Inverse Procedural Modeling of Trees

O. Stava[1], S. Pirk[2], J. Kratt[2], B. Chen[3], R. Měch[1], O. Deussen[2], B. Benes[4]

[1]Adobe Systems Inc., USA, [2] University of Konstanz, Germany
[3]Shenzhen Institut of Advanced Technology, China, [4]Purdue University, USA

Abstract

Procedural tree models have been popular in computer graphics for their ability to generate a variety of output trees from a set of input parameters and to simulate plant interaction with the environment for a realistic placement of trees in virtual scenes. However, defining such models and their parameters is a difficult task. We propose an inverse modeling approach for stochastic trees that takes polygonal tree models as input and estimates the parameters of a procedural model so that it produces trees similar to the input. Our framework is based on a novel parametric model for tree generation and uses Monte Carlo Markov Chains to find the optimal set of parameters. We demonstrate our approach on a variety of input models obtained from different sources, such as interactive modeling systems, reconstructed scans of real trees, and developmental models.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling; Computer Graphics [I.3.6]: Methodology and TechniquesInteraction Techniques Simulation and Modeling [I.6.8]: Types of SimulationVisual

1. Introduction

Plant modeling approaches can be categorized into three principal classes: reconstructions from existing real world data, interactive modeling methods, and procedural or rule-based systems. The level of realism of models produced by these methods is very high; however, many open problems still prevail.

Trees created using the reconstruction and interactive methods are typically static and do not react to varying environments. In contrast, procedural methods generate a wide variety of plants that can be sensitive to the environment and adapt their shape to changing conditions. However, the controllability of procedural models is low and is typically achieved by creating a set of input parameters, which is often a cumbersome manual process. This problem becomes even more severe with the increasing complexity of procedural models needed today for the creation of large realistic scenes. This drawback of procedural modeling can be alleviated by solving the inverse problem, i.e., by automatically computing the parameters of a procedural model for a given tree. Once these parameters are determined, the tree could be reconstructed and simultaneously enable all the advantages of procedural modeling for application to the production process. Recently, various inverse modeling approaches have been introduced. However, to the best of our knowledge, finding the parameters of a given stochastic procedural model of biological trees has not been addressed by the computer graphics community. We introduce a framework for stochastic inverse procedural modeling of biological trees that offers three main contributions:

- a new compact parametric procedural model that describes a wide variety of trees,
- an efficient similarity computation for the comparison of two trees, and
- a method for the automatic determination of input parameters of the procedural model for a given input tree.

As shown in Figure 1, the input of our framework is a single 3D polygonal tree model (or a set of such models). Our framework reproduces the visual appearance of the input as closely as possible using Monte Carlo Markov Chain (MCMC) optimization on the input parameters.

The input is approximated by the resulting procedural model with its set of input parameters. We demonstrate our framework on various tree models and show how their procedural representation can be used to generate more trees of the same species, trees of different ages, or trees grown under different environmental conditions.

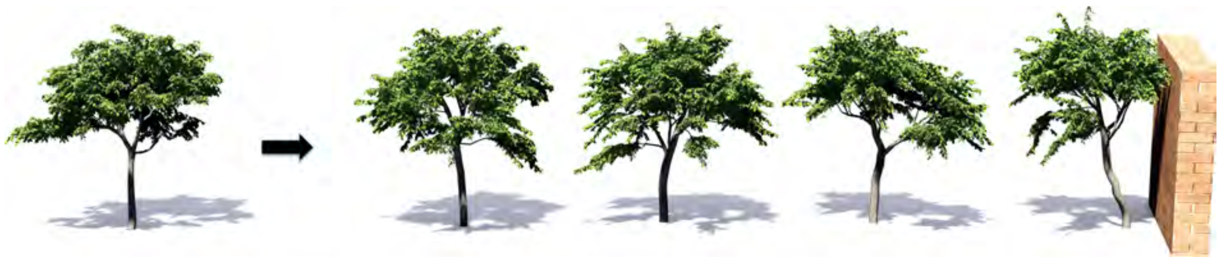

Figure 1: *An input polygonal tree model (a) has been processed by our inverse procedural modeling system and the input parameters of the developmental model have been estimated so that it can produce stochastically similar trees (b-d). The developmental model is able to produce environmentally sensitive trees models (e).*

2. Related Work

Plant modeling has been the focus of computer graphics research for a long time. Traditional methods are rule-based systems [Hon71, PL90], particle systems [RB85], environmental sensitive automata [Gre89], and space colonization methods [PHL*09, RLP07].

In the last decade some effort has been devoted to the reconstruction of real trees. Reche-Martinez et al. [RMMD04] use a set of registered images to generate volumetric representations of the tree canopy, and Neubert et al. [NFD07] work on loosely arranged inputs. In contrast to image-based methods, Livny et al. [LPC*11] reconstruct trees from laser-scanned point sets (see also [XGC07]).

Two recent approaches dynamically adjust tree models according to their graph skeleton. The first approach proposed by Pirk et al. [PNDN12] infers the early developmental stages from one input exemplar and the second approach of Pirk et al. [PSK*12] employs environmental sensitivity to static input trees and allows the user to automatically change the tree topology under varying environmental conditions. Both methods focus on an efficient processing; the underlying developmental models only utilize a subset of parameters required for a growth model and thus do not allow generation of new tree models or adding branches. Their methods do not allow for inferring all parameters that is required for a developmental model learning.

Sketch-based methods have been used to guide tree modeling [OOI06, CNX*08] by using biologically motivated rules to infer the 3D structure of a tree from 2D input [LRBP12]. Other interactive methods allow the user to control the modeling process by changing a set of parameters. One of the first user-aided methods is the work of Weber and Penn [WP95], and after it came the Xfrog modeling system [LD99], in which rule-based and procedural modeling are combined to directly affect the model of a plant. An overview of different modeling techniques has been presented by [DL04].

The difficulty of controlling the rewriting process and the low intuitiveness of the rule creation lead to works that focus on the problem of inverse procedural modeling. Ijiri et al. [IMIM08] propose a system that constructs a procedural model from an arrangement of 2D elements. Stava et al. [SBM*10] use transformation spaces to build procedural rules that describe an arbitrary vector image. An extension of this approach to 3D is presented by [BWS10]. In both of these works, convincing procedural models could be obtained only for highly regular and symmetric inputs, but not for stochastic data.

Finding a set of parameters that would generate a given input of a procedural model is a complex task. The parameters can be estimated using optimization techniques such as the MCMC sampling [MRR*53]. This approach was used for finding the procedural representation of facades [RB06] or for directing the rewriting process in procedural modeling [TLL*11]. Both methods rely on a known procedural model with predefined parameter values that were used to compute the best model fitting to the input data. Similar to our method is the approach of Vanegas et al. [VGDA*12] that estimates input parameters of a procedural model. However, their approach is aimed at interactive modeling of regular virtual cities in order to provide high-level control and at matching stochastic models. Moreover, Cournede et al. [CLM*11] present a method that automatically detects parameters of stochastic procedural plant models, but it relies on precise measurements of biomass distribution in real trees, which would be infeasible in computer graphics applications.

3. Method Overview

To find the computer graphics procedural representation for different input trees, we need a procedural model powerful enough to generate a variety of trees species. It is difficult to reverse engineer an entire model including its structural description but given an appropriate parametric model many species can be expressed. This reduces our problem to finding the right set of parameters for a given procedural model that is able to generate trees resembling the input.

Our modeling framework (see Figure 2) uses either a single or multiple geometric tree models such as LiDAR scans,

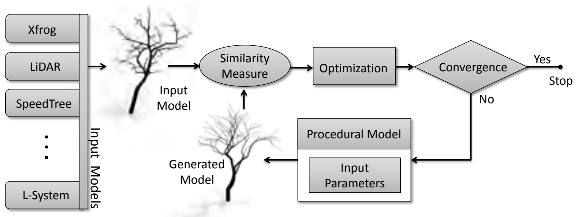

Figure 2: *Framework overview: Input parameters of the procedural model are estimated via optimization and similarity measures between the input and the generated model.*

Xfrog library models, SpeedTree models, or trees generated by Open L-systems. The model is converted into a tree graph representation [GC98] using an algorithm inspired by Au et al. [ATC*08].

We then use our procedural model that incorporates recent findings in plant biology and computer graphics (see Section 4). The input parameters for the model are determined by maximizing a similarity measure that computes the visual and structural resemblance of two trees (Section 5). This is done by MCMC-based optimization that proposes new parameter values that gradually increase the similarity of the generated trees and ends when the parameters can no longer be significantly improved (Section 5.4).

4. Used Procedural Model

Trees have a wide range of shapes corresponding to various tree species that can be represented by a developmental model with high expressive power. Various developmental plant models exist, but they often describe only a limited number of species.

We introduce a new parametric procedural model based on recent advances in plant biology [CH06] and computer graphics [PHL*09]. It uses a developmental formulation in which during each growth cycle the tree grows shoots from active buds. The new shoots then grow and create buds that can grow during the next cycle. The growth of individual shoots and flushing of buds is controlled using a light based model described in [PSK*12]. Note that one growth cycle does not necessarily correspond to one year because some species undertake multiple growth cycles per year [CH06].

4.1. Parameters

Our model is controlled by a set of 24 parameters $\bar{\varphi}$ describing effects of internal and external factors on the tree shape. The parameters belong to three groups: 1) geometric, 2) environmental, and 3) parameters determining bud fate. The geometric parameters define shapes of particular plant organs such as internode length or phyllotaxis and they are important for the overall plant shape. The actual branching

Table 1: *Summary of all parameters of the growth model.*

Params.	Name	Description
φ_{AAV}	Apical angle variance	Variance of the angular difference between the growth direction and the direction of the apical bud.
φ_{NLB}	Number of lateral buds	The number of lateral buds that are created per each node of a growing shoot.
φ_{BAM}, φ_{BAV}	Branching angle mean and variance	The mean and variance of the angle between the direction of a lateral bud and its parent shoot.
φ_{RAM}, φ_{RAV}	Roll angle mean and variance	The mean and variance of an angular difference orientation of lateral buds between two internodes.
$\varphi_{\text{AB\_D}}$, $\varphi_{\text{LB\_D}}$	Apical and lateral bud extinction rate	The probability that a given bud will die during a single growth cycle.
$\varphi_{\text{LF\_A}}$, $\varphi_{\text{LF\_L}}$	Apical and lateral light factor	The influence of light on the growth probability of a bud.
$\varphi_{\text{AD\_BF}}$, $\varphi_{\text{AD\_DF}}$, $\varphi_{\text{AD\_AF}}$	Apical dominance base, distance, and age factors	Control over the auxin production and transport within a tree.
φ_{GR}	Growth rate	The number of internodes generated on a single shoot during one growth cycle.
φ_{IBL}, $\varphi_{\text{IL\_AF}}$	Internode length and age factor	The base length of a single internode and its relation to the tree age.
φ_{AC}, $\varphi_{\text{AC\_AF}}$	Apical control level and age factor	The impact of the branch level on the growth rate φ_{GR} and its relation to the tree age.
φ_{PT}, φ_{GT}	Phototropism and Gravitropism	The impact of the average direction of incoming light and gravity on the growth direction of a shoot.
φ_{PF}	Pruning factor	The impact of the amount of incoming light on the shedding of branches.
$\varphi_{\text{LB\_PF}}$	Low branch pruning factor	The height below which all lateral branches are pruned.
φ_{GBS}, φ_{GBA}	Gravity-bending strength and angle factor	The impact of gravity on branch structural bending and its relation to branch thickness.
t	Growth time	Controls the age and size of the generated trees.

structure and density is controlled by the bud fate parameters. Lastly, environmental parameters are used to describe interaction of a tree with its environment, for example with light and gravity.

We have presented a moderately large parameter space for which it is difficult to find the appropriate parameters for a target species manually. Inverse modeling, with the help of a capable optimization algorithm with efficient sampling of the parameter space, provides an automatic solution.

Geometric parameters represent properties of different tree species on tree geometry. A group of parameters determines the *phyllotaxy* of a tree. They control number and relative orientation of lateral buds along a shoot as shown in Figure 4 and in Table 1. Additional parameters describe the variance of individual angular variables that add randomness into the generated trees. The apical bud is always located at the end of the shoot, and its orientation with respect to the parent shoot is determined by a spherical angle $(\theta, \phi); \{\theta \sim \mathcal{N}(0, \varphi_{\text{AAV}}), \phi \sim \mathcal{U}(0, 2\pi)\}$, where $\mathcal{N}$ and $\mathcal{U}$ are the normal and uniform distributions respectively and φ_{AAV} is the *apical angle variance* parameter. Increasing its value adjusts the branching structure as shown in Figure 3(a).

The shoot length is controlled by the *growth rate* (φ_{GR}) and the *internode base length* (φ_{IBL}), which together determine the distance between two nodes containing lateral buds. The number of internodes n_{IN} generated by a single shoot is computed as rounded growth rate $n_{IN} = \lfloor \varphi_{\text{GR}} \rceil$, and the length of a shoot is $n_{IN}\varphi_{\text{IBL}}$. The values of φ_{GR} and

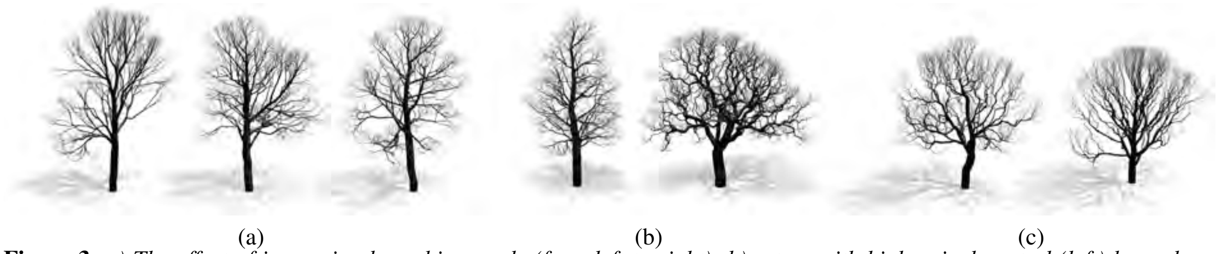

Figure 3: *a) The effect of increasing branching angle (from left to right); b) a tree with high apical control (left) has a long trunk and shorter side branches, whereas low apical control causes trees to have the typical spherical crown (right); (c) high apical dominance results in less-frequent branching enabling certain branches to become dominant (left). A tree with low apical dominance tends to branch very often, creating many equally dominant branches (right).*

φ_{IBL} can change based on the location and tree age. The initial internode length decreases with the increasing tree age [OL96] and is expressed by *internode length age factor* ($\varphi_{\text{IL\_AF}}$), which modifies the actual internode length because $\varphi'_{\text{IBL}} = \varphi_{\text{IBL}}\varphi^t_{\text{IL\_AF}}$, where t is the age of the tree. The age factor is usually smaller but close to one, which causes the length of internodes to gradually decrease as the tree gets older.

The growth rate of a shoot φ_{GR} is primarily affected by the dominance of its parent branch, described by *apical control* (φ_{AC}) [CBH09]. If its value is high, shoots growing from dominant branches grow faster than shoots growing from lateral ones, resulting in a tall tree with a significant trunk. Trees with a low apical control have a spherical crown as all branches grow with similar speed (see Figure 3(b)). Apical control directly modifies the growth rate

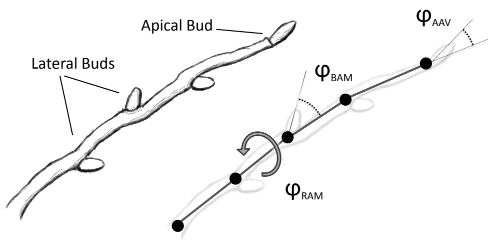

Figure 4: *Various parameters affect shoot growth: bending angle φ_{BAM}, roll angle φ_{RAM}, and apical angle variance φ_{AAV}.*

$$\varphi'_{\text{GR}} = \begin{cases} \varphi_{\text{GR}}/\varphi_{\text{AC}}^{\zeta_k} & \text{if } \varphi_{\text{AC}} > 1 \\ \varphi_{\text{GR}}/\varphi_{\text{AC}}^{\zeta_k - \zeta_{max}} & \text{otherwise} \end{cases} \quad (1)$$

ζ_k is the level of the parent branch [BAS09]. More dominant branches have lower levels. The value of apical control usually decreases with increasing tree age, which allows the tree to form a denser crown at the later stages of its growth [OL96]. We control this time dependence with *apical control age factor* ($\varphi_{\text{AC\_AF}}$), which modifies the original apical control as $\varphi'_{\text{AC}} = \varphi_{\text{AC}}\varphi^t_{\text{AC\_AF}}$.

The parameters from the previous part define the geometric properties of individual branch segments, but the internal tree structure is primarily defined by the fate of its buds that is controlled by **bud fate parameters**. A single shoot usually contains many buds, but not all of them flush and grow during the growth cycle. Some may stay dormant until the conditions are favorable, or they might die as a result of biological or environmental factors. During each growth cycle, we set the parameters $\varphi_{\text{AB\_D}}$ and $\varphi_{\text{LB\_D}}$, which specify the probability that a given apical ($\varphi_{\text{AB\_D}}$) or lateral ($\varphi_{\text{LB\_D}}$) bud died before the growth cycle started. Living buds can either form a new shoot or remain dormant as a function on illumination and the plant hormones [Sri02]. We represent the influence of light on apical buds with the *apical light factor* $\varphi_{\text{LF\_A}}$ and on lateral buds with the *lateral light factor* $\varphi_{\text{LF\_L}}$.

The fate of a bud depends on the hormone auxin, which is produced and transported towards the root when a bud starts growing into a new shoot. A high level of auxin inhibits lateral buds from flushing, effectively delaying the growth of side branches [CBH09]. This allows a tree to form more significant branches that hold smaller lateral branches. If the level of auxin is low, the tree branches more frequently, thus creating a regular structure with high density and without dominant branches (see Figure 3(c)). In real trees auxin also controls the growth rate that is considered constant in our model.

We encode such *apical dominance* in three parameters: $\varphi_{\text{AD\_BF}}$ is the *apical dominance base factor* corresponding to the amount of auxin produced by a flushing bud. The concentration of auxin decreases as it is transported toward the roots, described by *apical dominance distance factor* $\varphi_{\text{AD\_DF}}$. Apical dominance often decreases with increasing tree age, represented as *apical dominance age factor* $\varphi_{\text{AD\_AF}}$.

Apical dominance should not be confused with the apical control. Even though they are often used interchangeably, their cause and their effect are different [CH06]. Apical control influences the growth rate of a shoot, but apical dominance controls bud flushing. Apical control is used together with the light factors to compute the probability $P(F(\mathbf{b}_i))$ that a given bud $\mathbf{b}_i$ will flush. For apical buds the probability is

$$P(F(\mathbf{b}_i)) = I(\mathbf{b}_i)^{\varphi_{\text{LF\_A}}}, \tag{2}$$

where $I(\mathbf{b}_i)$ is the illumination of the bud. It is computed using the light model [PSK*12].

Lateral buds grow new shoots with probability

$$P(F(\mathbf{b}_i)) = I(\mathbf{b}_i)^{\varphi_{\text{LF\_L}}} \exp\left(-\sum_{\mathbf{b}_j \in \Lambda(\mathbf{b}_i)} \delta_j \varphi_{\text{AD\_BF}} \varphi^{t}_{\text{AD\_AF}} \varphi^{d(\mathbf{b}_i,\mathbf{b}_j)}_{\text{AD\_DF}}\right), \tag{3}$$

where $\Lambda(\mathbf{b}_i)$ is a set of all buds located above the given lateral bud, and $\delta_j = 1$ if the bud $\mathbf{b}_j$ flushed and otherwise zero. The value of $d(\mathbf{b}_i, \mathbf{b}_j)$ is the branch-wise distance (Section 5.2) between buds $\mathbf{b}_i$ and $\mathbf{b}_j$.

Environmental parameters define how the surrounding environment affects the plant. Light and gravity directly influence the growth direction of a shoot. In normal conditions, shoots grow in the direction defined by their source bud, but the growing shoot can bend toward or away from an impetus that is described by *tropisms*. *Phototropism* (φ_{PT}) is bending toward light, and *gravitropism* (φ_{GT}) is bending away from the gravity. The bending is simulated using the approach from Palubicki et al. [PHL*09], where the parameters φ_{PT} and φ_{GT} represent the strength of bending.

An increasing pruning factor φ_{PF} causes more intense shedding of shadowed branches. Many branches are also removed due to human or animal intervention, etc. We simplify these influences by using a single parameter called *low-branch pruning factor* ($\varphi_{\text{LB\_PF}}$). It specifies the maximal height up to which all side branches are pruned.

As a tree grows older, the branches bend down in a response to increasing weight of their child branches. The sagging of older branches is especially important for many coniferous and other softwood species (Figure 16). We propose a simplified model that simulates the bending at each node of the grown tree by using the orientation of the bent branch and the total mass of its child branches (see Appendix A).

4.2. Foliage

Instead of directly simulating the growth of foliage we use a procedural system from Livny et al. [LPC*11]. The leaves are located along the terminal branches. A certain species is assigned manually to a model and additional branchlets are generated that hold the leaves. This results in a higher foliage density, which is favorable for scanned models of real trees because they usually contain only the main branches. We used a set of leaf templates for many deciduous and coniferous trees, including willow, mahogany, pine, and thuja species. The leaves are generated at the end of the growth cycle.

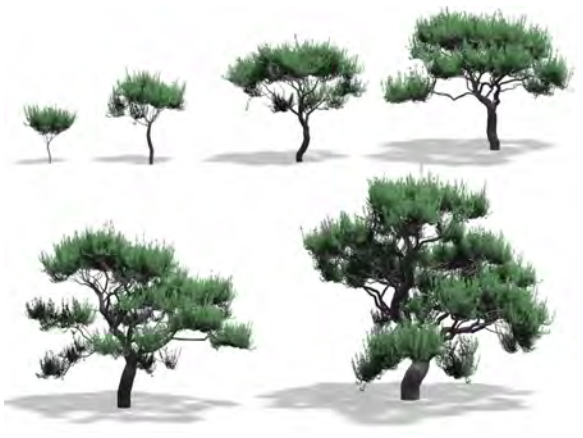

Figure 5: *Growth of a tree generated with our developmental model at ages (left to right): 5, 10, 15, 20, 25, and 30 years.*

4.3. Example Trees

Figure 5 shows the development of a tree achieved by our procedural model. The individual images were captured with a time difference of five years. Figure 6 shows several examples that were generated using our approach by directly setting the parameters (see Table 2). The variety of shapes has been achieved by changing the parameter values, and each tree was generated using a trial-and-error approach because the parameter space is large, and expressing the designer intention in this way is difficult.

Table 2: *Parameters for trees generated by the developmental model (see Table 1).*

Param	F6a	F6b	F6c	F6d	F6e	F6f
φ_{AAV}	38	0	10	5	2	12
φ_{NLB}	4	2	2	1	60	2
φ_{BAM}	38	41	51	45	3	43
φ_{BAV}	2	3	4	5	130	3
φ_{RAM}	91	87	100	130	10	80
φ_{RAV}	1	2	30	3	0	4
$\varphi_{\text{AB\_D}}$	0	0	0	0	0.018	0
$\varphi_{\text{LB\_D}}$	0.21	0.21	0.015	0.01	0.03	0.21
$\varphi_{\text{LF\_A}}$	0.39	0.37	0.36	0.5	0.21	0.36
$\varphi_{\text{LF\_L}}$	1.13	1.05	0.65	0.03	0.5	1.05
$\varphi_{\text{AD\_BF}}$	3.13	0.37	6.29	5.59	0.55	0.38
$\varphi_{\text{AD\_DF}}$	0.13	0.31	0.9	0.5	0.91	0.31
$\varphi_{\text{AD\_AF}}$	0.82	0.9	0.87	0.979	2.4	0.9
φ_{GR}	0.98	1.9	3.26	4.25	0.4	1.9
φ_{IBL}	1.02	0.49	0.4	0.55	0.97	0.51
$\varphi_{\text{IL\_AF}}$	0.97	0.98	0.96	0.95	5.5	0.98
φ_{AC}	2.4	0.27	6.2	5.5	0.92	0.25
$\varphi_{\text{AC\_AF}}$	0.85	0.90	0.9	0.91	0.05	0.70
φ_{PT}	0.29	0.15	0.42	0.05	0.15	0.15
φ_{GT}	0.61	0.17	0.43	-0.01	0.22	0.13
φ_{PF}	0.05	0.82	0.12	0.48	1.11	0.80
$\varphi_{\text{LB\_PF}}$	1.3	2.83	1.25	5.5	0.32	2.90
φ_{GBS}	0.73	0.195	0.94	0.09	0.12	0.19
φ_{GBA}	0.05	0.14	0.52	0.89	0.78	0.14
t	8	26	14	28	10	30

5. Similarity Measure and Optimization

Having described our parametric model, we now want to focus on how to find a set of parameters that maximizes the

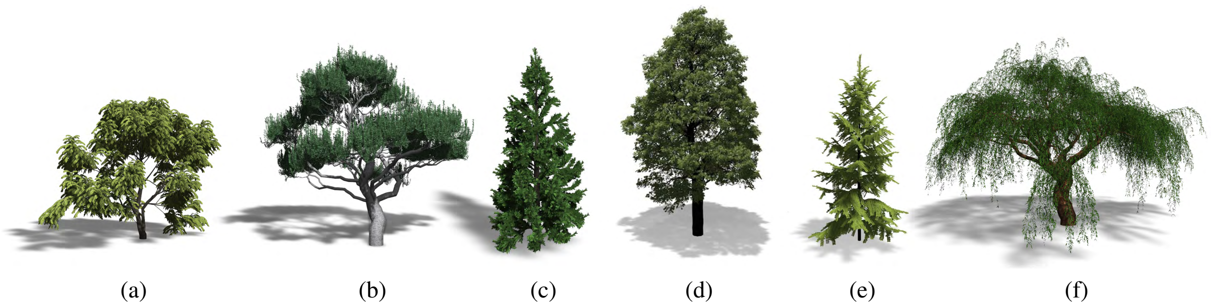

Figure 6: *Different species generated with the procedural model. The corresponding prameter values can be found in Table 2.*

similarity measure between the input data and the generated trees. Although possible, we make no attempt to optimize the environmental parameters of the procedural model that are used for the synthesis of trees interacting with the environment.

Various similarity measures have been used in computer graphics and in biology. Some of them concentrate on the branching skeleton [FG00] whereas others focus on the foliage rendering [NPDD11]. In contrast, we introduce a new similarity distance that incorporates shape, geometry, and structure and evaluates the visual and structural differences between two botanical tree models τ_1 and τ_2. Our similarity distance is composed of three different components that measure the differences between the two trees at three different quantitative levels: 1) the shape distance $d_S(\tau_1, \tau_2)$ measures the difference between the overall shapes of the trees; 2) the geometric distance $d_G(\tau_1, \tau_2)$ compares the global geometric branching properties; and 3) the structure-based distance $d_T(\tau_1, \tau_2)$ detects local differences between individual branches based on their position within the trees.

Pirk et al. [PNDN12] proposed a geometric approach for estimation of tree parameters from polygonal models. Although estimating some values directly from the model could speed up the optimization, we observed that the optimization time is not a bottleneck and decided to use full optimization of all parameters from the scratch. Moreover, our optimization works with any input model and is not limited to models compatible with approaches presented by [PNDN12].

5.1. Shape Distance

The shape distance function evaluates several descriptors that define the shape, as illustrated in Figure 7. The height of the tree τ is described by $\lambda_{S,h}(\tau)$. The crown shape is affected by the distribution of leaf branches along the vertical axis of the tree. To capture the variances in this distribution, we divide the tree into three horizontal slabs as shown in Figure 7.

To find the separating planes, we first compute the geometric mean of the crown μ_c that divides the entire crown into two halves. The location of the separating planes is then defined by the geometric means μ_{uh} and μ_{lh} of the upper and lower halves, respectively.

For each slab, we compute a set of shape descriptors $\{\lambda_{S,c}\}$ that capture their height h_i, the average radius in the horizontal principal directions of the crown $r_{\min}$ and $r_{\max}$, and their leaf-branch density, where the leaf-branches are all terminal branches. Having two trees τ_1 and τ_2, we first compute the differences δ_{λ_S} between all descriptors λ_S

$$\delta_{\lambda_{S,i}} = 1 - \exp\left(-\frac{\left(\lambda_{S,i}(\tau_1) - \lambda_{S,i}(\tau_2)\right)^2}{2\sigma^2_{\lambda_{S,i}}}\right), \tag{4}$$

where $\sigma_{\lambda_{S,i}}$ is a normalization factor that ensures comparability of different descriptors. Based on our experiments we set the normalization factor to $\sigma_{\lambda_{S,i}} = \frac{1}{5}\min\left(\lambda_{S,i}(\tau_1), \lambda_{S,i}(\tau_2)\right)$, which resulted in a fast convergence during the optimization process.

The shape distance is then computed as the Minkowski distance of order p of the vector $\bar{\delta}_S$ to the origin. If $p = 1$, the distance is essentially just an averaged sum of the differences, and with increasing p, the weight of larger differences increases. From our experiments we set $p = 2.5$, which gives a noticeably higher importance to large differences without significantly degrading the impact of the lower ones.

5.2. Geometric Distance

The shape of two trees can be similar even if there are noticeable differences in the geometric properties of their branching structures. The geometric properties of a tree are defined

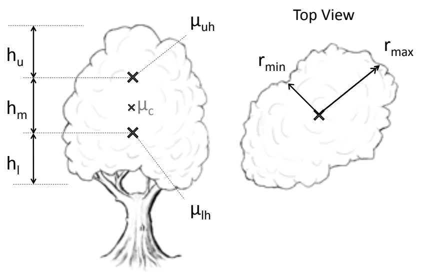

Figure 7: *The tree crown is divided into three parts that are then compared independently between two trees.*

by the statistics of the geometry of its branches and by their relative orientation.

Branch Geometry. Properties of branch geometry are computed from the tree graph. Each branch is described by a set of properties $\{b_i\}$ (see Table 3) that are computed from the geometric information of the branches, as illustrated in Figure 8.

We obtain **geometric descriptors** from the geometric properties described in Table 3. All geometric properties, except the branch length and thickness, are calculated by using sample points over the tree. The descriptors are defined as the weighted mean and variance of these samples.

Table 3: *Geometric properties of a single node in the branch graph.*

Sym.	Name	Description	Formula
b_L	Length	Total length of the branch.	$\sum_{i=1}^{k} d_i$
b_D	Thickness	Maximal thickness of the branch.	$\max_{\forall d_i} t_i$
b_α	Deformation	Accumulated angular deflection.	$\sum_{i=1}^{k-1} \alpha_i$
b_T	Straightness	Ratio between the end-point distance and the real length of the branch.	$\frac{\lvert\bar{d}_{SE}\rvert}{b_L}$
b_S	Slope	Angle between the direction of branch end points and the horizontal plane.	$\angle \bar{d}_{SE}$
b_{AS}	Sibling angle	Angle between the branch and its nearest sibling.	β_S
b_{AP}	Parent angle	Angle between the branch and its parent branch.	ω_S

The sample weight of the geometric attributes is computed using the length and the thickness of the given branch and reflects the different impacts of smaller and larger branches on the similarity between the two trees. For angular properties (sibling and parent angles), the weights are computed directly from the branch thickness as either $w'_T = b_D$ or $w''_T = b_D^2$. The remaining branch properties are weighted with $w'_L = b_D b_L$ and $w''_L = b_D^2 b_L$. The weighting scheme w'' denotes the dominant variation that puts more focus on the main branches, while the weights w' capture characteristics of smaller branches that form the foliage. By using two different weights for each branch property, we compute two different sample means and variances for every geometric attribute.

The sample means and sample variances are then used as geometric descriptors λ_G that are used to compute the geometric distance d_G using the Minkowski length, as shown in Equation 4.

5.3. Structural Distance

The third form of metrics evaluates the difference between individual branches with respect to their position within the

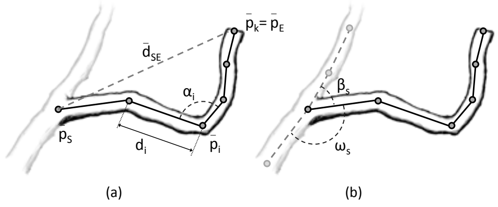

Figure 8: *Geometric properties of a single graph node.*

tree. Ferraro and Godin [FG00] evaluate distance between two tree graphs as a minimal cost of transforming all nodes of one tree graph into nodes of the other (the so-called edit distance). The nodes could be transformed by χ: *assign, insert, and delete* operators with a cost γ defined by the geometric properties of the transformed nodes. Operator *assign* maps one node from the source tree to a single node of the target tree while operators *insert* and *delete* create or erase a node from the target tree respectively. The distance between the two trees Γ is given by a sequence of operators S that transform one tree into the other one at the minimal cost: $\Gamma = \min_{\forall S} \sum_{\chi_i \in S} \gamma(\chi_i)$ (see an example in Figure 9). This

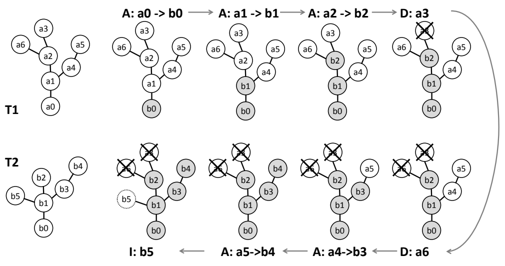

Figure 9: *Edit distance: a sequence of assign, delete, and insert operations that transform tree τ_1 into tree τ_2.*

method works correctly when the distribution of nodes is uniform but it quickly loses its accuracy when the geometric resolution between the two tree graphs differs, i.e., when one model is modeled in much coarser resolution than the other.

Split and Merge Operators. To address the problem of different geometric resolutions in structural tree distances [FG00], we propose new operators that compare tree graphs with inconsistent and irregular topologies. We define two new operators *split* and *merge* that replace the operators *insert* and *delete*:

- *Split:* $\chi_S(t_1, t_2, t_{2,i})$ divides one node t_1 from the first tree τ_1 into two nodes that are assigned to two nodes of the second tree: t_2 and its i-th child node $t_{2,i}$. All subtrees $T_{t_{2,j}}$ with roots in all remaining child nodes $t_{2,j}, \{j \neq i\}$ are then inserted into the first tree.
- *Merge:* $\chi_M(t_1, t_{1,i}, t_2)$ is the inverse operator of the split operator. It merges a node t_1 from the first tree with one of its child nodes $t_{1,i}$. The merged nodes are then assigned to a single node t_2 from the second tree. All remaining subtrees $T_{t_{1,j}}$ rooted in child nodes $t_{1,j}, \{j \neq i\}$ are deleted.

An example of the application of operators on a tree graph is in Figure 10. Because both operators can be applied repeatedly, they effectively connect chains of subsequent nodes.

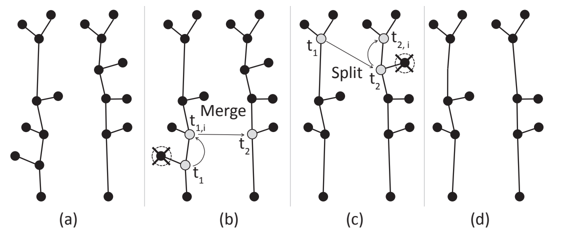

Figure 10: *To map the left input tree to the right one (a), we first apply the merge operator to fuse two nodes of the left tree and map them onto a single node of the right tree (b). We then split one node of the left tree to map the result to two nodes of the right tree (c). To compute the cost of the split operation, we apply the node joint $\mathcal{J}$ to the two nodes of the right tree and the resulting node is compared to the original node of the left tree (d).*

Operator Cost Functions. From a structural point of view, the new operators provide no new benefits because the same output can be achieved by a sequence of insert, delete, and assign operations. However, each use of a split or merge operator effectively changes the geometric properties of the transformed nodes. By changing the geometric properties, we modify the cost of the implicit assign, insert, and delete operations that are performed during each split and merge step. The cost of the split (γ_S) and merge (γ_M) operators is

$$\gamma_S(t_1,t_2,t_{2,i}) = \gamma_A(t_1,\mathcal{J}(t_2,t_{2,i})) + \sum_{\forall t_{2,j}, j\neq i} \sum_{\forall t_k \in T_{t_{2,j}}} \gamma_I(t_k) \tag{5}$$

$$\gamma_M(t_1,t_{1,i},t_2) = \gamma_A(\mathcal{J}(t_1,t_{1,i}),t_2) + \sum_{\forall t_{1,j}, j\neq i} \sum_{\forall t_k \in T_{t_{1,j}}} \gamma_D(t_k), \tag{6}$$

where γ_A, γ_I, and γ_D are the assigning, inserting, and deleting costs. The operator $\mathcal{J}(t_a,t_b)$ is a so-called *node join* operator that fuses two nodes t_a and t_b and creates a new node $t_{a\to b}$ with combined geometric properties of the original nodes (see Appendix B).

Edit Distance. The distance between two tree graphs T_1 and T_2 is equal to the minimal cost of transforming one tree graph into the other. The cost of such a sequence can be computed by a recursive algorithm [Zha96] that evaluates the so-called edit distance $d_N(t_1,t_2)$ where t_1 is the root of T_1 and t_2 is the root of T_2. Computing the edit distance between unordered trees is generally an NP-complete problem, but it can be solved in polynomial time when proper constraints are defined [Zha96] (see Appendix B for a discussion of the constraints used in our implementation.)

To compute $d_N(t_1,t_2)$, we modified the algorithm. When either t_1 or t_2 is an empty node, we compute the edit distance using the original approach, but the edit distance between two nonempty nodes t_1 and t_2 is now defined using the new operations split and merge

$$\begin{aligned} d_N \;=\; & \min\big(\gamma_A(t_1,t_2) + d_F(t_1,t_2), \\ & d_F(\varepsilon,t_2) - \max_{\forall t_{2,i}}\big(d_N(t_1,\mathcal{J}(t_2,t_{2,i})) - d_N(\varepsilon,t_{2,i})\big), \\ & d_F(t_1,\varepsilon) - \max_{\forall t_{1,i}}\big(d_N(\mathcal{J}(t_1,t_{1,i}),t_2) - d_N(t_{1,i},\varepsilon)\big)\big), \end{aligned}$$

where $d_F(t_1,t_2)$ is the distance between two forests that are created from subtrees of nodes t_1 and t_2 after the nodes have been removed. We compute $d_F(t_1,t_2)$ by solving the min-cost max-flow problem on a bipartite graph as described by Zhang [Zha96].

The structure-based distance $d_T(\tau_1,\tau_2)$ between trees τ_1 and τ_2 (and corresponding tree graphs T_1,T_2 with roots t_1,t_2) is

$$d_T(\tau_1,\tau_2) = \frac{d_N(t_1,t_2)}{2\max\left(d_N(t_1,\varepsilon), d_N(\varepsilon,t_2)\right)}. \tag{7}$$

The final **similarity measure** $D_T(\tau_1,\tau_2)$ is the sum of shape-based, geometry-based, and structure-based distances with corresponding weights $w_S, w_G, w_T \in [0..1], w_S + w_G + w_T = 1$. A value $D_T = 0$ reflects the exact similarity and 1 no similarity at all. All our results were generated for equal weights.Better results can be achieved if the weights are selected by the user for a specific problem. For example, if she is interested in the similarity of the tree crowns, the shape-based weight w_S can be increased while the remaining two weights are decreased.

5.4. Optimization of Parameters

A procedural model $\mathcal{M}$ is essentially a stochastic system, and each tree τ generated by the model is a realization of this system that is controlled by the parameters $\bar\varphi$ and the time t: $\tau^{\mathcal{M}}(\omega) \sim \mathcal{M}(\bar\varphi, t)$, where $\omega\{\omega \in \Omega_{\mathcal{M}}\}$ is the state variable of $\mathcal{M}$ defined in the state space $\Omega_{\mathcal{M}}$. In general, there can be infinite number of trees that can be generated by the system $\mathcal{M}$, where the probability that a given tree is generated is determined by some density function $p(\omega)$.

To find the model parameters that maximize the resemblance of the generated model to the input tree, we maximize the similarity measure by minimizing the distance function $D_T(\tau_1,\tau_2)$. The optimization problem is then defined as

$$\operatorname*{argmin}_{\bar\varphi_{\mathcal{M}},t} \left(\int_{\omega} D_T\left(\tau^r, \tau^{\mathcal{M}}(\omega)\right) p(\omega)\, d\Omega^{(\mathcal{M},\bar\varphi_{\mathcal{M}},t)} \right), \tag{8}$$

where τ^r is the reference input tree. The above formulation searches the optimal parameter values $\bar\varphi$ and the optimal growth time t that minimizes the distance for all trees that can be generated by the parametric model $\mathcal{M}$. To compute the distance of all trees, we need to integrate over all possible states ω that are allowed by the model $\mathcal{M}$ with parameters $\bar\varphi$ and time t.

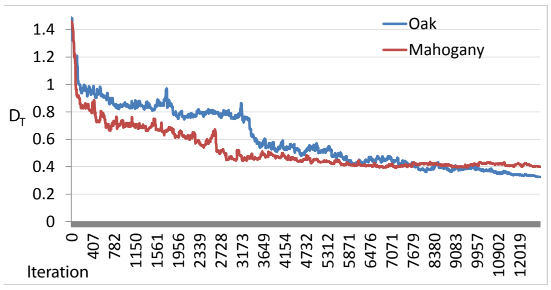

Figure 11: *Optimization of the tree distance.*

The integral of the distance function over all possible generated trees cannot be solved analytically, but its solution can be approximated by the Monte Carlo numerical integration technique. The approximated solution is then computed by randomly sampling the state space $\Omega_{\mathcal{M}}$, where each sample represents a single generated tree $\tau^{\mathcal{M}}$. Since the random samples follow the probability density function $p(\omega)$, the approximated optimization problem can be written as

$$\operatorname*{argmin}_{\bar{\varphi}_{\mathcal{M}},t} \left(\sum_{\omega_j} D_T \left(\tau^r, \tau^{\mathcal{M}}(\omega_j) \right) \right), \tag{9}$$

where ω_j are the random samples following the density $p(\omega)$ that is specific for every model $\mathcal{M}(\bar{\varphi}, t)$. The number of samples ω_j determines the accuracy of the approximated solution, as discussed in Section 6.

Because the relation between the model parameters $\bar{\varphi}$ and the distance D_T is unknown, we have to solve the optimization problem from Equation 9 indirectly using one of the stochastic or heuristics optimization methods. In our work we use the Simulated Annealing algorithm that is based on stochastic sampling of the parameter space using the Metropolis-Hastings sampling strategy [MRR*53]. The probability that a given parameter sample $\bar{x}_i = \{\bar{\varphi}_i, t_i\}$ is going to be accepted is then

$$\alpha = \left(\frac{p(\bar{x}_i)}{p(\bar{x}_{i-1})} \right)^{\frac{1}{\zeta_i}}, \tag{10}$$

where $\bar{x}_{i-1}$ is the previous parameter sample and, according to Equation 9:

$$p(\bar{x}) = \exp\left(-\sum_{\omega_j} D_T \left(\tau^r, \tau^{\mathcal{M}}(\omega_j) \right) \right). \tag{11}$$

The value of ζ_i is the temperature specific for a given iteration i. The Simulated Annealing algorithm starts with ζ_0 set to a maximum temperature ζ_{max} and the temperature is then linearly decreased until it reaches 0 for the last iteration i_{last}. For all our examples we used values $\zeta_{max} = 4$ and $i_{last} = 7500$. The convergence of our method for two results is shown in Figure 11.

6. Results

Here we show the results of our method on input models from various sources. The time required to process the input models is summarized in Table 4. The optimization was performed by generating eight sample trees for each input model (see Equation 9) that provided enough variation to sufficiently capture the shape of the input trees.

Figure	1	12	13(a)	13(b)	14	15 (left)
$t[min]$	85	43	53	12	270	45
nodes	587	359	464	298	2307	521

Table 4: *Processing time for different tree models.*

6.1. Input Models

Plant Libraries. Plant libraries created by using interactive systems belong among the most common sources of tree models. To capture this resource we applied our method on trees that were obtained from the Xfrog library [DL04]. Because of the nature of direct modeling techniques, the trees often contain repetitive regular small branches that cannot be represented exactly by a stochastic model. Although the structure of the smaller branches might not be recreated properly, it was possible to faithfully model the overall appearance of these inputs, as shown in Figure 14.

Scanned Tree Data. An important class of models consists of polygonal models of real trees that were reconstructed from scanned data [LYO*10, LPC*11]. They represent mostly young trees with relatively sparse branching structures; otherwise they could not have been effectively scanned. The input data typically contains only a few most significant branches because the smaller ones were difficult to retrieve. Although the missing data poses a problem to inverse procedural modeling, we were able to capture the shapes and structures of many of the analyzed models, as seen in Figures 1 and 13.

Developmental Models. Tree models can be also created using developmental rule-based systems that are also used in biology. We have tested an Open L-system approach by Měch and Prusinkiewicz [MP96]. Although the Open L-system description is developmental and uses internally a set of rules with input parameters, our system was able to find a set of parameters that generates models with high similarity using our developmental model, as shown in Figure 12.

Environmental Interaction. An important property of our framework is that the trees are represented as a growth model with a set of parameters. This allows the tree to develop while interacting with the surrounding environment, as shown in Figure 15. When trees are reconstructed while standing in close proximity, the competition for light prunes the colliding branches as shown on the middle tree. If a tree grows close to an obstacle, any shadow cast by the obstacle will cause significant shape alterations. This adds an important developmental property to otherwise static input models.

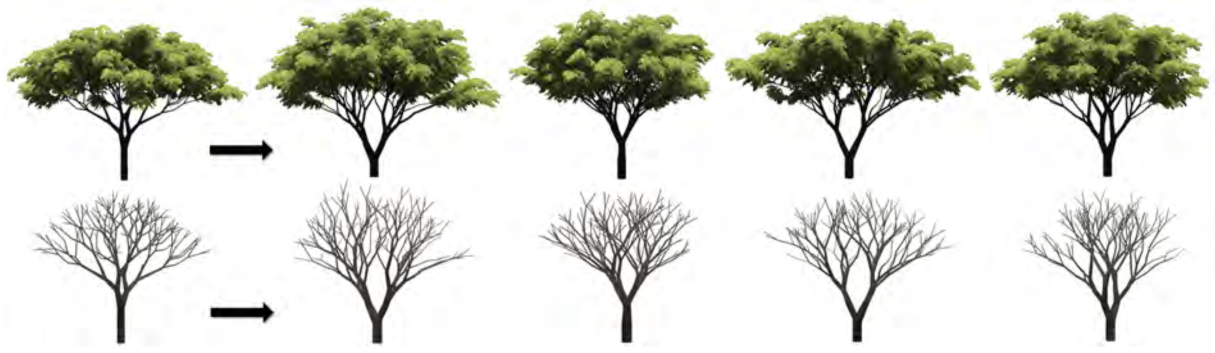

Figure 12: *A model generated using Open L-systems (left) has been reconstructed using our inverse approach (right).*

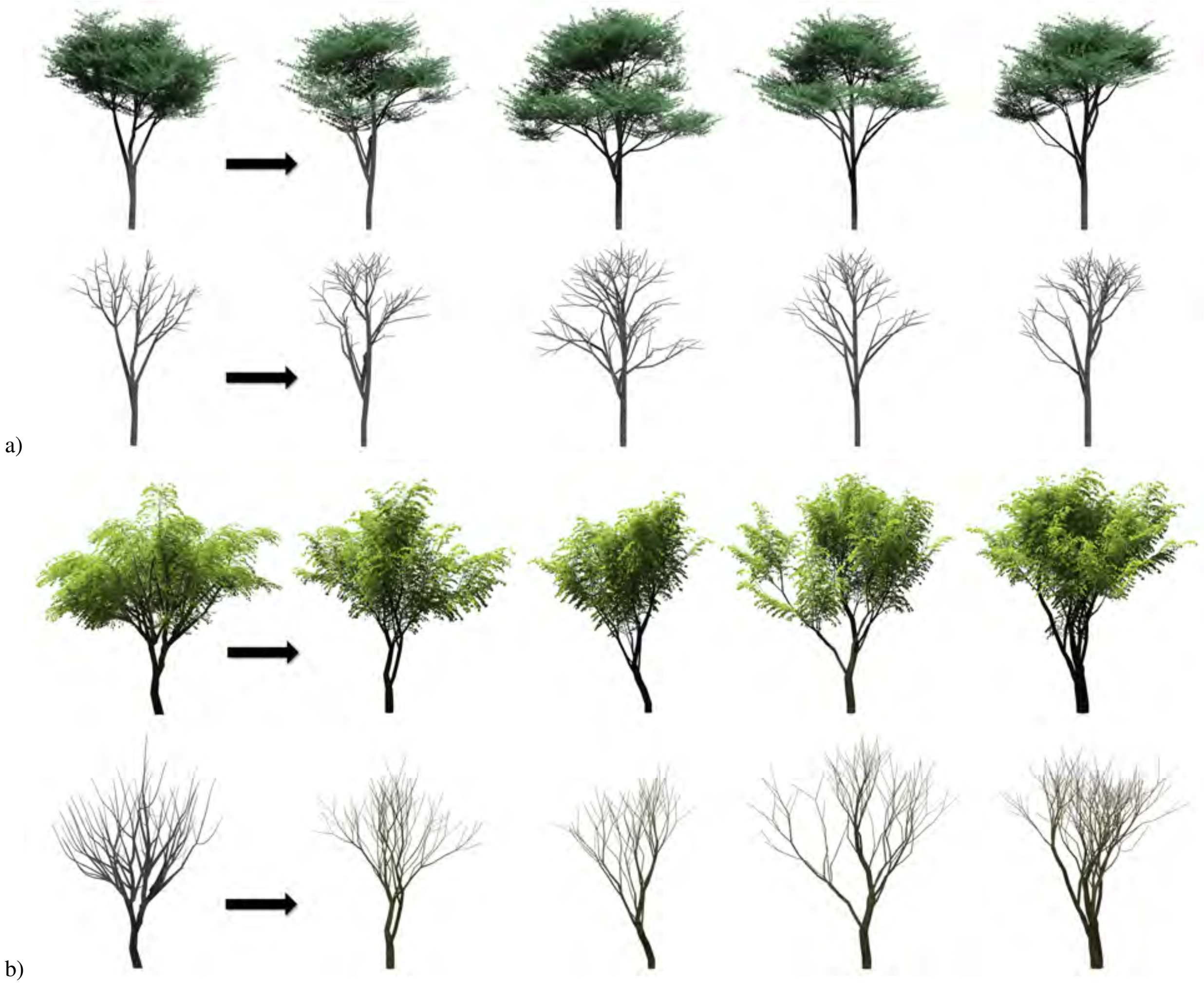

Figure 13: *Two models obtained from scanned data. Mahogany (a) and Ficus Virens (b).*

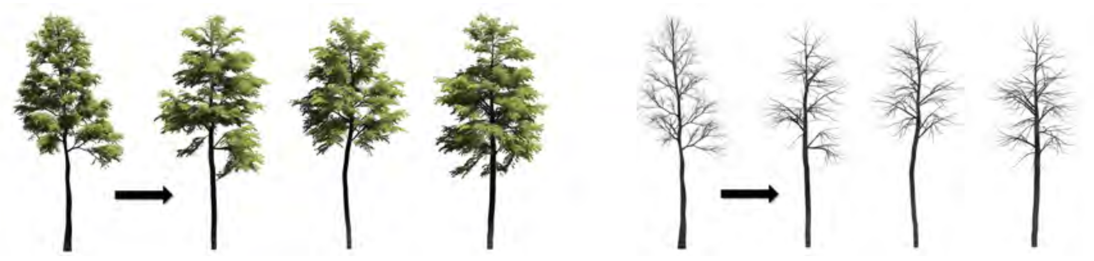

Figure 14: *A model from the Xfrog library. Left: The input model and three reconstructions. Right: The skeletal structure of the same models.*

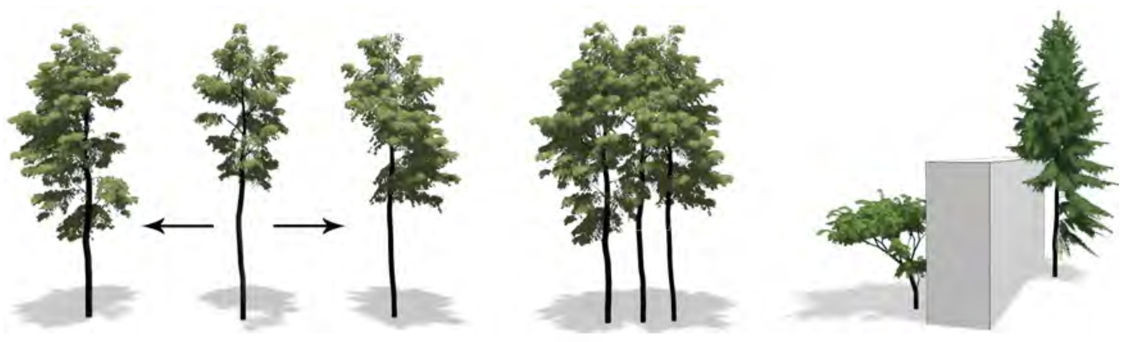

Figure 15: *Environmental interaction: Left: An input model was reconstructed, and its three instances are grown together. Middle: Competition for light prunes the branches that collide, resulting in a significantly different geometry that was not captured by the input; Right: Two reconstructed trees automatically adapt their shapes to obstacles.*

6.2. Limitations

Our technique has been successfully applied to obtain a procedural representation of many input geometric trees. Nevertheless, our method can be thought of as an example-based approach, and it fails to produce good results when the input data does not provide sufficient information, such as missing branches or incorrect topology. These problems could be resolved, at least partially, if the distance functions are modified to provide explicit information according to the specific characteristics of such input data.

The analysis of trees from model libraries revealed other issues. The inverse method was unable to accurately capture the regular nature of some input trees. To solve this problem, it would be necessary to add a support for these branching patterns into the developmental model. However, such regularity can make the reusability of resulting procedural models problematic.

Since our procedural model is stochastic it can generate a wide variety of trees for a given set of parameters. However, it cannot reproduce the input model exactly. Furthermore, the inverse procedural approach is tied to our model. Our set of parameters allows generation of a large amount of output trees. However, there may be a tree species that is not captured by our procedural model. If such a case arises, it would be necessary to extend the procedural model accordingly.

7. Conclusion and Future Work

We have introduced an approach to inverse procedural modeling of trees. Our approach uses an ad hoc developmental model that captures a wide variety of tree species. Our inverse procedural framework automatically detects the parameters that generate a tree by maximizing similarities between the input trees and generated tree. We have shown that our approach can be used to obtain the procedural representation of various tree species using inputs ranging from developmental models using L-systems, to scanned and reconstructed data, and even to hand-modeled trees from plant libraries.

Because the underlying procedural model is stochastic, for a given set of parameters it generates a set of similar trees. Because of this property, the inverse system cannot precisely reproduce the same structure of the input tree. To achieve this, we would need to control the stochastic growth of the procedural model so that all random parameters generated during application of the rules were replaced by fixed values. This is essentially another optimization problem, and it was addressed in a similar context by Talton et al. [TLL*11].

Acknowledgements

This work has been supported by the NSF IIS 0964302, Adobe Inc., NVIDIA research gifts and the DFG Research Training Group GK-1042 "Explorative Analysis and Visualization of Large Information Spaces", University of Konstanz, Germany.

References

[ATC*08] Au O. K.-C., Tai C.-L., Chu H.-K., Cohen-Or D., Lee T.-Y.: Skeleton extraction by mesh contraction. *ACM Trans. Graph. 27*, 3 (2008). 3

[BAS09] Benes B., Andrysco N., Stava O.: Interactive modeling of virtual ecosystems. In *NPH* (2009), Galin E., Schneider J., (Eds.), Eurographics Association, pp. 9–16. 4

[BWS10] Bokeloh M., Wand M., Seidel H.-P.: A connection between partial symmetry and inverse procedural modeling. *ACM Trans. Graph. 29*, 4 (July 2010), 104:1–104:10. 2

[CBH09] Cline M. G., Bhave N., Harrington C. A.: The possible roles of nutrient deprivation and auxin repression in apical control. *Trees 23* (Dec 2009), 498–500. 4

[CH06] Cline M. G., Harrington C. A.: Apical dominance and apical control in multiple flushing of temperate woody species. *Canadian Journal of Forest Research* (Jan 2006), 74–83. 3, 4

[CLM*11] Cournède P.-H., Letort V., Mathieu A., Kang M. Z., Lemaire S., Trevezas S., Houllier F., De Reffye P.: Some Parameter Estimation Issues in Functional-Structural Plant Modelling. *Mathematical Modelling of Natural Phenomena 6*, 2 (2011), 133–159. 2

[CNX*08] Chen X., Neubert B., Xu Y.-Q., Deussen O., Kang S. B.: Sketch-based tree modeling using markov random field. *ACM Trans. Graph. 27*, 5 (Dec. 2008), 109:1–109:9. 2

[DL04] Deussen O., Lintermann B.: *Digital Design of Nature: Computer Generated Plants and Organics*. SpringerVerlag, 2004. 2, 9

[FG00] Ferraro P., Godin C.: A distance measure between plant architectures. *Annals of Forest Science 57*, 5/6 (2000), 445–461. 6, 7

[GC98] Godin C., Caraglio Y.: A Multiscale Model of Plant Topological Structures. *Journal of Theoretical Biology 191*, 1 (Mar. 1998), 1–46. 3

[Gre89] Greene N.: Voxel space automata: modeling with stochastic growth processes in voxel space. *SIGGRAPH Computer Graphics 23*, 3 (1989), 175–184. 2

[Hon71] Honda H.: 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. *Journal of Theoretical Biology 31* (1971), 331–338. 2

[IMIM08] Ijiri T., Měch R., Igarashi T., Miller G.: An example-based procedural system for element arrangement. *Computer Graphics Forum 27*, 4 (2008), 429–436. 2

[LD99] Lintermann B., Deussen O.: Interactive modeling of plants. *IEEE Comput. Graph. Appl. 19*, 1 (Jan. 1999), 56–65. 2

[LPC*11] Livny Y., Pirk S., Cheng Z., Yan F., Deussen O., Cohen-Or D., Chen B.: Texture-lobes for tree modelling. *ACM Trans. Graph. 30*, 4 (July 2011), 53:1–53:10. 2, 5, 9

[LRBP12] Longay S., Runions A., Boudon F., Prusinkiewicz P.: Treesketch: interactive procedural modeling of trees on a tablet. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling* (2012), SBIM '12, pp. 107–120. 2

[LYO*10] Livny Y., Yan F., Olson M., Chen B., Zhang H., El-Sana J.: Automatic reconstruction of tree skeletal structures from point clouds. *ACM Trans. Graph. 29* (2010), 151:1–151:8. 9

[MP96] Měch R., Prusinkiewicz P.: Visual models of plants interacting with their environment. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), SIGGRAPH '96, pp. 397–410. 9

[MRR*53] Metropolis N., Rosenbluth A. W., Rosenbluth M. N., Teller A. H., Teller E.: Equation of state calculations by fast computing machines. *Journal of Medical Physics 21*, 6 (1953), 1087–1092. 2, 9

[NFD07] Neubert B., Franken T., Deussen O.: Approximate image-based tree-modeling using particle flows. *ACM Trans. Graph. 26*, 3 (July 2007). 2

[NPDD11] Neubert B., Pirk S., Deussen O., Dachsbacher C.: Improved model- and view-dependent pruning of large botanical scenes. *Comput. Graph. Forum 30*, 6 (2011), 1708–1718. 6

[OL96] Oliver C. D., Larson B. C.: *Forest Stand Dynamics*. McGraw-Hill, 1996. 4

[OOI06] Okabe M., Owada S., Igarashi T.: Interactive design of botanical trees using freehand sketches and example-based editing. In *ACM SIGGRAPH 2006 Courses* (New York, NY, USA, 2006), SIGGRAPH '06, ACM. 2

[PHL*09] Palubicki W., Horel K., Longay S., Runions A., Lane B., Měch R., Prusinkiewicz P.: Self-organizing tree models for image synthesis. *ACM Trans. Graph. 28*, 3 (2009), 1–10. 2, 3, 5

[PL90] PRUSINKIEWICZ P., LINDENMAYER A.: *The Algorithmic Beauty of Plants*. Springer–Verlag, New York, 1990. 2

[PNDN12] PIRK S., NIESE T., DEUSSEN O., NEUBERT B.: Capturing and animating the morphogenesis of polygonal tree models. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 169:1–169:10. 2, 6

[PSK*12] PIRK S., STAVA O., KRATT J., SAID M. A. M., NEUBERT B., MĚCH R., BENES B., DEUSSEN O.: Plastic trees: interactive self-adapting botanical tree models. *ACM Trans. Graph. 31*, 4 (July 2012), 50:1–50:10. 2, 3, 5

[RB85] REEVES W. T., BLAU R.: Approximate and probabilistic algorithms for shading and rendering structured particle systems. *SIGGRAPH Computer Graphics 19*, 3 (1985), 313–322. 2

[RB06] RIPPERDA N., BRENNER C.: Reconstruction of facade structures using a formal grammar and RjMCMC. In *Pattern Recognition*, vol. 4174 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2006, pp. 750–759. 2

[RLP07] RUNIONS A., LANE B., PRUSINKIEWICZ P.: Modeling trees with a space colonization algorithm. In *Proceedings of the Eurographics Workshop on Natural Phenomena*, (2007), Eurographics Association, pp. 63–70. 2

[RMMD04] RECHE-MARTINEZ A., MARTIN I., DRETTAKIS G.: Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 720–727. 2

[SBM*10] STAVA O., BENES B., MECH R., ALIAGA D. G., KRISTOF P.: Inverse procedural modeling by automatic generation of L-systems. *Comput. Graph. Forum 29*, 2 (2010), 665–674. 2

[Sri02] SRIVASTAVA L.: *Plant growth and development: hormones and environment*. Academic Press, 2002. 4

[TLL*11] TALTON J. O., LOU Y., LESSER S., DUKE J., MĚCH R., KOLTUN V.: Metropolis procedural modeling. *ACM Trans. Graph. 30* (2011), 11:1–11:14. 2, 11

[VGDA*12] VANEGAS C. A., GARCIA-DORADO I., ALIAGA D. G., BENES B., WADDELL P.: Inverse design of urban procedural models. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 168:1–168:11. 2

[WP95] WEBER J., PENN J.: Creation and rendering of realistic trees. In *Proceedings of SIGGRAPH '95* (1995), pp. 119–128. 2

[XGC07] XU H., GOSSETT N., CHEN B.: Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Trans. Graph. 26* (October 2007). 2

[Zha96] ZHANG K.: A constrained edit distance between unordered labeled trees. *Algorithmica 15*, 3 (1996), 205–222. 8, 13

Appendix A: Computing the Bending of a Branch

To compute the bending of a branch (see Figure 16) at node p_0 with an orientation $\bar{h}$, we first compute the bending force f_b on node p_0 as

$$f_b = \varphi_{\text{GBS}} m_c \left|(\bar{c} - \bar{p}_0)\,\bar{h}_H\right| \left(1 - |\bar{h}\bar{g}|\right),$$

where $\bar{p}_0$ is the location of the node p_0, m_c is the mass of supported branches with its center at point $\bar{c}$, vector $\bar{h}_H$ is the horizontal portion of the normalized branch direction $\bar{h}$, and $\bar{g}$ is the direction of gravity. Lastly, φ_{GBS} denotes the *gravity-bending strength*. The actual effect of the bending force on the orientation of the branch depends on the thickness at the given node p_0 and on the accumulated bending angle $\beta^{(t-1)}(p_0)$ that the branch has already been subjected to until time t.

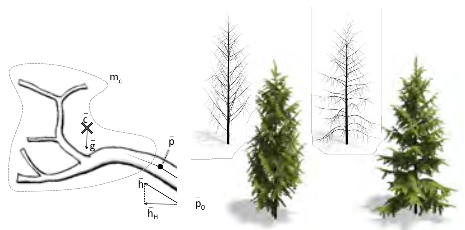

Figure 16: *Heavy branches can bend down because of the weight.*

The thickness $d(p_0)$ is used to compute the maximal bending angle of a single branch as

$$\beta_{\max}(p_0) = \frac{\pi}{2}\varphi_{\text{GBA}}^{d(p_0)},$$

where φ_{GBA} is a parameter called *gravity bending angle factor*.

The new bending angle $\beta(p_0)$ applied to the branch is

$$\beta(p_0) = \max\left(0, \beta_{\max}\exp\left(-\frac{|f_b|}{\beta_{\max}}\right) - \beta^{(t-1)}\right),$$

and the new accumulated bending angle $\beta^{(t)}(p_0)$ is then

$$\beta^{(t)}(p_0) = \beta^{(t-1)}(b_k) + \beta(p_0).$$

With the increasing accumulated bending, subsequent bending becomes more difficult. Also, as the tree grows older, the bending is hampered by the increasing thickness of the older branches.

Appendix B: Operator Cost Function Details

To compute the structural distance in polynomial time, the cost of individual operators needs to be properly constrained [Zha96]. In our implementation the cost of joining two nodes $\mathcal{J}(t_a,t_b)$ has to be defined so that $\gamma_D(\mathcal{J}(t_a,t_b)) \geq \gamma_D(t_a) + \gamma_D(t_b)$, and $\gamma_I(\mathcal{J}(t_a,t_b)) \geq \gamma_I(t_a) + \gamma_I(t_b)$. In other words, deleting or inserting a combined node should not cost less than the total cost of deleting or inserting its original components. We compute the geometric properties of $\mathcal{J}(t_a,t_b)$ by applying the formulas from the last column of Table 3 on the fused tree graphs. Geometric properties such as length, thickness, and slope of the branches are used to weight the costs of $\gamma_A(t_k)$, $\gamma_I(t_k)$, and $\gamma_D(t_k)$.

The cost of deletion or insertion of node t_k is computed from the total length $b_L(t_k)$ and thickness $b_T(t_k)$ of branches that are contained within the node as $\chi_I(t_k) = \chi_D(t_k) = b_L(t_k)b_D(t_k)$. To compute the cost of assigning node t_1 to node t_2, we first define an ordered pair of nodes $(t'_{\min},t'_{\max})$ where $(t'_{\min},t'_{\max}) = (t_1,t_2)$ if the total length of branches in node t_1 is smaller than in node t_2; otherwise $(t'_{\min},t'_{\max}) = (t_2,t_1)$. The assign cost is then

$$\begin{aligned}\chi_A(t_1,t_2) &= s(t_1,t_2)b_L(t'_{\min})\,(b_D(t_1)+b_D(t_2))+\\ &+\left(b_L(t'_{\max})-b_L(t'_{\min})\right)b_D(t'_{\max}), \qquad (12)\end{aligned}$$

where $s(t_1,t_2)$ is the dissimilarity factor between branches contained within nodes t_1 and t_2. The dissimilarity factor is the normalized Minkowski distance of a set of local differences δ_B that compare slope (b_S), straightness (b_T), and deformation (b_α) of the branches represented by nodes t_1 and t_2. The differences are computed using Equation 4. The dissimilarity factor is then

$$s(t_1,t_2) = \sqrt[p]{\frac{1}{k}\sum_{1}^{k}\delta_{B,i}^{p}},$$

where k is the number of used local differences and $p = 2.5$ is the order of Minkowski distance. With increasing dissimilarity or with increasing difference in the lengths b_L of the compared nodes, the value of $\gamma_A(t_1,t_2)$ converges to the combined cost $\gamma_D(t_1)+\gamma_I(t_2)$.