First publ. in: ACM Transactions on Graphics (TOG) ; 30 (2011), 4. - 53

http://dx.doi.org/10.1145/2010324.1964948

Texture-Lobes for Tree Modelling

Yotam Livny[1], Soeren Pirk[2], Zhanglin Cheng[1], Feilong Yan[1], Oliver Deussen[2], Daniel Cohen-Or [3], Baoquan Chen[1]

[1] SIAT, China, [2] University of Konstanz, Germany, [3] Tel Aviv University, Israel

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

Figure 1: *Reconstruction of a scanned tree using our lobe-based tree representation: a) photograph; b) point set; c) lobe-based representation with 24 lobes (22 kB in total); d) synthesized tree (25 MB in total).*

Abstract

We present a lobe-based tree representation for modeling trees. The new representation is based on the observation that the tree's foliage details can be abstracted into canonical geometry structures, termed lobe-textures. We introduce techniques to (i) approximate the geometry of given tree data and encode it into a lobe-based representation, (ii) decode the representation and synthesize a fully detailed tree model that visually resembles the input. The encoded tree serves as a light intermediate representation, which facilitates efficient storage and transmission of massive amounts of trees, e.g., from a server to clients for interactive applications in urban environments. The method is evaluated by both reconstructing laser scanned trees (given as point sets) as well as re-representing existing tree models (given as polygons).

Keywords: Plants synthesis and reconstruction, Point-based modeling, Rule-based tree modeling, Natural phenomena

1 Introduction

Trees are ubiquitous in nature and urban scenes and play an important role in enriching the realism of virtual environments. In past years many procedural methods have been developed for the design and creation of geometric tree models [Deussen and Lintermann 2005; Palubicki et al. 2009]. From a small set of rules, such as those used in L-systems, these techniques can create visually appealing tree models, which can be extremely complex in geometry and large in size. Given the high computation expense, such procedural operations cannot be performed during rendering time, such that applications have to deal with these heavy models. Furthermore, controlling the resulting geometric shape and conforming to specific characteristics of individual trees are still difficult issues [Stava et al. 2010; Benes et al. 2011; Talton et al. 2011]. A number of reconstruction methods have been developed that allow for modeling specific trees from real world data such as sets of photos [Reche-Martinez et al. 2004; Neubert et al. 2007] or 3D scans [Xu et al. 2007; Livny et al. 2010]. While the precise reconstruction of such models is steadily increasing, again, these methods tend to produce enormous amounts of geometry details representing the fractal structure of a tree.

In this paper, we present a novel representation of tree models, which captures the main characteristics of an individual tree and yet does not create too many structural nuances. The new representation is based on the observation that a tree's foliage details can be abstracted into canonical geometry parts, whose outer shapes we call lobe-geometry (or simply lobes). A tree can be simply represented by a set of lobes, which serve as a light weight intermediate representation, from which the full tree model can be efficiently synthesized by instancing (or texturing) the lobes with pre-defined patches.

The patches need to be stitched together to form a meaningful branching structure; this is inspired by patch-based texturing. In our case, however, the patches are small, predefined pieces of branch geometry that we combine using a discretization of botanic parameters such as branch width and vertical angle. The method therefore could also be seen as an intelligent instancing that is directed by botanic and geometric constraints.

Besides the overall shape of the foliage, the individual tree geometry is mostly determined by its main branching structure. This part of the model is encoded in the form of a skeletal graph with associated allometric information. The skeletal graph, together with the lobes and a set of associated species-specific parameters, forms what we call a lobe-based tree representation, which can be decoded and synthesized back to a full tree that resembles the original tree model. Figure 1 illustrates the process.

The main contribution of the paper is the introduction of the lobe-based representation of tree geometry and the further introduction of techniques to (i) approximate the geometry of given tree data and encode this representation, (ii) decode the representation and synthesize a fully detailed tree model that visually resembles the input. The achieved advantages are an extremely light-weight, yet highly representative representation of tree geometry, which not only facilitates efficient storage, transmission, and rendering of massive amounts of trees, but also eases tree modeling, whether for design or reconstruction.

2 Overview

Figure 2 illustrates the overall tree modeling and synthesis process: an input tree model, in this case a point set, is converted into the lobe-based representation – an encoding process, and subsequently reconstructed from it by texturing the lobes and producing the branch geometry – a decoding process. The encoded tree representation requires only a significantly small memory footprint for an individual tree. The efficiency of the decoding process, mainly lobe texturing, enables us to render many detailed trees.

A key factor for achieving high fidelity of the final tree representation is the construction and assignment of the lobe textures. It requires pre-generating a species library including parameters and sets of patches (textons), which are used for producing the lobe textures and reflect the characteristics of distinct species. The given tree data is initially classified into certain species based on its statistical properties. The result of this classification subsequently guides the assignment of proper parameters from the species library.

The remainder of the paper is organized as follows. After discussing related work, we start in Section 4 by describing the lobe-based representation, followed by model reconstruction (Section 5). In Section 6 we describe the applicability of our representation for modeling and rendering of large urban scenes acquired by LiDAR scanners. Trees in such a scene are automatically processed, classified and encoded. Later they are rendered interactively with high visual fidelity. We show a number of results in Section 7, followed by discussions and conclusions (Section 8).

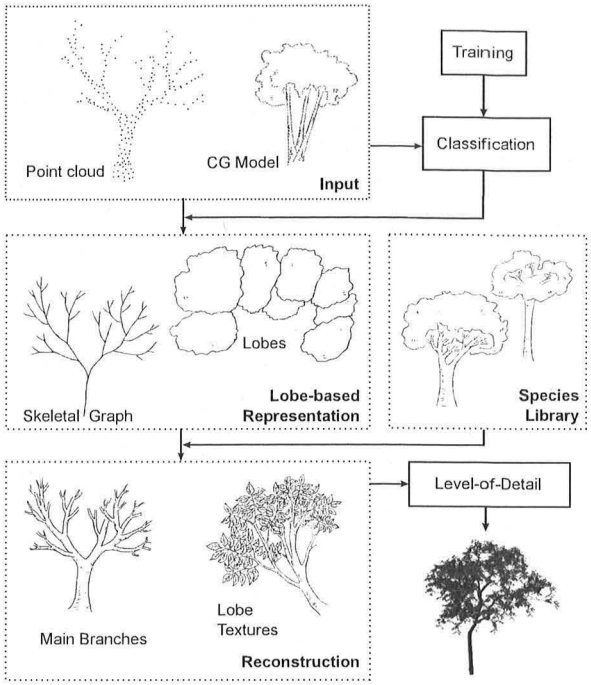

Figure 2: *Description of the tree modeling and reconstruction process: the lobe-based representation is computed from the input (point set or CG model). Additionally a classification subsystem is trained and later used to determine the species. A species library was built separately, including procedural elements and parameters for the different species. The tree is reconstructed by rebuilding the branching geometry from the skeletal graph and texturing the lobes using predefined elements from the species library. Level-of-detail, facilitated by lobe-based representation, is applied for interactive rendering.*

3 Related Work

Tree modeling has enjoyed considerable research attention in recent years, for both *designing* virtual trees and *reconstructing* real trees.

Rule-based systems are traditionally used for tree modeling [Honda 1971; Prusinkiewicz and Lindenmayer 1990], along with particle-systems [Reeves and Blau 1985], and space colonization [Greene 1989; Palubicki et al. 2009] frameworks. A good introduction is given by Deussen and Lintermann [2005], in which they also present the Xfrog modeling method which combines rule-based and procedural modeling.

In recent years efforts have been made to take guidance from either the user or the target constraints (photos or scans). Sketch-based approaches, such as [Ijiri et al. 2006; Anastacio et al. 2006; Tan et al. 2008; Wither et al. 2009], directly guide the design of tree objects. In [Okabe et al. 2006; Chen et al. 2008] sketching is enhanced by using examples taken from a library of tree models. Stava *et al.* [2010] use clustering techniques to detect patterns in a given 2D vector image and represent the patterns by rules of an L-System. A more challenging goal is inverse modeling to pinpoint the outcome towards a desired shape. Talton *et al.* [2011] have developed an optimization method to produce L-System rules that create a given shape. Their work is the first to enable inverse modeling; however, the computational cost of their technique is particularly high, suggesting that general inverse modeling is an extremely complex problem. Benes *et al.* [2011] use a connected set of guides to control local procedural environments in their behavior.

Reconstruction of a particular tree in the real world can be based on a collection of photographs. Reche-Martinez *et al.* [2004] use registered photos to generate a volumetric representation of the tree canopy and its branches and twigs, Neubert *et al.* [2007] use only loosely arranged input images. Other approaches [Shlyakhter et al. 2001; Tan et al. 2007] extract visual hulls from the input images and use L-Systems to synthesize branches within these hulls. Some methods [Runions et al. 2007; Tan et al. 2008] create an approximate but simple branching structure within envelope surfaces created from a single image or user sketch by applying some heuristics about the tree form.

With scanning technology becoming available, approaches for tree reconstruction from point sets were developed. Verroust and Lazarus [1999] as well as Xu *et al.* [2007] cluster edges in a spanning graph to reconstruct the tree skeleton, leaves are randomly added to the fine branches. Later approaches [Cheng et al. 2007; Zhu et al. 2008] focus on reconstructing tree properties such as main branches and crown shapes to overcome the insufficient sampling density for finer details of the tree. Bucksch *et al.* [2008; 2009] use space partitioning to cluster points and form a skeleton by connecting adjacent clusters. Côté *et al.* [2009] synthesize minor

tree and leaf geometry on the reconstructed branches based on light scattering properties obtained from different intensities of points. Livny *et al.* [2010] use global optimizations for automatically reconstructing the branching structure of multiple overlapping trees.

As we can see, the above tree modeling methods are moving in a direction that favors higher level abstraction and control of the intended geometry. Trees are complex natural phenomena due to their structural nuances and random nature. Skeletal structures are highly abstracted properties of trees, but using them alone or augmenting them with random leaves is insufficient in conveying full visual realism. Here we introduce sets of lobes and their texturing to augment skeletal structures and to encapsulate structural nuances. Such a representation strikes a balance between high level control, and low level botanical commitment.

4 Lobe-based Tree Representation

Let us assume for now that the tree data is given in form of a point set. If a complete virtual tree model is given, its connectivity information will better facilitate the generation of lobe-based representation, as we will comment on in the following description. Also, we assume a species library containing structural parameters is given.

4.1 Skeletal Structure

First we create a representation for the skeletal structure. There are several existing methods that extract such structures. The main branches of a tree are described using a graph with spline functions, which are associated with allometric values for their diameters along the axes. A typical skeletal graph consists of a few dozens of edges.

For the completeness of the description, also to facilitate discussions of lobe-geometry generation, we briefly describe the process of reconstructing skeletal structures. Similarly to Livny *et* al. [2010] we first connect neighboring points and construct a shortest-path tree. Each edge (u, v) is assigned an edge weight $\|u - v\|^{\beta}$ which determines how likely its points belong to the same branch. The parameter β allows for leveraging the edge lengths within Dijkstra's algorithm, which is applied in the next step to get the main tree structure. If β is close to one, the weights reflect the Euclidian distances of the corresponding points. Higher values will assign larger weights on longer edges and thus create a more compact tree structure.

Table 1: *Typical parameter values of β, γ, and f_s.*

Tree Species	β	γ	f_s
Mahogany	1.7	1.3	0.975
Bischofia polycarpa	1.8	1.5	0.986
Delonix	1.7	1.5	0.982
Lagerstroemia	1.6	1.5	0.976
Ailanthus altissima	1.5	1.9	0.980
Palm	1.8	1.3	0.800
Terminalia	1.5	1.5	0.970
Pine	1.3	1.2	0.980
Willow	1.5	1.9	0.980
Ficus Virens	2.0	1.6	0.975

While Livny *et* al. [2010] tried to obtain this parameter automatically, which does not work well for leafy trees, we determine it manually for every species and store it along with other parameters in the species library (Table 1). However, if no species information is given, we use a default value. In our application work (Section 6) we use a classifier to determine the species.

A second parameter is important for computing the lobe-based representation: given the diameter of the trunk at the tree base (d_{root}), we obtain a formula for tree allometry describing the decrease of branch diameters with their distance to the tree base. The general mechanism was already described by Da Vinci who stated that the diameter of a branch is the sum of all diameters of branching branches (see [Jaccard 1913]). Based on this observation we can compute the diameter $d(u)$ for a node u in the spanning tree:

$$d(u) = d_{root} \left(\frac{l(u)}{l(root)} \right)^{\gamma} \tag{1}$$

where $l(u)$ is the sum of length of all edges in the subtree and with root node u, $l(root)$ being the sum of the length of all the edges in the complete spanning tree.

Studies support this relation with $\gamma \approx 1.5$ for many cases [Xu et al. 2007]; however, in practice it also depends on the species and on environmental factors such as snow mass, wind strength and elevation of the tree stand. Based on 8–10 scans for each tree species we manually determined typical values for γ (see Table 1) and stored them in our library.

4.2 Lobe Geometry

The tree-graph and associated parameter γ enables us to select all edges for which the average distance of their assigned points is higher than the diameter of the branches. There is only a low probability for these points to be directly connected in the original tree structure, and therefore we have low confidence in the generated edges.

We follow the edges in the tree-graph from the root to the leaves until we reach such a low-confidence edge (at a point p_i) and mark the so far traversed edges as the main branching structure. The points that belong to the remaining edges are all connected to one of the p_i and thus are collected to form a cluster C_i, which is now represented as a lobe.

In practice, however, in order to determine the lobes, the branch diameter and respective maximal edge-length has to be modified since it is much harder to scan the interior of leafy trees, due to occlusion of the foliage, in comparison to sparse models. Therefore we add the parameter f_s into Eq. (1):

$$d'(u) = f_s \cdot d(u) \tag{2}$$

This parameter virtually reduces the diameter of the branch at a certain distance and thus enables us to modify the amount and size of lobes. Small values of f_s create large lobes, and for a tree without leaves, a value of $f_s \approx 1.0$ is used to generate many small lobes and reconstruct branches from the majority of the points (see Figure 3 and Figure 11).

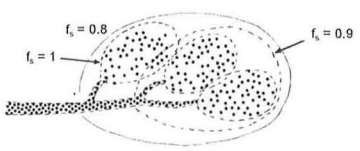

Figure 3: *Lobes for different values of f_s.*

We use the α-Shape approach [Edelsbrunner and Mücke 1994; Zhu et al. 2008] for computing the lobe-geometry from the points of each cluster C_i. α-Shapes are extensions of convex hulls and allow

non-convex envelopes to be created. The points are combined to hexagons, with α determining the radius of a virtual ball, which is used to delete all those hexagons that not fit into the ball.

An appropriate value of α plays an important role for balancing between capturing the accuracy of a lobe on the one hand and the size of the resulting geometry on the other. A large value of α will create the convex hull, a small value a surface with many holes. Since a good value for α dependents on the point density, we compute α_{min} as the value for which the points are represented by a single surface without holes and define our alpha as a multiple of this value. Our experiments show that independently from the species a value of $\alpha = 5.0 \cdot \alpha_{min}$ results in sufficiently accurate and simple hulls.

Figure 4: *Subset of predefined branch patches (small geometric elements) and a composed structure for Lagerstroemia. The dots denote predefined docking positions for adjacent patches.*

5 Model Reconstruction

Given the lobe-based representation and the species-dependent parameters from our library we are able to reconstruct the tree geometry. The main branching structure is produced by creating a generalized cylinder [Bloomenthal 1985] for each edge of the skeletal graph using the width information we obtain from the tree allometry. Branching is realized by computing smooth curves that connect the branches and by tessellating them later using the width information. Additionally, the lobes are represented by the triangulated surfaces produced from the α-Shapes, furthermore, we store the seed points on the skeletal graph for procedural texturing.

To synthesize a species we generate a set of branchlets and store them in a species library (see Figure 4). We use L-systems to form a collection of 20–30 complete branchlets for each species varying in thickness, shape, and orientation. Each of the branchlets is then partitioned in order to form smaller incomplete branchlets (patches). Docking points are inserted on branchlets where smaller twigs can be placed and each docking point has an assigned orientation and thickness.

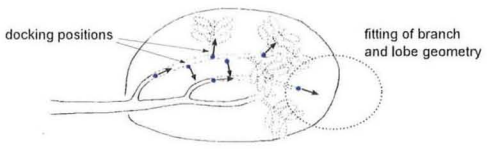

Figure 5: *Selection of patches for lobe texturing: patches are selected by branch diameter and vertical angle (blue dots with arrows) and have to match the lobe shape.*

The texturing of a lobe from seeds is performed by iterating two steps: patch selection and patch fitting. For patch selection we enforce thickness and orientation constraints to a seed point s. A subset S_s of patches is selected from the pre-defined patches by taking those with similar initial thickness and orientation to s.

The fitting of a patch to a docking position is computed by

$$d = \frac{\max(thickn_r, thickn_d)}{\min(thickn_r, thickn_d)} \times (v_r \cdot v_d)$$

where r is the root point of the patch, and d is the docking point, v_r, v_d are the orientation vectors of patch and docking position. For patch fitting we select the patch that additionally corresponds best to the local geometry of the lobe. We measure the distance of the patch geometry and the points of the lobe geometry. The selected patch is added to the lobe texture, while its docking positions, if there are any, are used for the next texturing iterations (see Figure 5). The smallest patches do not have docking positions and terminate the texturing process.

Patch fitting is very similar to patch-based texture synthesis [Efros and Freeman 2001] in which a patch (in this case part of an input image) is also selected in order to fit to a given local environment (the borders of a so far constructed texture). In texture synthesis, patches are selected and combined such that the texture is similar to the input image but has a larger size. In our case the "input image" is the pre-defined branching structure of the species and the "texture" is the reconstructed branching structure in the lobe.

Figure 6: *Texturing lobes: while the lobe geometry serves as a constraint it is only approximated by the branching structure due to limited lobe-patches provided by the library.*

Figure 6 shows two examples for the texturing of lobes. Starting from an initial branch patch, new ones are selected and added until the lobe geometry is filled, i.e., all docking positions are combined with patches.

6 Application in Acquisition and Modeling

To demonstrate the applicability of our tree representation we applied it to the modeling of massive amounts of trees acquired by a laser scanning device. The key to a high-quality representation with our lobe-based representation is the classification of the individual tree models. It helps us to retrieve appropriate parameters (e.g., allometric values) for generating the skeletal structure and for assigning the lobe-patches to lobe-geometry.

6.1 Tree classification

We have developed a supervised classification method that allows us to classify scans of trees taken from real urban environments. Note that tree classification remains a very complex and challenging problem, for which a general and robust algorithm is still hard to find.

We classify the scanned point sets by computing a number of features from the points and an approximate first reconstruction due

to Livny *et* al. [2010]. However, botanists consider also leaf shape and texture when identifying a tree species [Agarwal et al. 2006]. Bark texture and multispectral images of the tree canopy also provide additional clues [Key et al. 2001]. Thus our method cannot be generally applied to all kinds of trees and all kinds of contexts. Nevertheless, it works effectively considering the fact that only a limited number of species exist in a local area and there is a certain spatial coherence of trees.

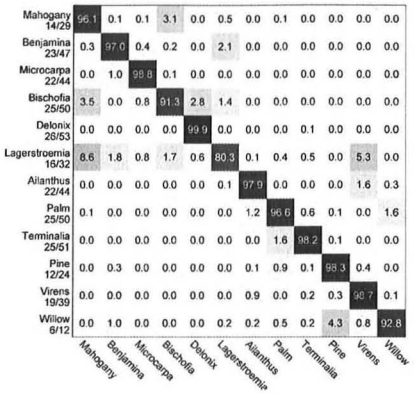

Figure 7: *Recognition rate for the Joint Boost Classification with 12 species and 12–53 scans for each (training set size and total set size under the tree names). Each row describes the classification results for point sets of one species.*

For each tree we compute over 200 features, which are useful to distinguishing different tree shapes. The values are combined to a parameter vector $\mathbf{p}$ and a Joint Boost classifier [Torralba et al. 2007] is trained with tree models that are typically found in urban environments. Joint Boost results in a vector $\mathbf{h}$ of probabilities describing the likelihood for a point set to belong to one of the species.

To evaluate the effectiveness of our classification method we used 12–53 scanned and manually classified individuals for twelve given species. Half of them were used as training set, the other half for testing. The average recognition rate was determined by 100 experiments where the models for the training sets and those for the test set were selected at random.

The average classification result is shown in Figure 7. Most species are correctly classified with an average recognition rate of 95.5%, which is high in comparison to usual rates in classification.

By analyzing the classification results we find the most effective features are related to the following geometric properties (as also illustrated in Figure 8):

- trunk width, trunk height, crown width, height, ratio between tree width and height
- distribution of normal directions (computed in a neighborhood of each point)
- density distribution of the point set in vertical direction (second/third/fourth central moment)
- inhomogeneity of point set, number of main stems

Furthermore, trees in urban areas are not planted at random but typically in arrangements of the same species. So we amended the Joint Boost classification by a simple spatial voting mechanism in order to improve the results.

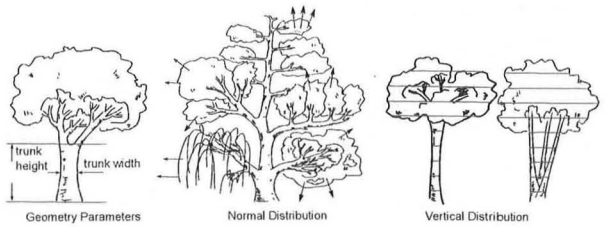

Figure 8: *Effective features used for classification: tree shape parameters, distribution of normals (trees with different shape structure vary in the normal distribution) and point set distribution.*

Confidence in a classification of Joint Boost is defined as when $h_1 > 1.5 \cdot h_2$ with h_1 being the highest probability value in the probability vector and h_2 the second highest. If we do not have confidence we collect all classification results for the neighboring trees within a radius of 25 meters (2–3 times the diameter of a typical tree) about which we can be confident and subsequently create a set of possible species. The candidate species with the highest probability is selected.

The method works very well for trees in an urban landscape, here in most cases we reach an average recognition rate of 98.8%. However, for mixed scenes with random species the results produce lower recognition rates, from an average of 95.5% to 89%.

6.2 Level-of-Detail Representation and Rendering

For the interactive rendering of dense scenes with many trees an efficient level-of-detail (LOD) mechanism is needed [Deussen et al. 2002]. The lobe texturing (see Figure 6) lends itself to a progressive LOD mechanism – patches are added or removed based on the distance of the lobe to the camera. To avoid popping with changing LOD, we apply stochastic pruning [Deussen and Lintermann 2005; Cook et al. 2007], which has proven to be an efficient LOD method for rendering trees, grass and other nature objects that consist of many small elements (leaves or grass blades). The method deletes part of the elements and enlarges the rest in a way that enables the overall appearance to be maintained.

In its original form the method requires that the complete geometry is produced before LOD can be applied. In our case, when a dense scene with many trees has to be displayed, this is not possible.

In this event, we use interactive geometry production on the GPU combined with a dynamic variant of stochastic pruning to avoid this problem. We benefit from the fact that our scene is assembled from a huge number of instances taken from a relatively small set of species and corresponding branch patches (20–30 per species). Each branching patch has a fixed number of leaves, which are stored in random order in a Vertex Buffer Object (VBO). During stochastic pruning, we render only a prefix of the VBO for each instance of the patch. The leaves are faded in and out to avoid popping artifacts.

Patch optimization. The overall number of patch instances affects the run-time performance on the GPU, where the patches have to be stored and accessed. For interactivity, it is desirable to minimize the number of instances. We do so by forming larger patches from a set of smaller ones belonging to the same lobe. These newly generated patches are used for the respective lobes to reduce the total number of GPU calls, thus improving the performance at the cost of memory. We strike a balance between performance and memory adapting to the available memory.

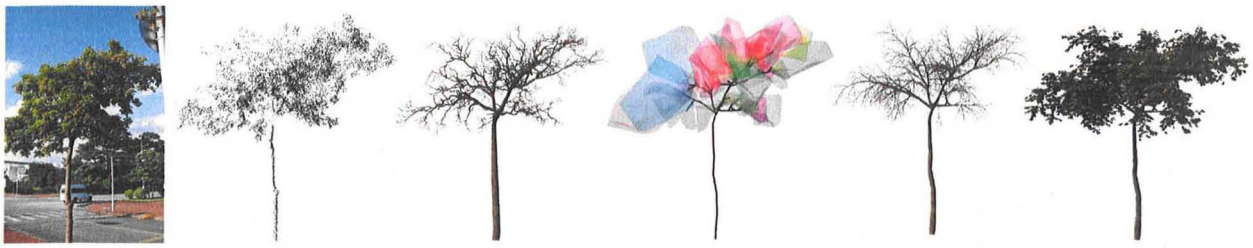

Figure 9: *Advantage of using lobe-based representation for laser scanned tree data (left to right: photo, point set, conventional skeletal structure reconstruction (e.g., Livny et al. [2010]), lobe-based representation, skeletal geometry from lobe-based representation, full tree geometry after lobe texturing. Results from lobe-based representation better captures the characteristics of the input tree.*

Skeletal Graph. The segments of the tree skeleton (the edges of the explicit graph) are sorted by their thickness and also stored in a buffer. We render a prefix of these edges based on their distance from the camera. On the GPU, furthermore the screen space length of an edge is used to control the amount of the generated geometry by the tessellation shader.

7 Results

We have tested our approach on point data obtained from laser scanning; the results are illustrated in Figure 9. An advantage of lobes in reconstruction from point sets is its proper synthesis of details that are not sufficiently captured in the original data, a common situation for scans. As long as points exist to define a lobe geometry, the internal structure of the tree can be synthesized by lobe texturing; oftentimes, the synthesized structures better capture the characteristics of the species. Reconstruction methods, such as global optimization [Livny et al. 2010], applied to the mere input of point sets may exhibit structures that appear foreign or even erroneous, due to noisy and insufficient data.

A faithful reconstruction corresponds to the proper selection of number and size of lobes. The number of lobes is mainly determined by the 3D arrangement of the main branches of the tree. As mentioned above, the number of lobes is achieved by using a clustering parameter for each species in our species library.

Figure 11 shows a tree with different amount of lobes generated by changing the parameter value. When the rightful number of lobes (in this case 20) is used, the overall tree structure is reconstructed faithfully. In this case, the split between the skeletal structure and lobe geometry matches what can be faithfully extracted from the point set and what has to be synthesized by lobe texturing. When only one lobe is used, the tree structure is over-synthesized – the texturing mechanism fails to fill in the whole lobe and the synthesis inevitably mismatches some prominent skeletal structure in the original data. When a very large number of small lobes is used, the tree structure is under-synthesized, resulting in missing structural details in the crown area. Figure 14 shows that for different species with different geometry structure, a different number of lobes is selected.

We have also applied our lobe-based representation to computer-generated plant models from the Xfrog library. We perform this experiment by simply converting the Xfrog model into a point cloud similar to laser scans, then apply the aforementioned method. Figure 12 shows the results that quite faithfully resemble the original models.

Our representation method allows us to reconstruct a variety of tree species ranging from leafy trees to special forms such as Weeping Willows, Palms, or Pine trees (see Figures 10, 14). In the paragraphs below, we present some numerical results obtained in our experiments.

Encoding. The description of a species in the library is about 200 kB and quite independent of the tree type. Less than one percent of the size is taken for the parameters of the system, the rest is needed for textures. As mentioned, the memory footprint of the lobe-based representation is quite small. For all species it is below 40 kB (see Table 2), allowing us to transmit many models per second over the internet.

Table 2: *Memory footprint and reconstruction time for different species. For similar size trees, each species has a typical number of lobes that represent each model.*

Species	# of Lobes	Model Size	Time Reconst.	Time LOD
Mahogany	29	15 kB	7 ms	1 ms
Bischofia polycarpa	31	30 kB	9 ms	2 ms
Delonix	39	39 kB	3 ms	0.5 ms
Lagerstroemia	24	22 kB	9 ms	1.5 ms
Ailanthus altissima	24	30 kB	12 ms	2 ms
Palm	1	3 kB	4 ms	0.5 ms
Terminalia	80	19 kB	11 ms	2.5 ms
Pine	86	20 kB	18 ms	4 ms
Ficus Virens	72	23 kB	16 ms	3 ms
Willow	9	5 kB	8 ms	0.5 ms

Decoding. The reconstruction of the geometry from the model is also very efficient. Column 4 of Table 2 shows the time for a full reconstruction, which is between 500K triangles for the Delonix tree and 3M triangles for the Terminalia. However, it has to be noted that during an interactive session due to the LOD only a small portion of the triangles are shown for each frame. Even if the viewer is directly in front of a tree model, its distant parts are reduced by the LOD rendering. This is why we show the times for reconstructing a full model (3 –18 ms) and the model with LOD enabled (1 – 4 ms). A consumer PC with GeForce GTX 480 graphics board was used for scene reconstruction.

Furthermore, in our implementation the tree models are not reconstructed for every frame but just updated, enabling the system to produce frame rates of about 20 – 30 fps with 40 different tree models and 10 – 15 fps with 250 trees in an urban scene as shown in the accompanying video. Figure 13 shows a screen shot of a scene to demonstrate the visual quality of the reconstructed LOD models.

Limitations. Our approach can reconstruct large sets of trees comprising a high variation of species and canopy shape. The pre-

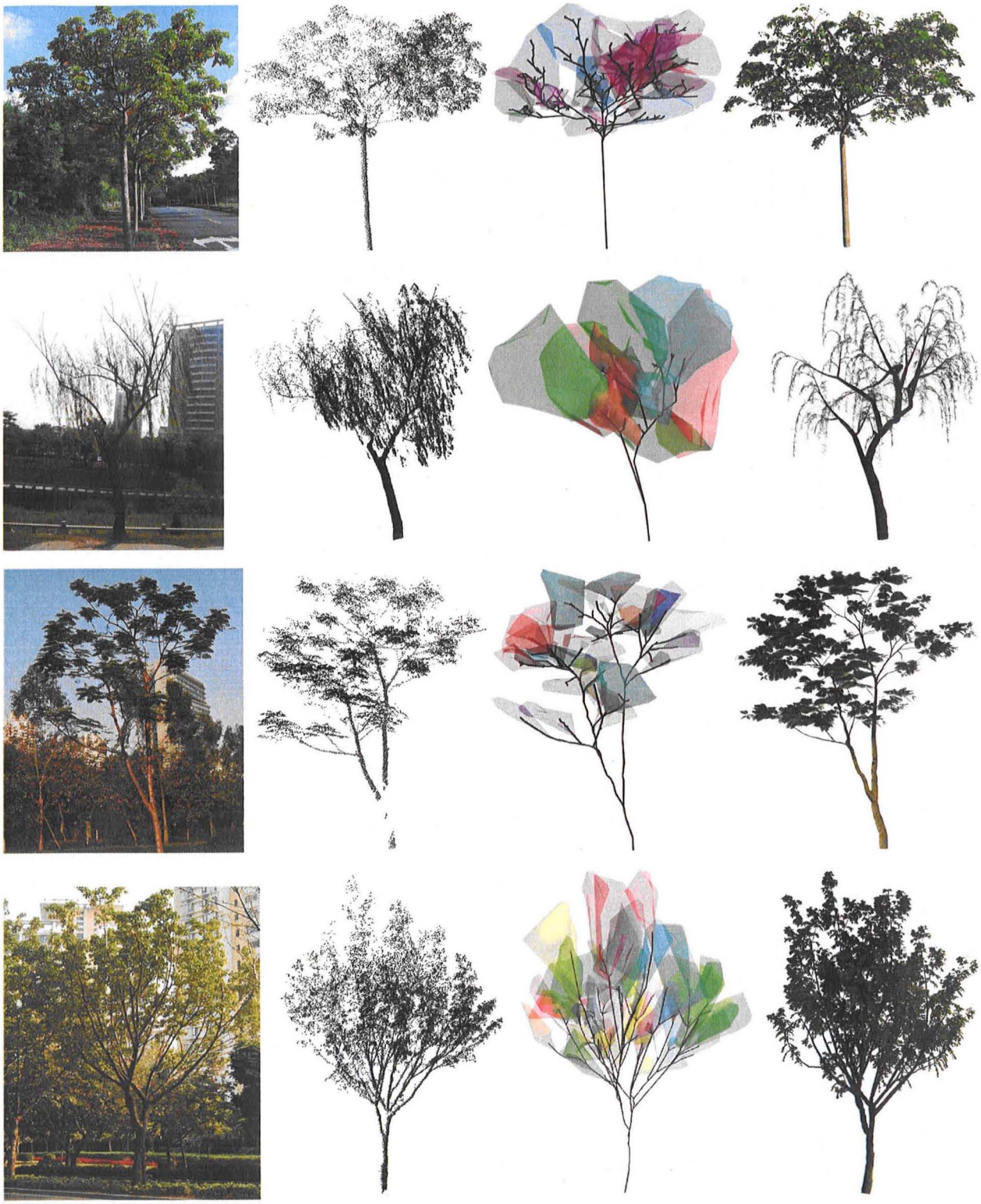

Figure 10: *Several results of reconstructed trees: Bischofia polycarpa, Willow, Delonix, Ficus Virens.*

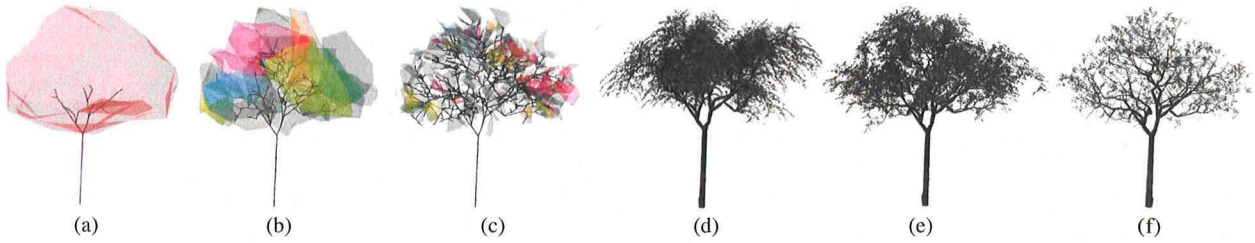

Figure 11: *Representation of a tree with a different number of lobes. If only one lobe is used (a)+(d), the representation of the tree canopy becomes unspecific and texturing fails to fill the large lobe completely,. It is designed for creating small random twigs and cannot build up large structures. With an appropriate number of lobes (b)+(e) details are represented and the overall shape is textured. If many are used, the occupied volume of the lobes becomes insignificant such that the detailed structure in the crown area is insufficiently synthesized (c)+(f).*

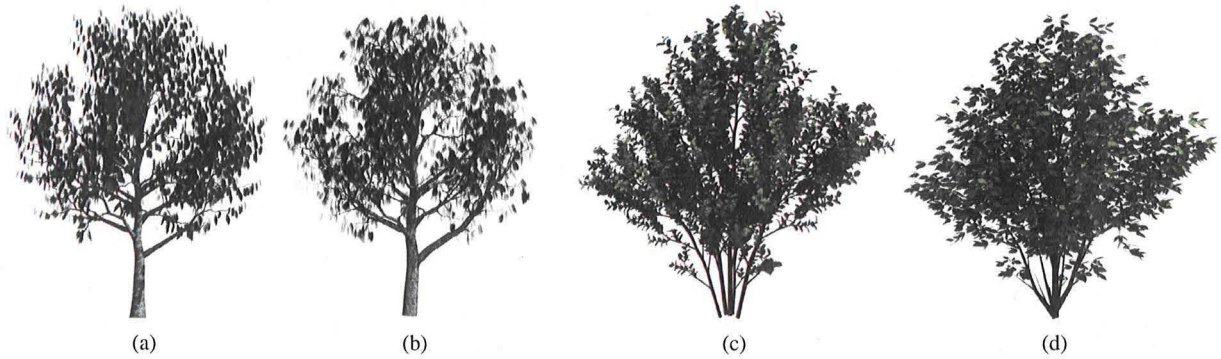

Figure 12: *Re-representation of two Xfrog plant models. For the two image pairs, the right image is a re-representation of the left one, achieving high fidelity, while significantly reducing representation size (60 kB instead of 56 MB for the tree, and 45 kB instead of 47 MB for the shrub).*

Figure 13: *A large scene shown in our interactive system. Thirty trees with the intermediate lobe-based representation (250kB) are synthesized on-the-fly with a total geometry of over 40M triangles, achieving 25 fps on a standard PC using NVidia GeForce GTX 480 graphics board.*

sented lobe-based representation enables trees to be stored, transmitted and rendered very efficiently, while maintaining a high degree of visual complexity. The lobe-based representation has however, some limitations caused by relying on the quality of a scanned point set. Some trees have dense foliage (e.g. Pine trees) and thus can only be scanned from a single side. In such a case, the point set lacks sufficient quality for the back parts, which results in an unbalanced canopy shape of the point cloud. The lobe-based representation reconstructs the canopy shape even in such a situation but discards the inner structure of tiny branches. The difference becomes most obvious when the canopy consists of long and sparse twigs (see Willow in Figure 10).

Figure 14: *Reconstruction of three tree models with special forms, the lobe-based representation adapts automatically to these situations.*

8 Conclusions

We presented a modeling paradigm for trees, which represents a given model by two types of entities: an explicit representation of the main branching structure in form of a graph with allometric parameters and an implicit representation of the finer branching details in biomass clusters by a set of lobes that are textured with predefined patches at rendering time. Using species information, the models can be produced automatically, and even more faithfully, from given point sets. This enables us to deal with scans of large areas and create models automatically for interactive visualization. We demonstrated this with our urban acquisition application.

The key to the success of our method is the classification of trees and the pre-generation of species information. In the future we want to extend our approach to a much larger number of trees allowing us to process most of the trees found in urban areas. Subsequently, classification would have to be adapted. An important extension would be animated models that are also able to display growth. Since the lobes in our model describe biomass clusters we are certain that there is a more compact implicit representation of the lobes, in comparison to using α-Shapes, which would be able to produce geometry fast. This would enable us to reduce our representation even further.

Another aspect that we have not realized yet is editing. The lobe-based representation allows the user to easily modify a tree model by simply redefining the lobes. The explicit graph can be changed, lobes either be merged or split and additionally, textures can be modified.

9 Acknowledgements

We thank the anonymous reviewers and Yiorgos Chrysanthou for their valuable suggestions. This work was supported in part by NSFC under Grant Nos. 60902104, 61025012, 61003190, CAS "One Hundred Scholar" Program, CAS Visiting Professorship for Senior International Scientists, CAS Fellowship for Young International Scientists, Shenzhen Science and Technology Foundation (JC201005270340A) , by the DFG Research Training Group GK-1042 "Explorative Analysis and Visualization of Large Information Spaces", University of Konstanz, and by Israel Science Foundation.

References

AGARWAL, G., BELHUMEUR, P., FEINER, S., JACOBS, D., KRESS, W., RAMAMOORTHI, R., BOURG, N., DIXIT, N., LING, H., MAHAJAN, D., ET AL. 2006. First steps toward an electronic field guide for plants. *Taxon 55*, 3, 597–610.

ANASTACIO, F., SOUSA, M. C., SAMAVATI, F., AND JORGE, J. A. 2006. Modeling plant structures using concept sketches. In *NPAR '06*, 105–113.

BENES, B., STAVA, O., MECH, R., AND MILLER, G. 2011. Guided procedural modeling. *Comput. Graph. Forum 30*, 2.

BLOOMENTHAL, J. 1985. Modeling the mighty maple. *SIGGRAPH '85 19*, 3, 305–311.

BUCKSCH, A., AND LINDENBERGH, R. 2008. Campino – a skeletonization method for point cloud processing. *ISPRS journal of photogrammetry and remote sensing 63*, 1, 115–127.

BUCKSCH, A., LINDENBERGH, R., AND MENENTI, M. 2009. Skeltre - fast skeletonisation for imperfect point cloud data of botanic trees. In *EG Workshop on 3D Object Retrieval*, 13–27.

CHEN, X., NEUBERT, B., XU, Y.-Q., DEUSSEN, O., AND KANG, S. B. 2008. Sketch-based tree modeling using markov random field. *ACM Trans. Graph. 27*, 5, 109–117.

CHENG, Z., ZHANG, X., AND CHEN, B. 2007. Simple reconstruction of tree branches from a single range image. *J. Comput. Sci. Technol. 22*, 6, 846–858.

COOK, R. L., HALSTEAD, J., PLANCK, M., AND RYU, D. 2007. Stochastic simplification of aggregate detail. *ACM Trans. Graph. 26*, 3, 79.

CÔTÉ, J.-F., WIDLOWSKI, J.-L., FOURNIER, R. A., AND VERSTRAETE, M. M. 2009. The structural and radiative consistency of three-dimensional tree reconstructions from terrestrial lidar. *Remote Sensing of Environment 113*, 5, 1067 – 1081.

DEUSSEN, O., AND LINTERMANN, B. 2005. *Digital Design of Nature: Computer Generated Plants and Organics*. Springer-Verlag New York, Inc.

DEUSSEN, O., COLDITZ, C., STAMMINGER, M., AND DRETTAKIS, G. 2002. Interactive visualization of complex plant ecosystems. In *Visualization '02*, 219–226.

EDELSBRUNNER, H., AND MÜCKE, E. P. 1994. Three-dimensional alpha shapes. *ACM Trans. Graph. 13*, 1, 43–72.

EFROS, A. A., AND FREEMAN, W. T. 2001. Image quilting for texture synthesis and transfer. In *SIGGRAPH '01*, 341–346.

GREENE, N. 1989. Voxel space automata: modeling with stochastic growth processes in voxel space. *SIGGRAPH '89*, 175–184.

HONDA, H. 1971. Description of the form of trees by the parameters of the tree-like body: Effects of the branching angle and the branch length on the shape of the tree-like body. *Theoretical Biology 31*, 331–338.

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

JACCARD, P. 1913. Eine neue auffassung ber die ursachen des dickenwachstums der bŁume. *Naturwiss. Z. fr. Landwirtschaft*, 13, 321–360.

KEY, T., WARNER, T., MCGRAW, J., AND FAJVAN, M. 2001. A Comparison of Multispectral and Multitemporal Information in High Spatial Resolution Imagery for Classification of Individual Tree Species in a Temperate Hardwood Forest. *Remote Sensing of Environment 75*, 1, 100–112.

LIVNY, Y., YAN, F., OLSON, M., CHEN, B., ZHANG, H., AND EL-SANA, J. 2010. Automatic reconstruction of tree skeletal structures from point clouds. *ACM Trans. Graph. 29*, 6, 151.

NEUBERT, B., FRANKEN, T., AND DEUSSEN, O. 2007. Approximate image-based tree-modeling using particle flows. *ACM Trans. Graph. 26*, 3, Article 71, 8 pages.

OKABE, M., OWADA, S., AND IGARASHI, T. 2006. Interactive design of botanical trees using freehand sketches and example-based editing. *Comput. Graph. Forum 24*, 3, 487–496.

PALUBICKI, W., HOREL, K., LONGAY, S., RUNIONS, A., LANE, B., MĚCH, R., AND PRUSINKIEWICZ, P. 2009. Self-organizing tree models for image synthesis. *ACM Trans. Graph. 28*, 58.

PRUSINKIEWICZ, P., AND LINDENMAYER, A. 1990. *The algorithmic beauty of plants*. Springer-Verlag New York, Inc.

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

REEVES, W. T., AND BLAU, R. 1985. Approximate and probabilistic algorithms for shading and rendering structured particle systems. *SIGGRAPH '85 19*, 3, 313–322.

RUNIONS, A., LANE, B., AND PRUSINKIEWICZ, P. 2007. Modeling trees with a space colonization algorithm. In *Proceedings of Eurographics Workshop on Natural Phenomena 2007*, 63–70.

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

STAVA, O., BENES, B., MECH, R., ALIAGA, D., AND KRISTOF, P. 2010. Inverse procedural modeling by automatic generation of l-systems. *Comput. Graph. Forum 29*, 2.

TALTON, J., LOU, Y., LESSER, S., DUKE, J., MĚCH, R., AND KOLTUN, V. 2011. Metropolis procedural modeling. *ACM Trans. Graphics 30*, 2.

TAN, P., ZENG, G., WANG, J., KANG, S. B., AND QUAN, L. 2007. Image-based tree modeling. *ACM Trans. Graph. 26*, 3.

TAN, P., FANG, T., XIAO, J., ZHAO, P., AND QUAN, L. 2008. Single image tree modeling. *ACM Trans. Graph. 27*, 5, 108.

TORRALBA, A., MURPHY, K. P., AND FREEMAN, W. T. 2007. Sharing visual features for multiclass and multiview object detection. *IEEE Trans. Pattern Anal. Mach. Intell. 29*, 854–869.

VERROUST, A., AND LAZARUS, F. 1999. Extracting skeletal curves from 3D scattered data. In *Proc. IEEE Conf. on Shape Modeling and Applications*, 194–201.

WITHER, J., BOUDON, F., CANI, M.-P., AND GODIN, C. 2009. Structure from silhouettes: a new paradigm for fast sketch-based design of trees. *Comput. Graph. Forum 28*, 2, 541–550.

XU, H., GOSSETT, N., AND CHEN, B. 2007. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Trans. Graph. 26*, 4, Article 19, 13 pages.

ZHU, C., ZHANG, X., HUAND, B., AND JAEGER, M. 2008. Reconstruction of tree crown shape from scanned data. *Technologies for E-Learning and Digital Entertainment*, 745–756.

A Used L-systems for Patches

Different species have different structural properties such as branching patterns and leaf arrangements. We used a relatively simple parameterized L-system with some rules to model the patches for each species. For a seed s with given thickness t and orientation v the system operate as follows (parameters see below):

```
Seed(t,v) -> Segment(t,v)*
Segment(t,v)
-> Edge(t,#Allometric)NewLeaf(v'')
    Transf(v,#Phototropism,#Gravitropism)
    NewSeed(t',v')
NewSeed(t,v)
-> if PlaceSeed(#InitThickness ,#NumTwigs)
    Seed(t,v)
NewLeaf(v)
-> if (PlaceLeaf(#NumLeaves)) Leaf(v)
```

The first rule creates a 3D chain of edges (number=#ChainPts) where each segment is transformed with respect to the preceding segment by a transformation that takes into account the orientation and the tropisms. The parameter "#Allometric" specifies how strong the branch reduces in diameter along the axis. "PlaceSeed" determines if a new seed point (docking position) will be added based on the branching characteristics and current thickness. The new orientation and thickness are computed from the parameters "#BranchingAngle", "#SprialAngle".

"PlaceSeed" determines if a new leaf will be added based on the plant characteristics (#NumLeaves). The leaf orientation is computed from the parameters "#LeafAngle", "#SpiralAngle' and orientation. "Leaf" produces a new leaf.

Our system uses species profiles that parameterize various botanical behaviors of complete branchlets such as allometric values, tropisms, branching angles, and leaf arrangement. The profiles (Lagerstroemia is given below) are stored in the species library.

```
- Branch modeling
  #ChainPts - 15
  #Phototropism - 0.2
  #Grapitropism - 0.3
  #Allometric - 1.2
- Leaf modeling
  #Phototropism - 0.2
  #Grapitropism - 0.7
  #FaceUp - 0.9
- Branching control
  #BranchDistr - 1 : 1
  #NumTwigs - 1
  #BranchingAngle - 30
  #SpiralAngle - 180
  #InitThickness - 1.0
- Leaf control
  #LeafDistr - 1 : 2
  #NumLeaves - 2
  #LeafAngle - 40
  #SpiralAngle - 120
  #OrientUp - 0.8
```