MVA2005 IAPR Conference on Machine VIsion Applications, May 16-18, 2005 Tsukuba Science City, Japan
3-11

Tree Segmentation from an Image

Chin-Hung Teng[†], Yung-Sheng Chen[‡], and Wen-Hsing Hsu[†]
[†]Department of Electrical Engineering, National Tsing Hua University, Hsinchu, Taiwan.
[‡]Department of Electrical Engineering, Yuan Ze University, Chung-Li, Taiwan.
d897908@oz.nthu.edu.tw, eeyschen@ee.yzu.edu.tw, whhsu@ee.nthu.edu.tw

Abstract

Tree is a very common object in nature, thus its segmentation provides quite important information for scene 3D reconstruction. In this paper, we present an algorithm for segmenting trees from a complex scene. The trunk and leaf regions of a tree can be individually identified and the trunk structure of the tree can also be extracted. The proposed algorithm is mainly composed of a preliminary image segmentation, a trunk structure extraction, and a leaf regions identification process. We model the extraction of trunk structure as an optimization problem, where an energy function is formulated according to the color, position, and orientation of the segmented regions. We propose an algorithm to minimize this energy function and thus extract the trunk structure of the tree. After obtaining the trunk structure, the leaf regions can then be easily identified by finding those leaf regions located above the trunk regions. This algorithm has been tested on some real images and the results indicated that our algorithm performed well for these images.

1 Introduction

As the progression of computer technology, the applications of virtual reality have been greatly increased. For virtual reality, scene 3D reconstruction plays a quite important role. Since tree is a very common object in natural environment, its 3D model creation is necessary for a natural scene 3D reconstruction. In fact, tree rendering is a research topic in computer graphics and some approaches have been developed to render and create realistic trees [1–4]. In computer graphics, trees are often synthesized by some algorithmic rules plus some random variations. Some researchers also considered botanical principles in their algorithm to create trees faithfully reflecting the natural phenomena. Although the trees generated by graphical methods are quite realistic, they are sometimes not what we want. For instance, if we want to create a 3D virtual scene of our surrounding environment, then the created tree should have a similar 3D trunk structure and global leaf appearance to the real one, not generated artificially. We should create the 3D tree model according to the real tree in natural environment. One method to achieve this goal may be accomplished by taking several pictures of the tree and exploiting the techniques of *structure from motion* to construct a

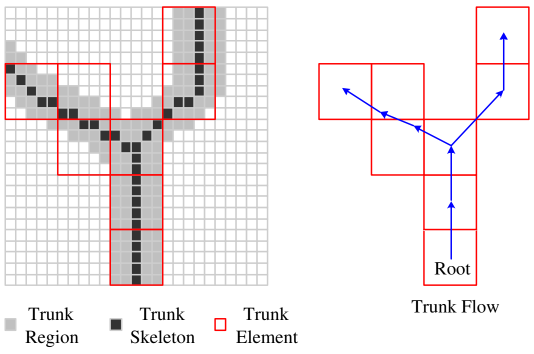

Figure 1. Trunk elements and trunk flow.

3D tree model similar to the real one. Specifically, we can first extract the trunk from the image and establish its 2D trunk structure. Subsequently, we can utilize trunk correspondences in different views to extend the 2D trunk structure to 3D trunk model. The leaves can be generated using graphical methods and pruned according to the tree in the images. The created 3D tree model is definitely not identical to the true one but is similar to it at least and for some applications this is enough. From this viewpoint, tree segmentation plays a rather important role for a similar 3D tree model construction. In fact, tree segmentation is also useful for other applications such as image retrieval and scene analysis. Therefore, in this paper, we develop an algorithm to segment trees from an image. This algorithm can not only identify the trunk and leaf regions from the scene but also extract the 2D trunk structure of the tree simultaneously. This algorithm consists of three units: a preliminary image segmentation, a trunk structure extraction, and a leaf regions identification process. We first segment the image using the well-known EM algorithm [5] and then formulate an energy function according to the segmented regions. The trunk structure is extracted by minimizing this energy function. After obtaining the trunk structure, the leaf regions can then be easily identified. We first merge the possible leaf regions using Bhattacharyya distance [6] and then draw a possible leaf area according to the extracted trunk structure. Those regions with partial area locating in the possible leaf area are then recognized as the leaf regions of the associated tree. In the following, we will describe the detailed procedures of how to extract the trunk structure from the segmented regions.

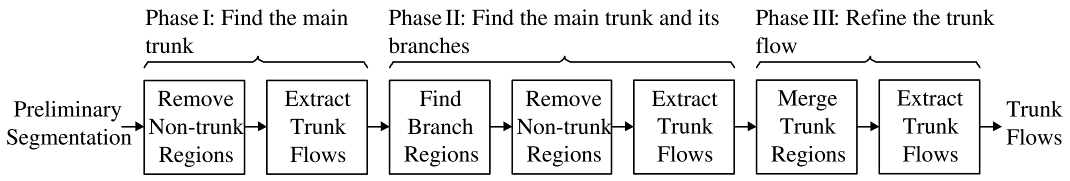

Figure 2. System diagram of the three-phase algorithm.

Table 1. Properties of a trunk element.

Property	Description	Definition
Region label, *L*	The region to which the trunk element belongs.	*L* is selected as the region label with the maximum number of skeleton pixels in the trunk element.
Direction, **v**	The direction of the skeleton in the trunk element.	**v** is defined as the unit eigenvector associated with the largest eigenvalue of **S** , where **S** is the covariance matrix of the position of skeleton pixels, from region *L*, in the trunk element.[a]
Color, **c**	The mean color of those region pixels in the trunk element.	$\mathbf{c} = \frac{1}{M_L}\sum_{i=1}^{M_L}\mathbf{c}_{L,i}$, where $\mathbf{c}_{L,i}$ is the color of pixel *i* with region label *L* and M_L is the number of pixels from region *L* in the trunk element.
Thickness, *T*	The thickness of the trunk.	$T = \frac{1}{N_L}\sum_{i=1}^{N_L}T_{L,i}$,where $T_{L,i}$ is the trunk thickness at skeleton pixel *i* from region *L* and N_L is the number of such skeleton pixels in the trunk element. The trunk thickness at skeleton pixel *i* is defined as 2×(number of thinning iterations to obtain the skeleton pixel *i*).
Centroid, **x**	The centroid of the skeleton pixels in the trunk element.	$\mathbf{x} = \frac{1}{N_L}\sum_{i=1}^{N_L}\mathbf{x}_{L,i}$, where $\mathbf{x}_{L,i}$ is the position of skeleton pixel *i* from region *L* and N_L is the number of such skeleton pixels in the trunk element.

[a] Generally, there is an ambiguity in the calculated direction. That is, **v** can be selected as positive or negative eigenvector. In our algorithm, we set **v** such that its y-component is greater than zero.

2 Extracting Trunk Structure

2.1 Three-phase algorithm

Generally, most trunk regions in an image have elongated shape, thus their skeletons provide a good feature in representing them. Hence, our basic idea is to trace the trunk by exploiting the skeletons of the segmented regions. However, the trunk of a tree is generally segmented into several regions, thus an efficient algorithm must be provided to connect these extracted skeletons. We partition the skeleton image into a number of small blocks called *trunk elements* as shown in Fig. 1. These small blocks are the basic elements for extracting trunk structure. We formulate the problem of extracting trunk structure as an optimization problem and define an energy function according to the color, position, and direction of the trunk elements. This energy function is the function of the connecting configuration of these trunk elements and can be minimized via connecting possible trunk elements. When the energy function is minimized, the trunk elements belonging to the true trunk regions are well-connected. The trunk structure can then be obtained by tracing these connected trunk elements from the root. These connected trunk elements are named *trunk flow* (see Fig. 1) in this paper. Because there are many non-trunk regions in the image, their skeletons will interfere with the connection of true trunk elements. Therefore, to correctly extract the trunk structure, these non-trunk regions should be removed in advance. In this research, we employ a systematic method to remove the non-trunk regions. Several rules are designed to test the color, orientation and position of the segmented regions and only the survived regions are processed in the subsequent procedures. However, this systematic method cannot fully filter out all the non-trunk regions. To reduce the interference of these survived non-trunk regions, we develop a three-phase algorithm to extract the trunk flow.

The system diagram of this three-phase algorithm is depicted in Fig. 2. We first concentrate on extracting the main trunk flow and identifying the main trunk regions in Phase I. Because only main trunk regions are considered, a stricter rule can be applied to remove the non-trunk regions. For example, those regions with horizontal skeleton can be removed because most main trunk regions have vertical orientation. This rule can greatly remove many undesirable regions and facilitate the searching of trunk regions. However, some branches of the tree will also be removed by this rule. Therefore, to retrieve the branch regions, we require a Phase II process. In Phase II, we detect the branch regions by finding those regions with similar color to the main trunk regions identified in Phase I and then compute the trunk flow of the entire tree again. By splitting the trunk regions identification into two phases, the influence of non-trunk regions can be extremely reduced and a more correct trunk structure can be extracted. However, sometimes the trunk will be segmented into several regions, thus the extracted skeletons will be slightly different from the true one. To solve this problem, we include a refinement phase in our algorithm. The main trunk regions and its branches are merged into one region and the trunk flow is recomputed for this merged region. The skeleton of this merge region is more accurate, thus more correct trunk flow can be calculated. In the following, we will present the details of the common blocks "Remove Non-trunk Regions" and "Extract Trunk Flows" in Fig. 2.

2.2 Remove non-trunk regions

In this algorithm, six rules are designed to remove the non-trunk regions. These rules are explained in the following:

1. **Trunk color**. Those regions with very different color from the trunk are removed.
2. **Relative position between trunk and leaf region**. Those regions without any leaf regions (the green regions are identified as possible leaf regions) above them are rejected.
3. **Direction of trunk element**. We remove those regions with horizontal orientation by this rule. This rule is only applied in Phase I for only the main trunk region can fulfill this condition.
4. **Thickness variation of trunk region**. If a testing region has a large thickness variation, then it may not be a trunk region and thus is removed.
5. **Thickness of trunk region**. If a testing region has a larger thickness than root thickness, then this region is not a trunk region and is rejected. This rule is only enabled in Phase II for, in this phase, the main trunk region has been identified in Phase I and therefore the root thickness can be calculated.
6. **Trunk region position**. Those regions with their positions below the root are not trunk regions and thus are removed. We enable this rule only in Phase II for the root position can be obtained from the identified trunk flow in Phase I.

2.3 Extract trunk flows

There are three steps in extracting trunk flows: 1) extract region skeleton, 2) compute the properties of trunk element and 3) connect trunk elements. We first extract the skeletons of the survived trunk regions and then determine the trunk elements from the extracted skeletons. We attach several useful properties for extracting trunk structure to each trunk element. These properties are summarized in Table 1. After computing the properties of each trunk element, we then formulate an energy function according to these properties and connect the trunk elements through minimizing this energy function. This energy function is defined as follows:

$$E(C) = \sum_{i=1}^{N(C)} \big(E_{i,r}+w_b E_{i,b}(C)+w_c E_{i,c}(C)+w_d E_{i,d}(C)\big), \tag{1}$$

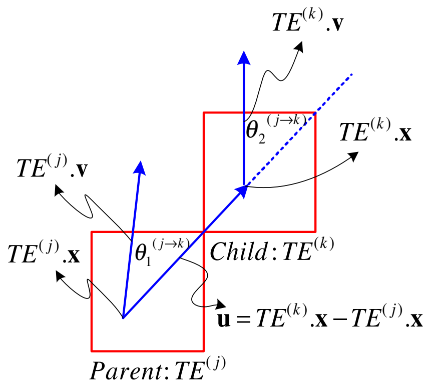

Figure 3. Illustration of $E_{i,d}$ and $E_{i,b}$.

where N is the number of trunk flows in the image. $E_{i,r}$ is a constant representing the root energy of i-th trunk flow. $E_{i,b}$, $E_{i,c}$, and $E_{i,d}$, which stand for the bend, color, and distance energies of i-th trunk flow, are defined in the following and illustrated in Fig. 3:

$$E_{i,b} = \sum_{\{j,k\}\in S_i} (\theta_1^{(j\to k)} + \theta_2^{(j\to k)}), \tag{2}$$

$$E_{i,c} = \sum_{\{j,k\}\in S_i} \left\| TE_i^{(j)}.\mathbf{c} - TE_i^{(k)}.\mathbf{c} \right\|, \tag{3}$$

$$E_{i,d} = \sum_{\{j,k\}\in S_i} \left\| TE_i^{(j)}.\mathbf{x} - TE_i^{(k)}.\mathbf{x} \right\|, \tag{4}$$

where S_i denotes the set of all the paired and connected trunk elements in i-th trunk flow. $TE_i^{(k)}.\mathbf{v}$, $TE_i^{(k)}.\mathbf{c}$, and $TE_i^{(k)}.\mathbf{x}$ represent the direction, color, and centroid of trunk element k. The parameters w_b, w_c, and w_d are the weights associated with $E_{i,b}$, $E_{i,c}$, and $E_{i,d}$, which are used to control the relative significance of these energies. In our algorithm, all the trunk elements are unconnected initially, thus each trunk element represents one trunk flow and all the connecting energies, which are the sum of bend, color, and distance energies, are zero. In this case, the total energy is contributed by all the root energies $E_{i,r}$. When two trunk flows are connected, one root energy is vanished and the connecting energy is increased. Thus, we can minimize this energy function by connecting those trunk flows with connecting energy smaller than the root energy. In this paper, we apply a heuristic algorithm to minimize the energy function given by (1). We connect the two trunk flows which lead to the maximum decreased energy and apply this procedure iteratively until the energy function is decreased no more. However, recall that there is an ambiguity in the direction of trunk element (see Table 1). The incorrect direction will prevent these trunk elements from connecting to the true trunk flows. To solve this problem, we include several additional steps in our algorithm. The overall algorithm is summarized in Fig. 4. After obtaining the trunk structure, the leaf regions can then be identified as described in section 1.

1. *Initialize trunk flows.* Set each trunk element as an individual trunk flow.
2. *Connect trunk flows.*
   - 2.1 Consider all the paired trunk flows *i* and *j*. Suppose trunk flow *j* is to be connected to trunk flow *i*. Calculate the decreased energy when connecting the root of trunk flow *j* to the optimal connecting point[a] in trunk flow *i*.
   - 2.2 Find the maximum decreased energy from the calculated energies. If the maximum decreased energy is greater than zero, then connect the two trunk flows which lead to this maximum decreased energy.
   - 2.3 Repeat steps 2.1 and 2.2 until no trunk flows can be connected.
3. *Break trunk flows.* Consider each trunk element in the image. If more than two trunk elements (the children) are connected to this trunk element (the parent), then break these connections. The parent becomes the end trunk element of its original trunk flow while the children change to the root of the new trunk flows.
4. *Identify root trunk flows.* If one trunk flow satisfies the following two conditions, it is identified as the *root trunk flow*.
   - C.1 The root of the trunk flow is in the lower part of the image.
   - C.2 There are no other trunk flows such that the root of this trunk flow can be connected to.
5. *Connect trunk flows with reverse operation.*
   - 5.1 For each root trunk flow *i*, consider every non-root trunk flow *j* in the image. Calculate the decreased energy when connecting the non-root trunk flow *j* to the root trunk flow *i*. Reverse the direction of the non-root trunk flow *j* and calculate the decreased energy again.
   - 5.2 Find the maximum decreased energy from the calculated energies. If the maximum decreased energy is greater than zero, then connect the non-root trunk flow to the root trunk flow which leads to this maximum decreased energy.
   - 5.3 Repeat steps 5.1 and 5.2 until no trunk flows can be connected.

[a]The optimal connecting point indicates that a minimum energy can be achieved when connecting these two trunk flows at this point.

Figure 4. The proposed algorithm for connecting trunk elements.

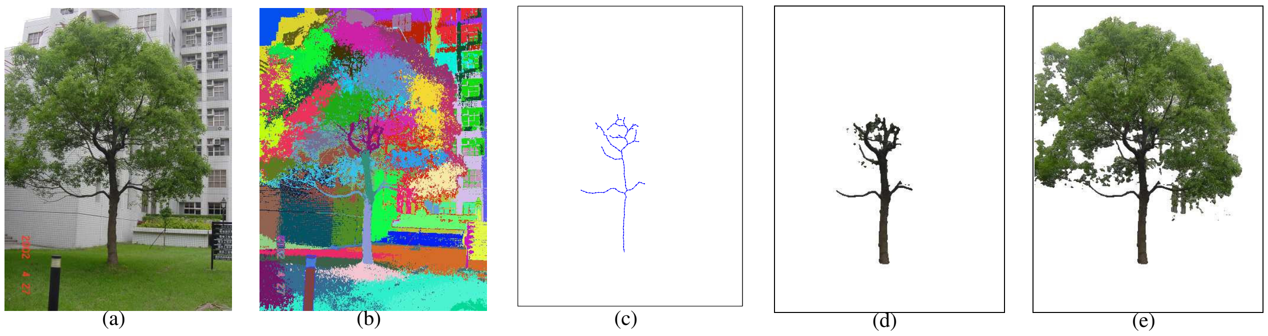

Figure 5. Illustration of tree segmentation using proposed algorithm: (a) input image, (b) preliminary segmentation, (c) extracted trunk flow, (d) extracted trunk region, and (e) final tree segmentation.

3 Experimental Results

In this section we illustrate the experimental results for applying our algorithm on several real images. One example for applying our algorithm on a real scene is shown in Fig. 5. Figure 5(a) is the original image, Fig. 5(b) is the preliminary segmentation and Fig. 5(c) is the extracted trunk flow. Although not so clear in Fig. 5(c), the extracted trunk flow is represented by the linked arrows just like Fig. 1. These linked arrows are started from the root and ended in the branches of the tree. By tracing the extracted trunk flow from the root, we can identify the trunk regions as revealed in Fig. 5(d). Clearly, the trunk of this tree except those hiding in the leaves is well segmented. After extracting the trunk regions, we then merge the possible leaf regions and find those regions located above the trunk regions. Combining the extracted trunk and leaf regions, the final tree segmentation can then be obtained as shown in Fig. 5(e). From Fig. 5(e), we can see that the entire tree in this image is perfectly extracted. In addition to segmentation of the tree, some other information about this tree can also be obtained from this algorithm. For example, we can understand where the trunk was and where the leaves were. We also know the position of root, branches, and how these branches were connected to the main trunk. Our approach is not merely a tree segmentation algorithm but also includes the understanding of this tree, just like humans recognize the tree.

We also tested our algorithm on several other real images. These images as well as their experimental results are

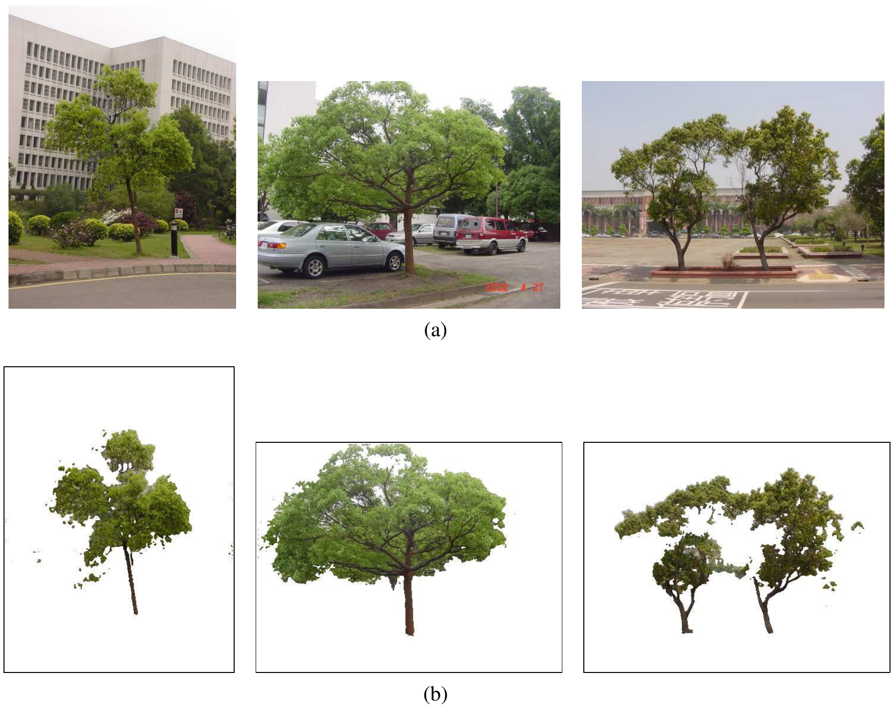

Figure 6. Experimental results for several tree images. (a) Original images, (b) final tree segmentations.

shown in Fig. 6. These images include those with only one tree[1] (the first and second images in Fig. 6), the image with two trees (the third image) and the image with very complex trunk structure (the second image). From the results, we can see that the trees in these images are correctly segmented even for those with the background similar to the extracted tree (e.g., the first and second images in Fig. 6). Although they are not shown in this figure, the trunk flows associated with these trees were also extracted. These trunk flows can provide further information about these trees in the images.

4 Conclusion and Future Work

In this paper we have proposed an algorithm for tree segmentation. The proposed algorithm consists of three units: preliminary segmentation, trunk structure extraction and leaf region identification. This algorithm has several merits: 1) we can understand where the trunk is and where the leaves are; 2) we can know the root position of this tree and the branch-points of the trunk; 3) we can distinguish different trees in the image for their trunk flows are extracted separately; and 4) the effect of complex background can be somewhat reduced for the leaves are located according to the extracted trunk flow so that those background regions similar to the leaves but far apart from the trunk can be well excluded. We have tested the proposed algorithm on some real images and the results were encouraging. The entire tree could be correctly segmented from the image and the trunk structure was also extracted. Our future work is to extend the extracted 2D trunk to 3D structure via structure from motion techniques and create a 3D tree model similar to the real tree in the images.

[1]Here, we only consider those trees with clear trunk. The trees with only leaves appearing in the image are neglected.

Acknowledgment

This work was supported in part by the National Science Council, Taiwan, Republic of China, under the grant numbers NSC 93-2213-E-007-013.

References

[1] A. R. Smith, “Plants, fractals, and formal languages,” *Computer Graphics*, vol. 18, no. 3, pp. 1–10, 1984.

[2] A. L. P. Prusinkiewicz and J. Hanan, “Developmental models of herbaceous plants for computer imagery purposes,” *Computer Graphics*, vol. 22, no. 4, pp. 141–150, 1988.

[3] J. Weber and J. Penn, “Creation and rendering of realistic trees,” in *Proceedings of the SIGGRAPH ’95*, 1995, pp. 119–128.

[4] B. Lintermann and O. Deussen, “Interactive modeling of plants,” *IEEE Computer Graphics and Applications*, vol. 19, no. 1, pp. 56–65, 1999.

[5] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Pearson Education, Inc., 2003.

[6] A. Bhattacharyya, “On a measure of divergence between two statistical populations defined by their probability distributions,” *Bull. Calcutta Math. Soc.*, vol. 35, pp. 99–110, 1943.