Erschienen in: IEEE Transactions on Visualization and Computer Graphics ; 22 (2016), 12. - S. 2608-2618
https://dx.doi.org/10.1109/TVCG.2015.2513409

Tree Modeling with Real Tree-Parts Examples

Ke Xie, Feilong Yan, Andrei Sharf, Oliver Deussen, Hui Huang, and Baoquan Chen

Abstract—We introduce a 3D tree modeling technique that utilizes examples of real trees to enhance tree creation with realistic structures and fine-level details. In contrast to previous works that use smooth generalized cylinders to represent tree branches, our method generates realistic looking tree models with complex branching geometry by employing an exemplar database consisting of real-life trees reconstructed from scanned data. These trees are sliced into representative parts (denoted as *tree-cuts*), representing trunk logs and branching structures. In the modeling process, tree-cuts are positioned in space in an intuitive manner, serving as efficient proxies that guide the creation of the complete tree. Allometry rules are taken into account to ensure reasonable relations between adjacent branches. Realism is further enhanced by automatically transferring geometric textures from our database onto tree branches as well as by guided growing of foliage. Our results demonstrate the complexity and variety of trees that can be generated with our method within few minutes. We carry a user study to test the effectiveness of our modeling technique.

Index Terms—Tree modeling, data-driven modeling, allometry

1 Introduction

Within computer graphics, tree models are of special interest due to their variety in nature, and have applications in many areas, including urban modeling, gaming, and movies. However, even with the significant advancement of 3D tree modeling, it still remains a challenge to easily create trees with high level of realism. The reasons to this are twofold: globally, trees have inherently complex branching structures and intricate topology, while locally branches are prevalent with fine-level geometric detail and nuances. Applications usually require the amount of user knowledge and interaction effort be kept modest opting for both flexibility and fidelity. Striking the balance among all these factors is still a major challenge for a majority of tree modeling applications.

Commonly, tree modeling approaches may be categorized into three major classes: procedural, sketch-based and data-driven. Procedural approaches utilize grammar-based parametric methods allowing the creation of trees with ever increasing branching detail [1]. These approaches typically require expert knowledge and are manually intensive, providing indirect means to control the tree modeling process. Sketch-based approaches [2], [3], [4] allow for fast interactive design of trees, however sketching the individual branches can be tedious, hence, other means have to be applied for generating complex branching and fine details. With the advance of 3D scanning, it has becoming a promising approach to scan real world trees and reconstruct their 3D models [5], [6]. With high fidelity reconstruction, the generated models capture branching structures and geometric nuances that are necessary for conveying life-like realism. In this work, we take advantage of such tree models captured from the real world to leverage the generation of new models that inherit natural realism and variety.

We introduce an example-based tree modeling technique which allows to design trees in an intuitive manner, to generate models of high realism with complex branching structures and fine-level geometric details, and yet without requiring a tedious interaction. At the core of our method is a novel editing operator utilizing trees parts exemplars representing trunk logs and branching structures (denoted *tree-cuts*).

We first build a data-base of realistic 3D trees captured with a range scanner. In a preprocessing step, we reconstruct the trees and compute tree-cuts which we extract from the tree branching structure and store in a repository. The tree editing operator allows the user to select and loosely position tree-cuts in 3D space. These tree parts exemplars serve as high-level geometric proxies, which locally constrain and guide the tree creation process. As the user selects and inserts new tree-cuts, the overall tree geometry and topology updates interactively. New branching structures that conform with recent edits of the user, automatically emerge and provide immediate visual feedback. The full tree is created through an optimization process which balances between the user-defined tree-cuts and allometric rules aiming at generating naturally-looking trees (Fig. 1).

To further enhance realism, we transfer fine branch details in the form of geometric textures from our database onto the new tree model branches. Additionally, the user models the foliage by drawing loose canopy profiles, which guide the foliage growth from branch ends (Fig. 2). In essence, the process provides a holistic modeling solution for naturally looking full trees. Our modeling tool accounts for both ease-of-use, as well as high levels of expressiveness and realism coming from the geometry of real trees.

Konstanzer Online-Publikations-System (KOPS)
URL: http://nbn-resolving.de/urn:nbn:de:bsz:352-0-324787

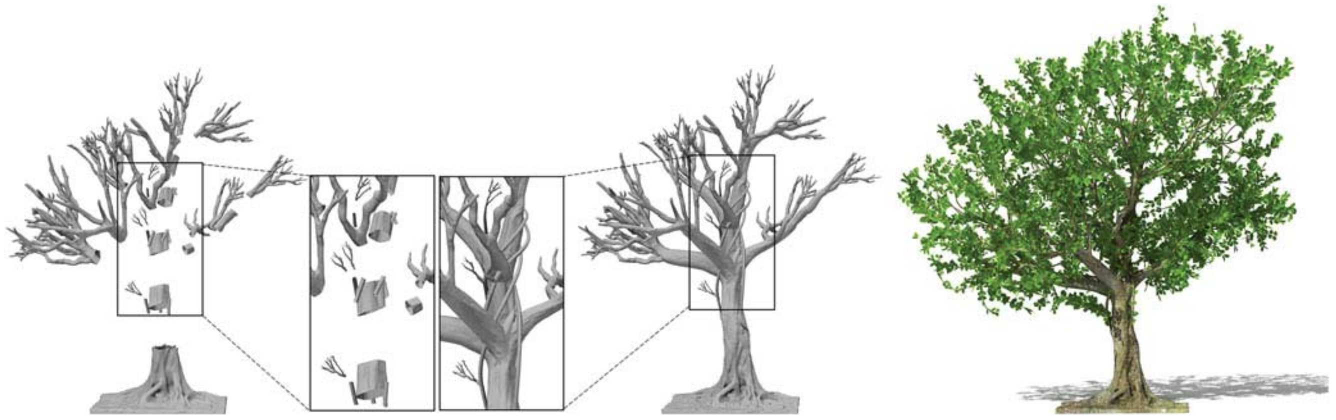

Fig. 1. Using tree parts from our data-base (left) we model a complete detailed ficus (mid), and increase its realism with geometric texture and foliage (right). The two zoomed images show the geometric complexity of the generated tree using only few tree-cuts.

2 Related Work

Early works on tree modeling use simple rules, functions and random variables to produce branching patterns [7]. In his pioneering work, Bloomenthal [8] takes a geometric approach for modeling of principal tree structures. He uses implicit functions to model branchings and full trees. Oppenheimer [9] uses a fractal model to describe the overall geometry and bark structure of a tree. Holton [10] uses Leonardo da Vinci's proportion rules to describe a botanically valid branching pattern. He defines a tree by strands growing from the root to leaves. The cross sectional area of a branch is defined by the sum of cross sectional areas of its contained strands. Boudon et al. [11] present a method for managing the multitude of parameters involved in procedural plant modeling. They represent trees as multi-scale graphs and design interactive tools to edit trees on them. Nevertheless, such generation methods are time consuming, hard to control and require a high level of expertise. In contrast, our method is straightforward and intuitive, allowing to easily guide and predict the resulting tree models.

There have been many attempts to utilize 3D reconstruction techniques from 2D photos in the context of tree modeling. Reche-Martinez et al. [12] use registered photos to generate a volumetric representation of the tree canopy and its branches and twigs. Neubert et al. [13] improve this by using only loosely arranged input images and a particle-system to produce small branches. Other approaches [14], [15] extract visual hulls from the input images and use L-Systems to synthesize branches within these hulls.

Other methods [16], [17] infer the branching structures within envelope surfaces created from a single image or user sketches by applying some heuristics on the tree form. Palubicki et al. [18] generate realistic models of trees and shrubs using a self-organizing process which simulates the competition of branches for light and space. Wither et al. [19] sketch foliage contours to guide the distribution of 3D branches while accounting for additional botanical constraints. Their sketching system applies from the scale of a leaf to the scale of an entire tree.

With scanning technology becoming available, approaches for 3D tree reconstruction from point sets emerged. Xu et al. [20] cluster edges in a spanning graph to reconstruct the tree skeleton while leaves are randomly added to the fine branches. Livny et al. [5] use this technology and reconstruct tree skeletons from point clouds of multiple trees by computing minimal spanning graphs. In a subsequent work, they separate the base tree skeleton and the procedurally reconstructible smaller parts of the tree [6]. Pirk et al. [21] convert this representation into a dynamic data structure that allows trees to react to their environment.

Bucksch et al. [22], [23] use a space partitioning method to cluster points and form a skeleton by connecting adjacent

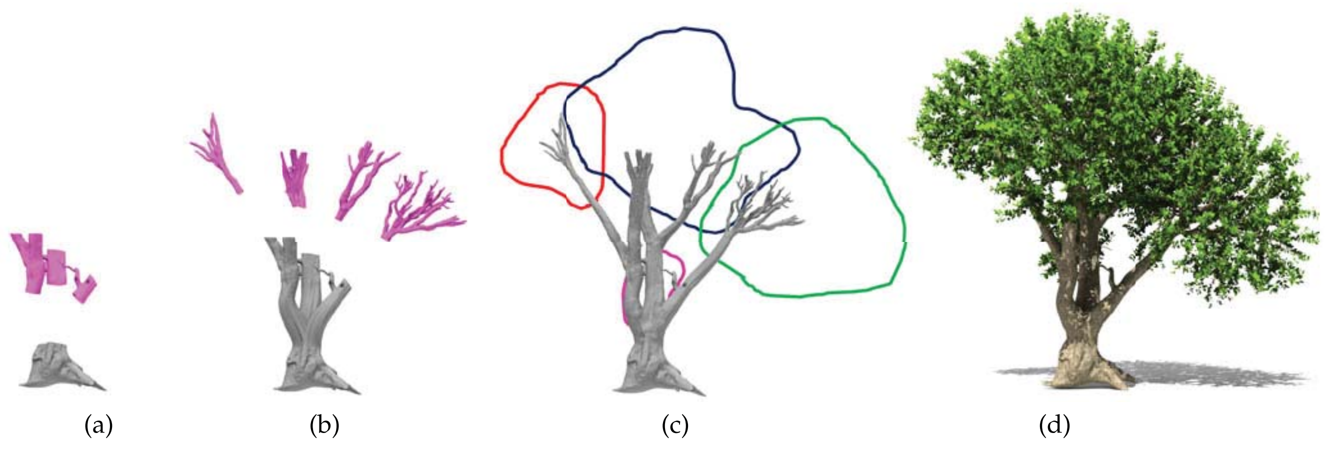

Fig. 2. Interactive modeling sequence of a Populus tree. Tree-cuts from a real tree are selected and placed in space (a) leading to a temporal trunk model (b), to which the user adds more cuts yielding a full tree model (c). Lobes are scribbled to guide the foliage growth in the final tree (d).

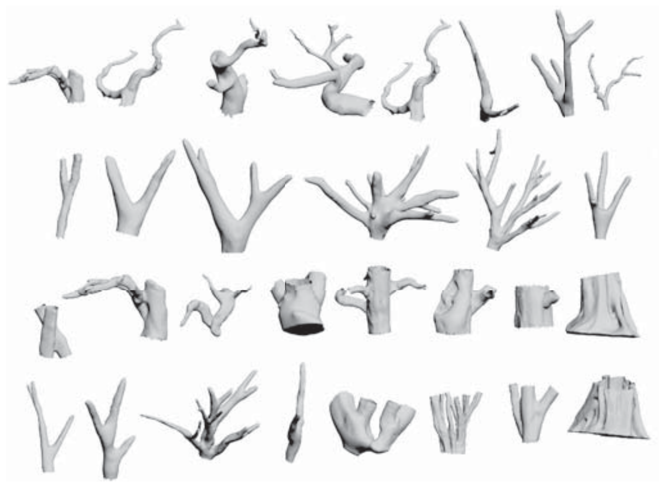

Fig. 3. A snapshot of part of our tree-cuts repository currently consisting of more than 200 exemplars from a variety of 15 tree species.

clusters. Côté et al. [24] synthesize minor tree and leaf geometry on reconstructed branches based on light scattering properties obtained from intensity data. Roumonen et al. [25] reconstruct branches by locally reconstructing patches that are then combined into the branches. Recently Zhang et al. [26] model a 3D tree by reconstructing cylindrical branches from a single scan of a tree combined with user assistance.

Two-dimensional sketches have been previously used for interactive design and editing of 3D trees and branchings. Okabe et al. [27] reconstruct 3D branching skeletons from 2D sketches by maximizing distances between branches. Additional gesture-based editing allows the user to apply further modification of the tree skeleton. Chen et al. [3] model trees from 2D sketches while accounting for additional tree constraints from a data-base using probabilistic optimization.

Recently, Longay et al. [28] combined procedural trees with user interaction to generate complex, realistic-looking tree models. Thus, models are generated through a self-organization procedural process which the user controls through several indirect parameters as well as directly guiding its overall form. In general, these methods utilize 2D sketches as means to guide tree model creation, however explicit control of the tree structure is limited. In contrast, our method is inherently 3D, using tree examples to explicitly control the tree creation process. Thus, the user directly models the complex branching structures through the utilization of existing tree-exemplars in a simple manner.

Within our framework we can utilize reconstructed trees as well as artificial ones as input in an example-based tree creation framework. Similarly, Sharf et al. [29] and Harary et al. [30] use shape parts as exemplars for hole filling and shape completion. In the same context, Funkhouser et al. [31], Merrel et al. [32] and Marechal et al. [33] synthesize the 3D geometry of parts for the creation of new shapes in an example-driven manner. Our work follows in this path, using real-life tree parts to enhance tree creation and leverage the level of realism. In contrast to general synthesis approaches, ours accounts for tree-specific properties such as cylinder-shaped branches, fine bark geometry and allometry rules.

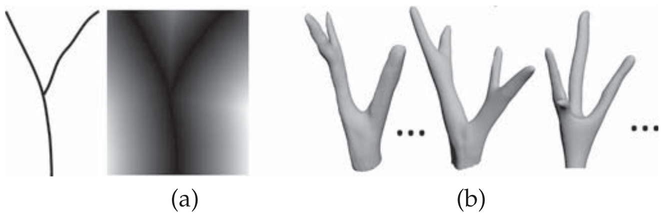

Fig. 4. Cut retrieval: from a simple user sketch (a) we compute a descriptor (a)-right which is used to retrieve a set of best matching cuts from the repository, sorted according their similarity (b).

3 Overview

In a preprocessing step, we create a database of 3D real-life trees which we capture using scanning devices and reconstruct into 3D meshes. Our method focuses on the modeling of branch geometries and therefore trees in the DB are stored without the foliage. We allow the user to create tree-branching exemplars simply by drawing 2D scribbles around tree parts and cutting them out. For each tree species, its tree-cuts (including branches and ramifications) are stored in a data repository and suggested to the user while modeling a tree. Having 100-200 cuts for each species is typically sufficient to create expressive trees with large variety and complex structures (Fig. 3).

Modeling operations merely consists of selecting tree-cuts from the repository and positioning them in 3D space. To effectively browse and suggest tree-cut candidates from the repository, the user may coarsely sketch a branch shape and topology which is then matched against our DB, retrieving the top matching candidates (Fig. 4).

As tree-cuts are selected and positioned in the scene, they connect to existing structures based on shortest-distance. Our method provides the user with immediate feedback by computing an interpolatory surface approximation of the final tree branch structure on-the-fly. Essentially, a tree-cut may connect to one or more cuts, forming various bifurcations in a generalized manner. During this process scale and orientations of tree-cuts are continuously updated while imposing allometry rules to account for botanical realism.

Once the required overall branching structure is reached, the user finalizes the model and a fine tree structure is generated. In this step, we compute a valid mesh by removing intersecting branches and remeshing discontinuous bifurcations. To increase the realism levels, branches' endpoints are enhanced with leaves and foliage. The user controls their growth by scribbling lobes that are automatically filled with foliage, in the spirit of [6]. Furthermore, we transfer geometric texture from real-life 3D bark exemplars in our repository onto the synthesized branches.

4 Technical Details

In a preprocessing step, we create a database of real-life 3D trees by scanning and reconstructing the tree models. We use a commercial handheld mid-range structure light scanner and capture a tree by manually scanning it from different views. We then register the scans together and reconstruct the 3D tree model using Poisson reconstruction [34]. Since trees vary to a large extent in their shape, even within one species, and to enrich our database we collect multiple tree models for each species. In practice we collect 15

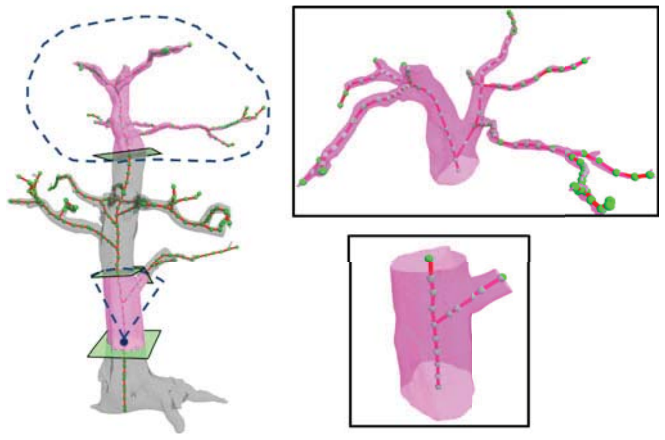

Fig. 5. Tree-cut generation. Cuts may be automatically generated by randomly cutting segments along the tree skeleton (bottom), or manually, using user scribbled loops (top).

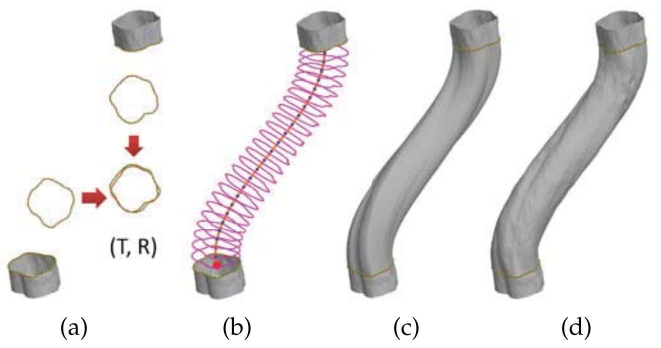

Fig. 6. Connecting tree-cuts. We compute correspondence and similarity transformation (T, R) between two branches contours (a). Next, we compute their interpolation (b) and smooth mesh (c). In (d) we transfer geometric texture onto the smooth surface.

different tree species, among others of a Cactus, Ficus, Cercis, etc. The whole DB creation was an elaborate process which took several days as tree scans were typically noisy, hard to register and properly reconstruct.

Our basic modeling operation involves manipulating small tree parts. Thus, we process tree models by cutting them into multiple parts which we store in our database repository. A tree-cut defines a cylinder-like connected part of tree, determined by at least two simple boundary-loops. Thus, the simplest tree-cut is of cylinder topology while complex cuts define arbitrary tree-like cylindrical branchings. Cuts may be of arbitrary size, ranging from small to large fractions of the tree.

Tree-cuts may be automatically generated by randomly cutting principled segments of the tree model. For an automatic production, we first compute the skeleton of a tree model using robust skeletonization [35] and define random segments along the tree skeleton yielding various subtrees. First, we sample a random starting point by climbing upwards from the root (we assume a known upward direction). A randomly given length defines the size of our subtree along the tree skeleton. The tree-cut is then produced by cutting the tree geometry at their endpoints with perpendicular planes to the local skeleton (Fig. 5).

In addition, we allow the user to manually create tree-cuts by loosely scribbling close loops on the tree. We project a loop onto the tree skeleton and find intersection endpoints. Note that our method requires that loop scribbles and their intersection with the tree skeleton yields a single connected subtree. We then proceed as in the automatic step and cut the tree geometry with perpendicular planes at the intersection endpoints. For each tree-cut, we compute its boundary loops of connected vertices and skeleton, and store them in the repository.

4.1 Modeling Interface

To model a tree, the user selects tree-cuts from the repository by browsing through the tree-cut gallery.

If a specific part or approximate structure is required, the system provides a sketch-based tree-cut retrieval tool (Fig. 4). Thus, a user may define the desired tree-cut shape and structure by loosely 2D sketching the branch shape and retrieving the best matching 3D shape from the DB. Since branching structures are typically complex, we develop a novel 2D sketch-based retrieval tool for this problem which is both easy-to-use and intuitive..

There are many works on 3D model retrieval which allow users to sketch the contour of their desired object and thus guide a DB search [36], [37], [38], [39], [40]. In these works, the user is required to sketch the 2D shape of the object as seen from one or more viewpoints. In our case, sketching the shape of tree branches would require some effort as well as certain drawing skills.

Thus, the user sketches a rough 2D approximation of the desired branch structure which is matched against 2D projections of tree-cuts skeletons from our DB. We compute a distance field originating from the user sketches using a $30X30$ grid resolution 2D image. We use this 2D field to search for best matching structures in the tree-cuts DB.

Since tree-cuts in the DB are essentially 3D, we project their skeletons onto 2D planes in a preprocessing step. To obtain a sufficient representation we compute projections of the skeleton from multiple views. In our experiments, five different views were sufficient to sample the view space around the tree-cut to a sufficient extent. Thus, we compute for each skeleton a set of five corresponding 2D distance fields. Our system then retrieves the top 10 best matching tree-cuts w.r.t. the distance between their distance-fields. Among these candidates, the user selects one tree-cut which may be loosely translated, rotated and scaled to a desired position in the 3D scene.

4.2 Connecting Tree-Cuts

Once a tree-cut is positioned, it is matched to other tree-cuts in the scene by computing the shortest Euclidean distance between the endpoints of the cuts. The user may also manually specify pairwise correspondences by clicking and marking the desired cuts.

Given two tree-cuts defined by their boundary loops (w.r.t. upward direction) (b_i, b_j), we compute the interpolating surface connecting them $\zeta(b_i, b_j)$ (Fig. 6). We uniformly resample the two boundary loops with an equal number of points (i.e., $|b_i| = |b_j| = 100$), and compute a correspondence between them. To establish a full correspondence between two boundary loops, we translate, rotate and scale the boundaries using the method presented by Horn [41].We select the best transformation (denoted T) by minimizing the sum of squared distances between the transformed

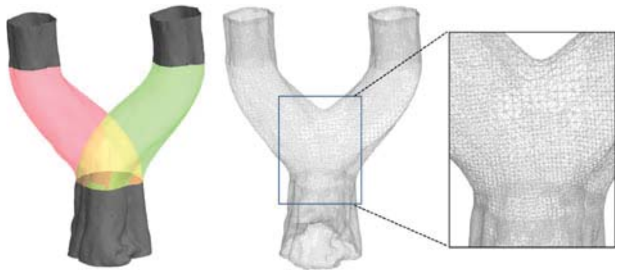

Fig. 7. Bifurcation generation. Original bifurcation with intersection (left), its mesh reconstruction (mid) and zoom in on remeshed intersection (right).

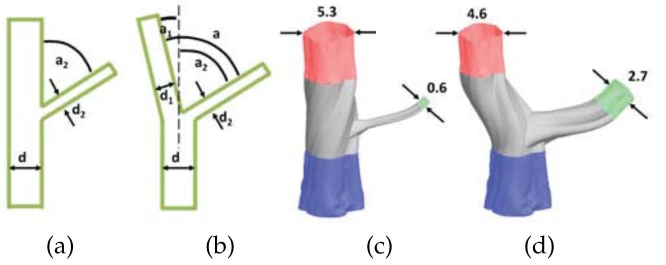

Fig. 8. Given an input bifurcation (a), we change the diameter and angle of its branches conforming to allometry constrains (b). In (c-d) we show their 3D counterpart.

points and their corresponding points on the other loop. We select the best matching pair of points in $T(b_i)$ and b_j and compute the full correspondence by walking on both boundary loops in counter-clockwise direction and matching sample points.

Next, we compute the interpolation surface $\zeta(B_1, B_2)$ by creating k intermediate cross sections that interpolate b_i and b_j. First, we compute a skeleton $\alpha(b_i, b_j)$ as the Hermite curve which interpolates the boundary loop centers and orientations. Then, we interpolate between b_i and b_j in a gradual manner and generate a set of intermediate boundary loops $\{b_{ij}^0, b_{ij}^2, \ldots, b_{ij}^k\}$.

We position the intermediate loops uniformly along $\alpha(b_i, b_j)$ yielding a set of interpolating cross sections. Thus, an intermediate loop center $center(b_{ij}^l)$ coincides with $\alpha(b_i, b_j)$ at a regular interval. In addition, we rotate the intermediate loop such that $center(b_{ij}^l)$ is perpendicular to the skeleton $\alpha(b_i, b_j)$ at the current position.

Finally, we compute the surface$\zeta(B_1, B_2)$ passing through boundary loops simply by connecting corresponding points between neighboring loops (specifically, we connect $p_m \in b_{ij}^{k\ 1}$ and $q_m \in b_{ij}^k$). This yields a quadrangulated boundary surface which may be further triangulated upon requirement.

4.3 Generation of Bifurcations

Branching bifurcations are created by connecting a tree-cut to two or more different cuts. The number of boundary loops in a tree-cut predominate the number of connections with other tree-cuts. Thus, a bifurcation can be seen as the generalization of connecting two boundary loops to a multiple cut connection. Bifurcations are modeled as superpositions of multiple connections between simple cuts which are independently matched and interpolated by their boundary loops.

Nevertheless, the generated interpolating surfaces may intersect each other and additional processing is required to generate a valid bifurcation surface. Thus, we search for mesh intersections, compute intersection curves on the surface and remove mesh parts in the intersection region (Fig. 7). To gracefully blend and connect intersecting branches, we apply Poisson reconstruction [34] in the intersection area.

4.4 Allometric Constraints

Our modeling process incorporates botanical constraints to enhance the realism of the final result. We utilize allometric rules to determine branching diameters and angles as defined by Holton [10]. Thus, our modeling scheme jointly accounts for user cuts and allometric rules to optimize branching diameters and angles of the tree (Figs. 8, and 9).

In real trees, branching diameter and angle are influenced by the species, the location of the tree, bifurcation orientation, and tropisms (growth tendencies). For simplicity and effective computation, we ignore these factors here.

Given a user-modeled tree structure (not necessarily final), we extract the father-son relations for its branches by simply traversing the tree skeleton starting from its root to the branch endpoints. Let d be the diameter of the father branch and d_1, d_2 the diameters of its children. Furthermore, let a be the angle between the child branches, and a_1,a_2 the angles between child branches and the main direction of the father branch (thus, $a = a_1 + a_2$). Having this, we follow Leonardo da Vinci's diameter rule [42]: $d^2 \approx d_1^2 + d_2^2$.

The relation between diameters and angles follows Holton's model which models branches as a set of strands [10] and is defined by:

$$a_1 = \frac{S_2}{S_1} a \quad \approx \quad \frac{A_2}{A_1} a \quad = \quad \left(\frac{d_2}{d_1}\right)^{\gamma} a \,, \tag{1}$$

where $A_i = \pi d_i^2/4$ is the cross-sectional area of a branch. γ is a species-specific constant close to two. We adapt Holton's model by assuming strands of equal diameter and define the cross-sectional area of a branch by the number of strands S. Simple rearrangement of the terms yields:

$$d_2 = \sqrt{\frac{a_1}{a}} d \,, \qquad d_1 = \sqrt{d^{\gamma} - d_2^{\gamma}} \,, \qquad a_2 = a - a_1 \,. \tag{2}$$

Having d, d_1, d_2 and a for the input bifurcation, Eq. (2) allows us to compute the new proportions for the branchings by

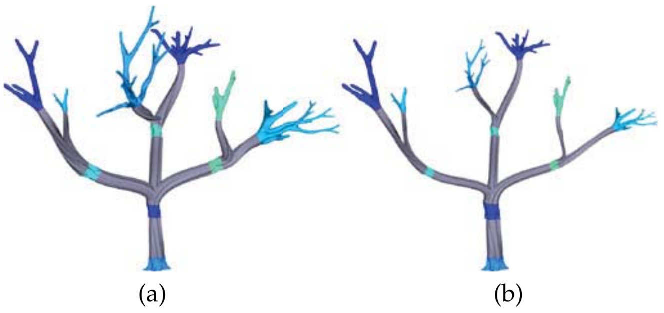

Fig. 9. Given an arbitrary tree-model (a), we apply allometry rules to further refining its branching structure (b).

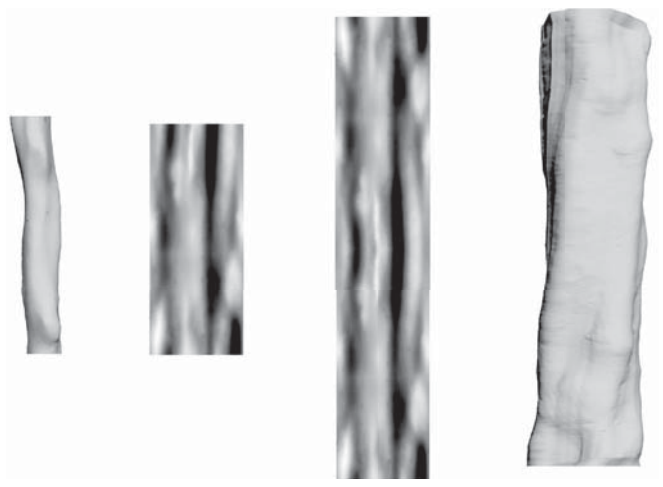
(a) (b) (c) (d)

Fig. 10. Given a tree-cut in our DB (a), we generate its geometric texture illustrated as a gray-scale height map (b). We scale the texture into a larger area (c) and transfer the geometric details onto a new cylindrical structure (d).

linking together a_1, a_2, and d_1, d_2. We apply allometry rules and adjust the tree structure on-the-fly after each user edit. Allometry typically yields a well-proportioned tree with thinner branches at higher branching levels. Figs. 8a and 8b show the global effect of applying allometric constraints on a bifurcation.

4.5 Geometry Transfer

Tree-cuts are connected using an interpolating surface that allows the creation of twisted and gnarled branches through modification of the tree-cut position and orientation. Nevertheless, the resulting interpolation is essentially a smooth generalized cylinder (inherently parameterized) that lacks fine details and realism. Thus, we enhance the interpolated surface with detailed geometry by applying geometric textures from our database. We do this by extracting the fine details from a given example and map them onto the smooth surface (Fig. 10).

Given a sample tree-cut from our repository, we transfer its fine geometric details into a texture which we then apply to our surface. First, we compute a cylindrical parametrization of the tree-cut by fitting cylinders along the tree-cut skeleton. We then project the fine geometry to this cylinder and obtain a cylindrical parametrization of the geometry which is represented as a displacement map from the cylindrical base. A similar cylindrical parametrization is applied to the branch surface, thus reducing detail transfer to a conventional texture mapping and scaling problem. We transfer a geometric texture simply by scaling it to fit the target cylinder. If scaling is too large, we simply tile the target cylinder with multiple instances. Since geometry is irregular, the transition between tiles is negligible. To avoid inconsistent results at tree-cut boundaries, we let textures of neighboring cuts overlap and blend the overlapping maps at the boundary regions.

4.6 Modeling of the Foliage

To finalize the tree model, we allow the user also to control the foliage production, i.e., the generation of twigs and leaves, on the so-far created branching structures.

To relieve the user from the tedious task of manually modeling the foliage we take a sketch-based approach similar to Longay et al. [28]. We define the tree foliage using the main branches and a set of envelope lobes [6]. Thus, a user loosely scribbles 2D loops to definemajor lobe contours and to guide foliage growth. While Longay's method utilizes sketches to control the overall tree from, our sketched lobes define only the space of twigs and leaves which grow from the nodes of the existing branch skeleton (Fig. 11).

To compute the shapes of the 3D lobes from the 2D scribbles, we simply extrude their volume in the view direction and round them to avoid sharp edges. Then lobes are filled by randomly distributed sample points, which represent the growth space of the foliage.

Twigs grow based on these sample points from nodes of the already existing branch skeleton similar to Runions et al. [16]. Specifically we use the skeleton nodes of the existing branches as buds and grow twigs from them. In a recursive manner, we use the new twigs as buds and grow additional twigs from their nodes.Thus, we generate twigs for several iterations, until all sample points are assigned to twigs (w.r.t. their distance). Finally, we add tiny branches and leaves in the spirit of Livny et al. [6] where a set of branchlets is defined by using simple L-Systems and is stored in a repository.

5 Results and Discussion

We demonstrate our method by modeling a large variety of trees, focusing on their complex branching structures and

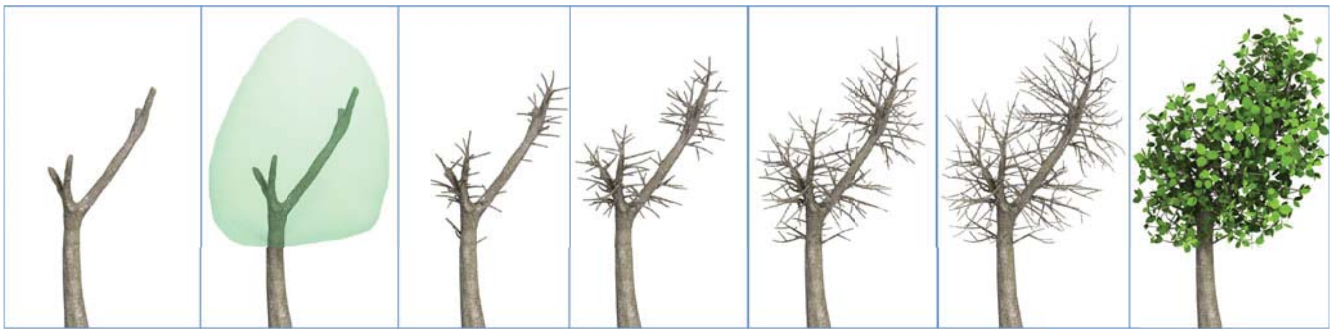
(a) (b) (c) (d)

Fig. 11. Interactive foliage modeling. Main branches computed from user tree-cuts (a) are enhanced with user sketched lobes (b). In the lobe volume we grow twigs in a recursive manner for four iterations (c). Tiny branchlets and leaves are added to twigs endpoints (d).

geometric details (see Fig. 1). All results were modeled by non-professionals with limited botanic and modeling expertise. To qualitatively and quantitatively evaluate our method, we provide some visual results and comparisons as well as performing a user study.

Modeling of complex trees consisting of detailed branches and geometrical structures stayed well below 20 minutes.This, however, did not include the foliage computation within the lobes, as it was done in a preprocessing step and took up to one hour using our non-optimized code.

Table 1 summarizes statistics of our modeling interaction. The number of cuts that the user was required to position is very low even for complex trees, demonstrating the expressiveness of our technique.We run our method on a Quad-Core Intel i5-3210M CPU 2.5 Ghz with 4 GB RAM. All trees are well detailed, represented by a large number of triangles.

TABLE 1
Modeling Time for the Different Trees

Figure	# Cuts	# Triangles	Time
1(ficus)	25	1,145,985	20 m
12 (ficus)	18	734,987	13 m
13 (bonsai)	16	81,435	12 m
13 (Cercis)	25	707,467	15 m

Fig. 1 shows excerpts from a modeling sequence demonstrating the mid-grain level of our tool, balancing between fine-level quality details and effectiveness. Here, only a small number of tree-cuts is sufficient to generate a detailed and complex branching structure. In Fig. 2 some more details about the tree-cuts and the 2D sketching for the lobe construction are given.

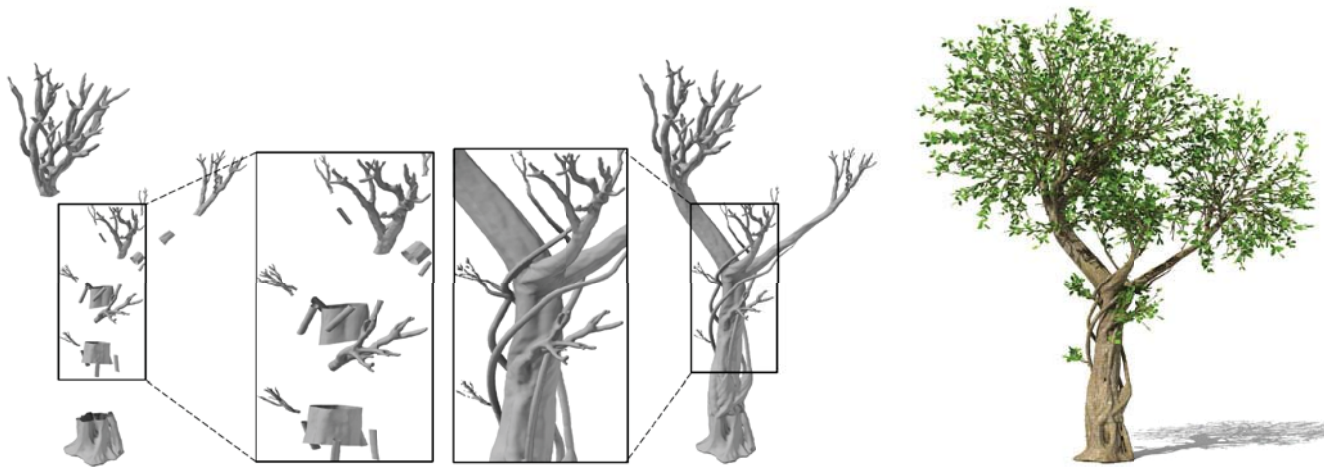

Fig. 12. Modeling a Ficus tree with complex branching structures and geometric variety. Left-to-right are the tree-cuts, detailed views on the cuts and their completion, full tree structure and complete tree with foliage.

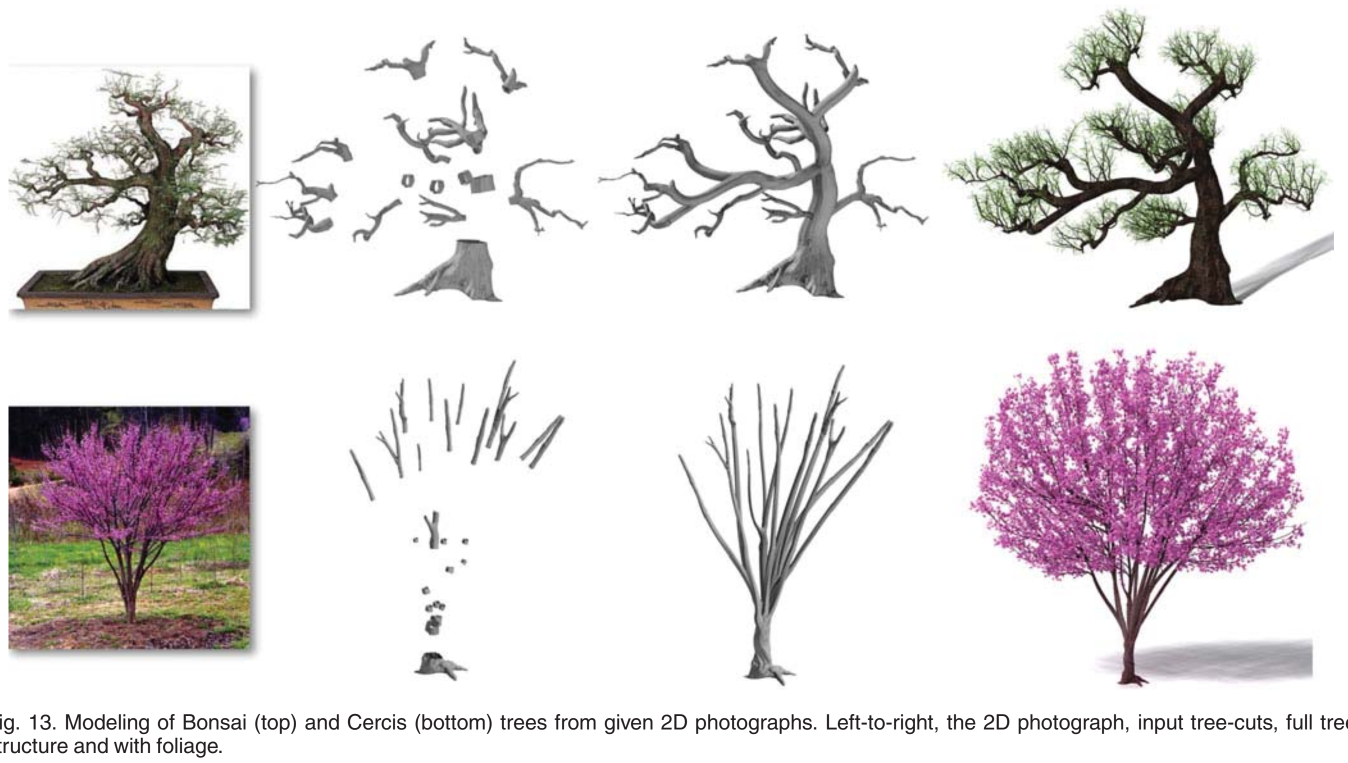

Fig. 13. Modeling of Bonsai (top) and Cercis (bottom) trees from given 2D photographs. Left-to-right, the 2D photograph, input tree-cuts, full tree structure and with foliage.

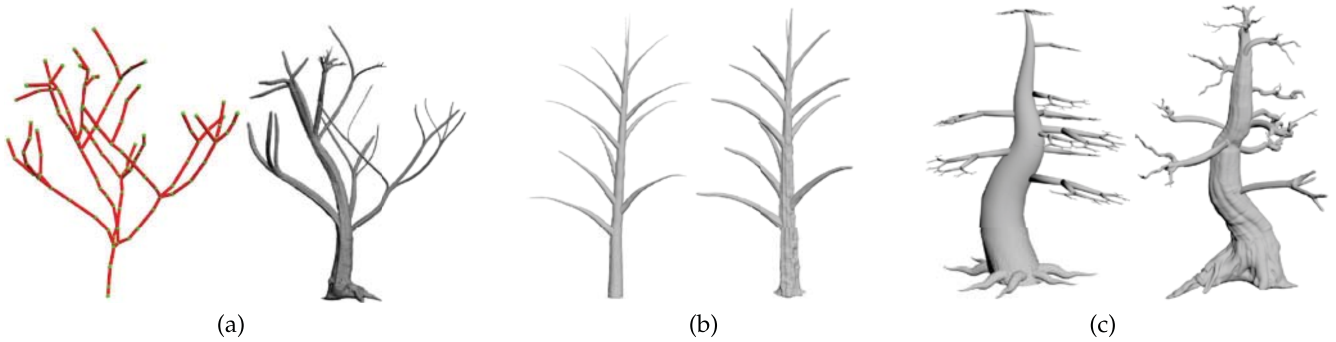

Fig. 14. Realism enhancement of existing trees: in (a) we enhance a simple skeleton (left) with geometric texture (right). Tree models from Xfrog (left of (b)) and Google Sketchup (left of (c)) are re-modeled using our tree-cuts and geometric detail (right of (b) and (c)).

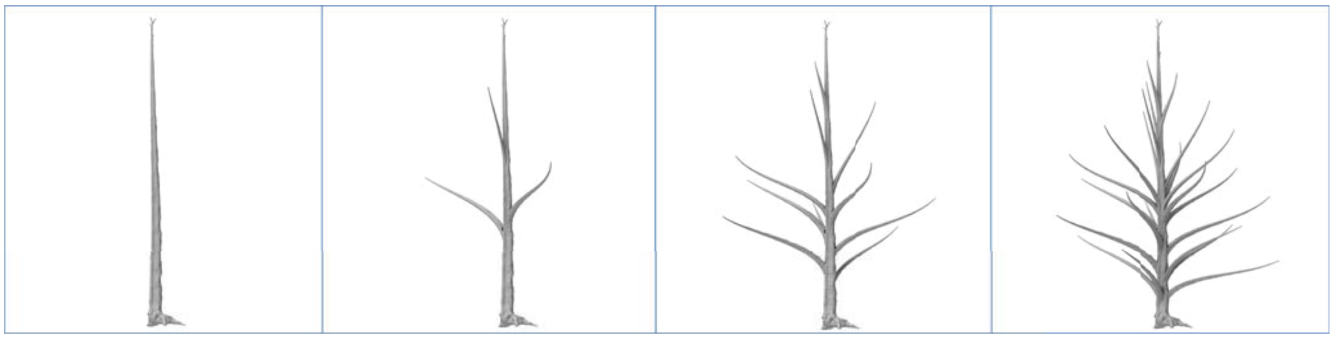

Fig. 15. Connecting branches arbitrary. Our system allows connecting branches to arbitrary surface regions. Here, the user loosely positions tree-cuts in space which connect regions in the surface of a simple trunk.

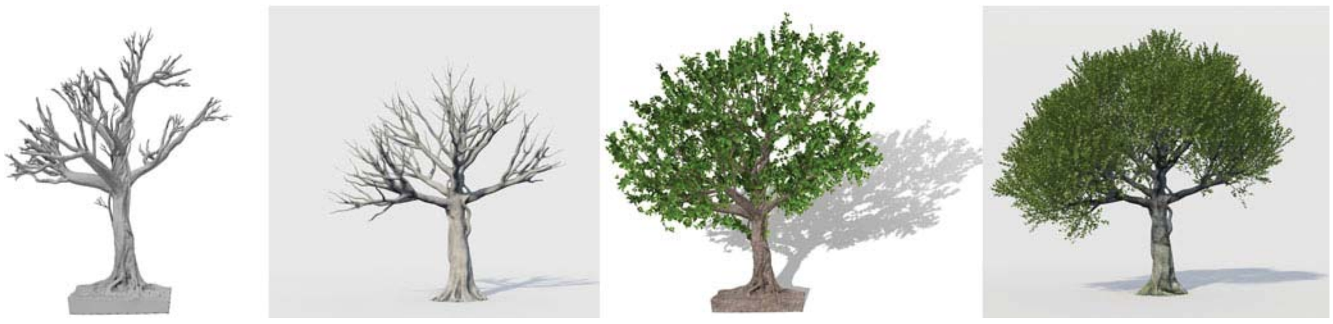

Fig. 16. We compare a tree created by our tool (a, b) with a tree created professionally by Xfrog (c, d). While both trees resemble in terms of their branching structures and geometric complexity, our tree was created faster and required less expertise.

Fig. 13 demonstrates the modeling power of our technique compared with real-life trees. In 13 (top) we model a bonsai tree by following and approximating its 3D structure from a 2D photograph.Similarly, in 13 (bottom), we model a dense Cercis tree. We are also able to reproduce even the highly dense branching structure at the top of the tree by simply positioning small branches without much effort.

In Figs. 1 and 12 we modeled two realistic Ficus trees, zooming into their geometric fine details and branch structures. From a relatively small number of interactive edits, mostly positioning of tree-cuts, we obtain a fairly complex branching structure with large variety and enough details to represent the Ficus faithfully.

We demonstrate that our method can be used to enhance the realism of results generated by state-of-art systems. In Fig. 14a we inflate a cylindrical skeleton and transfer geometric details from our database, resulting in a realistic tree with rich surface geometries. In Fig. 14b, we re-model a tree generated by the commercial software Xfrog. Obviously, since we use tree-cuts from real tree examples, our method generates more compelling, realistic and complex tree geometry. In Fig. 14c, we re-model a bonsai tree created using Google Sketchup. The side-by-side comparison suggests utilizing our method to transform low-quality tree models into realistic models with high-level details and complex structures.

Our tree-cut editing framework is not restricted to cut connecting operations. In Fig. 15 we demonstrate connecting a set of tree-cuts to arbitrary regions on the surface of a main trunk. Our method is general and merely requires to define positions on the surface. Our method then defines a local loop on the surface which connects to the tree-cuts using the the same process.

We also compare our method with professional tree editing systems. We asked an expert tree modeler to model a tree similar to Figs. 16a, 16b in terms of number of branches and their configuration. Using Xfrog and ZBrush it took the modeler more than 2 hours to create the tree model in Figs. 16c, 16d. Their main efforts lied in editing and

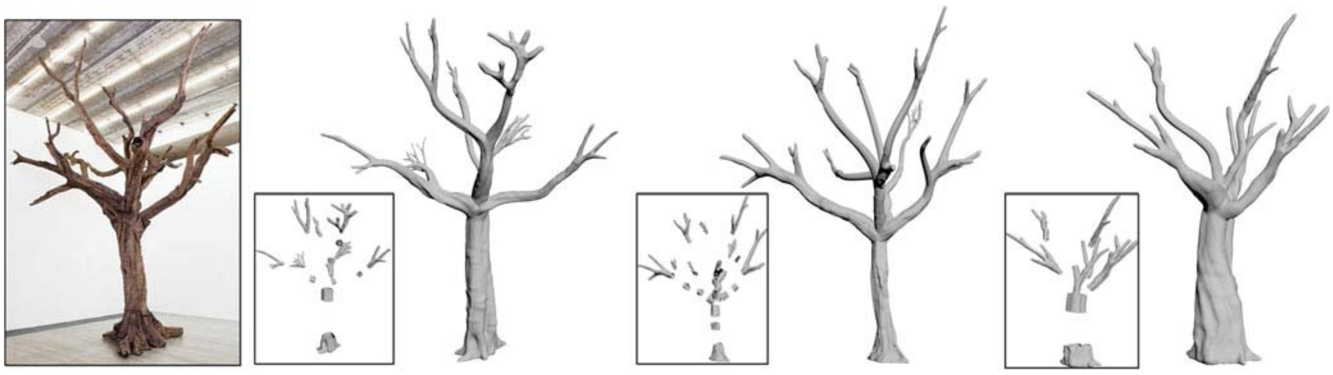
Fig. 17. User study excerpts, showing photography guided tree models achieved by users. Small figures left to trees depict the applied tree-cuts.

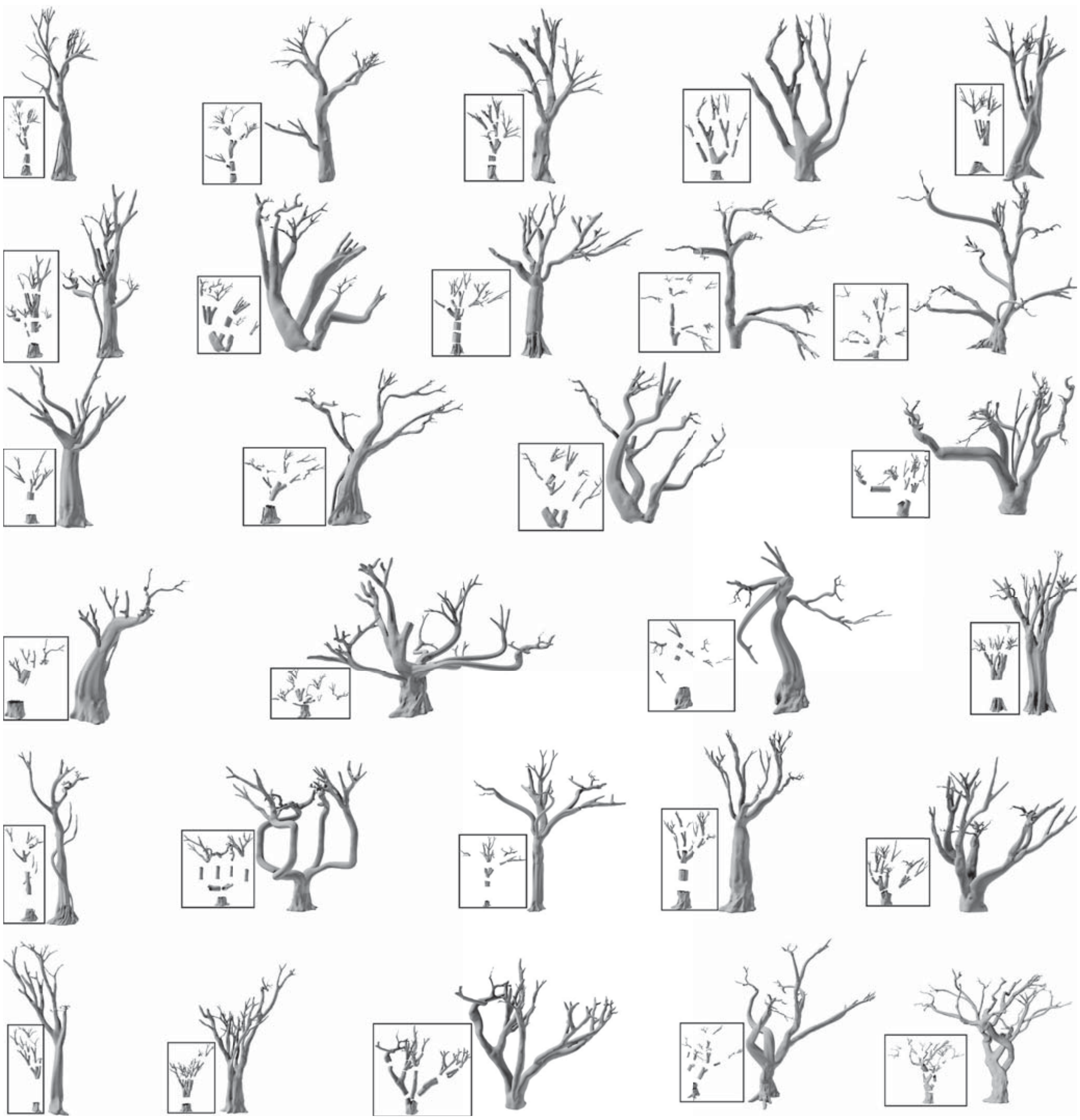
Fig. 18. A gallery of 28 complex tree models generated by 10 user study participants using our interactive tree modeling tool within 20 minutes.

adjusting the tree to match the target tree. In contrast, using our method, it took approximately 20 minutes to generate similar looking branching structures in terms of their configuration and complexity.

5.1 User Study

We performed a user study to obtain feedback and evaluate our system. Ten non-professional participants received a short introductory to the system (10 minutes). Our tool requires not more than two basic interaction actions of cut-positioning in 3D and specifying branch connections. The subjects had basic modeling experience, thus they familiarized easily with the system and reported no problems with positioning and orienting cuts in 3D. All subjects found the connection of the cuts and the lobe-based modeling of small structures easy and intuitive.

To evaluate controllability and expressiveness factors, we designed two tests for each participant. First, modeling a tree from a photo within a bounded time of 5 minutes. We observe that the resulting models bear good resemblance with the target photo (see Fig. 17). Second, creating at least two complex and expressive trees in a bounded time of 20 minutes. Fig. 18 summarizes these results where eight out of ten participants are able to generate three trees given 20 minutes. The gallery in Fig. 18 demonstrates the large expressivity of our tool as a large variety of complex trees was accomplished in a relatively short time.

6 Conclusions and Future Work

We presented an example-based method for generating realistic tree models with complex branching structures. From a collection of example tree models, sub-structures (named tree-cuts) are extracted such that a new tree model is produced after the user manually placed selected tree-cuts. Our system is capable of generating tree models that inherit geometry nuances of the example models, which are normally missing in the existing systems, but are important to human eyes for the perceived realism.

We were only able to acquire a subset of existing trees, due to the limitations in scanning large tree structures in uncontrolled environments. Additionally, our system currently ignores the interrelations between branches, which may be computed using geometrical, physical, or biological constraints. The system could be further amended by tropisms and other global factors.

There are several ways to further improve our system. First, our example-base can be enriched not only by scanning more trees from the real-world, but also by creative editing of existing examples. Second, we plan to further make the user interaction more intuitive and effortless, so that the user is able to concentrate more on creativity. This can be achieved by leveragingstate-of-the-art procedural methods and using the underlying procedural principles to suggest the placement of new tree-cuts. Third, our interaction could be further enhanced using a suggestive system to assist tree editing. Specifically, our system would analyze the tree on-the-fly and suggest tree-cuts at different locations based on the current status. Cuts may be automatically positioned and aligned at feasible locations, thus significantly reducing the user's workload.

Acknowledgments

The authors thank all the reviewers for their valuable comments and constructive suggestions. They are grateful to Jianwei Guo for his help on tree rendering. This work was supported in part by NSFC (61522213, 61232011, 61502471), 973 Program (2015CB352501), Guangdong Science and Technology Program (2015A030312015, 2014B050502009, 2014TX01X033), Shenzhen Innovation Program (JCYJ20151015151249564, CXB201104220029A), National Foreign 1000 Talent Plan (WQ201344000169), Leading Talents of GD Program (00201509) and the Israel Science Foundation. Hui Huang and Baoquan Chen are corresponding authors of this paper.

References

[1] O. Deussen and B. Lintermann, *Digital Design of Nature: Computer Generated Plants and Organics*. New York, NY, USA: Springer, 2010.
[2] M. Okabe, S. Owada, and T. Igarashi, "Interactive design of botan ical trees using freehand sketches and example based editing," *Comput. Graph. Forum*, vol. 24, no. 3, pp. 487 496, 2005.
[3] X. Chen, B. Neubert, Y. Q. Xu, O. Deussen, and S. B. Kang, "Sketch based tree modeling using Markov random field," *ACM Trans. Graph.*, vol. 27, pp. 109:1 109:9, 2008.
[4] (2002). Speedtree. Speedtree web site, interactive data visualiza tion, Inc. [Online]. Available: http://www.speedtree.com/
[5] Y. Livny, F. Yan, M. Olson, B. Chen, H. Zhang, and J. El Sana, "Automatic reconstruction of tree skeletal structures from point clouds," *ACM Trans. Graph.*, vol. 29, pp. 151:1 151:8, 2010.
[6] Y. Livny, S. Pirk, Z. Cheng, F. Yan, O. Deussen, D. Cohen Or, and B. Chen, "Texture lobes for tree modelling," *ACM Trans. Graph.*, vol. 30, pp. 53:1 53:10, 2011.
[7] P. Prusinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants (The Virtual Laboratory)*. New York, NY, USA: Springer, 1996.
[8] J. Bloomenthal, "Modeling the mighty maple," in *Proc. 12th Annu. Conf. Comput. Graph. Interactive Techn.*, 1985, pp. 305 311.
[9] P. E. Oppenheimer, "Real time design and animation of fractal plants and trees," in *Proc. 13th Annu. Conf. Comput. Graph. Interac tive Techn.*, 1986, pp. 55 64.
[10] M. Holton, "Strands, gravity and botanical tree imagery," *Comput. Graph. Forum*, vol. 13, pp. 57 67, 1994.
[11] F. Boudon, P. Prusinkiewicz, P. Federl, C. Godin, and R. Karwow ski, "Interactive design of bonsai tree models," *Comput. Graph. Forum*, vol. 22, no. 3, pp. 591 600, 2003.
[12] A. Reche Martinez, I. Martin, and G. Drettakis, "Volumetric reconstruction and interactive rendering of trees from photo graphs," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 720 727, 2004.
[13] B. Neubert, T. Franken, and O. Deussen, "Approximate image based tree modeling using particle flows," *ACM Trans. Graph.*, vol. 26, no. 3, p. 8, 2007.
[14] I. Shlyakhter, M. Rozenoer, J. Dorsey, and S. Teller, "Reconstructing 3d tree models from instrumented photographs," *IEEE Comput. Graph.*, vol. 21, no. 3, pp. 53 61, May 2001.
[15] P. Tan, G. Zeng, J. Wang, S. B. Kang, and L. Quan, "Image based tree modeling," *ACM Trans. Graph.*, vol. 26, p. 8, 2007.
[16] A. Runions, B. Lane, and P. Prusinkiewicz, "Modeling trees with a space colonization algorithm," in *Proc. Eurograph. Workshop Natu ral Phenomena*, 2007, pp. 63 70.
[17] P. Tan, T. Fang, J. Xiao, P. Zhao, and L. Quan, "Single image tree modeling," *ACM Trans. Graph.*, vol. 27, no. 5, p. 7, 2008.
[18] W. Palubicki, K. Horel, S. Longay, A. Runions, B. Lane, R. Měch, and P. Prusinkiewicz, "Self organizing tree models for image syn thesis," in *Proc. ACM SIGGRAPH*, 2009, pp. 58:1 58:10.
[19] J. Wither, F. Boudon, M. P. Cani, and C. Godin, "Structure from silhouettes: A new paradigm for fast sketch based design of trees," *Comput. Graph. Forum*, vol. 28, pp. 541 550, 2009.
[20] H. Xu, N. Gossett, and B. Chen, "Knowledge and heuristic based modeling of laser scanned trees," *ACM Trans. Graph.*, vol. 26, no. 4, p. 13, 2007.
[21] S. Pirk, O. Stava, J. Kratt, M. A. M. Said, B. Neubert, R. Měch, B. Benes, and O. Deussen, "Plastic trees: Interactive Self adapting botanical tree models," *ACM Trans. Graph.*, vol. 31, pp. 50:1 50:10, 2012.

[22] A. Bucksch and R. Lindenbergh, "Campino A skeletonization method for point cloud processing," *ISPRS J. Photogrammetry Remote Sensing*, vol. 63, no. 1, pp. 115 127, 2008.

[23] A. Bucksch, R. Lindenbergh, and M. Menenti, "Skeltre fast skele tonisation for imperfect point cloud data of botanic trees," in *Proc. Eurograph. Workshop 3D Object Retrieval*, 2009, pp. 13 27.

[24] J. F. Côté, J. L. Widlowski, R. A. Fournier, and M. M. Verstraete, "The structural and radiative consistency of Three dimensional tree reconstructions from terrestrial lidar," *Remote Sensing Envi ron.*, vol. 113, pp. 1067 1081, 2009.

[25] P. Raumonen, M. Kaasalainen, S. Kaasalainen, H. Kaartinen, M. Vastaranta, M. Holopainen, M. Disney, P. Lewis, et al. "Fast auto matic precision tree models from terrestrial laser scanner data," *Remote Sensing*, vol. 5, pp. 491 520, 2013.

[26] X. Zhang, H. Li, M. Dai, W. Ma, and L. Quan, "Data driven syn thetic modeling of trees," *IEEE Trans. Vis. Mach. Comput. Graph.*, vol. 20, no. 9, pp. 1214 1226, Sep. 1, 2014.

[27] M. Okabe, S. Owada, and T. Igarashi, "Interactive design of botan ical trees using freehand sketches and example based editing," in *Proc. ACM SIGGRAPH Courses*, article no. 28 (18 pages), 2006.

[28] S. Longay, A. Runions, F. Boudon, and P. Prusinkiewicz, "Treesketch: Interactive procedural modeling of trees on a tablet," in *Proc. Int. Symp. Sketch Based Interfaces Model.*, 2012, pp. 107 120.

[29] A. Sharf, M. Alexa, and D. Cohen Or, "Context based surface completion," *ACM Trans. Graph.*, vol. 23, pp. 878 887, 2004.

[30] G. Harary, A. Tal, and E. Grinspun, "Context based coherent sur face completion," *ACM Trans. Graph.*, vol. 33, no. 1, pp. 5:1 5:12, 2014.

[31] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin, "Modeling by example," in *Proc. ACM SIGGRAPH*, 2004, pp. 652 663.

[32] P. Merrell, "Example based model synthesis," in *Proc. Symp. Inter active 3D Graph. Games*, 2007, pp. 105 112.

[33] N. Marechal, E. Galin, E. Gurin, and S. Akkouche, "Component based model synthesis for low polygonal models," in *Proc. Graph. Interface*, 2010, pp. 217 224.

[34] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Trans. Graph.*, vol. 32, no. 3, p. 29, 2013.

[35] O. K. C. Au, C. L. Tai, H. K. Chu, D. Cohen Or, and T. Y. Lee, "Skeleton extraction by mesh contraction," *ACM Trans. Graph.*, vol. 27, pp. 44:1 44:10, 2008.

[36] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs, "A search engine for 3d models," *ACM Trans. Graph.*, vol. 22, pp. 83 105, 2003.

[37] J. Lee and T. Funkhouser, "Sketch based search and composition of 3D models," in *Proc. 5th Eurograph. Conf. Sketch Based Interfaces Model.*, 2008, pp. 97 104.

[38] S. M. Yoon, M. Scherer, T. Schreck, and A. Kuijper, "Sketch based 3D model retrieval using diffusion tensor fields of suggestive con tours," in *Proc. Int. Conf. Multimedia*, 2010, pp. 193 200.

[39] T. Shao, W. Xu, K. Yin, J. Wang, K. Zhou, and B. Guo, "Discriminative sketch based 3d model retrieval via robust shape matching," *Comput. Graph. Forum*, vol. 30, pp. 2011 2020, 2011.

[40] M. Eitz, R. Richter, T. Boubekeur, K. Hildebrand, and M. Alexa, "Sketch based shape retrieval," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 31:1 31:10, 2012.

[41] B. K. P. Horn, "Closed form solution of absolute orientation using unit quaternions," *J. Opt. Soc. America A*, vol. 4, no. 4, pp. 629 642, 1987.

[42] P. Jaccard, "Eine neue Auffassung ueber die Ursachen des Dick enwachstums der Baeume," *Naturwiss. Z. fuer. Landwirtschaft*, vol. 8, no. 13, pp. 321 360, 1913.