicking on any point of the current branch to a new position.

Edit radius. The radius is indicated as a circle (in 2D) or a sphere (in 3D). The user can enlarge or shrink the circle or sphere directly

Figure 5: Branch structure editing. The editable areas are shown: 2D area (left), 3D space (right). The user modifies the radii of the circles or spheres (shown in red) to change the thicknesses of branches.

on the interface, effectively increasing or reducing the radius, respectively.

Specify leaf. Each branch can be specified whether it can grow a leaf at its endpoint, as illustrated as green branches in Figure 5. A synthesized leaf will be the average leaf over all reconstructed leaves, scaled by the thickness of the branch.

Once the branching structure is finalized, each leaf is automatically connected to the nearest branch. The orientation of each leaf, initially determined using a heuristic as described in Section 5, is also automatically refined at this stage. The plant model is produced by assembling all the reconstructed branches and leaves.

8 Results

Figure 6: An indoor tree. Left: an input image (out of 45). Right: the recovered model, inserted into a simple synthetic scene.

We have reconstructed a variety of plants with different shapes and densities of foliage. In this section, we show results for four different plants: nephthytis, poinsetta, schefflera, and an indoor tree. Because of variation in leaf shape, size, and density, the level of difficulty for each example is different. We typically capture about 35 images, more for plants with smaller leaves (specifically the indoor tree). The captured image resolution is 1944×2592 (except for the poinsettia, which is 1200×1600). For the efficiency of structure from motion, we down-sampled the images to 583×777 (for the poinsettia, to 600×800). It took approximately 10 mins for about 40 images on a 1.9GHz P4 PC with 1 GB of RAM. On average, we reconstructed about 30,000 3D points for the foreground plant (see the accompanied video).

The recovered texture-mapped geometry models are rendered using Maya to produce images shown in this paper. The statistics associated with the reconstructions are summarized in Table 8. About 80 percent of the leaves were automatically recovered by our

	nephthytis	poinsettia	schefflera	indoor tree
# Images	35	35	40	45
# 3D pts	128,000	103,000	118,000	156,000
# FG pts	53,000	83,000	43,000	31,000
# Leaves	30	≈120	≈450	≈1500
(α,k)	(0,3)	(0.5,3)	(0.5,3)	(0.3,2)
# AL	23	85	287	509
# UAL	6	21	69	35
# ASL	0	10	18	492
All leaves	29	116	374	1036
BET (min)	5	2	15	40

Table 1: Reconstruction statistics. The foreground points are automatically computed as the largest connected component in front of the cameras; they include both the plant, pot, and sometimes part of the floor. The segmentation parameters α and k are defined in Section 5.1. Note: FG = foreground, AL = automatic leaves, UAL = user assisted leaves, ASL = additional synthetic leaves, BET = branch edit time.

system. Figure 6 shows a simple example of inserting the reconstructed model into a synthetic scene. We also show examples of plant editing: texture replacement (Figure 1), and branch and leaf cut-and-paste (Figure 7).

Nephthytis. This plant has large broad leaves, which makes modeling easy. The 3D points are accurate, and leaf extraction and recovery are fairly straighforward. Only the extraction of 6 leaves are assisted by user, and the reconstruction is fully automatic. The 3D points were sufficient in characterizing the spatial detail of the leaf shapes, as shown in Figure 8.

Poinsettia and schefflera. These plants have medium sized leaves; as a result, the recovered 3D points (see the video) on leaves were still of high quality. Occlusion is difficult when the foliage is dense, and the foliage for the poinsettia (see Figure 1) and schefflera is denser than that of the nephthytis. Leaf segmentation of the schefflera is complicated by the overlapping leaves and the small leaves at the tip of branches. We recovered about two-thirds of all possible leaves, and synthesized some of them (on the top most branch) using the branching structure, as shown in Figure 7.

Indoor tree. The indoor tree, with its small leaves, was the most difficult to model. First, the 3D points are less accurate because they were typically recovered from 2D points on occluding boundaries of leaves (which are not reliable). In addition, much more user interaction was required to recover the branching structure due to its high complexity. The segmentation was fully automatic using a smaller $k = 2$. And for each group containing more than 3 points, the same automatic leaf fitting procedure like all other examples except the generic model is much simpler, is used. But the geometric accuracy of the orientation of the recovered leaves are noticeably less reliable than that of the large leaves in the other examples. If the group contains fewer than 3 points, it is no longer possible to compute the pose of the leaf. In this case, the pose of each leaf was heuristically determined using geometry information of its nearest branch.

9 Concluding Remarks

We have proposed a general approach to modeling plants from images. The key idea is to combine both available reconstructed 3D points and the images to more effectively segment the data into individual leaves. To increase robustness, we use a generic leaf model (extracted from the same image dataset) to fit all the other leaves. We also developed a user-friendly branch structure editor that is also guided by 3D and 2D information. The results demonstrate the

Figure 7: Schefflera plant. Top left: an input image (out of 40 images). Top right: recovered model with synthesized leaves rendered at the same viewpoint as the top left. Bottom right: recovered model from images only. The white rectangles show where the synthesized leaves were added in the top right. Bottom left: recovered model after some geometry editing.

effectiveness of our system. We designed our system to be easy to use; specialized knowledge about plant structure, while helpful, is not required.

There are several straightforward improvements on our current implementation. For instance, the graph-based segmentation algorithm could be made more efficient by incorporating more priors based on real examples. Our current leaf reconstruction involves shape interpolation using the precomputed 3D points; better estimates may be obtained by referring to the original images during this process. Also, we use only one 2D boundary to refine the shape of the 3D leaf model. It may be more robust to incorporate the boundaries from multiple views instead. However, occlusions are still a problem, and accounting for multiple view substantially complicates the optimization. A more complex model for handling complex-looking flowers could be built as suggested in [Ijiri et al. 2005]. Finally, for enhanced realism, one can use specialized algorithms for rendering specific parts of the plant, e.g., leaf rendering [Wang et al. 2005].

Acknowledgements

We would like to thank Oscar Au for helping to render results for this paper and Yichen Wei for his technical assistance. The work was supported by Hong Kong RGC Grant HKUST6182/04E and HKUST6190/05E.

References

BESL, P., AND MCKAY, N. 1992. A method for registration of 3D shapes. IEEE *Transactions on Pattern Analysis and Machine Intelligence 14*, 2, 239–256.

BOYKOV, Y., VEKSLER, O., AND ZABIH, R. 2001. Fast approximate energy minimization via graph cuts. IEEE *Transactions on Pattern Analysis and Machine Intelligence 23*, 11, 1222–1239.

DE REFFYE, P., EDELIN, C., FRANCON, J., JAEGER, M., AND PUECH, C. 1988. Plant models faithful to botanical structure and development. *SIGGRAPH*, 151–158.

Figure 8: Nephthytis plant. An input image out of 35 images on the left, and recovered model rendered at the same viewpoint as the image on the left.

FAUGERAS, O., LUONG, Q., AND PAPADOPOULO, T. 2001. *The Geometry of Multiple Images*. The MIT Press, Cambridge, MA, USA.

HAN, F., AND ZHU, S.-C. 2003. Bayesian reconstruction of 3d shapes and scenes from a single image. In *Proc. IEEE Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis*, 12–20.

HARTLEY, R., AND ZISSERMAN, A. 2000. *Multiple View Geometry in Computer Vision*. Cambridge University Press, June.

IJIRI, T., OWADA, O., OKABE, M., AND IGARASHI, T. 2005. Floral diagrams and inflorescences: Interactive flower modeling using botanical structural constraints. *ACM Transactions on Graphics (SIGGRAPH) 24*, 3 (July), 720–726.

LHUILLIER, M., AND QUAN, L. 2005. A quasi-dense approach to surface reconstruction from uncalibrated images. IEEE *Transactions on Pattern Analysis and Machine Intelligence 27*, 3, 418–433.

LI, Y., SUN, J., TANG, C.-K., AND SHUM, H.-Y. 2004. Lazy snapping. SIGGRAPH *2004, Los Angeles, USA 23*, 3, 303–308.

MECH, R., AND PRUSINKIEWICZ, P. 1996. Visual models of plants interacting with their environment. *SIGGRAPH*, 397–410.

NOSER, H., AND THALMANN, D. 1994. Simulating life of virtual plants, fishes and butterflies. In *Artificial Life and Virtual Reality*, N. Magnenat-Thalmann and D. Thalmann, Eds. John Wiley and Sons, Ltd.

NOSER, H., RUDOLPH, S., AND STUCKI, P. 2001. Physics-enhanced L-systems. In *Procs. 9th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, vol. 2, 214–221.

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

PRUSINKIEWICZ, P., MUENDERMANN, L., KARWOWSKI, R., AND LANE, B. 2001. The use of positional information in the modeling of plants. *SIGGRAPH*, 289–300.

RECHE-MARTINEZ, A., MARTIN, I., AND DRETTAKIS, G. 2004. Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Transactions on Graphics (SIGGRAPH) 23*, 3 (August), 720–727.

SAKAGUCHI, T., AND OHYA, J. 1999. Modeling and animation of botanical trees for interactive virtual environments. In *Procs. ACM Symposium on Virtual Reality Software and Technology*, 139–146.

SAKAGUCHI, T. 1998. Botanical tree structure modeling based on real image set. In *SIGGRAPH 1998 (Tech. Sketch)*, 272.

SHI, J., AND MALIK, J. 2000. Normalized cuts and image segmentation. IEEE *Transactions on Pattern Analysis and Machine Intelligence 22*, 8, 888–905.

SHLYAKHTER, I., ROZENOER, M., DORSEY, J., AND TELLER, S. 2001. Reconstructing 3d tree models from instrumented photographs. *IEEE Computer Graphics and Applications 21*, 3 (May/June), 53–61.

VAN HAEVRE, W., AND BEKAERT, P. 2003. A simple but effective algorithm to model the competition of virtual plants for light and space. In *Procs. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'03)*.

WANG, L., WANG, W., DORSEY, J., YANG, X., GUO, B., AND SHUM, H.-Y. 2005. Real-time rendering of plant leaves. SIGGRAPH *2005, Los Angeles, USA 24*, 3, 712–719.

WEBER, J., AND PENN, J. 1995. Creation and rendering of realistic trees. In *SIGGRAPH*, 119–127.