Interactive Invigoration: Volumetric Modeling of Trees with Strands

BOSHENG LI, Purdue University, USA
NIKOLAS A. SCHWARZ, CAU, Germany
WOJTEK PAŁUBICKI, Adam Mickiewicz University, Poland
SÖREN PIRK, CAU, Germany
BEDRICH BENES, Purdue University, USA

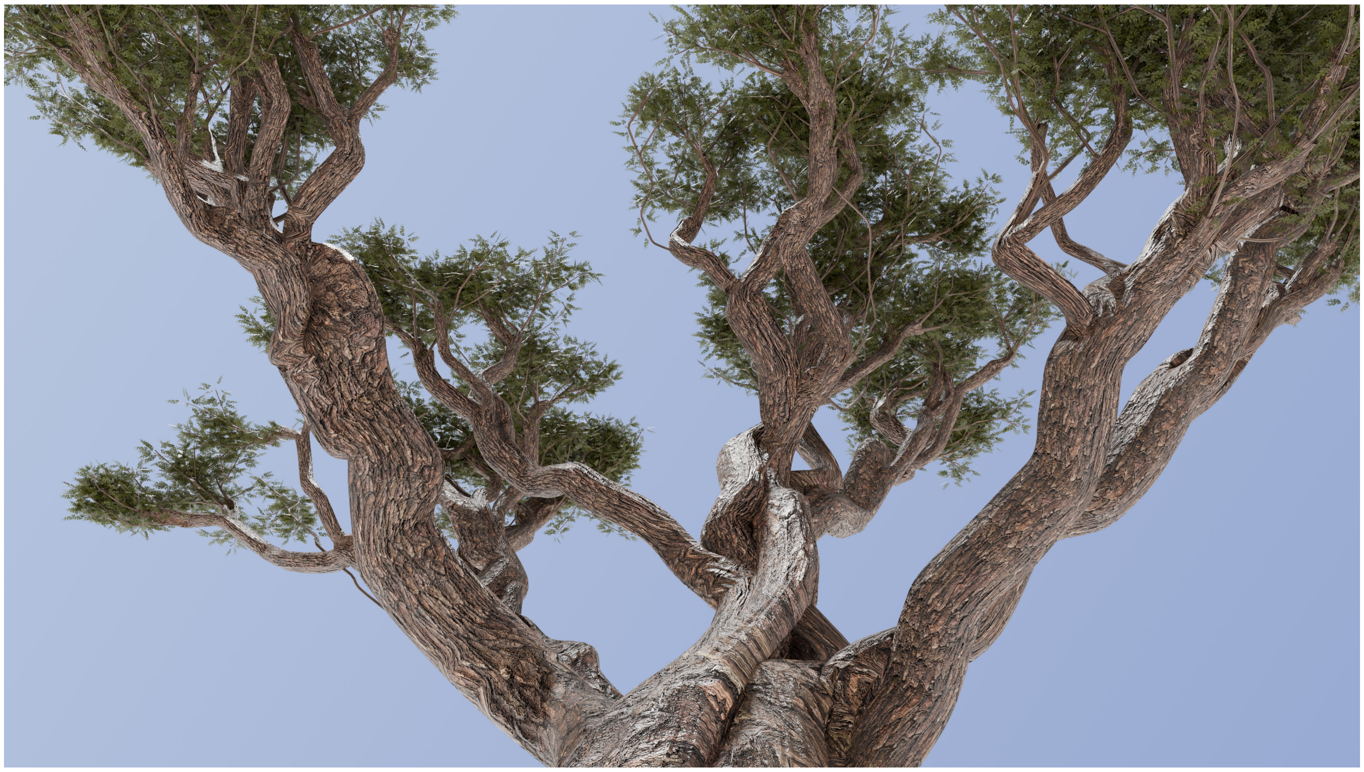


Fig. 1. The intricate branching structure of a mature acacia tree model created with our framework *Interactive Invigoration*.

Generating realistic models of trees and plants is a complex problem because of the vast variety of shapes trees can form. Procedural modeling algorithms are popular for defining branching structures and steadily increasing their expressive power by considering more biological findings. Most existing methods focus on defining the branching structure of trees based on skeletal graphs, while the surface mesh of branches is most commonly defined as simple cylinders. One critical open problem is defining and controlling the complex details observed in real trees. This paper aims to advance tree modeling by proposing a strand-based volumetric representation for tree models. Strands are fixed-size volumetric pipes that define the branching structure. By leveraging strands, our approach captures the lateral development of trees. We combine the strands with a novel branch development formulation that allows us to locally inject vigor and reshape the tree model. Moreover, we define a set of editing operators for tree primary and lateral development that enables users to interactively generate complex tree models with unprecedented detail with minimal effort.

CCS Concepts: • **Computing methodologies** → **Computer graphics**; **Volumetric models**.

Additional Key Words and Phrases: Geometric modeling, Tree Models, Strands, Interaction

ACM Reference Format:
Bosheng Li, Nikolas A. Schwarz, Wojtek Pałubicki, Sören Pirk, and Bedrich Benes. 2018. Interactive Invigoration: Volumetric Modeling of Trees with Strands. In . ACM, New York, NY, USA, 13 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Trees and plants are important visual assets in various application domains, from computer games and movies to urban planning, architectural design, and forestry simulators. Advances in procedural modeling enable an efficient generation of branching structures and plant development by considering biological findings. However, most tree modeling approaches only focus on generating the skeletal graph of a tree, while the lateral growth of branches is neglected. Moreover, the majority of tree models use boundary representation, neglecting the volumetric structure of the trees' inner parts. Creating complex forms, such as holes and twists, using boundary representation is a tedious manual task. Complex and realistic branch geometry is not only a key visual feature but also enables tree models for advanced applications, such as the interaction of plants with autonomous agents or the measurement of biomass in agriculture or forestry [Cieslak et al. 2024]. Therefore, methods for efficiently modeling the volumetric structure of trees seem like an important step for advancing tree modeling in computer graphics and related application areas.

A range of methods address the procedural and developmental modeling of tree form, either through sketch-based interfaces [Longay et al. 2012], with a focus on the environmental response of trees [Měch and Prusinkiewicz 1996; Pirk et al. 2012b], by modeling specific features of trees, such as roots [Li et al. 2023], for reconstructing trees from point sets or images of trees [Livny et al. 2011; Neubert et al. 2007], or by leveraging data-driven techniques to learn tree generation [Zhou et al. 2023]. Only a few approaches focus on modeling the secondary growth of trees [Kratt et al. 2015] or the modeling of trees with strands [Hädrich et al. 2020; Holton 1994] and none of them supports modeling the primary growth, while also maintaining a plausible volumetric structure for the branches.

This paper aims to advance tree modeling by introducing three novel concepts. First, we introduce *strand-based modeling* for defining branch volumes. Strands are defined as fixed-sized pipes running from the tip of small twigs all the way down to the root of a skeletal graph. As each strand occupies space, bundles of them directly define the volumetric shape. Additionally, we enable fast interactive sketching and manipulation of the branch profiles – the cross-section between branch segments – to provide users with a means of control. Second, we introduce *interactive invigoration*, a vigor-based model for the local development of branching structures that allows us to develop branching structures with plausible structural properties interactively. Third, we define operators for the vigor-based and strand-based models, allowing us to interactively design detailed and biologically plausible tree forms.

We describe procedural primary growth via a vigor-based model like other state-of-the-art methods. This ensures a balanced distribution of branches in space where meaningful parameters express distinctive traits of tree species, e.g., pines or oaks. In contrast to other methods, we integrated into such a biologically inspired model a method for describing realistic branch shapes via a novel strand-based model. Specifically, this allows us to capture a variety of naturally looking branching points, twisting branch shapes, and old and decaying branches, as well as the formation of adventitious buds and burls.

Figure 1 shows the branching structure of a mature oak tree model generated with our method. In summary, our contributions are: (1) we introduce interactive invigoration as a novel way to grow and reshape branching structures; (2) we use strands to model secondary branch growth while we let users define the shape of their cross sections; (3) we introduce a novel meshing algorithm for branching structures that operates on strands; (4) we propose a novel set of tree editing operators that enable to design a tree while maintaining its structural integrity interactively.

2 RELATED WORK

Researchers in computer graphics have invested a considerable amount of effort to model trees and plants. Early techniques represent branching structures using rule-based algorithms [Honda 1971; Prusinkiewicz and Lindenmayer 1990], grammars [Aono and Kunii 1984], particle systems [Reeves and Blau 1985], and fractals [Oppenheimer 1986]. Among these methods, L-systems have emerged as a prominent formal framework for modeling plant structures [Prusinkiewicz 1986]. However, although L-systems offer a robust method for generating plant structures, the intricate process of defining production rules to create complex trees is a labor-intensive task that requires a significant amount of expertise or applications of machine learning to detect the rules [Guo et al. 2020; Lee et al. 2024]. Consequently, this complexity often makes their application impractical. Procedural modeling techniques [Weber and Penn 1995] aim to overcome these limitations by combining rule-based approaches with geometric modeling and user interaction [Lintermann and Deussen 1996; Longay et al. 2012]. More recent procedural modeling approaches focus on establishing formalism for phenomenological or self-organizing growth [Guo et al. 2020; Palubicki et al. 2009; Runions et al. 2005], inverse procedural modeling [Stava et al. 2014], urban forests [Niese et al. 2022], dynamic environmental response [Pirk et al. 2012b], the procedural modeling of specific types of plants [Wong and Chen 2015], and learned representations for defining branching structures [Zhou et al. 2023] or foliage of trees [Kałużny et al. 2024; Liu et al. 2021b].

Recent approaches for modeling trees have also expanded toward modeling the dynamics of plants. This reaches as far as modeling animations of growth [Hädrich et al. 2017], the interaction of trees with wind [Habel et al. 2009; Pirk et al. 2014; Quigley et al. 2018; Shao et al. 2021], and fire [Pirk et al. 2017]. A few methods focus on modeling the biomechanic material properties of branching structures for generating realistic animations of tree dynamics [Wang et al. 2017; Zhao and Barbič 2013] or even simulate the effect of plant wilting [Maggioli et al. 2023]. While some approaches also focus on defining volumetric representations for branches, e.g., to enable the combustion of trees through slabs [Pirk et al. 2017], only a handful use strands to define branching [Hädrich et al. 2020; Holton 1994; Kleiberg et al. 2001].

As modeling branching structures is challenging, many methods also focus on reconstructing tree models from different sensor data. Approaches exist to reconstruct trees from one or multiple images [Neubert et al. 2007; Quan et al. 2006; Tan et al. 2008], point clouds [Liu et al. 2021a; Livny et al. 2011; Xu et al. 2007], and videos [Li et al. 2011]. Some methods also use intermediate data structures for the reconstruction process, such as segmentation masks [Argudo et al. 2016] or 3D bounding volumes [Li et al. 2021]. Bradley et al. [2013] utilize point clouds acquired from stereo images and employ a statistical model along with non-rigid mesh fitting to reconstruct dense foliage. In a more advanced approach, Li et al. [2013] take plant reconstruction a step further by incorporating the temporal dimension, tracking events like budding and bifurcation for enhanced accuracy.

Closest to our method are sketch-based techniques, a class of algorithms that aim to simplify the creation of tree models by allowing users to define gestures that guide the procedural modeling of branching structures [Chen et al. 2008; Okabe et al. 2007; Tan et al. 2008], as well as for plants [Anastacio et al. 2006] and flowers [Ijiri et al. 2006]. The most advanced approach is TreeSketch, which seamlessly integrates user-defined sketches with procedural modeling [Longay et al. 2012]. Additionally, sketches can serve as a means to define intermediary representations for tree modeling, such as envelope shapes [Benes et al. 2009], which can direct the modeling process [Wither et al. 2009] or guide particle flows [Neubert et al. 2007]. Pirk et al. [2012a] use growth spaces to first inversely compute growth to define new branching structures [2012a] then. Unlike the existing approaches, we use a recently proposed vigor-based signaling mechanism to grow branching structures [Li et al. 2023] procedurally and combine it with user-sketched profiles for defining the secondary growth of branches.

3 OVERVIEW

Our interactive method uses the model proposed by Li et al. [2023] to express the growth of trees as the result of long-distance signal processing within the tree architecture. This model captures essential biological processes such as phototropism, gravitropism, or bud suppression by existing branches (apical dominance). This growth model describes biological development as the iterative addition and removal of nodes and edges of the plant graph (PG) describing the branching structure. Furthermore, node attributes are recomputed in each iteration as part of a model for long-distance signaling to express physiological changes in response to the environment and the adaptation of the plant structure. The output of this model is a skeleton of the tree with additional attributes, such as the width, age, position, and orientation.

Our goal is to combine the development of a skeletal graph with a strand-based representation to model the volume of branches. A strand is a fixed-radius generalized cylinder running from an endpoint of the skeletal graph down to the root node. We update the strands after each developmental iteration to maintain a plausible volumetric representation. We compute a planar profile that defines the branch cross-section for each node in the PG. To compute the location of a strand within the profile, we compute the packing of strand positions to avoid collisions. Strands from multiple child branches are packed into the parent branch profile by updating their positions with position-based dynamics (PBD) [Bender et al. 2017; Müller et al. 2007]. Furthermore, strand positions are updated based

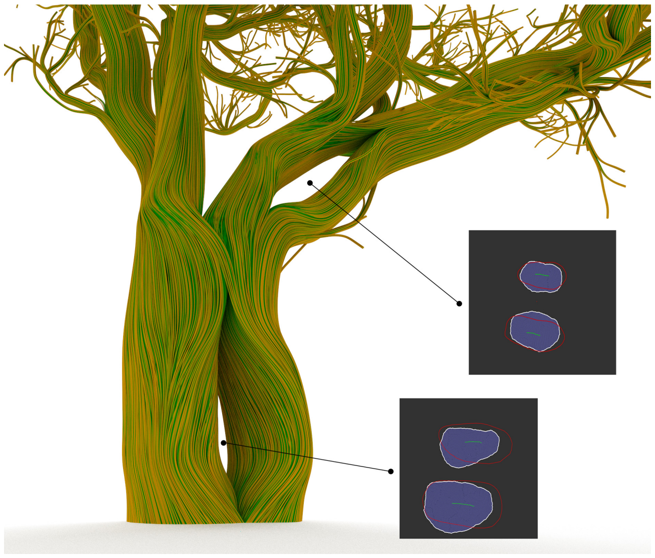


Fig. 2. Multiple branch profiles: our method allows the definition of branch profiles for any internode in the branch graph. Here, we show an example where branch profiles have been defined for the trunk and a major branch.

on user-defined properties. This results in strands with unique positions at every branching point. We then generate strand geometry by treating the strand positions at branching segments as control points for B-splines. This leads to the expansion of the skeletal graph to a 3D tree volume.

Our interactive framework allows users to modify the tree shape during simulation with several intuitive operators defined for either the plant graph or the strands. We allow the modification of signal concentrations in the developmental simulation to invigorate branches. Additionally, parameters from both representations can be adjusted during simulation time, allowing the capture of a wide range of developmental phenomena inherent to plants. This ranges from rare events such as the merging of branches (inosculation) to spiral growth facilitated through user-defined profiles (Fig. 2).

After the strands have been created, we generate a mesh based on the geometric information of all strands, automatically identifying any branching points. The interactivity of our framework extends to the mesh generation phase. Users can fine-tune the smoothness of the mesh to achieve the desired level of detail.

4 METHOD

This section outlines the individual components of our interactive tree editing framework. First, we describe a procedural model of tree development, then provide a detailed account of integrating our novel strand-based model. Finally, we explain how to create a mesh efficiently based on the strand representation.

4.1 Tree Graph Development

We procedurally compute a PG (inset figure) to represent tree development using a recently proposed vigor-based signaling model [Li

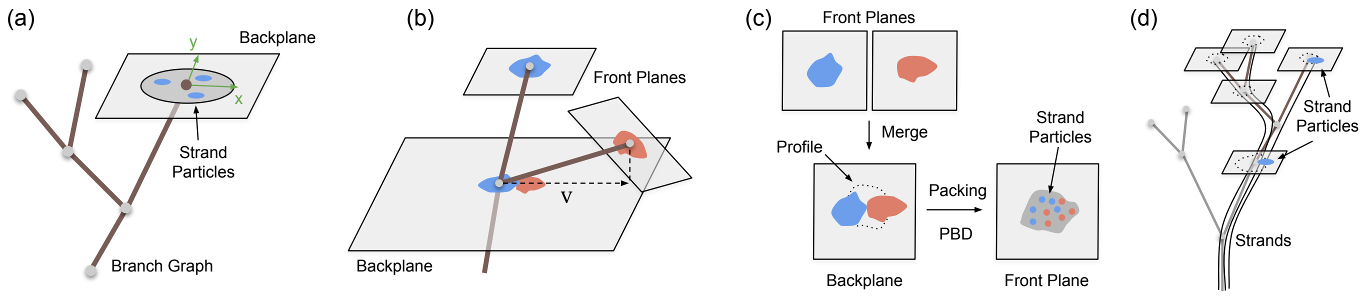
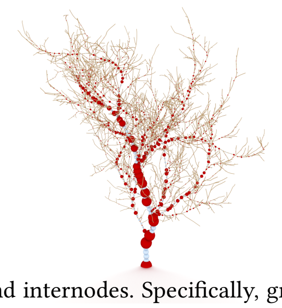


Fig. 3. Overview of the strand particle placement and dynamics in tree branch modeling: (a) Initial placement of strand particles on the 2D local coordinate system at the end nodes of the branch graph. Strand particles are placed within the circular profile boundary. (b) Projection process of strand particles from the 'front planes' of smaller branches onto the 'backplane' of the main branch, with vector **v** illustrating the direction of displacement. (c) Integration and packing of strand particles on the backplane. Strand particles from different branches are merged onto a single plane, followed by a PBD process to resolve collisions and enforce packing constraints within the profile boundary on the front plane. This leads to the final positioning of strand particles that are ready for further geometric processing. (d) After all strand particle positions have been computed for all branch segments, we generate strand geometry using B-splines as trajectories for generalized cylinders.

et al. 2023]. We have adapted the model for the above-ground architecture, as the root development and environmental interactions, are beyond the scope of this work. Specifically, our model defines a PG to represent the plant's architecture at the scale of individual segments.



We define the PG as an acyclic, directed graph $G = \{N, E\}$, where N are the nodes and E the edges connecting the nodes. Our model simulates plant growth through a series of simulation steps that update the attributes of the PG nodes. These attributes encompass the size and state of plant organs, including apical buds and internodes. Specifically, graph nodes define the attributes of position x, diameter d, length l, shoot flux signal S_F, activation signal A, leaf number N_l, lateral bud number N_b, vigor V, optimal growth direction g, and orientation o. We update the node attributes in a sequence of simulation steps, which expresses long-distance signaling within a plant in the model. While the developmental model introduced by Li et al. [2023] accounts for environmental factors through coupled soil and atmospheric models, our implementation does not include these components, as we aim at interactive design. Consequently, our model does not simulate the below-ground root development or the plant's direct interaction with varying environmental stimuli such as light, water, and nutrients.

4.2 Volumetric Modeling with Strands

The volumetric modeling of tree branches using strands begins by establishing local coordinate systems at each node of the PG, which we refer to as backplanes (see Fig. 3a). Each backplane is defined by the position of a node (serving as the origin), the direction of the outgoing edge from that node, and the direction pointing 'upwards,' which gives us the x and y axes. We determine the up vector of the local frame, defining the backplane using parallel transport frames to minimize differences in up-direction vectors. This defines a two-dimensional frame of reference for strand placement around each node. We initiate the process by computing 2D positions $p_i \in P$ of disk-shaped particles with a radius r_i representing future strand positions at various branching segments. We position these particles for a set of strands $S = \{s_1, s_2, \ldots, s_n\}$ within a circular area centered on each node's backplane (see Fig. 3a). For each node, we start by assigning a user-specified number of strands. These strands are then projected first to a second plane of the same branch node called the 'front plane'. Then, in case the underlying branch node is not a branching point (i.e., connected only to one edge from above in the PG), we project the strand particle positions from the front plane to the backplane of the underlying branching node. We continue this process along the chain of edges in the PG down to the root node.

Multiple front planes coming from different branches at each branching point need to be merged into a single backplane via a different method. To merge strands from multiple front planes, we project the strand positions from the front planes of the overhanging branches (child branches) onto the backplane of the underlying main branch. The branch with the highest number of strands is positioned around the origin of this main branch's backplane. For branches with fewer strands, their strand positions are projected further from the origin. The direction of this displacement is determined by the edge projection of the overhanging branch onto the main branch's backplane (see Fig. 3b): $\mathbf{v} = B\mathbf{e}$. Note that $\mathbf{e}$ is the vector representing the edge of the overhanging branch, and B is the 2×2 matrix representing the backplane. The projection gives us a vector in the plane of the main branch, indicating the direction in which the strands from the smaller branch should be displaced. The distance of displacement for these strands is then calculated based on two factors: the furthest extent of strand positions already projected onto the main branch from the larger branch and the largest extent of the strand positions from the currently projected smaller branch.

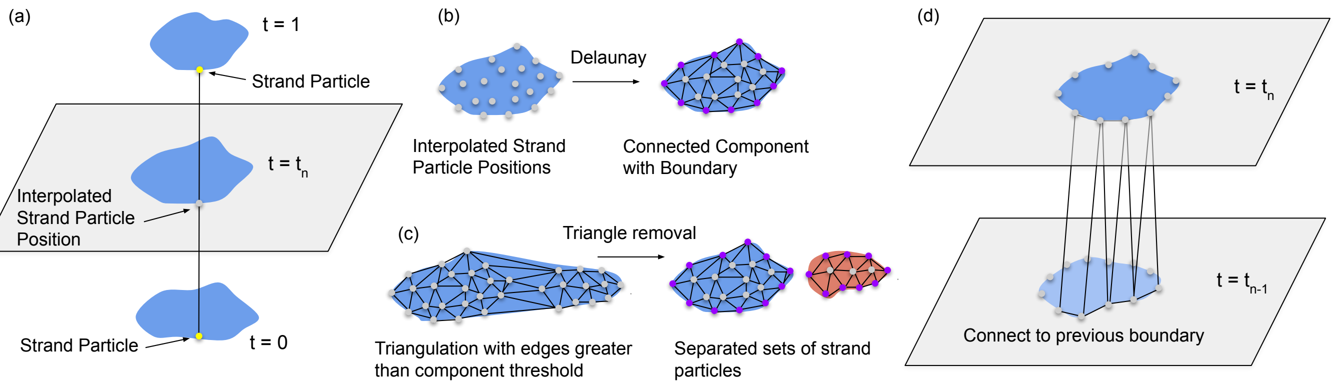

(a)
t = 1
Strand Particle
t = t_n
Interpolated Strand Particle Position
Strand Particle
t = 0
(b)
Delaunay
Interpolated Strand Particle Positions
Connected Component with Boundary
(c)
Triangle removal
Triangulation with edges greater than component threshold
Separated sets of strand particles
(d)
t = t_n
t = t_n-1
Connect to previous boundary

Fig. 4. Schematic representation of mesh construction via strand particle positioning for branch segmentation. (a) Demonstrates the spatial positioning of strand particles, where t=1 represents the upper branch segment, t=0 signifies the lower branch segment, and t=n is an intermediate position showing the interpolated strand particle position between the two segments. (b) Showcases the interpolation of strand particle positions and subsequent Delaunay triangulation, leading to a connected component with a clearly defined boundary. (c) Illustrates the removal of triangles based on edge length criteria exceeding a specific threshold, isolating strand particles into separate groups. (d) Shows the connection of the computed boundaries at spatial position t=n to the previous boundary at t=n-1, thereby constructing a continuous mesh for branch segments.

The displacement for each strand $p_i \in P$ is calculated as:

$$d_i = D_{\text{large}} + D_{\text{small}}, \tag{1}$$

$$p_i' = p_i + d_i \cdot \frac{\mathbf{v}}{\|\mathbf{v}\|}, \tag{2}$$

where p_i is the original particle position, $\mathbf{v}$ the projected vector, D_{large} the furthest extent of strand particle positions from the larger branch, D_{small} the largest extent of the strand particle positions from the smaller branch, and p_i' is the new displaced particle position. Essentially, this means that the strands from the smaller branch are positioned so that they do not overlap with those from the larger branch but instead are adjacent to them (see Fig. 3b, blue and red regions after projection on the backplane).

The disk-shaped 2D particles facilitate the avoidance of intersections between strands by using PBD. During the PBD process, if strands are found outside the user-defined or default profile $\mathcal{B}$ (circular in default cases), they are moved towards the closest point on the profile's boundary. In case they are inside $\mathcal{B}$, they are moved towards attractors and/or the medial axis of $\mathcal{B}$ (see Alg. 1). Attractors are defined by the user as a sequence of line segments within the 2D profile. The medial-axis is computed using the method by Au et al. [2008]. The PBD process may move a border particle in one plane out of the border in the following plane and vice versa, thus moving some strands inside the tree and others outside. To resolve this, we connect all matching strands, and a walker step sweeps over all adjacent planes that might contain unmatched strands to connect them. We use a user-defined maximum number of steps to finalize the strand particle positions for all front planes of all branch nodes. After finalizing the strand segment positions through the PBD process, we generate the geometry of the strands. This is accomplished by treating the finalized positions as control points for B-spline curves, which we use to determine the direction of the generalized cylindrical shapes of strands s_i.

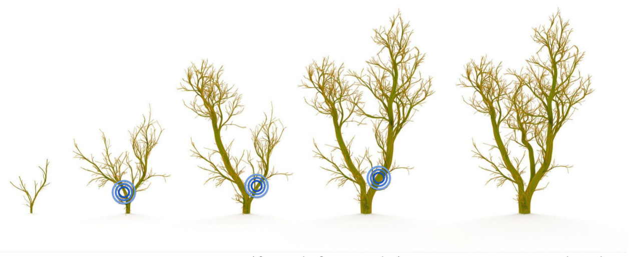


Fig. 5. Interactive invigoration (from left to right): a user continues developing a young tree by injecting vigor (blue circles) into individual branches.

4.3 Strand-based Mesh Generation

Our mesh generation method leverages the spatial information of strand particles to generate a coherent mesh representing the continuity of the branch segments. We use parametric linear interpolation with the parameter t within a branch segment to label the spatial locations of interpolated strand particles, with $t = 0$ corresponding to the position of the strand particle of a lower branch segment and $t = 1$ representing the end of the branch segment (coinciding with the beginning of the next). We interpolate the strand particle positions for all strand particles at user-defined intervals $t \in \{t_1, t_2, \ldots, t_p\}$ to cover the entire length of the branch (Fig. 4a). Following this, a Delaunay triangulation of the strand particle positions is performed. We traverse the particle boundary and match all particles belonging to the same strand. In the case of a single branch segment, the subsequent triangulation results in a triangular network connecting the particle points (Fig. 4b).

We then remove from the triangular network any triangles where the length of at least one triangle edge is larger than a user-specified threshold α. After the triangle removal, the graph may degenerate into multiple connected components, indicating tree branching. We determine this by randomly selecting interpolated strand particles

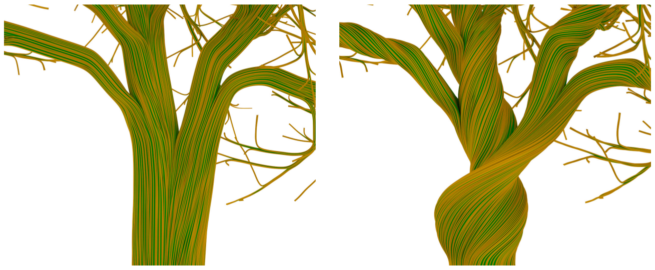


Fig. 6. Twisting operator: branches can be generated mimicking spiral branch growth (right) and for comparison without twisting applied (left).

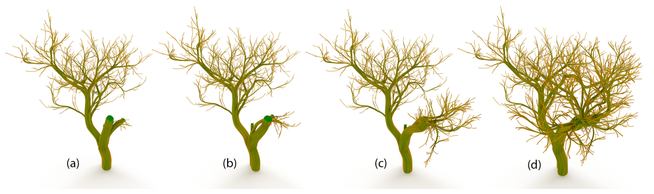


Fig. 7. Adventitious Bud Formation: A branch of a developed tree model is pruned (a). By locally injecting vigor, our method enables mimicking traumatic reiteration (forming new branches from older, already developed branches) by developing adventitious buds (b). After the tree has developed new branches, they continue to develop, leading to a complex tree shape (c, d).

from a set of unvisited particles U to compute the connected component, where we flag all interpolated particles as visited until U becomes empty. The boundary of each connected component is found by a traversal around it (Fig. 4c). The final step connects the boundaries of the interpolated strand particle positions at each level to obtain a surface mesh for the current branch segment (Fig. 4d). This is achieved by first matching all boundary strands in both planes. However, these matches can sometimes cross each other. To resolve this, we search for such crossings and swap the matched strands until this no longer occurs. Finally, we create triangles between two matched strand particles by sweeping over the boundary, always maintaining two active vertices, one on the upper and one on the lower plane, and connecting them to the next available particle to form a triangle. This vertex then becomes the new active vertex in its corresponding plane. We repeat this process for all branch segments contained in the PG until meshes for all branches have been generated.

5 INTERACTIVE TREE MODELING

Our implementation of natural tree development is computationally efficient enough to allow users to interactively control the tree model generation. We introduce several tree editing operators that allow the intuitive adjustment of tree-shape features. We focus on a number of natural phenomena important for tree development, including branch twisting, invigoration, pruning, and formation of adventitious buds. In particular, the operators that we defined are:

Interactive Invigoration. An important aspect of interactive tree modeling is the ability of users to influence the growth and shape of tree models directly [Longay et al. 2012]. The *Interactive Invigoration* operator is designed to inject vigor into the tree growth simulation, allowing users to enhance the growth of branches at specific positions. This is achieved by integrating a tree graph operator that responds to user input within the graphical interface. When a user clicks on a particular location on the tree model, the *Interactive Invigoration* operator interprets this action as a signal to initiate new growth. Specifically, the operator adds vigor to the selected node corresponding to the click duration. The increased vigor at the nodes subsequently results in longer internode formation and a higher amount of lateral buds as described in Li et al. [2023]. This allows for the real-time creation and modification of tree models by amplifying the growth of arbitrary branches, which conform to our model's biologically plausible growth rules (Fig. 5).

Adventitious Bud Formation. Another operator in our framework simulates the spontaneous generation of buds on mature sections of the tree. This phenomenon, often observed in nature as a response to environmental stress or injury, is replicated in our model to increase the detail of the tree structure. Upon clicking on a node with the operator selected, new lateral nodes are added to the PG connecting to the clicked node. We calculate the growth direction and other node attribute values based on the default setting of our model. The growth directions of the new branches are determined by the local surface normal and the phototropism factor that simulates the branch's tendency to grow towards light used by Li et al. [2023]. These new nodes develop into branches as part of the subsequent tree growth simulation in a spiral phyllotactic pattern from around the main branch segment associated with the node (Fig. 7).

Branch Twisting Operator introduces a twist to the branches, mimicking the natural torsion observed in many tree species. This operator allows users to apply a rotational transformation to the front planes of neighboring branch nodes, creating the characteristic spiraling effect of the bark and branching structure. The operation is performed by first converting the strand particle positions from Cartesian to polar coordinates relative to the origin of the front plane. A user-defined rotation angle is then added to the polar angle component of the upper branch node (effectively rotating the backplane). The subsequent B-spline calculation to define strand geometry results in the branch's spiraling appearance around its longitudinal axis. Users can apply varying degrees of twists across different branches to achieve a diverse range of natural appearances and specify them also for selected branches only (Fig. 6).

6 IMPLEMENTATION AND RESULTS

In this section, we describe implementation details and modeling results. We implemented our method in C++ with OpenGL for interactive visualization, and photorealistic results were rendered using Blender. We used a computer with an i9-10850k, RTX 3090, and 32GB of memory to generate all models presented in this work.

We employ the method of Runions et al. [2007] based on parallel transport frames to generate leaf positions and orientations. We apply a rotation to the local frame of the branch segment to obtain a new orientation for leaf geometries. The geometry of a leaf is defined as a quad, which is textured using alpha blending.

We used the following parameter value settings to implement the PBD step. Delta time is set to 0.002 with a damping factor of

ALGORITHM 1: Algorithm for Volumetric Modeling with Strands and Profiles

```
Input: Skeletal graph representing tree topology.
Output: Volumetric model of the tree with detailed branch
        structures.
Initialization:
    Initiate strand particles with a random position at end nodes of
    the skeletal graph.
for each node $N$ in skeletal graph do
    Trace strands from $N$ to root, positioning along branch paths.
end
for each branch segment $B$ do
    Resolve local interactions and non-overlapping strand placement
    at $B$ (Eqs. 1 and 2).
    Employ Particle-Based Dynamics for collision resolution.
end
Profile Integration:
    Apply user-defined or default profiles for cross-sectional shapes.
    Adjust strands to adhere to profiles using PBD, aligning with
    profile boundary or attractors as necessary.
for each particle $p_i$ in $P$ do
    if $p_i$ is outside boundary $\mathcal{B}$ then
        Let $q_i$ be the point in $\mathcal{B}$ closest to $p_i$
        Move $p_i$ towards $q_i$ by a factor of $\beta$:
        $p_i \leftarrow p_i + \beta(q_i - p_i)$
    else
        for each attractor $a_k$ in $A$ do
            Move $p_i$ towards $a_k$ by a factor of $\gamma_k$:
            $p_i \leftarrow p_i + \gamma_k(a_k - p_i)$
        end
        Let $m_i$ be the closest point on medial axis $M$ to $p_i$
        Move $p_i$ towards $m_i$ by a factor of $\delta$:
        $p_i \leftarrow p_i + \delta(m_i - p_i)$
    end
end
Strand Geometry Generation:
    Generate B-spline control points from strand positions.
    Create final strand geometry along each branch segment.
Finalization:
    Assemble complete tree model from all generated branch
    segments.
end
```

0.02 to reduce oscillations. The maximum velocity is capped at 60 to handle cases of overlapping particles and maintain the simulation's integrity under intensive scenarios. Particle softness is set to 0.1 for minimal overlap and energy absorption during collisions. The attraction acceleration is kept high at 40,000 to facilitate rapid particle arrangement. The maximum number of iterations is assigned to five per particle, balancing the resolution and computational load. The simulation's timeout for branching and junction packing is set to range from 300 to 6,000 steps, depending on the complexity of the branch internodes. For profiles altered by users (e.g., by providing contours), the timeout is extended to a range of 1,500 to 10,000 steps to accommodate additional geometric complexities. For the results shown in this paper, we distributed between one and five strands per end node and generated five or more intervals ($p \geq 5$) per branch segment. Finally, we apply Laplacian mesh smoothing for the final mesh to remove mesh artifacts.

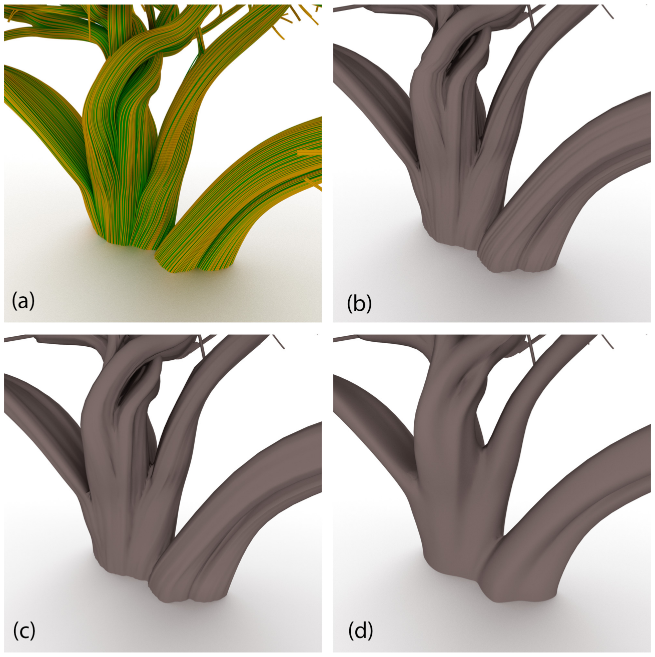


Fig. 8. Having a single skeletal graph strand representation (a), the user can control the separation of branches at bifurcation points using small values for α resulting in early separation (b), medium values for medium separation (c), and high values for α resulting in late separation (d).

6.1 Algorithm Details

We provide the pseudo-code of our interactive strand-based method for generating tree models in Alg. 1. We first initialize the PG (lines 1-9), then we compute the positions of the strand particles (lines 10-27), and finally, we compute the geometry of strands (lines 28-33). Please note that the symbols used in the description of the pseudo-code are: q_i the closest point on $\mathcal{B}$ to p_i, used for aligning particles outside the boundary; the coefficient β controls the rate at which particles outside $\mathcal{B}$ are moved towards q_i. For particles inside $\mathcal{B}$, $A = \{a_1, a_2, \ldots, a_m\}$ denotes the set of attractor points that exert an influence on the particles, with γ_k being the individual influence coefficients for each attractor point a_k; The medial axis of the profile is represented by M, and m_i is the closest point on M to p_i, with δ as the coefficient determining the influence of M on the particles' movement. Finally, the updated position of a particle after adjustment is given by p_i.

Particles in the system move at relatively low speeds during the PBD collision resolution step. Therefore, we do not use substep or continuous collision detection to improve the stability and accuracy of the simulation, which requires more calculation time. The number of steps in our implementation is user-specified. Higher step numbers allow the modeling of smoother branch shapes.

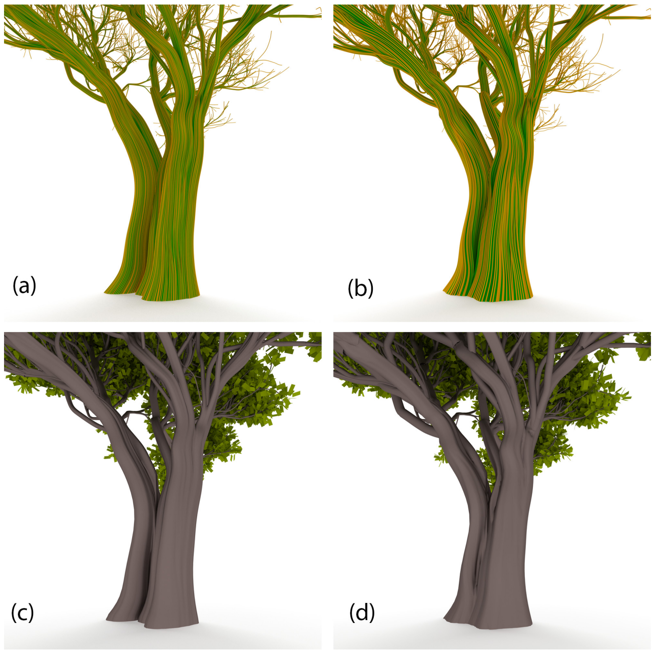


Fig. 9. Closeups of trunks for a tree model with a fixed number of nodes (21K) with one (b,d) and four strands per end node (a, c). Using more strands gives a smoother and more separated branch appearance.

6.2 Results

User-defined Profiles. Our framework allows users to sketch 2D profiles for backplanes of individual nodes interactively. This allows for the generation of a variety of different branch shapes with our strand-based model. In Fig. 13a and b, we show a tree with elongated trunks from two perspectives. By sketching a c-shaped profile, users can model cavities Fig. 13c. Users can model complex inosculations Fig. 13d and e if multiple profiles are provided. Precise control over profiles becomes particularly important when designing the shape of mature trees, which are often characterized by intricate trunk shapes Fig. 16.

Strand-based Control of Branch Shape. Our strand-based model relies on a few parameters, one of them is the maximum number of PBD steps during the strand particle packing step. By selecting different values, our method can express a variety of branching point shapes. In Fig. 14c-f, we show the same branching point in a tree model once with a low number of maximum steps (c, d) and a high number of steps (e, f). The branches separate earlier in the former and later in the latter, leading to visibly different yet plausible organic shapes of branches.

Ablation Study. Our volumetric approach considerably increases the realism and controllability of branch shapes. Fig. 10 shows a comparison to Li et al. [2023] who use generalized cylinders to compute the tree mesh. For the same skeletal graph, our method produces more detailed and realistically bent branch shapes (Fig. 10, right). In contrast, a method based on generalized cylinders produces less natural branch shapes (Fig. 10, left).

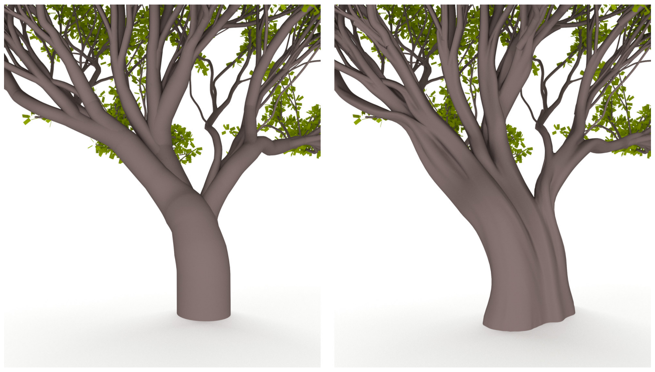


Fig. 10. Comparison of our method (right) with the generalized cylinder method (left) by Li et al. [2023]. The tree model used to calculate the two meshes depends on the same skeletal graph. Our method produces more complex and natural-looking branch shapes.

Branch Point Modeling. The PBD-based mesh generation allows the user to specify the shape of a branch bifurcation, which is an important aspect of accurately representing detailed tree forms [Galbraith et al. 2004]. The tree models in this example use the same skeletal graph and strand model (Fig. 8a), but the shape is controlled by a single parameter. Specifically, we use the parameter α to determine a threshold controlling the removal of triangles based on their edge lengths during the Delaunay triangulation. A low value for α specifies an early separation of the mesh branches as shown in Fig. 8b. Increasing the values of α delays the separation (Figs. 8c and d) as observed in many oak or elm trees.

Complex Examples. The combination of procedural modeling of tree development and our detailed volumetric representation allows the synthesis of a large variety of plausible shapes. In Fig. 11, we demonstrate our framework's capability to construct diverse tree architectures. Fig. 11a exemplifies how the Interactive Invigoration operator can guide the development of branches, allowing users to influence tree growth actively. Variations in naturalistic tree responses are evident in Fig. 11b, c, where the Twisting operator is employed. Fig. 11d showcases a complex branching structure that can be achieved by carefully applying user-defined profiles. In Fig. 11e, we show the design of an unusual tree shape by employing a combination of the Twisting operator and sketched profiles. A Joshua tree's stark and characteristic shape is captured in Fig. 11f, highlighting the framework's versatility. Fig. 11g, h depict the complex and mature growth patterns that our system can replicate. In Fig. 14a, b, we use a combination of the Invigoration and Twisting operators to craft a tree with a split trunk, demonstrating the detailed control our framework offers for tree modeling.

6.3 Runtime Performance

We analyzed the computational performance of our tree generation method across various metrics such as number of nodes, strands, particles, profile calculation time (s), and mesh generation time (s) as shown in Table 1. The data indicates that the time required for profile calculation and meshing increases as tree complexity increases.

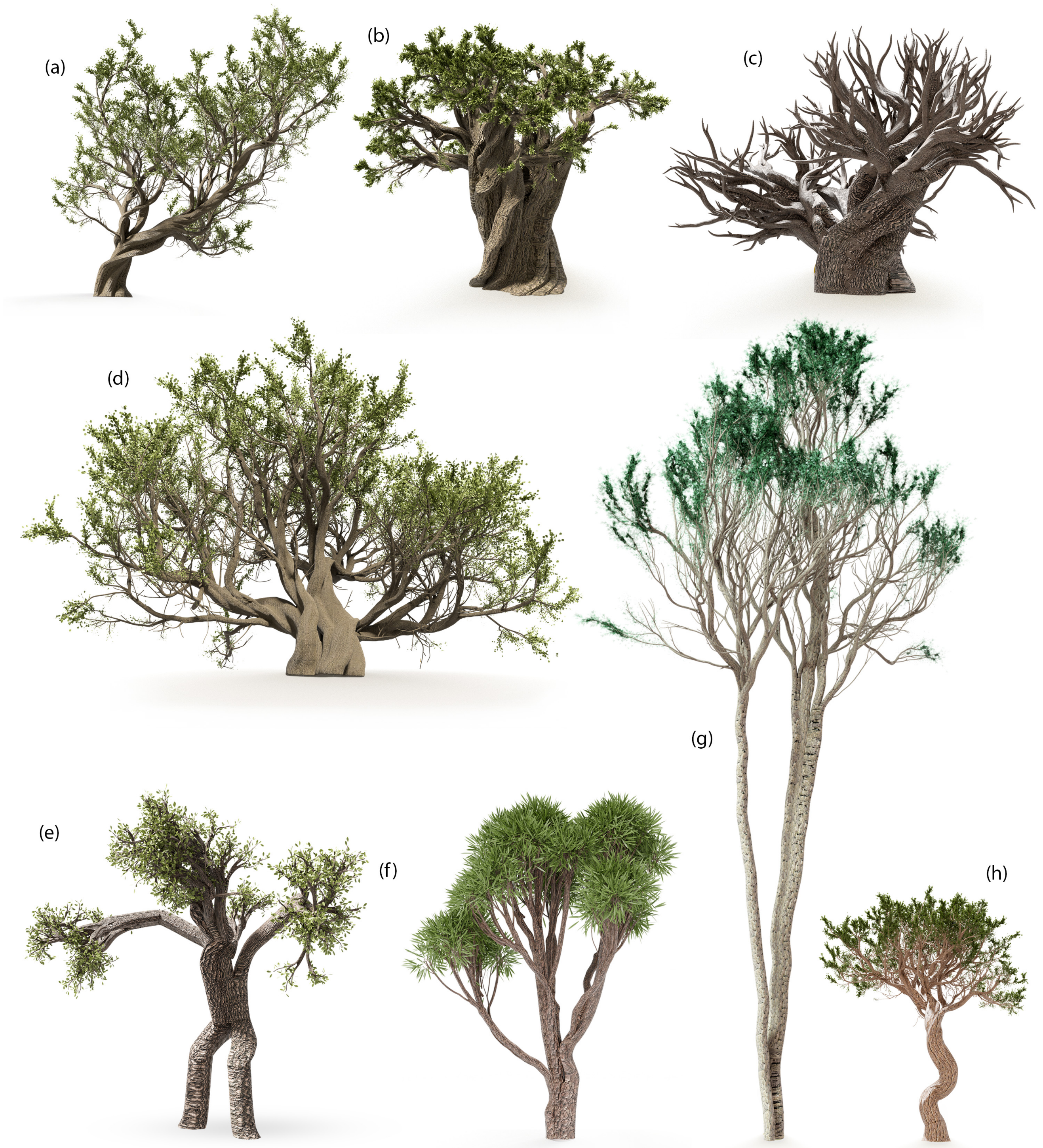


Fig. 11. An array of complex tree structures generated by our interactive tree modeling framework. (a) Shows user-guided branch growth through the Interactive Invigoration operator. (b) and (c) features the dynamic forms achievable with the Branch Twisting Operator. (d) Highlights the intricate branching structure of a mature tree model. (e) Features an unusual tree form achievable through user-defined profiles. (f) Captures the stark silhouette of a Joshua tree. (g) represents a result achieved using the Adventitious Bud Formation and Twisting operator to generate a Juniper tree model. (h) Illustrates the application of both the invigoration and user-defined profiles to model a tree with a split trunk.

Table 1. Performance characteristics of tree models with different levels of complexity. PC = Profile calculation time in seconds; M = time in seconds for computing a surface mesh.

Figure	# Profile Nodes	# Strands	# Particles	PC	M
Fig. 9a	21,345	6,240	564,632	3,38	2.75
Fig. 9b	21,354	24,960	2,258,528	13,28	10.31
Fig. 12a	5,668	2,149	95,976	0.81	0.48
Fig. 12b	15,219	4,541	357,276	2.13	1.51
Fig. 12c	21,425	6,252	585,672	3.29	2.38
Fig. 12d	32,256	9,389	1,079,838	5.26	4.16
Fig. 12e	41,470	12,206	1,602,270	8.73	6.09

For example, trees with 5,668 nodes took approximately 0.81 seconds for profile calculations and 0.48 seconds for meshing, whereas trees with 41,470 nodes required around 8.73 seconds for profile calculations and 6.09 seconds for meshing. The number of required strands scales approximately linearly with the number of nodes for an experiment on the temporal progression of tree development (Fig. 12). We also evaluated the runtimes for a tree model with a fixed node number (21K) and the increasing number of strands introduced per endpoint ranging from 1 to 4 (Fig. 9). Trees with four strands per end node exhibit the highest complexity, totaling 24,960 strands and requiring over 13 seconds for profile calculations and 10 seconds for meshing. Here, we observe an approximately linear increase in mesh generation and profile computation times. The total time to generate a tree depends on the mesh and profile computation times and the time needed for an artist to design a tree. For the results in this paper, we used between 30 seconds and 10 minutes to design tree models.

7 DISCUSSION AND LIMITATIONS

While innovative and robust, our method also has limitations. For one, the detailed and realistic modeling of trees, especially when considering secondary growth and complex branching structures, can be computationally intensive. We refrained from modeling the annual growth of the cambium in a temporally coherent manner and chose a profile-driven approach instead. This approach forfeits the biological plausibility and accuracy of describing tree growth phenomena for interactive control. Second, every time the skeletal graph changes, all strands have to be recomputed, and further code optimizations could be implemented to alleviate this problem. Third, modeling individual trees with high fidelity is feasible, but scaling this to large environments like forests can be challenging. This includes not only the computational complexity but also ensuring that the ecological interactions and variations across a large number of trees are realistically represented. While our model simulates interactions with environmental factors, the dynamic integration of these factors (like changing weather, seasons, or soil conditions) is complex and may not always yield realistic results. Other methods employ mesh refinement approaches to introduce detailed botanical features to tree models [Bloomenthal 1985; Xie et al. 2015]. A volumetric approach like ours, while computationally more demanding, allows us to model more complex branch geometries, including holes, as shown in Figs. 2 and 14.

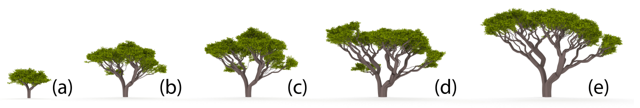


Fig. 12. Tree models of different levels of complexity: 2K strands (a), 4K strands (b), 5K strands (c), 9K strands (d), 12K strands (e). More details for each of these models are provided in Tab. 1.

8 CONCLUSIONS AND FUTURE WORK

We presented a novel approach for tree modeling in computer graphics, addressing the need for realistic and biologically plausible tree forms in various applications. Compared to existing approaches, we focus on both primary and secondary growth of trees, introducing the concepts of interactive invigoration and strand-based modeling, along with a robust meshing algorithm and a set of tree editing operators. The strand-based modeling approach, a significant advancement over traditional methods, allows for the detailed representation of branch volumes. This technique captures the natural appearance of branching points, such as twists in branch shapes. The meshing algorithm we introduced efficiently leverages the strand-based representation to obtain detailed structural models in a graphics-ready state. Finally, our novel set of tree editing operators enables users to interactively design and modify tree forms while preserving their structural and biological plausibility. This feature is particularly important for applications requiring specific tree shapes, such as urban planning or forestry simulations.

Given the current state of our work, there are several avenues for future research: For one, we aim at quantitative comparisons with observations of real trees to validate the results of the strand-based model, i.e., assessing how approximate our methods for describing wound healing or burl formation are. Second, integrating our tree modeling approach with dynamic environmental simulation tools could lead to even more realistic models. This includes the interaction of trees with varying weather conditions, soil types, and interaction with other flora and fauna. Finally, incorporating machine learning techniques could further refine the procedural generation of tree models, potentially allowing for the automated generation of species-specific traits based on a set of biological parameters.

ACKNOWLEDGMENTS

This research was supported by the Foundation for Food and Agriculture Research Grant ID: 602757 to Benes. The content of this publication is solely the responsibility of the authors and does not necessarily represent the official views of the Foundation for Food and Agriculture Research. This project was also sponsored by USDA NIFA, Award #2023-68012-38992 grant "Promoting Economic Resilience and Sustainability of the Eastern U.S. Forests" to Benes. This work is based upon efforts supported by the EFFICACI grant, #NR233A750004G044, under USDA NRCS to Benes. The views and conclusions contained herein are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government or NRCS. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes, notwithstanding any copyright annotation therein.

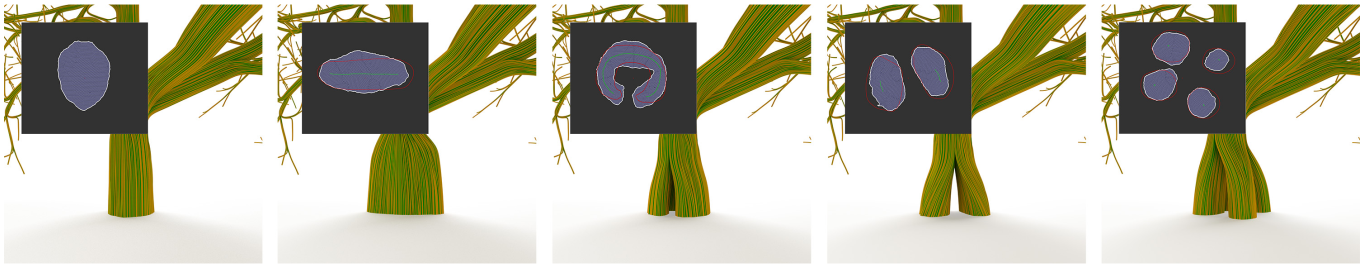


Fig. 13. Branch profile sketches: our method enables users to sketch branch profiles for any node in a branch graph. Possible profiles range from single branches and branches with cavities to split branches.

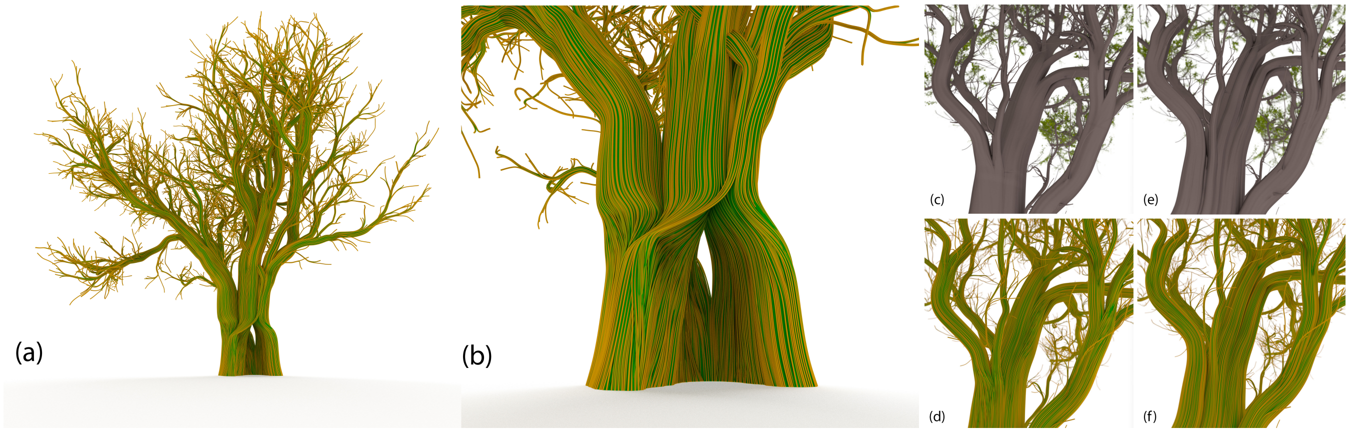


Fig. 14. (a, b): Example of strand-based tree shape editing: complex tree models can be generated conveniently by interactively injecting vigor and drawing branch profiles. (c-f): our strand-based modeling together with the meshing procedure enables generating different branch bifurcations.

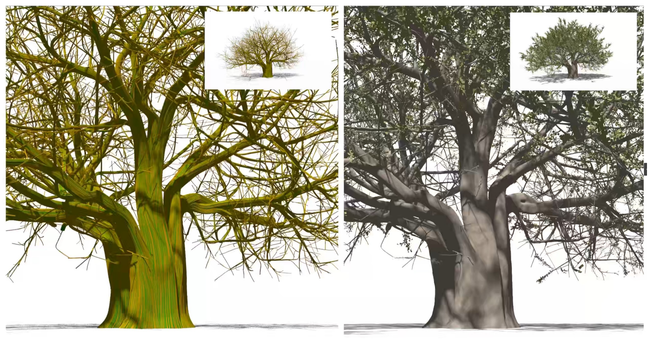


Fig. 15. An old tree generated with our framework. Using strand-based modeling allows for defining plausible branching structures and complex secondary growth.

REFERENCES

F. Anastacio, M. C. Sousa, F. Samavati, and J. A. Jorge. 2006. Modeling Plant Structures Using Concept Sketches *(NPAR '06)*. ACM, 105–113.

M. Aono and T.L. Kunii. 1984. Botanical Tree Image Generation. *IEEE Computer Graphics and Applications* 4(5) (1984), 10–34.

O. Argudo, A. Chica, and C. Andujar. 2016. Single-picture Reconstruction and Rendering of Trees for Plausible Vegetation Synthesis. *Comput. Graph.* 57, C (2016), 55–67.

O. K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, and T.-Y. Lee. 2008. Skeleton extraction by mesh contraction. In *ACM SIGGRAPH 2008 Papers* (Los Angeles, California) *(SIGGRAPH '08)*. Association for Computing Machinery, New York, NY, USA, Article 44, 10 pages. https://doi.org/10.1145/1399504.1360643

J. Bender, M. Müller, and M. Macklin. 2017. A survey on position based dynamics, 2017. In *Proceedings of the European Association for Computer Graphics: Tutorials* (Lyon, France) *(EG '17)*. Eurographics Association, Goslar, DEU, Article 6, 31 pages.

B. Benes, N. Andrysco, and O. Šťava. 2009. Interactive Modeling of Virtual Ecosystems. In *Proceedings of the Fifth Eurographics Conference on Natural Phenomena* (Munich, Germany) *(NPH'09)*. Eurographics Association, Goslar, DEU, 9–16.

J. Bloomenthal. 1985. Modeling the mighty maple. *SIGGRAPH Comput. Graph.* 19 (July 1985), 305–311. Issue 3. https://doi.org/10.1145/325165.325249

D. Bradley, D. Nowrouzezahrai, and P. Beardsley. 2013. Image-based Reconstruction and Synthesis of Dense Foliage. *ACM TOG* 32, 4, Article 74 (2013), 74:1–74:10 pages.

X. Chen, B. Neubert, Y.-Q. Xu, O. Deussen, and S. B. Kang. 2008. Sketch-Based Tree Modeling Using Markov Random Field. *ACM TOG* 27, 5, Article 109 (Dec. 2008).

M. Cieslak, U. Govindarajan, A. Garcia, A. Chandrashekar, T Hädrich, A. Mendoza-Drosik, D. L. Michels, S. Pirk, C.-C. Fu, and W. Palubicki. 2024. Generating Diverse Agricultural Data for Vision-Based Farming Applications. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop: Vision for Agriculture* (2024).

C. Galbraith, L. Mündermann, and B. Wyvill. 2004. Implicit Visualization and Inverse Modeling of Growing Trees. *Computer Graphics Forum* 23, 3 (2004), 351–360.

J. Guo, H. Jiang, B. Benes, O. Deussen, X. Zhang, D. Lischinski, and H. Huang. 2020. Inverse Procedural Modeling of Branching Structures by Inferring L-Systems. *ACM Trans. Graph.* 39, 5, Article 155 (June 2020), 13 pages. https://doi.org/10.1145/3394105

R. Habel, A. Kusternig, and M. Wimmer. 2009. Physically Guided Animation of Trees. *Comput. Graph. Forum* 28, 2 (2009), 523–532.

T. Hädrich, B. Benes, O. Deussen, and S. Pirk. 2017. Interactive Modeling and Authoring of Climbing Plants. *Comput. Graph. Forum* 36, 2 (2017), 49–61.

T. Hädrich, J. Scheffczyk, W. Palubicki, S. Pirk, and D. L. Michels. 2020. Interactive Wood Fracture. In *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation - Posters*. The Eurographics Association.

M. Holton. 1994. Strands, Gravity and Botanical Tree Imagery. *Computer Graphics Forum* 13(I) (1994), 57–67. https://doi.org/10.1111/1467-8659/1310057

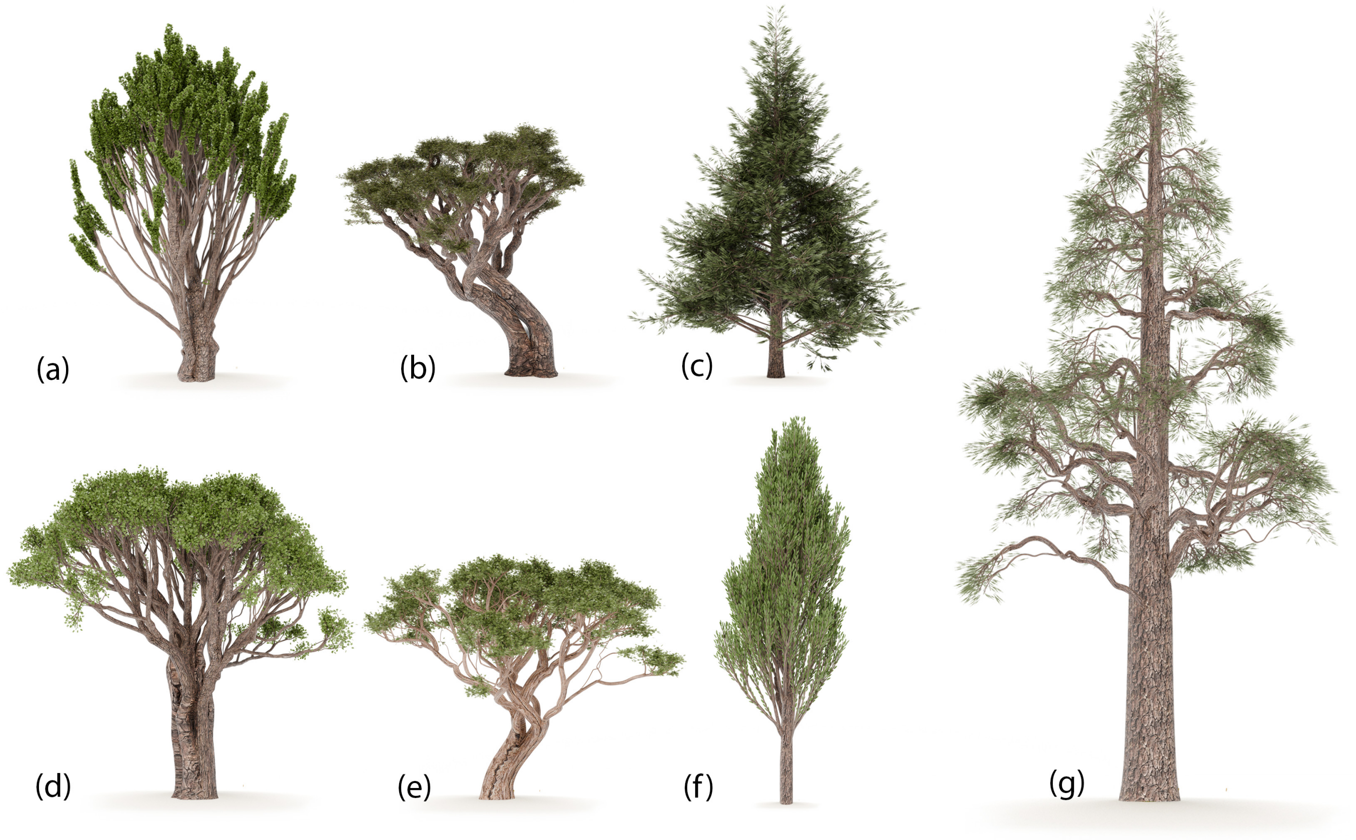


Fig. 16. A number of different tree species generated with our framework ranging from hazel (a), elm (d), and poplar (f) trees to different spruces (c, g) and acacia trees (b, e). Our method is able to generate plausible shapes for younger tree models (c, f) as well as older trees with more defined trunks (a, b, c, e, g).

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

T. Ijiri, S. Owada, and T. Igarashi. 2006. Seamless Integration of Initial Sketching and Subsequent Detail Editing in Flower Modeling. *Comput. Graph. Forum* 25, 3 (2006), 617–624.

J. Kałużny, Y. Schreckenberg, K. Cyganik, P. Annighöfer, S. Pirk, D. Michels, M. Cieslak, F. Assaad, B. Benes, and W. Palubicki. 2024. LAESI: Leaf Area Estimation with Synthetic Imagery. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshop: Synthetic Data for Computer Vision* (2024).

E. Kleiberg, H. Van de Wetering, and J. Van Wijk. 2001. Botanical visualization of huge hierarchies. In *IEEE Symposium on Information Visualization, 2001. INFOVIS 2001.* IEEE, 87–94.

J. Kratt, M. Spicker, A. Guayaquil, M. Fiser, S. Pirk, O. Deussen, J. C. Hart, and B. Benes. 2015. Woodification: User-Controlled Cambial Growth Modeling. *Comput. Graph. Forum* 34, 2 (May 2015), 361–372.

J. J. Lee, B. Li, and B. Benes. 2024. Latent L-Systems: Transformer-Based Tree Generator. *ACM Trans. Graph.* 43, 1, Article 7 (2024), 16 pages.

B. Li, J. Kałużny, J. Klein, D. L. Michels, W. Pałubicki, B. Benes, and S. Pirk. 2021. Learning to Reconstruct Botanical Trees from Single Images. *ACM Transaction on Graphics* 40, 6, Article 231 (12 2021).

B. Li, J. Klein, D. L. Michels, B. Benes, S. Pirk, and W. Pałubicki. 2023. Rhizomorph: The Coordinated Function of Shoots and Roots. *ACM Trans. Graph.* 42, 4, Article 59 (jul 2023), 16 pages.

C. Li, O. Deussen, Y.-Z. Song, P. Willis, and P. Hall. 2011. Modeling and Generating Moving Trees from Video. *ACM TOG* 30, 6, Article 127 (2011), 127:1–127:12 pages.

Y. Li, X. Fan, N. J. Mitra, D. Chamovitz, D. Cohen-Or, and B. Chen. 2013. Analyzing Growing Plants from 4D Point Cloud Data. *ACM TOG* 32, 6, Article 157 (2013).

B. Lintermann and O. Deussen. 1996. Interactive Modelling and Animation of Branching Botanical Structures. In *Computer Animation and Simulation'96 (Springer Computer Science)*. Springer–Verlag Wien New York, 139–151.

Y. Liu, J. Guo, B. Benes, O. Deussen, X. Zhang, and H. Huang. 2021a. TreePartNet: Neural Decomposition of Point Clouds for 3D Tree Reconstruction. *ACM Transaction on Graphics* 40, 6, Article 232 (Dec. 2021), 16 pages.

Z. Liu, K. Wu, J. Guo, Y. Wang, O. Deussen, and Z. Cheng. 2021b. Single Image Tree Reconstruction via Adversarial Network. *Graphical Models* 117 (2021), 101115.

Y. Livny, S. Pirk, Z. Cheng, F. Yan, O. Deussen, D. Cohen-Or, and B. Chen. 2011. Texture-lobes for tree modelling. In *ACM SIGGRAPH 2011 papers* (Vancouver, British Columbia, Canada) *(SIGGRAPH '11)*. ACM, New York, NY, USA, Article 53, 10 pages.

S. Longay, A. Runions, F. Boudon, and P. Prusinkiewicz. 2012. TreeSketch: Interactive Procedural Modeling of Trees on a Tablet. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling* (Annecy, France) *(SBIM '12)*. Eurographics Association, Goslar, DEU, 107–120.

F. Maggioli, J. Klein, T. Hädrich, E. Rodolà, W. Pałubicki, S. Pirk, and D. L. Michels. 2023. A Physically-inspired Approach to the Simulation of Plant Wilting. In *SIGGRAPH Asia 2023 Conference Papers (SA '23)*. ACM, Article 66, 8 pages.

R. Měch and P. Prusinkiewicz. 1996. Visual Models of Plants Interacting with Their Environment. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96)*. Association for Computing Machinery, New York, NY, USA, 397–410. https://doi.org/10.1145/237170.237279

M. Müller, B. Heidelberger, M. Hennix, and J. Ratcliff. 2007. Position based dynamics. *Journal of Visual Communication and Image Representation* 18, 2 (2007), 109–118.

B. Neubert, T. Franken, and O. Deussen. 2007. Approximate Image-Based Tree-Modeling using Particle Flows. *ACM Trans. Graph. (Proc. of SIGGRAPH 2007)* 26, 3 (2007).

T. Niese, S. Pirk, M. Albrecht, B. Benes, and O. Deussen. 2022. Procedural Urban Forestry. *ACM Trans. Graph.* 41, 2, Article 20 (March 2022), 18 pages.

M. Okabe, S. Owada, and T. Igarashi. 2007. Interactive Design of Botanical Trees Using Freehand Sketches and Example-based Editing. In *ACM SIGGRAPH Courses* (San Diego, California). ACM, Article 26.

P. E. Oppenheimer. 1986. Real time design and animation of fractal plants and trees. *SIGGRAPH Comput. Graph.* 20, 4 (1986), 55–64. https://doi.org/10.1145/15886.15892

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

S. Pirk, M. Jarząbek, T. Hädrich, D. L. Michels, and W. Palubicki. 2017. Interactive Wood Combustion for Botanical Tree Models. *ACM Trans. Graph.* 36, 6, Article 197 (Nov. 2017), 12 pages.

S. Pirk, T. Niese, O. Deussen, and B. Neubert. 2012a. Capturing and Animating the Morphogenesis of Polygonal Tree Models. *ACM Trans. Graph.* 31, 6, Article 169 (Nov. 2012), 10 pages.

S. Pirk, T. Niese, T. Hädrich, B. Benes, and O. Deussen. 2014. Windy Trees: Computing Stress Response for Developmental Tree Models. *ACM TOG* 33, 6, Article 204 (2014), 204:1–204:11 pages.

S. Pirk, O. Stava, J. Kratt, M. A. M. Said, B. Neubert, R. Měch, B. Benes, and O Deussen. 2012b. Plastic Trees: Interactive Self-adapting Botanical Tree Models. *ACM Trans. Graph.* 31, 4, Article 50 (July 2012), 10 pages.

P Prusinkiewicz. 1986. Graphical Applications of L-systems. In *Proceedings on Graphics Interface '86/Vision Interface '86* (Vancouver, British Columbia, Canada). Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 247–253.

P. Prusinkiewicz and Aristid Lindenmayer. 1990. *The Algorithmic Beauty of Plants.* Springer-Verlag New York, Inc.

L. Quan, P. Tan, G. Zeng, L. Yuan, J. Wang, and S. B. Kang. 2006. Image-Based Plant Modeling. *ACM TOG* 25, 3 (July 2006), 599–604.

E. Quigley, Y. Yu, J. Huang, W. Lin, and R. Fedkiw. 2018. Real-Time Interactive Tree Animation. *IEEE TVCG* 24, 5 (2018), 1717–1727.

W. T. Reeves and R. Blau. 1985. Approximate and Probabilistic Algorithms for Shading and Rendering Structured Particle Systems. *SIGGRAPH Comput. Graph.* 19, 3 (July 1985), 313–322.

A. Runions, M. Fuhrer, B. Lane, P. Federl, A. Rolland-Lagan, and P. Prusinkiewicz. 2005. Modeling and visualization of leaf venation patterns. *ACM Trans. Graph.* 24, 3 (2005), 702–711.

A. Runions, B. Lane, and P. Prusinkiewicz. 2007. Modeling Trees with a Space Colonization Algorithm. (2007), 63–70. https://doi.org/10.2312/NPH/NPH07/063-070

H. Shao, T. Kugelstadt, T. Hädrich, W. Pałubicki, J. Bender, S. Pirk, and D. L. Michels. 2021. Accurately Solving Rod Dynamics with Graph Learning. In *NeurIPS*.

O. Stava, S. Pirk, J. Kratt, B. Chen, R. Mech, O. Deussen, and B. Benes. 2014. Inverse Procedural Modelling of Trees. *Computer Graphics Forum* (2014), n/a–n/a.

P. Tan, T. Fang, J. Xiao, P. Zhao, and L. Quan. 2008. Single Image Tree Modeling. *ACM TOG* 27, 5, Article 108 (2008), 7 pages.

B. Wang, Y. Zhao, and J. Barbič. 2017. Botanical Materials Based on Biomechanics. *ACM Trans. Graph.* 36, 4, Article 135 (jul 2017), 13 pages.

J. Weber and J. Penn. 1995. Creation and Rendering of Realistic Trees. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. ACM, New York, NY, USA, 119–128. https://doi.org/10.1145/218380.218427

J. Wither, F. Boudon, M.-P. Cani, and C. Godin. 2009. Structure from silhouettes: a new paradigm for fast sketch-based design of trees. *Comput. Graph. Forum* 28, 2 (2009), 541–550.

S.-K. Wong and K.-C. Chen. 2015. A Procedural Approach to Modelling Virtual Climbing Plants With Tendrils. *Comput. Graph. Forum* (2015).

K. Xie, F. Yan, A. Sharf, O. Deussen, H. Huang, and B. Chen. 2015. Tree modeling with real tree-parts examples. *IEEE Transactions on Visualization and Computer Graphics* (2015).

H. Xu, N. Gossett, and B. Chen. 2007. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM TOG* 26, 4 (2007), Article 19, 13 pages.

Y. Zhao and J. Barbič. 2013. Interactive Authoring of Simulation-ready Plants. *ACM TOG* 32, 4, Article 84 (2013), 12 pages.

X. Zhou, B. Li, B. Benes, S. Fei, and S. Pirk. 2023. DeepTree: Modeling Trees with Situated Latents. *IEEE Transactions on Visualization and Computer Graphics* (2023), 1–14.