

# Image-based Plant Modeling by Knowing Leaves from Their Apexes

Wei Ma<sup>1</sup>, Hongbin Zha<sup>1</sup>, Jia Liu<sup>2</sup>, Xiaopeng Zhang<sup>2</sup>, and Bo Xiang<sup>2</sup>

Key Laboratory of Machine Perception (Ministry of Education), Peking University, P. R. China<sup>1</sup>,  
LIAMA-NLPR, CAS Institute of Automation, P. R. China<sup>2</sup>

{mawei, zha}@cis.pku.edu.cn, {jliu, xpzhang}@nlpr.ia.ac.cn, helenxiang@gmail.com

## Abstract

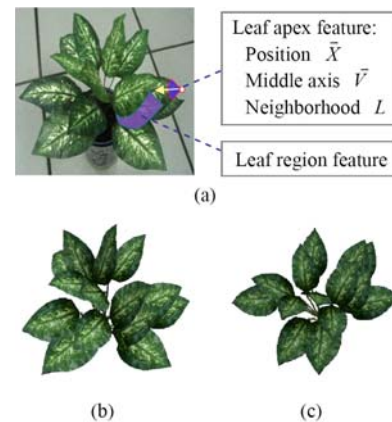
*In the paper, we present a novel approach to modeling plants from images by detecting apex features. First, an effective algorithm is proposed to extract apex features in volumetric data recovered from the images. It provides position and pose information for assigning 3D generic leaves. Then, the 3D leaf shapes are determined by an optimization based on the volume. Finally, Branches are modeled by using a particle flow approach. The proposed method is simply with limited manual intervention and has the obvious benefit of knowing a leaf by its visible apex part.*

## 1. Introduction

Modeling plants is a challenging task due to their complex geometry with serious occlusions. The modeling targets in the paper are plants with relative large leaves. The method we choose is image-based because relative to laser data, image data are noiseless and easily acquired. For reconstruction, the leaves in images should be identified as individuals. An intuitive method is to extract each leaf region through segmentation. However, intelligent segmentation of leaves is another big challenge in computer vision [4] since they are compact with similar appearances, as seen in Figure 1. Even a leaf region is correctly indicated (as shown in Figure 1(a)), the partial visible information can hardly determine the full leaf shape. Fortunately, we notice that as local features lying at the tipping points of the leaves, leaf apexes are more likely visible and recognizable than the leaf region features (as seen from Figure 1). If we know leaf shapes by their apexes, the plant can be reconstructed easily.

Motivated by the observations, we propose an apex detection algorithm in volumetric data which are recovered from a set of images. The algorithm as well obtains the pose information about leaves. Next, a generic leaf

is modeled by images and attached to each apex. An optimization process is then performed to further adjust the leaves in geometry. Finally, particle flows starting from the leaf bases are used to model branches. The following sections explain in detail our modeling approach of knowing leaves by their apexes.



**Figure 1. (a) An input image with two kinds of features; (b) and (c) show the model constructed by detected apexes.**

## 2. Related work

In recent years, plant modeling based on visual information (images or laser data) attracts many researchers due to its potential in generating realistic models. Shlyakher et al. [6] introduced image information to pilot the L-system. Xu et al. [9] constructed branches from 3D data based on knowledge and heuristics. Han et al. [1] modeled branches from a single image using a Bayesian approach. Neubert et al. [3] used particle flows to simulate branches by images. Tan et al. [8] modeled trees based on structure from motion. These methods focus more on the realistic effects

of branches. By contrast, individual leaf modeling for plants with relative large leaves should be paid more attention, as done in [4] and this paper. Quan et al. [4] combine 2D gradient with 3D depth information to segment each observed leaf with interactive assistance. In contrast to their method, the paper obtains 3D leaf shapes by extracting and analyzing simple apex features with limited user intervention which can be finished in seconds.

### 3. Apex feature extraction

The input for apex feature extraction is a continuously-connected volumetric model. Here, we choose the voxel coloring [5] approach with a photo-consistency constraint to get a geometric volume. The photo-consistency constraint we adopt is ZNCC, which is not so sensitive to varying lighting conditions.

#### 3.1. Sharp feature evaluation

It is observed that an apex lies in the tipping point of a leaf. Therefore, in this section, we present a geometric evaluation for the local sharpness property of each voxel in the volume to find apex positions.

Given a voxel  $C_0$  with spatial coordinates  $\vec{X}_0$ , we choose its continuously connected  $L$ -ring neighborhood (the 1-ring neighborhood of the voxel is its 26 closest neighbors). Assuming that the coordinates of a voxel in ring  $l$  are  $\vec{X}_n^l$ , where  $n \in N^l$  and  $N^l$  is the number of voxels in ring  $l$ , its extended length cumulated from  $\vec{X}_0$  to  $\vec{X}_n^l$  ring by ring along a direction vector  $\vec{V}_i$  is:

$$r_n^l(\vec{V}_i) = (\vec{X}_n^l - \vec{X}_0)^T \vec{V}_i. \quad (1)$$

The sharpness of  $C_0$ , denoted as  $S$ , can be evaluated intuitively by averaging the extended quantities of the voxels in the  $L$  neighborhood rings along an optimal direction vector  $\vec{V}_0$ , i.e.:

$$\vec{V}_0 = \underset{\vec{V}_i}{\operatorname{argmax}}(S(\vec{V}_i)), \quad (2)$$

where,

$$S(\vec{V}_i) = \sum_{l=1}^L \frac{1}{N^l} \sum_{n=1}^{N^l} r_n^l(\vec{V}_i), \quad (3)$$

with constraints

$$\vec{V}_i^T \vec{V}_i = 1 \quad (4)$$

$$r_n^l(\vec{V}_i) > 0, \forall \vec{X}_n^l. \quad (5)$$

By maximizing Eq. 3 with the constraints Eq. 4 and Eq. 5, we obtain the sharpness score  $S$ , along with the

vector  $\vec{V}_0$ .  $S$  represents the probability of  $C_0$  as a sharp apex. As explained in the following,  $\vec{V}_0$  will be the middle axis of the leaf indicated by  $C_0$ , when  $C_0$  is determined to be an apex voxel. Assuming that  $C_0$  is an apex and the neighborhood voxels involved in evaluating  $S$  are uniformly distributed in an axial-symmetric flat cone volume (as shaped near real leaf apices) with the vertex  $C_0$ ,  $\vec{V}_0$  can be easily proved in geometry the symmetric axis. On the other hand, leaves are generally symmetric about their middle axes determined by the leaf apices and bases. Therefore, in this case,  $\vec{V}_0$  can be treated as the leaf's middle axis direction.

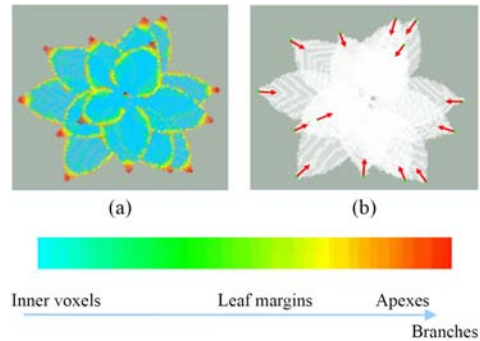
However, the uniform distribution does not exist due to the cubic voxelization. To address this issue, we recompute the cumulate extended length  $r_n^l$  ring by ring and replace the real distance from a voxel to one of its closest neighbors (face, edge or corner neighbors) in the upper ring with a unit 1:

```

 $r_n^l = 0$ 
 $l' = l$ 
while  $l' > 0$ 
  find a closest neighbor  $\vec{X}_n^{l'-1}$  for  $\vec{X}_n^{l'}$  in ring  $l'-1$ 
   $r_n^l = r_n^{l'} + \frac{(\vec{X}_n^{l'} - \vec{X}_n^{l'-1})^T \vec{V}_i}{\|\vec{X}_n^{l'} - \vec{X}_n^{l'-1}\|}$ 
   $l' = l' - 1$ 
endwhile

```

The scores of all the voxels form a spatial field, as shown in Figure 2(a), from which, we can see that large values are localized at convex leaf margins, especially on leaf apices. The optimal vectors are demonstrated in Figure 2(b) the right middle axes of leaves.



**Figure 2. Apex detection.** (a) A score distribution visualized with the pseudo color bar at the bottom representing ascending scores from left to right; (b) the extracted apex features each accompanied by a direction vector. Each voxel is rendered with  $\alpha=0.3$ .

### 3.2. Apex feature selection

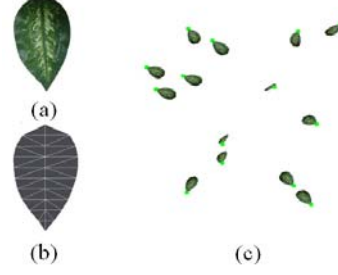
According to the definition of the scores in Eq. 3, bare and spindly branches gain the highest scores, and then come cone leaf apices (as seen from Figure 2(a)). Based on the regular distribution, we extract apex features as follows: 1) A local suppression (in the  $L$ -ring neighborhood) is performed to eliminate the competition from voxels near real apices. 2) The features surviving from the above process are put into a queue in order of the descending scores. 3) An appropriate range covering correct apex features is indicated by users in our 3D visual interface. In this way, we can obtain correct apices, each has a vector indicating the middle axis of a corresponding leaf. However, mistakes in the queue may happen for the reason that the geometric volume may be noisy and the score of a stubbed branch may be lower than that of a leaf apex. For robustness, we provide an extra interface for users to remove or add apex features by clicking corresponding positions in the volume. The interaction for indicating the score range and remedying the mistakes is limited and can be finished in seconds. For plants with multi-apex leaves, a simple clustering can be used to group sharp features according to some criterions established based on the special relationship among apices in the same leaf.

### 4. Plant reconstruction

To model the leaves of plants, we first generate a flat generic leaf model (as shown in Figure 3) by segmenting a leaf from its most frontal and parallel image. Next, we put the generic leaf at each apex position by the corresponding leaf middle axis. The orientation  $\vec{O}$  of the leaf plane is determined by analyzing the  $L$ -ring neighborhood voxels using PCA. After leaf assignment (Figure 3), the geometry of each flat leaf is adjusted as follows.

#### 4.1. Leaf scaling

To compute the scale of each leaf, we gather more neighbor voxels by iteratively increasing ring by ring with the initial  $L$  rings. At each iteration, a new ring is added and the most appropriate scale is computed with current neighbors by a discrete and finite searching in a certain scale range. The process stops until the optimal scale in a following iteration brings an fitting error larger than a given threshold. The error is computed by first projecting neighbors onto the leaf plane, and then fitting the projections determining leaf margins to the margin of that flat leaf.



**Figure 3. Leaf assignment. (a) An extracted leaf image; (b) a generic flat leaf model; (c) leaves arranged by apex features (the green squares).**

#### 4.2. Leaf deformation

In this section, each leaf  $F$  is locally deformed in its 3D shape by an optimization process. An energy function is defined as:

$$E(F) = w_1 E_b(T) + w_2 E_s(T) + w_3 E_m(M). \quad (6)$$

$E_b(T)$  is a balance term, which is given by:

$$E_b(T) = \sum_{T_i \in T} \sum_{\vec{X}_j^{T_i} \in \vec{X}^{T_i}} ((\vec{X}_j^{T_i} - \vec{C}_{T_i})^T \vec{O}_{T_i})^2. \quad (7)$$

$T$  is the triangle set of  $F$ .  $\vec{X}^{T_i}$  denotes the voxels whose projections on the plane of triangle  $T_i$  lie in  $T_i$ .  $\vec{C}_{T_i}$  is the center of  $T_i$ . The term will be at the minimum when voxels above and below the triangles reach a balance.  $E_s(T)$  is a smooth term, defined as:

$$E_s(T) = \sum_{T_i \in T} \sum_{T_j \in Nei^{T_i}} (\tan \frac{\theta_{i,j}}{2})^2. \quad (8)$$

$Nei^{T_i}$  means the 1-ring neighbor triangles of  $T_i$ .  $\theta_{i,j}$  denotes the inter-angle between the normals of  $T_i$  and  $T_j$ . The last term  $E_m(M)$  is a middle axis constraint (as defined in [2]) used to eliminate undesirable deformations on the middle axis:

$$E_m(M) = \sum_{M_i \in M} ((\frac{M_i M_{i+1}}{M_{avg}})^2 - 1)^2. \quad (9)$$

$M_i$  is a point segmenting the middle axis  $M$ .  $M_i M_{i+1}$  is the interval between two consecutive points in  $M$ .  $M_{avg}$  is the average interval. The energy will be at the minimum when  $M_i M_{i+1}$  equals to  $M_{avg}$ . The weights  $w_1$ ,  $w_2$  and  $w_3$  are used to modulate the three terms.

As done in Section 4.1, neighbors used here should be gradually increased with the initial  $L$  rings by checking in each iteration whether the thickness of the newly

added ring is smaller than a threshold which is approximated by both the real thickness of the leaves and the size of the voxels. In our experiments, even for leaves incompletely carved out in the volumetric data, which results in few neighbors used for deformation, the algorithm is effective due to the existence of the smooth term and the middle axis term.

### 4.3. Branch modeling

Once the leaves are reconstructed, we model branches using the particle flow approach in [3] by replacing their parallel projection to be a perspective one and their initial seeds to be the bases of our constructed leaves.

## 5. Results

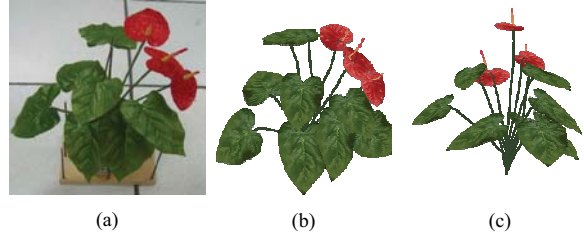
We test our approach on more than ten plants. Due to the limited pages, we present the results of three of them, including a Dieffenbachia (Figure 1), a Nephthytis (Figure 4) and a Anthurium scherzerianum (Figure 5). We take about 20-30 pictures for the plants. The parameter  $L$  used for controlling the size of neighborhood takes around 10. The manual intervention in constructing the leaves is limited to selecting a generic leaf, indicating the score range (as described in Section 3.2), and adding or removing no more than four apices. Users can finish the operations in seconds. The experiments demonstrate that our approach of knowing leaves by their apices can generate realistic models conveniently.



**Figure 4. Nephthytis. (a) An original photo; (b) reconstructed model rendered at nearly the same viewpoint.**

## 6. Discussions

We have presented an approach to modeling plants by detecting apex features in geometric volumes. Future work will focus on quantitative evaluation of plant modeling approaches and rendering of natural dynamic plants in virtual scenes.



**Figure 5. Dieffenbachia. (a) An original photo; (b) constructed models rendered at nearly the same viewpoint; (c) the model rendered at a new viewpoint.**

## 7. Acknowledgement

This work was supported in part by NKBRPC (No. 2004CB318000), NHTRDP 863 Grant (No. 2006AA01Z301, No. 2006AA01Z302 and No. 2007AA01Z336), Key Grant Project of Chinese Ministry of Education (No. 103001).

## References

- [1] F. Han and S. C. Zhu. Bayesian reconstruction of 3d shapes and scenes from a single image. In *Proceedings of IEEE Workshop on Higher-Level Knowledge in 3D modeling and Motion Analysis*, pages 12–20, 2003.
- [2] A. G. Manh, G. Rabatel, L. Assemat, and M. J. Aldon. Weed leaf image segmentation by deformable templates. *Journal of Agricultural Engineering Research*, 80(2):139–146, 2001.
- [3] B. Neubert, T. Franken, and O. Deussen. Approximate image-based tree modeling using particle flows. *ACM Transactions on Graphics*, 26(3):88–1–88–8, 2007.
- [4] L. Quan, P. Tan, G. Zeng, L. Yuan, J. Wang, and S. Kang. Image-based plant modeling. *ACM Transactions on Graphics*, 25(3):599–604, 2006.
- [5] S. M. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 25(3):151–173, 1999.
- [6] I. Shlyakher, M. Rozenoer, J. Dorsey, and S. Teller. Reconstructing 3d tree models from instrumented photographs. *IEEE Computer Graphics and Applications*, 21(3):53–61, 2001.
- [7] I. Soner and A. M. Day. Modeling trees and their interaction with the environment: a survey. *Computer & Graphics*, 29(5):805–817, 2005.
- [8] P. Tan, G. Zeng, J. D. Wang, S. B. Wang, and L. Quan. Image-based tree modeling. *ACM Transactions on Graphics*, 26(3):87–1–87–7, 2007.
- [9] H. Xu, N. Gossett, and B. Chen. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Transactions on Graphics*, 26(4):19–2–19–13, 2007.