Erschienen in: ACM Transactions on Graphics (TOG) ; 26 (2007), 3. - 88
https://dx.doi.org/10.1145/1276377.1276487

Approximate Image-Based Tree-Modeling using Particle Flows

Boris Neubert Thomas Franken Oliver Deussen
University of Konstanz

Figure 1: A tree is modeled using a set of input photographs. We show some examples of input and resulting 3D tree models. If image information is not available, e.g. the foliage is missing, the user is able to sketch it (right). The models approximate the input images while forming botanically plausible branching structures.

Abstract

We present a method for producing 3D tree models from input photographs with only limited user intervention. An approximate voxel-based tree volume is estimated using image information. The density values of the voxels are used to produce initial positions for a set of particles. Performing a 3D flow simulation, the particles are traced downwards to the tree basis and are combined to form twigs and branches. If possible, the trunk and the first-order branches are determined in the input photographs and are used as attractors for particle simulation. The geometry of the tree skeleton is produced using botanical rules for branch thicknesses and branching angles. Finally, leaves are added. Different initial seeds for particle simulation lead to a variety, yet similar-looking branching structures for a single set of photographs.

Keywords: Image-Based Modeling, Plant models, Botanics

1 Introduction

Modeling complex botanical tree geometry has posed a challenge for computer graphics for decades. Beginning with abstract branching structures, the complexity and visual appearance has been enhanced over the years in such a way that today many tree models appear photo-realistic to us. However, creating this type of models is still cumbersome. To mimic a specific tree or a given tree shape, usually many parameters must be manually adjusted. Image-based modeling methods try to overcome this problem by using a set of photographs to create the geometry directly.

In recent years some techniques have been published that try to create exact 3D representations for given trees. In contrast to these methods our approach produces qualified approximations. Thus, we avoid registration and many numerical problems while still achieving plausible branching structures. The models still show differences to the input; however, we are able to create a variety of similar models easily by changing parameters in our system.

The input is a small set of photographs of a tree taken from different views. Usually two images are sufficient for a good approximation. In contrast to other approaches, we do not need an exact registration of these images and interactively arrange them in our system. Since the images typically contain unwanted background we separate the trees using common algorithms for alpha matting.

The general idea of our approach is to combine a bottom-up construction with internal and external constraints. We compute a voxel-model of the tree volume with each voxel containing a density estimate of the trees biomass. Proportional to this density particles are produced and traced downwards to the tree basis using a 3D flow simulation. Simple rules direct and force them to form twigs and subsequent branches. This already creates nice-looking tree skeletons and, by later adding leaves, complete trees.

To achieve models that are similar to the given input photographs, particle simulation is directed by the main branching structures in the input images. Therefore, if possible, we extract the trunk and the main branches from the photographs and construct two-dimensional attractor graphs for each input image. These attractor graphs are combined to influence the 3D particle simulation by modifying their directions. Subsequently, a triangular mesh is built around the resulting 3D graph using allometric rules. While we primarily concentrated on automatic image-based construction, it is also possible to interactively guide the method to a desired result. By painting densities and by changing directions for particle simulation, the produced geometry can be modified in various ways without the need to adjust many parameters.

2 Previous Work

Classical tree modeling is rule-based or uses modeling procedures. While in L-Systems [Prusinkiewicz and Lindenmayer 1990] formal rules are applied to an initial state, most procedural approaches use parameterized algorithms [Oppenheimer 1986; de Reffye et al. 1988; Holton 1994; Weber and Penn 1995]. These algorithms also encode rules, but in a more specific notion. The xfrog system [Lintermann and Deussen 1999] tries to combine both approaches.

In a classical L-System, the rule basis has to be written by the user. Since rules work locally, small changes of values might cause large changes in the overall shape. Such behavior makes modeling quite cumbersome. Various extensions of L-Systems have been proposed since then, e.g. parametric [Prusinkiewicz and Lindenmayer 1990], open [Měch and Prusinkiewicz 1996] and differential L-Systems [Prusinkiewicz et al. 1993]. These extensions are able to create a variety of effects, but also result in additional parameters. Prusinkiewicz *et al.*[2002] present a modeling interface for L-Systems to enhance the modeling ease, but still a large set of parameters has to be defined by the user.

Procedural approaches are usually restricted to produce a limited number of forms. They are also able to limit the amount of adjustable parameters for the user. However, with increasing model complexity, this amount also increases. While Oppenheimer [1986] used only some basic parameters, later approaches such as the one presented by Weber and Penn [1995] have dozens.

In the xfrog system, procedural elements are combined using a simple rule system, which allows faster modeling. However, the number of parameters is still large. Ijiri *et al.*[2005] use interactive editors based on botanical rules to create plant models. While these editors allow for an efficient production of flowers and phyllotaxis, the production of complex trees is not their strength. Okabe *et al.*[2005] present a sketch-based interface for trees. Here, the user draws the outline of a tree skeleton and its shape. However, again many parameters have to be adjusted to achieve specific species.

Image-based modeling: Shlyakter *et al.*[2001] direct the growth of an L-System by given photographs. A visual hull is reconstructed from the registered input images. The medial axis diagram of the hull is constructed and used as the tree skeleton. The smaller branches and leaves are added using an L-System. Our approach is different in several aspects: Instead of using the medial axis, we generate the tree skeleton using a particle simulation and are thereby able to introduce numerous constraints. In contrast to this approach, we estimate the density of the foliage also using the input photographs. Instead of several hours, our models are computed within seconds.

A very precise, though complex image-based modeling approach is described by Reche-Martinez *et al.*[2004]. In this case a set of carefully registered photographs is used to determine the volumetric shape of a given tree. The volume is divided into cells; for each cell a valid visual representation is computed by a set of textures. The complete set of textures represents the tree quite faithfully. However, a large amount of texture space in the order of tens of megabytes is needed. Also, it is not easy to show the tree under various lighting conditions since the lighting is already incorporated into the textures. In our approach we avoid this by creating a 3D surface model using the images. Also we do not need an exact registration of the input photographs and instead create an approximate shape of the given tree.

Another image-based method for modeling smaller plants was presented by Quan *et al.*[2006]. Here, an image sequence is recorded and the model is computed from these images in a semi-automatic way. Although the results are of high quality, the amount of required user input prevents modeling complex objects where such a method would be particularly interesting.

An important inspiration for our approach are particle simulations. Reeves [Reeves 1983; Reeves and Blau 1985] used them for producing trees. They were also utilized for creating other ramified patterns such as river beds or lightning [Viennot et al. 1989; Ebert et al. 2003]. Rodkaew *et al.*[2003] describe an automatic modeling method that creates a branching skeleton from given random leaf positions. Particle simulation is used to trace particles from the leaf positions downwards to the base of the tree. This way it is possible to yield realistic looking tree models. The authors also perform particle simulation for the creation of vessels in leaves, which results in nice images; however, is computationally quite expensive. Furthermore, the resulting branching structures depend strongly on the initial positions of the particles in the simulation. In our approach we modify the particle traces to create results that are similar to the given images. This is done by extending the particle simulation by direction fields computed from the input images. Additionally, for the branches biological constraints are imposed, i.e. we are able to specify the branching angle or the direction in which branches primarily grow.

3 Overview

Our approach therefore can be seen as particle simulation with external constraints from input images and internal botanical restrictions. It can be divided into five steps, which are computed one after the other. The outline below also reflects the further structure of the paper:

1. **Pre-processing:** For each given input image the tree is separated from the background and a 2D attractor graph is computed.
2. **Creation of the voxel model:** By back-projecting to the input images, we fill a voxel-grid with density values. The values of the voxels are an estimate of the tree density.
3. **Computation of direction fields:** To incorporate information of the input images, we use the 2D attractor graphs to create direction fields. These fields provide direction vectors for each input image plane. Combining these vectors for all image planes yields vectors that direct the particles in the subsequent 3D flow simulation.
4. **Particle simulation:** In proportion to the density values we produce random initial positions for particle flow simulation. The traces of the particles are influenced by forces of neighbors and by the direction fields. As a result we obtain the main tree skeleton in form of a 3D graph.
5. **Production of geometry:** The tree skeleton is now converted into 3D geometry using allometric rules. Tiny branches and leaves are added and create the final foliage.

4 Pre-processing

Usually our input images contain background objects. Therefore, we initially need to separate the tree from the background. This is particularly complicated for natural objects with many holes. Fortunately, in recent years a number of methods for performing alpha matting have been published [Ruzon and Tomasi 2000; Pérez et al. 2003; Sun et al. 2004]. Usually the methods are not fully automatic. A common setup lets the user create an initial trimap that specifies the pixels of the object, pixels of the background and an uncertainty region. In our case this is done by selecting pixels with ap-

propriate colors in the input images. The separation algorithm fills the uncertain regions with either opaque (foreground), transparent (background) or partly transparent pixels based on an interpolation, which takes into account the image gradients. We interpret the result from the matting procedure as described by Sun *et al.*[2004] as a density estimation of the tree for the corresponding view (see Figure 2(b)).

In a second step, the images are used to sketch an estimate of the underlying tree skeleton in the corresponding view. We use the Livewire approach [Chodorowski et al. 2005] for this purpose. The algorithm needs a target point – usually the foot point of the tree or the position of the first branching on the trunk (see Fig. 2(b)). Additionally, seed points for the branches have to be determined. This can be done automatically by randomly selecting points on the tree silhouette or by manually introducing seed points which usually creates better results.

(a) (b)

Figure 2: (a) Input image; (b) tree density estimation with corresponding attractor graph.

The attractor graph is now computed with every branching forming a node. The algorithm starts at each seed point and finds a path through the visual structures of the input image to reach the target point (this is in fact another kind of particle simulation). The result reflects the main branching structure of the tree in the image. This is done for all photographs, we call the resulting graphs *attractor graphs*, see Figure 2(b). If no information is contained in the image, an arbitrary skeleton is produced or the user has to draw it manually.

5 Computing the Tree Density

For now let us assume the camera model of the input images to be a parallel projection and to have two input images at a right angle. In the previous step we obtained the alpha values of the input photographs, now we construct an initial 3D estimation of the plant volume that encompasses a voxel grid. Initial density values for the voxels are estimated from the input photographs. A discretized version of the volume rendering equation allows us to compute the desired solution. Solving for a least-squares solution results in a refined density volume that is later used for the branching structure and as a bounding volume for the final foliage. For each voxel V_i in the grid we assign a density value α_i. After initializing the values we use an iterative algorithm to refine them. In the final density distribution, voxels with higher α values will belong to parts of the plant that contain many leaves or branches.

A well known method to generate images from volumetric representations is found in [Sabella 1988; Max 1995] using a discrete form of the volume rendering equation for an emission-absorbtion model without scattering:

$$I(s_n) \quad = \quad \sum_{k=0}^{n} b_k \prod_{j=k+1}^{n} \theta_j \tag{1}$$

The value θ_j is the transparency of voxel j and b_k is the light emitted from the k-th voxel.

Considering the given input images as a solution for Eq. 1, it is possible to reconstruct the density values of the volume grid V. Reche-Martinez *et al.*[2004] use a similar approach to reconstruct the volume model for their image-based rendering algorithm and rely on the same assumptions. However, it is important to note that in our case an accurate reconstruction is not needed as we do not render images directly from the volumetric reconstruction. Instead we use the density information to subsequently produce the density of a surface representation of the plant. This allows us to apply a simple reconstruction method and a relatively coarse grid with typically $25 \times 25 \times 25$ voxels.

To initialize the density α_i for the i-th voxel V_i we project the voxel back onto each input image. We determine the average density of the projected area and use the minimum value as an initialization for α_i. Note, that the resolution of the voxel grid is typically much coarser than the input images we use, so many pixels are combined for one projected area. This initial value would only be correct if all other contributing voxels had zero density and therefore is an upper limit for the transparency. Voxels with initial density values below a predefined threshold are rejected and not considered in the following steps. This can be seen as implicit space carving and reduces the complexity in the following steps considerably without reducing the quality of the resulting density estimate.

We solve the resulting system of linear equations in an iterative way similar to Reche-Martinez *et al.*[2004]. In Figure 3(a) a set of three input density images is shown. The initial alpha values of the voxel grid can be seen in Figure 3(b), the least squares solution of the equation system is shown in Figure 3(c). In the solution we consider negative results as to be empty voxels. In subfigure (d) the computed density values for one image plane are shown. Their values are indicated by the size of grey squares.

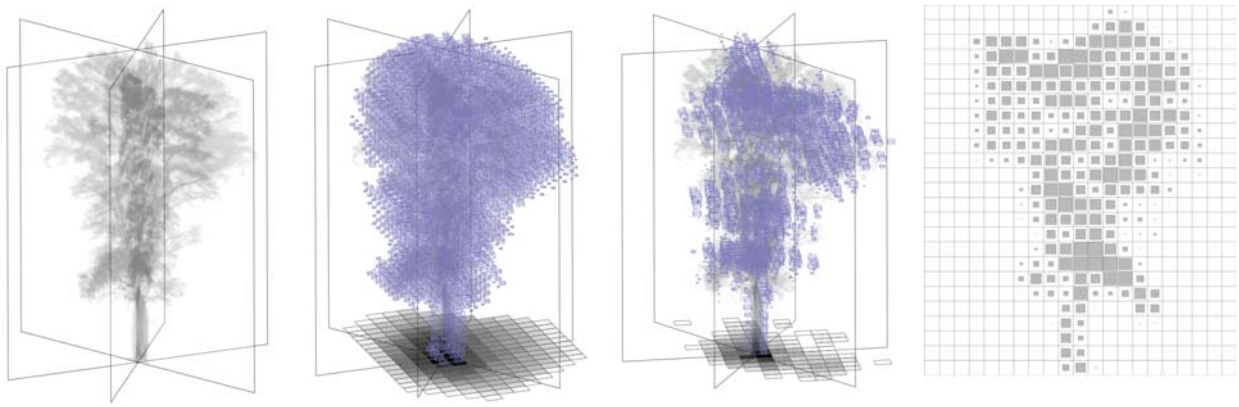
6 Particle Tracing

Using the density values we create initial particle positions for the main tree skeleton. The particles are placed randomly in the voxels in proportion to their density. Since we do not have specific botanic measurements about the total number of branches in different tree species, we use heuristics that are manually adapted if necessary. For medium-sized trees we use between 500 and 1000 particles, for large models between 1000 and 2000.

As mentioned above, we extend the method by Rodkaew *et al.*[2003] in order to achieve specific tree skeletons and shapes according to our input photographs. This is done by introducing the attractor graphs to particle tracing and by establishing additional rules. First, we would like to describe the general simulation followed by our refinements.

A particle p_i is represented by a position x_i, velocity x_i', and mass m_i. It moves under the influence of time-dependent forces represented by $f_i(x_i, x_i', t)$. The Newtonian law gives us $f_i = m_i \cdot x_i''$ and $x_i'' = f_i/m_i$ which is a well studied differential equation of second order [Hockney and Eastwood 1988; Witkin and Baraff 1997] and usually written in the form of coupled first order differential equations:

$$[x_i, x_i'] = [x_i', f_i/m_i] \tag{2}$$



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

Figure 3: Estimating the tree density: (a) Initial density values for three input images; (b) voxel grid by back-projection; (c) refined voxel grid; (d) density values for one image plane. High density values are marked by large squares.

This system can be iteratively solved using an explicit Runge-Kutta method as found in Press et al. [1992, p. 707ff.]. During simulation the positions of the particles are updated according to their mass and external forces. The mass can be seen as some kind of reluctancy of a particle to change the current orientation and/or velocity.

One important aspect of the force is to implement the particle attraction. This attraction is needed to form the tree skeleton. Particles close to each other are forced to join and subsequently move forward together. This is implemented by searching the nearest neighbor for each particle and combining them if their distance is below a given threshold. The basic particle tracing mechanism can be written as follows:

Procedure `ParticleTrace`

```
Initialize particle positions according to voxel density
while particles not at tree basis do
    forall particles p_i do
        determine forces f_i
        determine velocities and positions (Eq. 2)
    determine nearest neighbor and join if close enough
```

The particle traces are stored as a 3D branching graph, which describes the main tree skeleton. This basic simulation is now refined in order to match our modeling requirements.

7 The Direction Field

The simulation so far combines the particle traces to small branches that merge and form the skeleton. The particles are directed towards the tree basis and towards their respective nearest neighbors. Additionally, for each 2D attractor graph from the input images (see Section 4) we create a two-dimensional discrete vector field in the according image plane. Our goal is to direct the particles in such a way that they simultaneously move towards the attractor graphs in any of the image planes that correspond to these input images.

We project the particle position into each of the image planes using the respective projection. Then, we determine the direction vector from the vector fields in the global coordinate system. With two input images at a right angle, we would obtain two vectors perpendicular to each other. Generally speaking, we get a set of direction vectors for which we compute the average. This vector, when being projected back to the image planes, is the average direction and is used as part of the external force that is applied to the particle. Theoretically there is a chance that all direction vectors might sum up to zero, however our practical experiments have never supported such a case.

The direction field is computed by applying a distance transform to the attractor graph. For each position $x_{i,j}$ in the direction field the closest point on the attractor graph $g_{i,j}$ is computed, let $\mathbf{v}_{i,j} = g_{i,j} - x_{i,j}$ be the vector pointing towards the $g_{i,j}$. Additionally we compute the tangential vector of the graph $\mathbf{t}_{i,k}$ at position $g_{i,j}$.

Normalized versions of these two vectors, $\bar{\mathbf{v}}_{i,j}$ and $\bar{\mathbf{t}}_{i,j}$, rsp., are used for computing the forces on a particle close to $x_{i,j}$. We modeled this force as a weighted sum using the blending function $h(d)$ which depends on the distance $d = |\mathbf{v}_{i,j}|$ to the graph:

$$f_{i,j} = h(d) \cdot \bar{\mathbf{v}}_{i,j} + (1 - h(d)) \cdot \bar{\mathbf{t}}_{i,j} \tag{3}$$

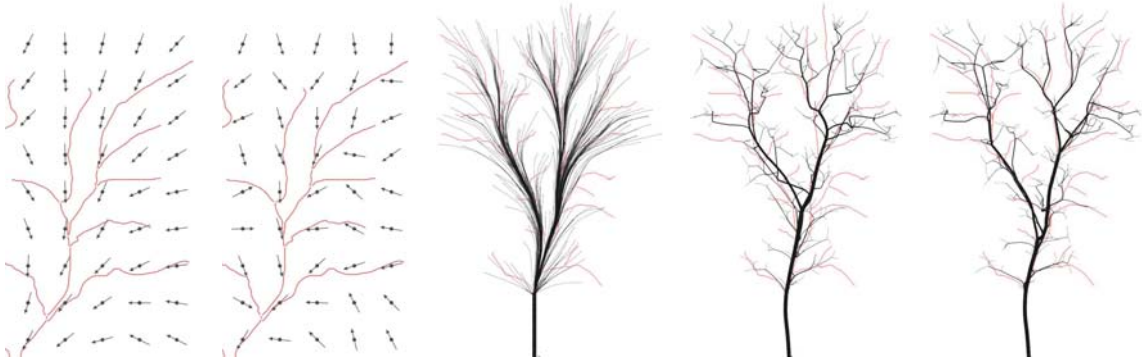
If the blending function is linear, particles far away from the graph are directed towards $g_{i,j}$ and particles close to the graph into the tangential direction $\mathbf{t}_{i,j}$ in $g_{i,j}$. If instead we use a constant function $h(d)$ for all $d > 0$, the particles are directed at a fixed angle towards the closest point $g_{i,j}$ on the nearest graph segment.

The direction vectors $f_{i,j}$ are computed for a two-dimensional grid which is embedded in each of the image planes. The resolution for these grids does not need to match the resolution of the voxel grid nor the resolution of the images. In our practical tests a grid of 100 by 100 proved to be sufficient.

In Figure 4 we demonstrate the particle flow for a two-dimensional case. Figure 4(a) shows a direction field for a linear blending function $h(d)$, in (b) for a constant function. Subfigure (c) shows the result for a flow simulation using the field of subfigure (a), but without attraction between particles. In (d) the field of (b) is used, this time including attraction. The branches now have a nearly fixed branching angle due to the given field. Finally, subfigure (e) shows particle flows with linear $h(d)$ and attraction, the mostly used case. In this case the graph follows the given direction field quite well.

8 Creating the Tree Geometry

So far we have created a 3D graph that represents the main branching structure of our desired model. For each of the segments within



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

Figure 4: (a) Direction field for a given graph and linear blending function $h(d)$, the vectors close to graph segments point down the graph; (b) for a constant blending function, the vectors point at a given angle towards the nearest segment; (c) result for linear blending without particle attraction; (d) result for constant blending including particle attraction; (e) result for linear blending including particle attraction.

the graph we store a number indicating how many particles were combined to form this segment. The graph is converted into geometry by assigning each branch a thickness and by introducing appropriate geometry to the branchings.

In our approach, we determine the thickness of the branches based on a rule that has already been discovered by Da Vinci in the 16th century and holds for many trees (see [Deussen and Lintermann 2005, pp. 32ff]). It was introduced to computer graphics by Holton [1994]. The rule specifies the relation between the diameter of a branch and the diameters of its children:

$$r^2 = a \cdot \sum r_i^2$$

where r is the radius of the main branch, r_i are the radii of the branching twigs, a is a constant. We compute the thickness of a branch by using the stored number of particles. They reflect the overall number of sub-branches and therefore can be used to determine the cross-section [Holton 1994]. The branch geometry is now created by connecting discs of the required thickness that are positioned along the branch in certain distances and that are oriented perpendicular to the branch. These discs are triangulated and textured.

(a) (b)

Figure 6: (a) Creation of the tiny twigs; (b) leaf primitives.

The process allows us to create convincing structures of several hundred branches. However, the geometric complexity of natural trees consists in large part of tiny branches and twigs that branch from larger ones in a more regular way. Instead of creating these tiny branches using our global particle simulation we decided to give the user the opportunity to gain finer control over the appearance of these branches. This control can also be achieved using a sketch based system as presented by Okabe *et al.*[2005], adapted in a way that the user sketches examples only for higher branching levels that are subsequently applied to the whole plant or using a parameterized approach that allows us to incorporate further botanic laws. One of these laws is the Golden Section placement rule, which arranges leaves at a deviation angle about 137.5° along a twig; other prominent angles are 90°, 180°. For many species plagiophototropism directs the leaves perpendicular to the sunlight [Deussen and Lintermann 2005, pp. 24ff] (see Fig. 6(a)). The parameters that are interactively chosen by the user during this final modeling step are length, rigidity, and frequency of the small twigs, leaf size and texture.

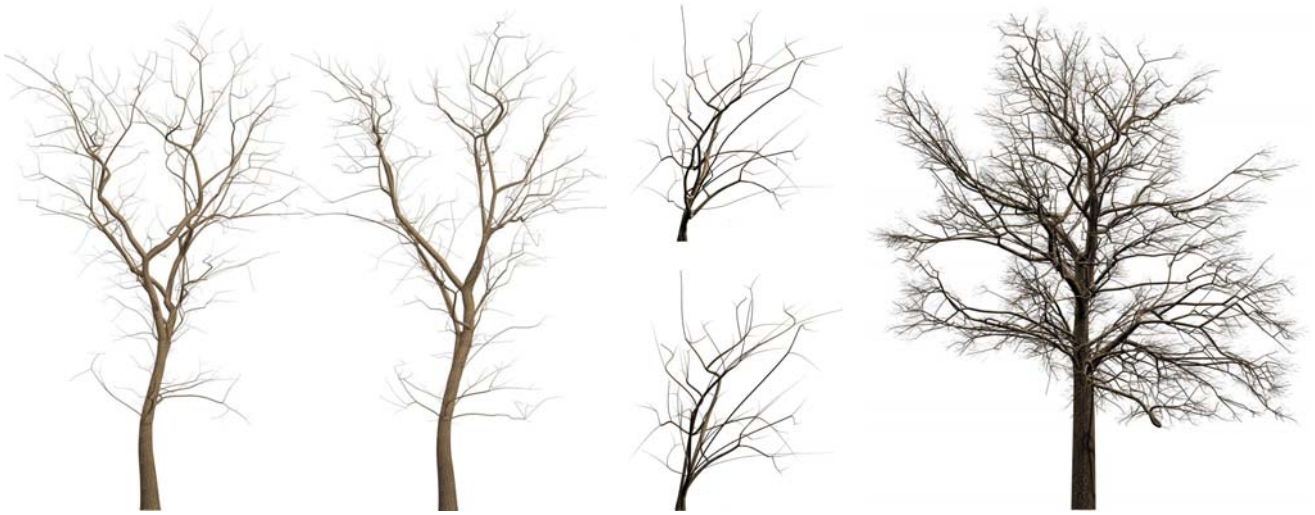
To create the foliage texture we use photographs of natural leaves, add an alpha buffer and use a quad or a small set of triangles to support the texture (see Fig. 6(b)). The positions of the leaves are determined by the density values of the voxels resulting in models where parts with high density values will contain many leaves. For trees as in Figure 2 the leaves appear in spatially clustered regions that we cannot represent with our course grid - therefore we refine the leaf arrangement by projecting the position of each leaf to the image planes and discarding leaves that are projected to empty or very sparse regions.

9 Results and Discussion

Figure 7 shows two tree models with their corresponding input photographs. Animations of the models can be found in the accompanying video. The geometric complexity of the models is 555,000 triangles for the first tree and 285,000 triangles for the second. In both cases the given shape and structure of the input is approximated quite faithfully by the models. In Figure 8 a pine tree is created from two photographs that show significant occlusion and other trees in the background.

The modeling works in most parts at interactive rates. The preprocessing separates the tree in the input images and creates the attractor graph interactively. The volume model is created within a few seconds. The flow simulation of the particles needs about 5–10 seconds for 1000–2000 particles and 200 iterations. The branch geometry can be produced within a matter of seconds and the leaves are also added within seconds even for complex models. All times were recorded on a standard PC with 3 GHz.

In many cases the trees in the images are partly occluded by other objects or have a shape that is deformed due to natural reasons. In



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

Figure 5: (a)-(b) Two results for a given direction field and different initial positions; (c) detailed view to chiseled and smoothed branches; (d) complete tree skeleton of an oak tree (see Figure 9).

this case it is easy to fix the problem in the input images by drawing and removing density. Another situation is due to the seasons. We had three photographs of an oak at winter time. So we created a tree skeleton (Figure 5) and added density to the images by simply painting some strokes. The result is a tree model that looks natural and reflects the sketched density by its branches and foliage.

9.1 Limitations of the Method

While creating nice, high quality models, our method has some limitations: As outlined above, the branching patterns at smaller branches are influenced not only by branching angles but also by specific patterns in which branches appear. Such branching patterns are hard to simulate using particle flows since in this case complicated interaction of particles during the flow simulation is needed or a post-processing step after simulation has to be performed. A moved branch might interact with other branches, or curvature might need to be adapted, etc. We are not able to solve this completely, however, to minimize these problems we introduced the above mentioned compound leaves: a number of simple leaves arranged procedurally on a small twig.

We were also not able to create all tree species as convincingly as others. Some trees change their shape and structure among branching levels. This is hard to simulate without creating different direction fields and particle rules for different branching levels. Also we found it hard to create very steep branching angles with our simulation since particles merge when getting too close.

10 Conclusions and Future Work

We have presented a new image-based modeling method for 3D tree geometry. By imposing image constraints in the form of captured branching patterns and density distributions we are able to adapt particle simulation to a given set of input images. This enables us to create convincing tree models that approximate a given shape. Using different initial positions for the particles, different but similar tree skeletons can be produced. Tiny twigs and leaves are added, the method runs at interactive rates and allows also for the modeling of trees by manually altering input images and direction fields for the flow simulation. We tested the approach with several sets of input images.

In the future we plan to couple the approach with Level-of-Detail data structures since the produced models are often too complex for interactive applications. Our branching graph already incorporates a hierarchy since it has larger and smaller branches. For distant models only the large branches might be shown together with an visual approximation of the foliage. Smaller branches and leaves might be generated on the fly when needed.

Acknowledgements

This work was supported in part by the BW-FIT research cluster “Gigapixel displays” as well as the Karl Steinbuch Foundation, the German Israel Foundation GIF and the DFG Graduiertenkolleg/1024 “Explorative Analysis and Visualization of Large Information Spaces” at the University of Konstanz.

References

CHODOROWSKI, A., MATTSSON, U., LANGILLE, M., AND HAMARNEH, G. 2005. Color lesion boundary detection using live wire. In *Proceedings of SPIE Medical Imaging: Image Processing vol. 5747*, 1589–1596.

DE REFFYE, P., EDELIN, C., FRANCON, J., JAEGER, M., AND PUECH, C. 1988. Plant models faithful to botanical structure and development. In *Computer Graphics (SIGGRAPH ’88 Proc.)*, J. Dill, Ed., vol. 22, ACM SIGGRAPH, 151–158.

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

EBERT, D., MUSGRAVE, K., PEACHEY, P., PERLIN, K., AND WORLEY, S. 2003. *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmman.

HOCKNEY, R. W., AND EASTWOOD, J. W. 1988. *Computer Simulation using Particles*. Taylor & Francis, Inc., Bristol, PA, USA.

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

Figure 7: One example of the created tree models: (a) input photographs; (b) 3D model in view corresponding to upper photograph; (c) view corresponding to lower photograph; (d) another view.

HOLTON, M. 1994. Strands, gravity and botanical tree imagery. *Computer Graphics Forum 13*, 1, 57–67.

IJIRI, T., OWADA, S., OKABE, M., AND IGARASHI, T. 2005. Floral diagrams and inflorescences: interactive flower modeling using botanical structural constraints. *ACM Transactions on Graphics 24*, 3, 720–726.

LINTERMANN, B., AND DEUSSEN, O. 1999. Interactive modeling of plants. *IEEE Computer Graphics and Applications 19*, 1, 56–65.

MAX, N. 1995. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 1*, 2, 99–108.

MĚCH, R., AND PRUSINKIEWICZ, P. 1996. Visual models of plants interacting with their environment. In *SIGGRAPH 96 Conf. Proc.*, ACM SIGGRAPH, 397–410.

OKABE, M., OWADA, S., AND IGARASHI, T. 2005. Interactive design of botanical trees using freehand sketches and example-based editing. *Computer Graphics Forum 24*, 3, 487–496.

OPPENHEIMER, P. 1986. Real time design and animation of fractal plants and trees. In *Computer Graphics (SIGGRAPH 86 Conf. Proc.)*, vol. 20, 55–64.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. *ACM Transactions on Graphics 22*, 3 (July), 313–318.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. 1992. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press.

PRUSINKIEWICZ, P., AND LINDENMAYER, A. 1990. *The Algorithmic Beauty of Plants*. Springer-Verlag, New York.

PRUSINKIEWICZ, P., HAMMEL, M. S., AND MJOLSNESS, E. 1993. Animation of plant development. *Computers Graphics (SIGGRAPH 93 Conf. Proc.)*, 351–360.

PRUSINKIEWICZ, P., MÜNDERMANN, L., KARWOWSKI, R., AND LANE, B. 2002. The use of positional information in the modelling of plants. In *SIGGRAPH 2001 Conf. Proc.*, 289–300.

QUAN, L., TAN, P., ZENG, G., YUAN, L., WANG, J., AND KANG, S. B. 2006. Image-based plant modeling. *ACM Transactions on Graphics 25*, 3 (July), 599–604.

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., AND BLAU, R. 1985. Approximate and probabilistic algorithms for shading and rendering structured particle systems. In *Computer Graphics (SIGGRAPH '85 Conf. Proc.)*, vol. 19, 313–322.

REEVES, W. 1983. Particle systems – a technique for modeling a class of fuzzy objects. *Computer Graphics 17*, 3, 359–376.

RODKAEW, Y., CHONGSTITVATANA, P., SIRIPANT, S., AND LURSINSAP, C. 2003. Particle systems for plant modeling. In *Plant Growth Modeling and Applications*, 210–217.

RUZON, M., AND TOMASI, C. 2000. Alpha estimation in natural images. In *IEEE Conference on Computer Vision and Pattern Recognition, Volume I*, 18–25.

SABELLA, P. 1988. A rendering algorithm for visualizing 3D scalar fields. In *Computer Graphics (SIGGRAPH 88 Conf. Proc.)*, 51–58.

SHLYAKHTER, I., ROZENOER, M., DORSEY, J., AND TELLER, S. 2001. Reconstructing 3D tree models from instrumented photographs. *IEEE Computer Graphics and Applications*, 53–61.

SUN, J., JIA, J., TANG, C., AND SHUM, H. 2004. Poisson matting. *ACM Transactions on Graphics 23*, 3 (July), 315–321.

VIENNOT, X., EYROLLES, G., JANEY, N., AND ARQUÉS, D. 1989. Combinatorial analysis of ramified patterns and computer imagery of trees. In *Computer Graphics (SIGGRAPH '89 Conf. Proc.)*, vol. 23, 31–40.

WEBER, J., AND PENN, J. 1995. Creation and rendering of realistic trees. In *SIGGRAPH 95 Conf. Proc.*, 119–128.

WITKIN, A., AND BARAFF, D., 1997. Physically based modeling: Principles and practice. Siggraph '97 Course notes.

Figure 8: Pine tree generated using our approach. Left: input photographs.

Figure 9: Sketch-based modeling of an oak. Left: input photographs; middle: sketched density; right: resulting model. Please note the hole in the left part of the tree that can also be found in the sketched density of the corresponding view. The skeleton of the oak is shown in Figure 5(d).