Mid-Air Finger Sketching for Tree Modeling

Fanxing Zhang*[1] , Zhihao Liu*[1], Zhanglin Cheng†[1], Oliver Deussen[1,2], Baoquan Chen[3], Yunhai Wang†[4]

[1] Shenzhen VisuCA Key Lab, SIAT [2] University of Konstanz [3] Peking Unitversity [4] Shandong University

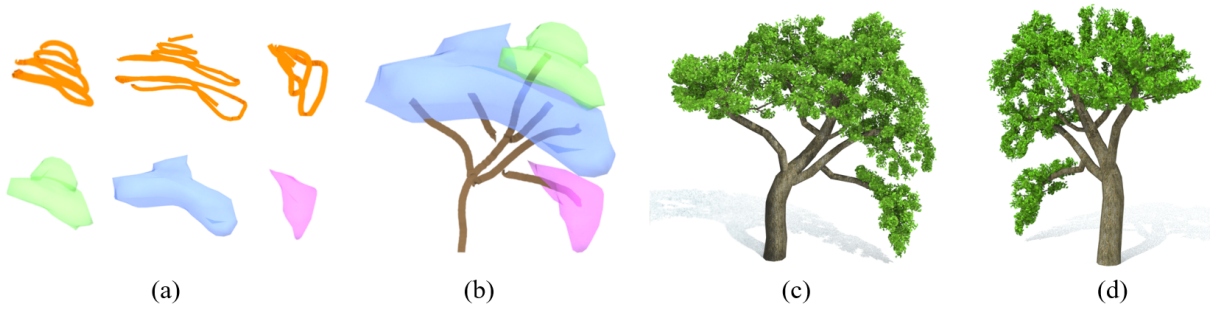


Figure 1: Tree modeling via mid-air finger sketching. (a) The user draws a set of 3D strokes (top row) to produce tree lobes (bottom row). (b) Lobes are arranged in 3D space and main skeletons are drawn to connect them. (c-d) Small branches and leaves are recursively populated in lobes to generate a full 3D tree model: (c) rendering result from front view, (d) side view.

Abstract

2D sketch-based tree modeling cannot guarantee to generate plausible depth values and full 3D tree shapes. With the advent of virtual reality (VR) technologies, 3D sketching enables a new form for 3D tree modeling. However, it is labor-intensive and difficult to create realistically-looking 3D trees with complicated geometry and lots of detailed twigs with a reasonable amount of effort. In this paper, we explore the use of mid-air finger 3D sketching in VR for tree modeling. We present a hybrid approach that integrates freehand 3D sketches with an automatic population of branch geometries. The user only needs to draw a few 3D strokes in mid-air to define the envelope of the foliage (denoted as lobes) and main branches. Our algorithm then automatically generates a full 3D tree model based on these stroke inputs. Additionally, the shape of the 3D tree model can be modified by freely dragging, squeezing, or moving lobes in mid-air. We demonstrate the ease-of-use, efficiency, and flexibility in tree modeling and overall shape control. We perform user studies and show a variety of realistic tree models generated instantaneously from 3D finger sketching.

Index Terms:

Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Virtual reality; Computing methodologies—Computer graphics—Shape modeling

1 Introduction

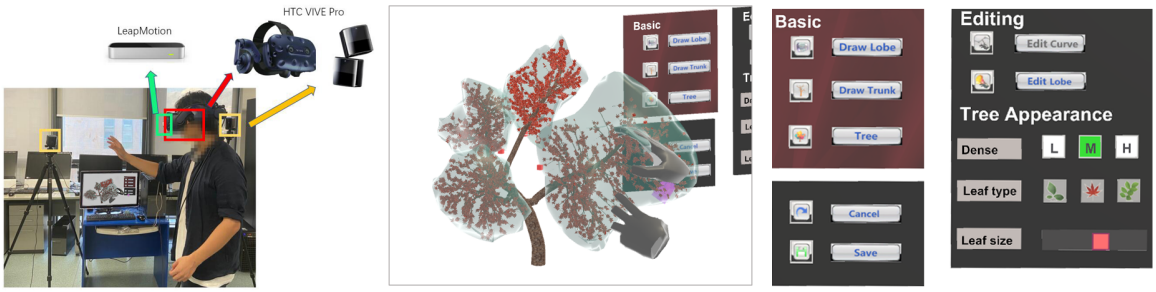
Realistic 3D tree models play an important role in movies, games, and applications such as city and landscape planning. In the past decades, a number of 3D tree generation methods have been developed. These methods can be roughly divided into three categories: procedural tree modeling [26,29], sketch-based tree modeling [6,21], and data-driven tree reconstruction [20,36]. For novice users, it is not easy to generate 3D tree models using procedural modeling or data-based reconstruction methods since these two types of approaches either require specialized knowledge of biology or expensive equipment to capture data from real trees. Therefore, sketch-based tree modeling is a much more intuitive alternative to freely creating 3D tree models. However, previous sketch-based methods are all based on 2D sketches drawn on tablets or desktop screens and suffer from generating plausible depth values and full 3D tree shapes. The ease-of-use and flexibility on shape control of generated tree models remain challenging, especially for content designers or artists during the conceptual design stage. Also, the instantaneous feedback - 3D display of the generated intermediate tree models following through sketching - is essential for creating a fluent user experience.

With the advent of VR technologies, it is possible to directly draw in 3D and intuitively see the drawing result from different views. This enables the user a new form for 3D tree modeling. To the best of our knowledge, this paper presents the first time the use of mid-air finger 3D sketching in VR for tree modeling and editing. Since trees can be complex in geometry with many structural nuances and foliage details, it is extremely labor-intensive and difficult, if not impossible, to draw all the details using a conventional VR modeling system [1, 10, 13, 30, 32]. Fortunately, the most important characteristics of a tree are its main branches and the envelopes that describe dense regions of the foliage (denoted as lobes) [19]. Based on this observation, we present a novel 3D-sketch-based tree modeling method that combines freehand 3D sketches of the user with rule-based branch and foliage generation. The user only needs to focus on the high-level inputs by simply sketching in mid-air to control the overall tree shape (skeletons and lobes), while the detailed branches and leaves are automatically synthesized in the lobes using a rule-based method. We use finger sketching instead of more accurate pen drawing because finger interaction is more natural and agile and it is just enough to accurately depict the overall shape of a tree.

As shown in Figure 2, the user freely draws in mid-air using his/her fingers. The movement of the fingers is tracked by a Leap Motion controller attached to the front of a VR headset and converted to a series of 3D strokes. These strokes are then stitched to generate tree lobes. Subsequently, the user only needs to sketch some more 3D strokes to depict the main branches of a tree. Small branches and twigs in the lobes are synthesized using a rule-based tree growth

* F. Zhang and Z. Liu are joint first authors and contribute equally.
† Z. Cheng and Y.Wang are joint corresponding authors.
Email:{fx.zhang, zh.liu3, zl.cheng}@siat.ac.cn, oliver.deussen@uni-konstanz.de, baoquan@pku.edu.cn, cloudseawang@gmail.com.

2642-5254/21/$31.00 ©2021 IEEE
DOI 10.1109/VR50410.2021.00110



(a) (b) (c)

Figure 2: Our mid-air sketching based tree modeling system. (a) Hardware setup: an HTC VIVE VR glass with Leap motion. (b) The workspace from a user's perspective. The 3D movements of hands are directly captured by Leap motion. (c) The control panels that allow the user to switch the modeling stages and manipulate the tree appearance.

method. In this paper, we design a simple but effective shape-guided method to generate these structures instantaneously during sketching (see an example in Figure 1). The main contributions of our method are:

- An immersive mid-air finger sketching interface for easy-to-use and real-time tree modeling and editing. To the best of our knowledge, this is the first exploration of a VR-based interface for realistic tree modeling.
- A hybrid 3D-sketch-based tree modeling approach that combines the user's high-level inputs (several 3D strokes to define the overall tree shape) and the automatic population of branch geometries. This enables even novice users without botanical skills to focus on the overall tree shapes, which are then automatically evolved into detailed and realistic tree models.
- A flexible tree editing tool that enables local and global shape control based on the abstract representation of a tree model using lobes and skeletons.

2 Related Works

Tree modeling. Trees can be synthesized automatically using rule-based procedural modeling approaches. L-systems [16] and their extensions [27–29] are the most famous recursive algorithms that utilize a set of generative rules to grow complex branching structures. Such algorithms can yield a wide variety of tree species [7]. However, it is difficult to devise a collection of rules to simulate particular tree structures, especially for modelers without understanding the intricate growing processes and their software programming. To circumvent this, Deussen and Lintermann [8, 17] developed the Xfrog technique that combines the power of rule-based algorithms with geometric modeling. Instead of directly growing trees with given rules, Stava et al. [35] presented an inverse procedural method to estimate the optimal parameters for growing trees into a target shape.

To simulate a biologically plausible growth process, Runions et al. [31] proposed the space colonization algorithm by considering the competition of branches for space. Palubicki et al. [26] exploited Runions et al.'s idea and devised a self-organization algorithm to generate more realistic tree models. These methods were subsequently extended to model trees with desired shapes by properly distributing the initial points within a specific envelope [21]. Instead of filling the space with a huge number of attractor points, Wang et al. [38] presented a method by formulating tree growth as an optimization process. All these methods, however, do not meet artists' requirements, since they work fully automatically. Moreover, growth simulations cost lots of computing time and thus are not suitable for real-time applications.

On the other hand, data-driven synthesis is a recent trend in tree modeling that constructs trees from real-world data such as photographs [11, 36], videos [15], and 3D scanned point clouds [20, 41]. But such methods are devised for real tree reconstruction, and it is outside their scope to model virtual trees with the desired shape.

Interactive Tree modeling. Due to the complex branching and foliage structure, the fully manual creation of trees is almost impossible for modelers. Therefore, several interactive sketch-based approaches were proposed to reduce the effort of users. Okabe et al. [23] generate 3D branches from 2D sketches by maximizing distances between branches. Ijiri et al. [12] develop the Sketch L-system where the growth of a tree is controlled by a single user-drawn stroke. Chen et al. [6] infer a full 3D model from freehand sketches using probabilistic optimization with a database of trees as priors. Wither et al. [39] propose a sketch-based interface to construct full 3D branching structures from 2D silhouettes of foliage. TreeSketch [21] is a more advanced system for modeling trees with a multi-touch tablet based on procedural methods. This system provides a set of interactive tools to control the form of the tree, e.g., brushes and lasso tools. However, all these methods infer the 3D information indirectly from a collection of 2D strokes, which is inconvenient for artists that want to directly control the 3D form of a tree. A method to allow this is presented by Onishi et al. [25]. They propose an interactive L-system to create tree models with 3D gesture input. However, this method only supports 3D sketching of main branches; Creating free-form 3D silhouettes of tree crowns is beyond its capability.

3D modeling systems in VR. Sketch-based modeling is getting mature over the past decades [24]. The growth of consumer-grade VR devices enable users to sketch 3D strokes directly in mid-air. As a result, researchers have developed a range of techniques for 3D modeling based on immersive VR systems. Industrial software, such as OculusMedium [22] and ShapeLab [33], allows users to create and manipulate sophisticated shapes with multiple sculpting tools. Other systems [10, 14, 32] enable the modeler to directly draw a range of sweeping surfaces in 3D space. However, these systems require modeling expertise and it costs a lot of time to create complex shapes since the users have to accurately draw all details of the target objects. Recently, Google released the TiltBrush [37] which allows users to create surfaces by collections of nearby 3D strokes. Rosales et al. [30] extended this method and proposed the SurfaceBrush technique for converting 3D strokes to a manifold surface. These methods are also laborious since the users have to draw a mass of stroke ribbons to fully cover the surface. Some researches [2, 3] indicate that drawing precisely using a VR controller is challenging due to the inaccuracy in depth and the user's general lack of spatial orientation. This also makes it difficult to directly sketch details of complex shapes.

Several 3D sketching approaches were proposed to reduce user workload and improve user experience. A 3D sketching system using sketch planes leverages a generative design algorithm to inspire the design, but the success of such interaction depends to a large extent on the accuracy and speed with which users can choose their desired sketch planes [13]. Bohari et al. [4] studied how to recognize a user's stroke-hover intention for mid-air sketching. Arora et al. [1] built the SymbiosisSketch system that combines 3D mid-air interactions with precise 2D sketching to create detailed 3D shapes. However, modeling objects with intricate details like trees is still laborious using this approach. Xing et al. [40] proposed the HairBrush system to model 3D hair. The user just draws several guiding strips, and then a neural network will predict the most plausible hairstyles. Inspired by this idea, we design a hybrid tree modeling approach that enables users to sketch several 3D strokes to depict the overall shape of the foliage (lobes) and the main branches of a tree. The detailed tree structure is then synthesized inside the lobes by a generative algorithm automatically.

3 System Overview

As shown in Figure 1, the user freely draws in mid-air using his/her fingers. The movement of fingers is tracked by a Leap Motion controller attached to the front of a VR headset and converted to a series of 3D strokes. These strokes are then connected to generate the tree lobes. In addition, the user only needs to sketch some more 3D strokes to depict the main branches of a tree.

Figure 2 shows the actual hardware environment and the user interface. Using a VR headset the user freely sketches in mid-air using his/her fingers and hands. The movements are tracked by a Leap Motion controller attached to the front of the VR headset and converted to a series of 3D strokes, then the 3D strokes are connected to generate the tree lobes and finally the full 3D tree model. All the functions are illustrated in the accompanying video.

The user interface contains a number of buttons that are frequently used while sketching or manipulating the tree (cf. Figure 2 (c)). The three buttons of the *Basic* panel are used for switching between the different modeling phases: lobe sketching, trunk sketching, and tree synthesis. If the lobe shapes do not meet the requirements of the user, two editing tools are provided in the *Editing* panel for dragging or squeezing the strokes and lobe meshes. The *Appearance* panel provides options to change the appearance and rendering parameters of the created tree.

The remainder of the paper is organized as follows: we start in Section 4 by describing lobe generation from 3D sketching, followed by our approach for lobe-based tree modeling (Section 5). In Section 6, we discuss the modeling results and user evaluation, followed by conclusions, limitations, and future work (Section 7).

4 Lobe Generation from 3D Sketching

To model a tree, the user first creates a set of lobes to depict the overall shapes of tree crown. A number of VR techniques [10, 14, 30, 37] have been presented for producing 3D shapes by employing a mass of 3D strokes, ribbons or sweeping surface. All these methods, however, are designed to intently create 3D models with fine details from a lot of accurate user sketches that are very time consuming. To improve the speed and ease of use of tree modeling, we develop a simple but efficient modeling algorithm that only requires a few 3D strokes for constructing the mesh of a lobe. The user sketches 3D strokes continually and the strokes are stitched automatically one by one until the desired lobe is created. The mesh of a lobe is updated immediately once a new stroke is added. Finally, a remeshing technique [5] is adopted for obtaining high-quality triangle meshes.

4.1 Stroke Generation and Sampling

A stroke is described as a set of successive points $P = \{p_1, p_2, ..., p_N\}$, which are recorded sequentially when the 3D movement of the finger is larger than a given minimum distance. However, for connecting the strokes and obtaining a triangle mesh, directly using such a large number of points is quite inefficient. Therefore, we utilize a sampling process to obtain a few key points that can preserve the overall shape of the stroke well. Our sampling algorithm is based on the principle that more points are needed at the segments with sharp bends, but less at smooth segments with low curvature. Therefore, we utilize curvature as the sampling weight. Given a stroke consisting of a dense point set P, we first compute the curvature κ_i at each point p_i. Starting from a random point, we collect key points by visiting the point set P sequentially with a variable moving step. The moving step at point p_i is computed as:

$$\Delta(p_i) = \lfloor k \cdot log_2(2.0 + 1/\rho) \rfloor \quad (1)$$

where $\rho = max(\kappa_i, 0.01)$. Thus, the next visited index of a point after the current point p_i is $(i + \Delta(p_i))$. The logarithmic function is used for slowing down the growth of steps when the curvature decreases. The default value of k is 2, and the range of steps is within the interval $[2, 13]$ obtained by the sharpest and the smoothest position ($\kappa_i = +\infty$ and 0.01, respectively). Figure 3 (1.a) and (2.a) show examples of stroke sampling results, where the red dots are the sampled key points.

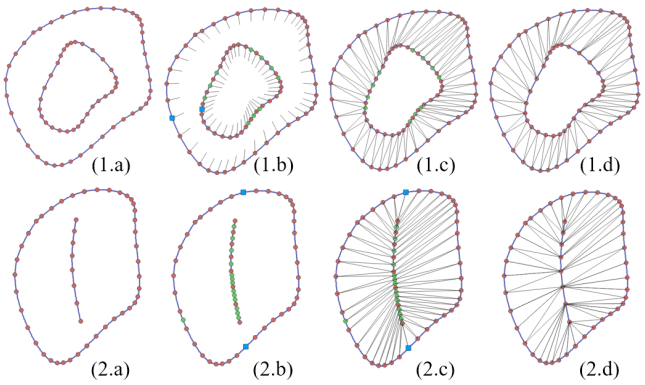


Figure 3: Mesh strip generation from two closed strokes (top row), an open and a closed stroke (bottom row). (a) Two input strokes and sampling points (red dots); (b) Interpolate points (green dots), starting points (blue rectangles), and barycentric directions (gray lines); (c) Initial mesh stripes by connecting point pairs; (d) Final mesh after snapping the interpolated points to their neighboring sampling points.

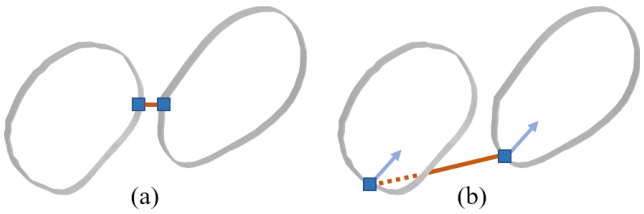


Figure 4: Finding starting points (blue rectangles) to stitch two strokes. (a) Bad starting points: the pair of points have the nearest Euclidean distance; (b) Good starting points: the pair of points have the closest barycentric directions.

4.2 Stroke Stitching and Lobe Generation

Then the lobes can be produced by generating mesh strips between two neighboring strokes. Figure 3 illustrates the process of generating mesh strips between two strokes. There are two types of strokes: closed and open ones. A stroke is considered closed if its starting and end points are nearby. The open stroke is used for enclosing

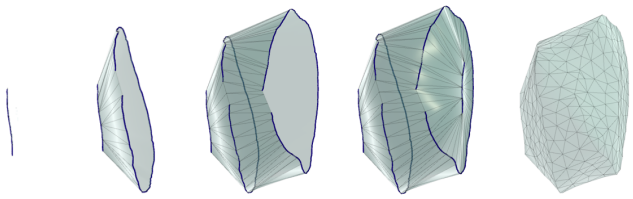


Figure 5: Our system updates the mesh strips progressively according to the user input. From left to right: the first stroke, three progressively added strokes and corresponding mesh strips, the generated lobe after remeshing.

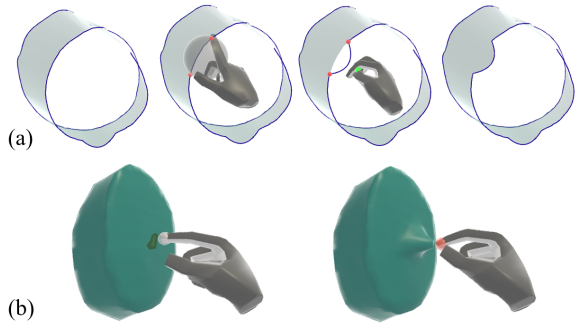


Figure 6: Editing tools. (a) Stroke editing: the user first selects a segment of stroke using the thumb and the index finger, then adjusts the stroke by moving a green control point. (b) Lobe mesh editing using a pinching gesture.

the hole formed by the closed stroke, thus the modelers are able to create a sealed silhouette if needed. Below, we explain how to form a manifold mesh in different conditions.

Condition 1: two closed strokes. The red dots are sampling points. To facilitate the later point-to-point matching, we first equalize the number of points on the two strokes. Specifically, for the stroke with fewer points, we repeatedly insert a new point (in green) between two adjacent points at the maximum gap using cubic interpolation until the numbers of points become the same. Then we match points in pairs and form the initial triangle strip in sequential order. Before that, we need to determine a good starting point pair on the two strokes that are first connected. An intuitive method is to choose two points that have the nearest Euclidean distance [9]. However, this method is not robust and only works when two strokes are close. As shown in Figure 4, the bottom of the right stroke is far away from the bottom but near to the upper of the left stroke, Figure 4 (a) shows a bad point pair with the nearest Euclidean distance. The two starting points should have similar features that reflect the overall shape and trend of the stroke. We found that the barycentric direction is a good feature to find the start point pair. The barycentric direction of a point p_i can be calculated approximately as the average of the directions formed by p_i and the other points on the stroke:

$$dir_{barycentric}(p_i) = normalize(\sum_{k=0}^{N} p_k - p_i), k \neq i \qquad (2)$$

A good starting point pair for stroke stitching can be set as points with the closest barycentric direction, as shown in Figure 4 (b). As shown in Figure 3(c), after finding the starting points (blue rectangles) with the closest directions, the mesh strips are formed by sequentially connecting points in the strokes. Finally, after getting the initial mesh strips, we move each interpolated point (green dot) p_{new} to its nearest sampling point (red dot) p_{ori}. The edges attached to p_{new} are directly devolved to p_{ori}. This operation is to ensure that

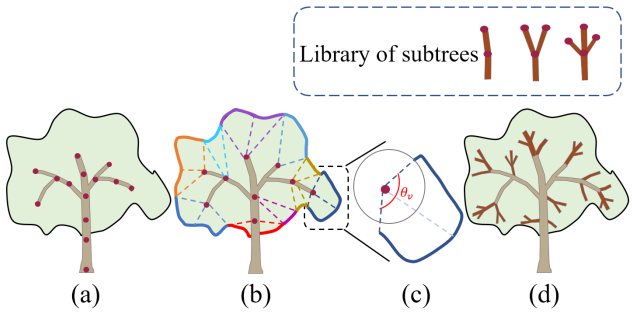


Figure 7: Process of branch growth under the guidance of the lobe for a single iteration. (a) Lobe and the initial main skeleton; (b) The points of the lobe are clustered into groups by associating them to the nearest branch node. (c) A second partition for large clusters. (d) Nodes produce new subtrees directed by the clusters.

the generated mesh is a manifold when stitching 3 or more strokes. The lobe mesh is updated in real time when the modeler sketches new strokes. The added new stroke is stitched to it's closest stroke.

Condition 2: an open and a closed stroke. As shown in the bottom row of Figure 3, given an open stroke A and a closed stroke B, we should find two starting points on Stroke B corresponding to the two endpoints of Stroke A. Such two matching points of Stroke B have the barycentric directions closest with the extension directions of two endpoints of stroke A. Thus, stroke B is divided into two parts located at the two sides of stroke A. Subsequently, we interpolate new points on stroke B to create the same number of vertices as for stroke A on the two sides of stroke B. Then, triangle strips can be generated by connecting each point of stroke A with the matching points on the two parts of B. Finally, we adopt the mesh reduction step mentioned above to remove the interpolated points. Figure 3 (2.d) shows an example of the resulting surface.

With the above strategies, mesh strips can be generated between any two neighboring strokes. For multiple strokes, our system updates the meshes progressively according to the user input. As shown in Figure 5, a new stroke will be connected to the existing mesh immediately once drawn. Note that the drawing order of strokes can be varying. When the user draws a new stroke between two existing strokes, our system will update the local mesh stripes according to the current adjacency. And the final mesh is obtained by incremental remeshing [5], which takes as input a mesh and feature edges (the user-drawn strokes in our case) for producing high-quality meshes.

Lobe editing. We provide several editing tools for the modeler to further adjust and manipulate the shape of lobes. As shown in Figure 6 (a), the user can adjust the stroke using special gestures. The region of interest is determined by two fingers that indicates the endpoints (red dots) of a stroke segment. Then the user can deform the stroke by moving a control point (in green) based on the Bezier curve. Moreover, we also allow users to directly edit the lobe mesh as shown in Figure 6 (b). A pinching gesture is used for adjusting the mesh based on Laplacian surface editing [34, 42].

5 Lobe-based Tree Modeling

Once the lobes have been obtained, we are able to generate the full 3D tree model. Existing shape-guidance methods [21, 38] focus on modeling quality and are very time-consuming. Considering the real-time interaction requirements of VR applications, we developed a simple but effective 3D shape guided approach for modeling a tree in seconds.

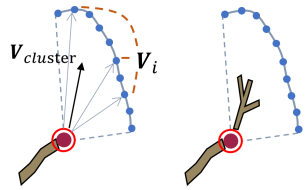


Figure 8: The optimal growth direction $\boldsymbol{V}_{cluster}$ is computed as the weighted average of the normalized vectors $\boldsymbol{V}_i$ formed by the node and attractor point i.

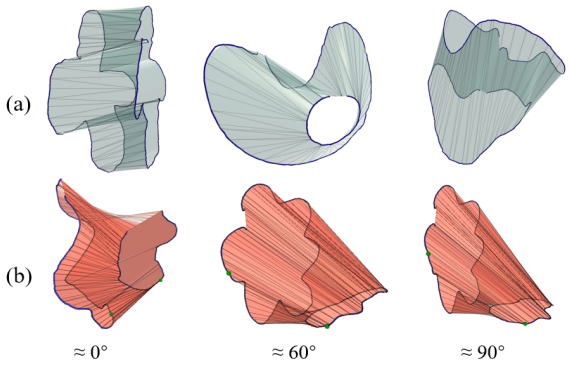


Figure 9: Examples of mesh generation from two free-form strokes (blue lines). (a) Meshes generated from different strokes; (b) Meshes generated from two strokes that are placed with different included angles (0, 60, and 90 degree angles or so).

5.1 Initialization

A tree can be described as a hierarchically organized structure [7,26]. A *node* is a point composing a branch and probably attached by leaves. A *branch* is the part of stem between two successive nodes. The nodes that have the probability to generate new branches are called *active nodes*.

Library of subtrees. In the pre-processing step we built a library that provides a set of predefined elementary subtrees for handling large varieties of trees (see Figure 7 top). To complete a tree, our synthesis process iteratively produces a subtree for each active node. Note that the initial length of all the library subtrees is set to a unit length. During tree growth, subtrees are scaled adaptively according to the size of the tree at the current iteration.

Sketching the main trunk. To better control the branching structure and specify the connections among multiple lobes, the user can also sketch several 3D strokes for depicting main trunks, as shown in Figure 1 (b). To construct a tree skeleton from such 3D strokes, we first take the stroke that has the lowest point as the initial main branch, and then select the most adjacent stroke from the remaining strokes to add into the current tree skeleton successively, until all strokes are used.

5.2 Tree Growth

Our tree growing approach uses a recursive generative algorithm. The basic idea is that each active node will iteratively produce a subtree until the branches fully fill the given lobes. To produce a plausible branching structure, we use the vertices of the lobes as *attractor points* for guiding branch growth. Figure 7 illustrates the growth procedure for a single iteration. It consists of the following two steps:

First step: At the beginning of each iteration, we associate each attractor point of the lobe with the nearest branch node (red dots), this allows us to cluster attractor points into groups. Figure 7 (b) shows the clustering result in which the clusters are marked with different colors. The nodes that are allocated with clusters of attractor points are considered as active nodes, which are able to produce new branches in the current iteration. However, especially during early growth stages, when the number of branch nodes is still small, this tends to produce large clusters. Thus, large clusters are divided into vertical and horizontal directions. Figure 7 (c) describes the vertical split in which the cluster is projected onto a unit sphere. θ_v and θ_h denote the vertical and horizontal span angles. If θ_v or θ_h is larger than a given threshold t, the cluster will be partitioned uniformly using the corresponding direction. Therefore, a large cluster can produce $\lceil \theta_h/t \rceil \times \lceil \theta_v/t \rceil$ pieces. The user can also adjust the value of threshold t to manipulate the density of branches.

Second step: Once the clusters are obtained, we then produce a subtree for each active branch node. As shown in Figure 8, the blue dots are the attractor points associated with the red branch node. $\boldsymbol{V}_{cluster}$ denotes the growth direction, calculated by the weighted average of the normalized vectors $\boldsymbol{V}_i$ formed by the node and the attractor point i, that is, $\boldsymbol{V}_{cluster} = \frac{\sum d_i \times \boldsymbol{V}_i}{\sum d_i}$, where d_i denotes the Euclidean distance from the node to point i. Using the distance as the weight can help to guide the new branch to grow towards bulgy and distant regions and this way balance the branching structure. In addition we consider effects such as gravity, the actual growth direction is calculated as: $\boldsymbol{V}_A = \boldsymbol{V}_{cluster} + \alpha \boldsymbol{V}_{default} + \beta \boldsymbol{V}_{gravity}$, where $\boldsymbol{V}_{default}$ is the head orientation of the node, and $\boldsymbol{V}_{gravity} = (0,-1,0)^T$ is the direction of gravity. The default value of α and β is set to 0.8 and -0.3, respectively. The length of the subtree is proportional to the distance between the node and the lobe surface. A branch node stops growing when its distance to the lobe surface is less than a threshold value of 1e-5 (set heuristically). Note that the clusters of attractor points that are attached to a stop node will not participate in the next iteration.

Branch geometry and leaf population. The tree branching geometry is the polygon model using a set of generalized cylinders. The branch diameters are calculated basipetally and accumulated along tree axes [26]. The leaves (quads with alpha textures) are arranged on branches according to specified tree species, and distributed at the tip nodes along branch directions with a random deviation angle. The users are able to control the tree appearance by adjusting the parameters, e.g., the average leaf length and width, and the number of leaves at one node.

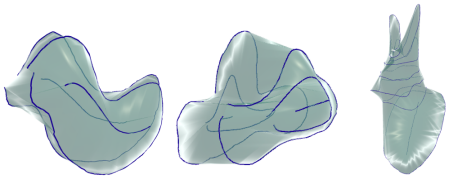


Figure 10: Example lobes generated from multiple strokes (blue lines).

6 Modeling Results and User Evaluation

Our tree modeling system was implemented in C# with Unity. All the results were obtained on a PC with 3.6 GHz i7-4790 CPU.

6.1 Modeling Results

Results of lobe generation. We first discuss the effectiveness of our lobe generation algorithm. Figure 9 shows several example meshes generated by connecting two irregular strokes. The generated surfaces keep manifolds and non-overlapping well. Figure 9 (b) also illustrates the effects when the modeler places two strokes with different included angles. Note that, even if the strokes are almost orthogonal, mesh strips can also be well generated. Figure 10 shows

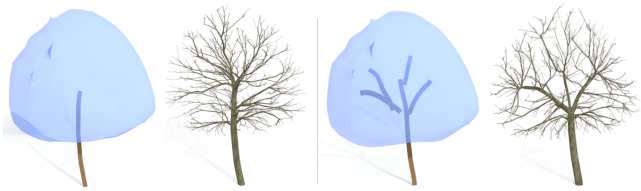


Figure 11: Different main trunks can affect the branching structure in the lobe.

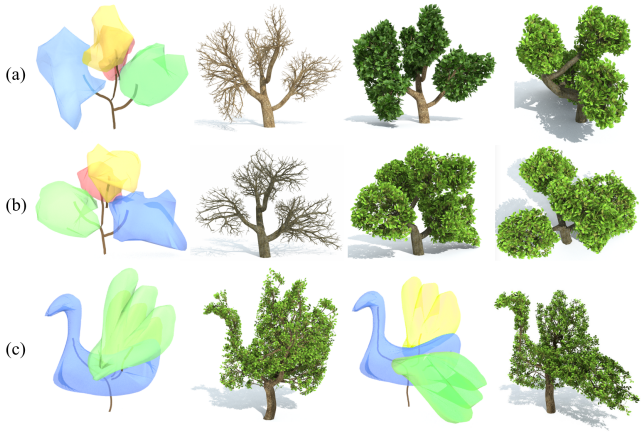


Figure 12: Modeling results with various placements of the lobes.

3 complex-shaped lobes that are constructed only from a small number (4, 6, and 9) of strokes.

Results of tree modeling. To demonstrate the modeling abilities of our approach, we reconstructed a variety of trees with diverse branching types and foliage. Figure 19 shows several modeling results with reference to the photographs of Japanese bonsai trees. Modeling such trees is quite difficult for existing systems since their main trunks are bent heavily and the foliage consists of a number of complicated lobes. Our method benefits from the direct 3D input, and thus it can produce high-quality tree models that resemble the target photos.

The branching structure generated within the lobe can be changed by using different main trunks (see Figure 11). Moreover, the users can also move and rotate the lobes freely to generate trees with different overall shapes. Figure 12 (a-b) shows trees that adopt different placements of lobes. Figure 12 (c) shows two swan-shaped tree models flapping their wings to different angles. Figure 13 shows a specially shaped tree targeted to letters "V" and "R" from front and side view, respectively. The models are easy to create by moving and rotating the four simple lobes in the upper left of Figure 13.

More tree modeling results with four different artistic shapes are illustrated in Figure 20, where trees with rabbit, apple, tank, and goblet shapes are constructed. We also provide an option for users to get plausible trees directly from the silhouette points of external 3D models, as shown in Figure 14.

Overall, each tree takes about 1-5 minutes to create, which depends on the complexity of the lobes. Since the modeling process consists of three steps, including lobe sketching, main trunk sketching, and tree growth (branch propagation), we recorded the time of each stage for several typical trees in Table 1. We noticed that the most time-consuming stage is to design the lobes, especially for the trees with desired artistic shapes. For example, the swan-shaped tree in Figure 12 (c) takes the longest time among all our results for

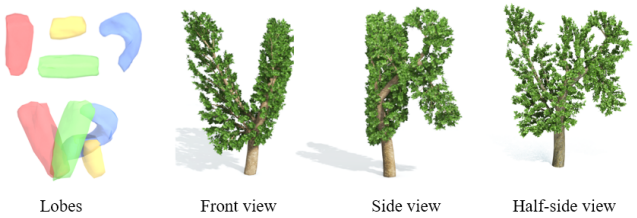


Figure 13: A tree with "V" and "R" shape when viewed from the front and side views.

Figure 14: Sample trees generated directly from external 3D models (Stanford bunny, Nefertiti Bust and a cartoon cow).

designing lobes, since the user needs to intently draw and adjust the strokes to make the appearance similar to a swan. Note that the tree growth will be real-time once the lobes and main trunks are created. Moreover, we found that users tend to decompose a complex shape into a set of small lobes. Table 1 also recorded the stroke number of lobes. Each lobe was just formed by the average of 5 strokes, even for those in complex shapes like the swings and body of swan. This demonstrates our system is able to depict free-form shapes using only several interactions.

Table 1: Time spent in the three stages of tree modeling including lobe generation, main trunk sketching, and tree growth.

	Lobe num.	Stroke num. of lobes	Lobe sketching	Trunk sketching	Tree growth
Fig. 11 (a)	1	5	0.3min	2s	0.096s
Fig. 12 (a-b)	4	22	1.2min	15s	0.310s
Fig. 12 (c)	3	16	4.6min	10s	0.172s

6.2 Comparison to Prior Methods

We compare the modeling results of our approach with prior sketch-based tree modeling methods. Typical sketch-based methods [6, 23] utilize 2D sketches for design and editing of tree branches, and suffer from inferring 3D shape from 2D sketches. Such 2D sketch-based methods require the user to carefully draw a large number of 2D strokes to represent the branch skeleton as shown in Figure 15(a), so the interaction is laborious for users. Moreover, it's impossible for the user to control or edit the 3D tree structure freely since the 3D information is inferred automatically based on simple botanical assumption or a given database. By contrast, with the benefits of direct 3D interaction, the users just need to define the overall shape of foliage by several strokes instead of drawing numerous branching details, and produce similar tree models.

Figure 16 compares our method with TreeSketch [21]. In TreeSketch, trees are generated within an envelope that is inflated from a 2D stroke drawn by the user. Thus, these tree models often look flat and implausible from a side view. However, our result, with the benefit of mid-air sketching, can be designed more artistically and plausibly.

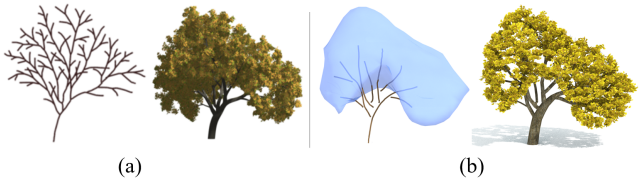


Figure 15: Similar tree models generated by different methods. (a) A tree generated by Chen's method [6] requires a large number of 2D strokes (over 80 strokes) to represent the branch skeleton. (b) A similar tree generated from a single lobe (4 strokes) and a few trunks (9 strokes) using our method.

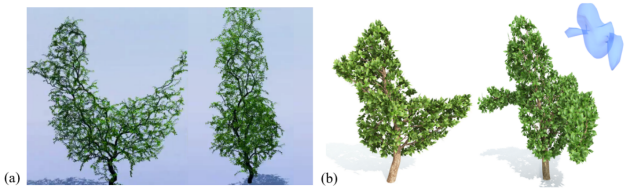


Figure 16: Comparison with TreeSketch [21]. (a) The result of TreeSketch looks flat when observed from a side view. (b) The resulting tree with similar shape created by our system.

6.3 User Evaluation

To obtain feedback and evaluate our system, we invited 10 participants to freely model the trees they wanted to create using our system as well as two sketch based tree modeling systems, namely Okabe et al.'s system [23] (a re-implemented version [18]) and TreeSketch [21] (downloaded from app store). All participants had experience with HMD VR applications. A brief tutorial regarding all functions of the systems was provided before the task. We had a short interview after the task and asked them to fill out the questionnaires for subjective evaluation.

The statistical results are shown in Figure 17. We use the questionnaire to gather subjective ratings of users on three questions [Rating: 1 = Strongly disagree; 10 = Strongly agree]. We are happy to observe that the results indicate our system was consistently better rated than others. All the participants stated that our system was easy to use, and the intuitive mid-air sketching and editing facilitated them to create 3D trees even if they don't have too much modeling experience. Several participants especially mentioned that the mid-air sketching made them feel more creative than 2D sketching, and that making lobes instead of making trees directly did reduce the user workload and improve the modeling efficiency. We collected reasons why they gave lower ratings to other methods. For Okabe et al.'s method, they did not like to intently draw numerous smooth strokes to represent branches. And they thought the 2D interactions in TreeSketch were limited to express free-form shapes, since its brush tool was just implemented by moving a sphere, and the lasso tool only produced flat trees. By contrast, the greatest advantage of our method is that the participants can directly design shapes in 3D and just need to sketch several lobes and main trunks to obtain expressive 3D tree models.

In addition, some participants pointed out the weakness and provided useful suggestions for our system. One of them criticized that the mid-air drawing might lead to slight fatigue when the arms are kept lifted for a long time. Another participant suggested us to provide more parameter control in the future for further adjusting the branching distribution in lobes.

To observe how modelers communicate shapes, the participants were also asked to create a 3D tree model according to the left photo

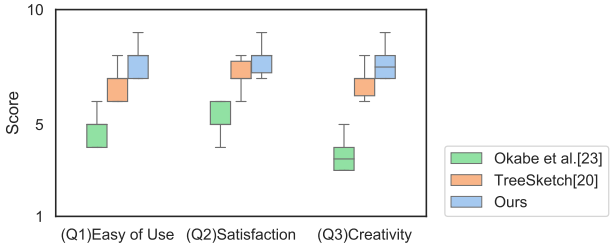


Figure 17: Subjective responses to three questions. (Q1) Easy of use: The interaction is easy to use. (Q2) Satisfaction: I am satisfied with the modeling results. (Q3) Creativity: I felt creative while using this system.

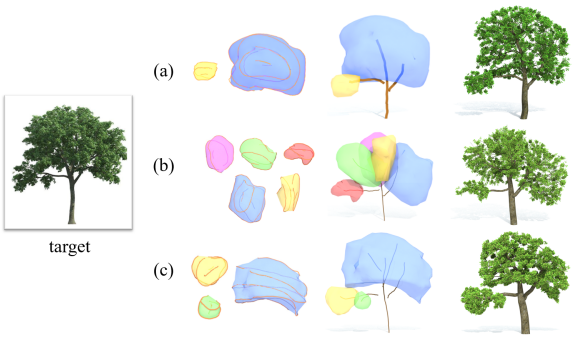


Figure 18: User study excerpts, showing the tree models created by users according to a target photo.

in Figure 18. All the resulting models bear good resemblance with the target photo. We notice that the participants depicted the tree freely using diverse shapes and numbers of lobes, due to the personal difference of observation. And the drawing styles of the same lobe can be also varying. For example, the two blue lobes that represent the same part of foliage in Figure 18 (a) and (c) were designed with different directions of strokes. Therefore, the mid-air interaction can give full play to the imagination and creativity of the users.

7 Conclusions, Limitations, and Future Work

We introduced a novel 3D sketch-based tree modeling system that generates realistically-looking tree models by sketching in mid-air. The system combines easy-to-use input methods in VR with rule-based generation for the heavy amount of details of a tree model. Only a few 3D strokes are needed to define the lobes and main branches of a tree.

A number of problems remain open for further research. The method works well for all those kinds of trees in which the foliage can easily be described by lobes, this is not the case for some species. The current version of the system also does not allow users to edit the bark or other details of the branches. And the immersive VR environment might cause vertigo for some people. As for the future work, we would like to explore more interactive functions, and we will enable users to change more parameters to design more species of trees.

Acknowledgments

This work was supported in part by NSFC (61972388), Shenzhen Basic Research Program (JCYJ20180507182222355), the Leading Talents of Guangdong Program (00201509), the CAS grant (GJHZ1862), and DFG Center of Excellence 2117 "Centre for the advanced Study of Collective Behaviour" (ID: 422037984).

Figure 19: Reconstructed tree models with reference to the photograph of Japanese bonsai trees. From left to right: photograph; lobes and main skeleton; bare tree models without leaves; trees rendered from front and side views.

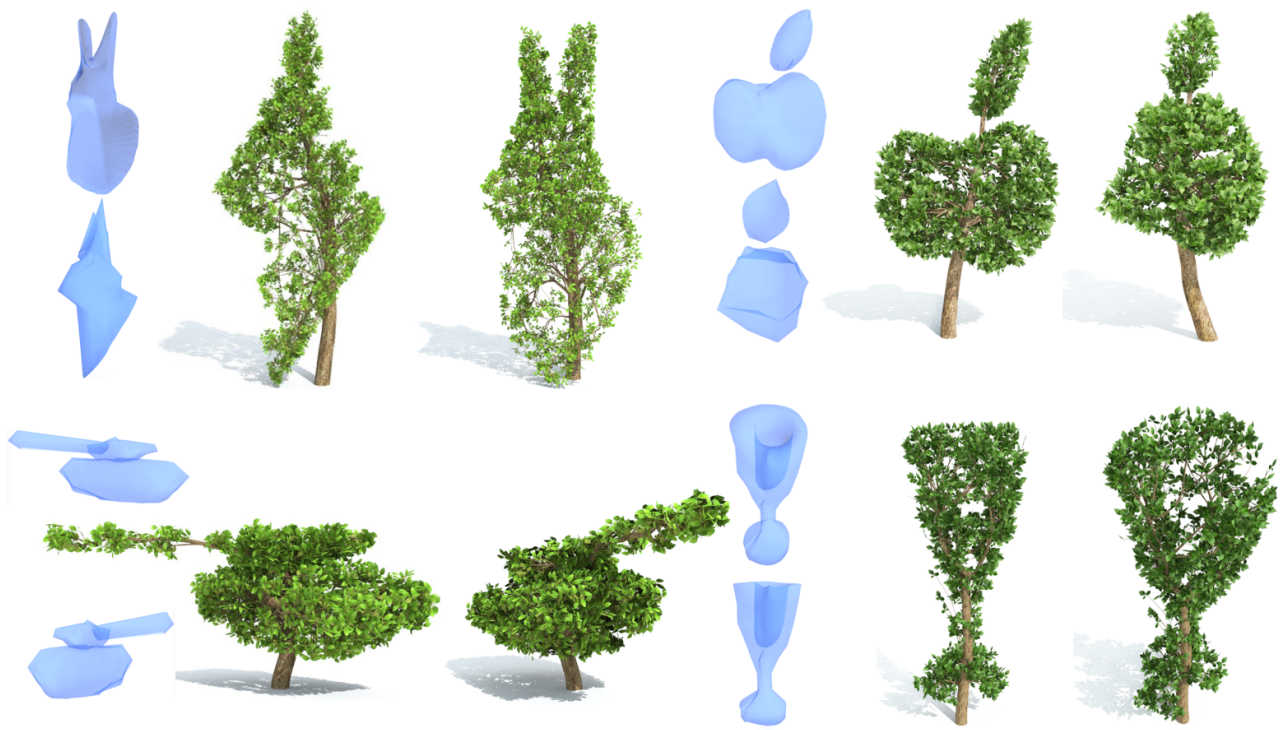


Figure 20: Several tree modeling results with artistic shapes.

REFERENCES

[1] R. Arora, R. Habib Kazi, T. Grossman, G. Fitzmaurice, and K. Singh. Symbiosissketch: Combining 2d & 3d sketching for designing detailed 3d objects in situ. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pp. 1–15, 2018.

[2] R. Arora, R. H. Kazi, F. Anderson, T. Grossman, K. Singh, and G. W. Fitzmaurice. Experimental evaluation of sketching on surfaces in vr. In *CHI*, vol. 17, pp. 5643–5654, 2017.

[3] M. D. Barrera Machuca, W. Stuerzlinger, and P. Asente. The effect of spatial ability on immersive 3d drawing. In *Proceedings of the 2019 on Creativity and Cognition*, pp. 173–186. 2019.

[4] U. Bohari and T.-J. Chen. To draw or not to draw: recognizing stroke-hover intent in non-instrumented gesture-free mid-air sketching. In *23rd International Conference on Intelligent User Interfaces*, pp. 177–188, 2018.

[5] M. Botsch and L. Kobbelt. A remeshing approach to multiresolution modeling. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pp. 185–192, 2004.

[6] X. Chen, B. Neubert, Y.-Q. Xu, O. Deussen, and S. B. Kang. Sketch-based tree modeling using markov random field. In *ACM SIGGRAPH Asia 2008 papers*, pp. 1–9, 2008.

[7] P. de Reffye, C. Edelin, J. Françon, M. Jaeger, and C. Puech. Plant models faithful to botanical structure and development. In *Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '88, pp. 151–158. ACM, New York, NY, USA, 1988.

[8] O. Deussen and B. Lintermann. *Digital design of nature: computer generated plants and organics*. Springer Science & Business Media, 2006.

[9] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin. Modeling by example. In *ACM SIGGRAPH 2004*, p. 652–663, 2004.

[10] GravitySketch. Gravitysketch. `https://www.gravitysketch.com/`, 2018.

[11] J. Guo, S. Xu, D.-M. Yan, Z. Cheng, M. Jaeger, and X. Zhang. Realistic procedural plant modeling from multiple view images. *IEEE transactions on visualization and computer graphics*, 2018.

[12] T. Ijiri, S. Owada, and T. Igarashi. The sketch l-system: Global control of tree modeling using free-form strokes. In *International Symposium on Smart Graphics*, pp. 138–146. Springer, 2006.

[13] R. H. Kazi, T. Grossman, H. Cheong, A. Hashemi, and G. W. Fitzmaurice. Dreamsketch: Early stage 3d design explorations with sketching and generative design. In *UIST*, vol. 14, pp. 401–414, 2017.

[14] D. F. Keefe, D. A. Feliz, T. Moscovich, D. H. Laidlaw, and J. J. LaViola Jr. Cavepainting: a fully immersive 3d artistic medium and interactive experience. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pp. 85–93, 2001.

[15] C. Li, O. Deussen, Y.-Z. Song, P. Willis, and P. Hall. Modeling and generating moving trees from video. *ACM Transactions on Graphics (TOG)*, 30(6):1–12, 2011.

[16] A. Lindenmayer. Mathematical models for cellular interactions in development i. filaments with one-sided inputs. *Journal of theoretical biology*, 18(3):280–299, 1968.

[17] B. Lintermann and O. Deussen. Interactive modeling of plants. *IEEE Computer Graphics and Applications*, 19(1):56–65, 1999.

[18] Z. Liu. A re-implemented version of okabe et al.'s sketch-based tree modeling system. `https://github.com/RyuZhihao123/Sketch-based-Tree-Modeling`, 2019.

[19] Y. Livny, S. Pirk, Z. Cheng, F. Yan, O. Deussen, D. Cohen-Or, and B. Chen. Texture-lobes for tree modelling. *ACM Trans. Graph.*, 30(4), July 2011. doi: 10.1145/2010324.1964948

[20] Y. Livny, F. Yan, M. Olson, B. Chen, H. Zhang, and J. El-Sana. Automatic reconstruction of tree skeletal structures from point clouds. In *ACM SIGGRAPH Asia 2010 papers*, pp. 1–8. 2010.

[21] S. Longay, A. Runions, F. Boudon, and P. Prusinkiewicz. Treesketch: Interactive procedural modeling of trees on a tablet. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*, SBIM '12, pp. 107–120. Eurographics Association, Goslar Germany, Germany, 2012.

[22] OculusMedium. Oculusmedium. `https://oculus.com/medium/`, 2016.

[23] M. Okabe, S. Owada, and T. Igarash. Interactive design of botanical trees using freehand sketches and example-based editing. In *Computer Graphics Forum*, vol. 24, pp. 487–496. Wiley Online Library, 2005.

[24] L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge. Sketch-based modeling: A survey. *Computers & Graphics*, 33(1):85–103, 2009.

[25] K. Onishi, N. Murakami, Y. Kitamura, and F. Kishino. Modeling of trees with interactive l-system and 3d gestures. In *International Workshop on Biologically Inspired Approaches to Advanced Information Technology*, pp. 222–235. Springer, 2006.

[26] W. Palubicki, K. Horel, S. Longay, A. Runions, B. Lane, R. Měch, and P. Prusinkiewicz. Self-organizing tree models for image synthesis. In *ACM Transactions on Graphics (TOG)*, vol. 28, p. 58. ACM, 2009.

[27] P. Prusinkiewicz, M. Hammel, J. Hanan, and R. Mech. L-systems: from the theory to visual models of plants. In *Proceedings of the 2nd CSIRO Symposium on Computational Challenges in Life Sciences*, vol. 3, pp. 1–32. Citeseer, 1996.

[28] P. Prusinkiewicz, M. James, and R. Měch. Synthetic topiary. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pp. 351–358. ACM, 1994.

[29] P. Prusinkiewicz, A. Lindenmayer, and J. Hanan. The algorithmic beauty of plants. *The virtual laboratory (USA)*, 1990.

[30] E. Rosales, J. Rodriguez, and A. Sheffer. Surfacebrush: from virtual reality drawings to manifold surfaces. *arXiv preprint arXiv:1904.12297*, 2019.

[31] A. Runions, B. Lane, and P. Prusinkiewicz. Modeling trees with a space colonization algorithm. In *Proceedings of the Third Eurographics conference on Natural Phenomena*, pp. 63–70. Eurographics Association, 2007.

[32] S. Schkolne, M. Pruett, and P. Schröder. Surface drawing: creating organic 3d shapes with the hand and tangible tools. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 261–268, 2001.

[33] ShapeLab. Shapelab. `https://store.steampowered.com/app/571890/ShapeLab/`, 2018.

[34] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pp. 175–184, 2004.

[35] O. Stava, S. Pirk, J. Kratt, B. Chen, and B. Benes. Inverse procedural modelling of trees. *Computer Graphics Forum*, 33(6):118–131, 2014.

[36] P. Tan, G. Zeng, J. Wang, S. B. Kang, and L. Quan. Image-based tree modeling. *ACM Trans. Graph.*, 26(3), July 2007.

[37] TiltBrush. Google tiltbrush. `https://tiltbrush.com/`, 2018.

[38] R. Wang, Y. Yang, H. Zhang, and H. Bao. Variational tree synthesis. In *Computer Graphics Forum*, vol. 33, pp. 82–94. Wiley Online Library, 2014.

[39] J. Wither, F. Boudon, M.-P. Cani, and C. Godin. Structure from silhouettes: a new paradigm for fast sketch-based design of trees. In *Computer Graphics Forum*, vol. 28, pp. 541–550. Wiley Online Library, 2009.

[40] J. Xing, K. Nagano, W. Chen, H. Xu, L. Y. Wei, Y. Zhao, J. Lu, B. Kim, and H. Li. Hairbrush for immersive data-driven hair modeling. In *the 32nd Annual ACM Symposium*, 2019.

[41] H. Xu, N. Gossett, and B. Chen. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Trans. Graph.*, 26(4), Oct. 2007.

[42] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, and H.-Y. Shum. Large mesh deformation using the volumetric graph laplacian. In *ACM SIGGRAPH 2005 Papers*, pp. 496–503. 2005.