DOI: 10.1111/cgf.13088

COMPUTER GRAPHICS *forum*
Volume 00 (2017), number 00 pp. 1–16

Tree Branch Level of Detail Models for Forest Navigation

Xiaopeng Zhang[1], Guanbo Bao[1], Weiliang Meng[1], Marc Jaeger[2,3], Hongjun Li[1,4], Oliver Deussen[5,6] and Baoquan Chen[7]

[1]NLPR-LIAMA, Institute of Automation, CAS, China
[2]AMAP, Cirad, France
[3]LIRMM-ICAR, Montpellier University, France
[4]Beijing Forestry University, China
[5]VCC, Shenzhen Institute of Advanced Technology, CAS, China
[6]Universitfät Konstanz, Germany
[7]Shandong University, China

Abstract

We present a level of detail (LOD) method designed for tree branches. It can be combined with methods for processing tree foliage to facilitate navigation through large virtual forests. Starting from a skeletal representation of a tree, we fit polygon meshes of various densities to the skeleton while the mesh density is adjusted according to the required visual fidelity. For distant models, these branch meshes are gradually replaced with semi-transparent lines until the tree recedes to a few lines. Construction of these complete LOD models is guided by error metrics to ensure smooth transitions between adjacent LOD models. We then present an instancing technique for discrete LOD branch models, consisting of polygon meshes plus semi-transparent lines. Line models with different transparencies are instanced on the GPU by merging multiple tree samples into a single model. Our technique reduces the number of draw calls in GPU and increases rendering performance. Our experiments demonstrate that large-scale forest scenes can be rendered with excellent detail and shadows in real time.

Keywords: level of detail, virtual forests, real time, branch models, simplification

ACM CCS: I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

Realistic, real-time rendering of vegetation remains a key challenge in computer graphics. Trees are major components of outdoor scenes and have complex topological structures and geometrical shapes that require a great amount of small graphical primitives to be visually convincing. Figure 1 provides an example as well as its computer graphics representation and shows the complexity we dealt with using our interactive rendering method.

The development of modern techniques in plant shape modelling and simulation has led to increasing complexity and realistic simulation of growth and dynamic behaviour. Software systems such as *L-systems* [PL90, MP96], *AMAP* [dREF*88], *Xfrog* [LD99], *Onyxtree* [ony] enable users an easy modelling of complex structures. These techniques cover a wide range of applications, reaching from games to scientific simulations. Virtual gardens, forestry, landscapes [JT03], and ecosystems [DCSD02, DHL*98] are typical examples. Real-time rendering of such scenes is a major challenge for such applications [DN04, GMN05].

Unfortunately, the geometrical complexity of vegetation over large areas still exceeds today's hardware rendering capabilities. Several strategies have been developed to address this issue: (1) simplifying meshes to reduce the number of polygons; (2) replacing polygons with more simple primitives such as points and lines; and (3) replacing 3D geometry with textures or billboards. These methods have all been effective in improving the efficiency of rendering large plant ecosystems.

For navigation in large-scale forests, several specialized techniques have been developed: (1) compressing the geometrical complexity of forest scenes using simplification [GCRR11, NPDD11] or volumetric textures plus aperiodic tiling: [DN04]; (2) instancing LOD models for trees and creating LOD shadow maps by utilizing the capacity of modern GPUs[BLZD12]. Realistic rendering of large forests using light fields and view dependent texture shaders was introduced by Bruneton *et al.* [BN12].

In this paper, we present a continuous level of detail (LOD) method for tree branches which, when combined with methods

(a) A rendered image of virtual defoliated trees

(b) A photo of real defoliated trees

Figure 1: *Leafless trees in winter. Figure 1(a) is a rendered image of virtual leafless trees in the winter with our new technique, and Figure 1(b) is a photo of real leafless trees showing rich details.*

for processing tree foliage such as those presented in [DZYJ10], facilitates real-time forest walk-throughs and fly-overs.

Starting from a specifically designed representation of the tree skeleton, we first simplify the tree geometry by replacing consecutive branch segments with a single one, modelled by a single prism or a line. We then perform a topology simplification by deleting or merging branch segments. The generation of complete branch LOD sets is guided by rigorous error metrics that are used to ensure smooth transitions between adjacent LODs.

Our tree models are generated by the *AMAP-Genesis* software that is based on botanical rules [dREF*88, dRDJ90], reflecting a plant's growing process and its spatial occupation, and conforming to real measured data. Other plant models, however, may also be used as long as skeleton information is provided, such as data from *L-systems, Xfrog* or *Onyx-tree*. These procedural generators lead to complex geometrical objects. Other models, where branches are represented by very simple single straight cylinders[LVM04, LCV03, HB05] are not sufficient for immersive views, and are thus not considered here.

2. Related Work

Several strategies have been developed for improving the rendering efficiency of branching models and forests on large geometrical scales. These techniques can be divided into four categories that consider general objects, plants, branches and large-scale objects.

Geometry processing techniques are efficient for reducing complex geometry. They can be applied to plant objects although their overall efficiency is limited.

While *mesh simplification* is useful in polygonal decimation [GS02] of smooth surfaces, it is inefficient for trees in complex ecosystems [Max90] since such techniques do not maintain details of the tree branching structures. *Primitive replacement* is a useful technique for polygons that are replaced with simpler primitives such as points and lines. The technique was applied to plants by Deussen *et al.* [DCSD02]. After their replacement, however, lines are not further simplified. *Object replacement* can be useful for replacing geometry with textures [MNP01] or billboard clouds [BCF*05]. In [KCD*14], implicit surface computations define tree model occlusions from which billboards are generated. Problems often involve the transition between the different representations, and the memory footprint is relatively large.

Tree model representations and processing techniques were specifically developed for compressing the data of plant objects. *View-dependent foliage pruning* [GCRR11, NPDD11] is a special form of stochastic simplification [CHPR07], i.e. invisible parts of the foliage are pruned for real-time rendering. A problem with this approach is that real-time shadows are hard to compute. *Multi-resolution foliage* models (cf. [DZYJ10]) can unify foliage polygons, based on simple metrics. A GPU-oriented design of the corresponding LOD structure is presented to decrease the communication between CPU and GPU. Unfortunately, semi-transparent objects, which are important for such geometry, are not considered. *Texture-lobe models* are a recent approach for representing foliage envelopes [LPC*11]. They divide tree models into visually important and less important parts; a tree is represented by a cluster of lobes that are procedurally filled with geometry as the process is executed. This is a compact representation and thus very efficient for data transmission. [DLX*15] propose a hybrid representation (HR) of a simplified tree model, which allows to adaptively select simplified models according to the resolution of different devices. A saliency map is extracted for simplifying the tree crown.

Branch model processing approaches act on the branching skeleton as the visually most prominent structure of trees. Specific modelling and rendering methods have been developed based on an understanding of these features. Modelling branches using *generalized cylinders* is a simplistic but popular way to model branch shapes from skeletons [Blo90, Blo95, BL99]. Specifically, prisms as simplified generalised cylinders are widely used in tree generators [Blo85, Blo95]. In [LVM04] a single polygonal mesh is used to represent the structure of a tree with smooth transitions in bifurcations. Continuous re-grouping of branches is used to construct LODs using error control [BK04]. A specialised metric is used in [LCV03] to select a LOD model of plant branches. The generic tree model by Prusinkiewicz *et al.* [PMKL01] uses three levels of plant shapes to deliver a range of representations from silhouettes to full detailed models.

Large-scale scenes are used in many applications, especially those involving urban or landscape planning. *Self-similarity* and the hierarchy of plants were investigated in [DHL*98, GMN05], where the plant model is broken down into components that are instanced. Thus, rendering is accelerated by a simple lookup table

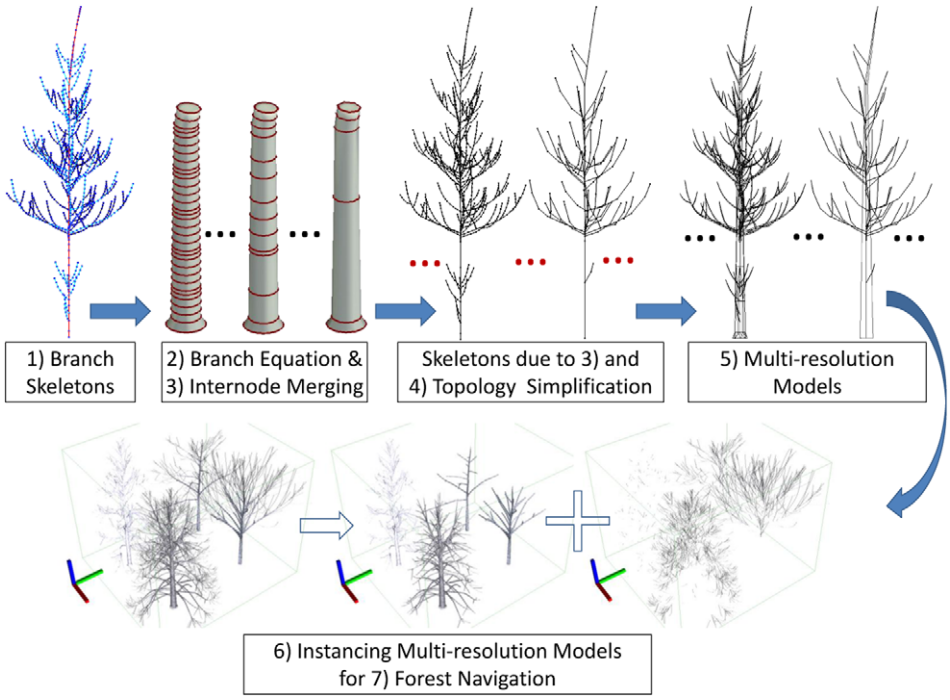


Figure 2: *Main steps of branch LODs for navigation.*

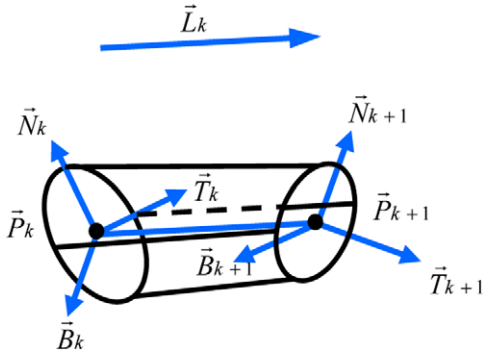


Figure 3: *An internode and its local geometry: two nodes $\vec{P}_k$ and $\vec{P}_{k+1}$, their skeletal vectors $\vec{G}_k$ and $\vec{G}_{k+1}$, and the direction vector of the skeleton segment $\vec{L}_k$.*

in graphics memory. After a tree is procedurally defined, the inherent tree hierarchy can be further reduced to speed up rendering [SG97, LP02]. Large-scale instancing of trees, supported by hardware, is presented in [BLZD12] for forests with highly detailed geometry and used for real-time navigation. A shadow map LOD strategy is used and maps are filtered entirely on the GPU to reduce shadow aliasing.

[BN12] presents a real-time realistic lighting model for realistic and large-scale forest rendering of scenes in real-time. Unfortunately, only a limited number of tree samples can be used due to memory costs. This model is therefore more suitable for flight simulators with bird's-eye views. In [ACV*14], two orthogonal photos are used to model and render realistic vegetation distributions along roads paths, where the vegetation is created procedurally on-the-fly and leaves are rendered using OpenGL 4 tessellation and geometry shaders. This vegetation model is very simple with relatively few predefined trunk models.

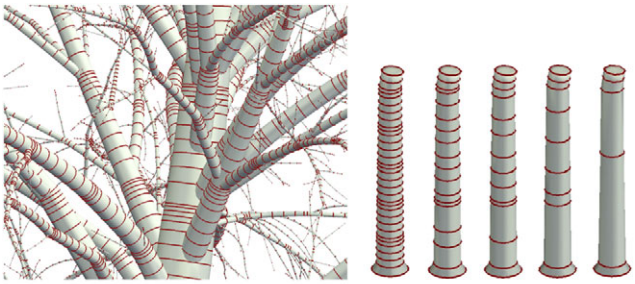
Lastly, the rendering of large graphs set is presented in [ZBDS12], involving a novel combination of edge accumulation with density-based node aggregation. Such image-based methods could also be used for rendering forests.

In contrast to most earlier works, our LOD model focuses on tree models including error-controlled direct construction of a simplified representation of the tree branch geometry with prisms and thin branches, along with hardware instancing, allowing fast and realistic rendering of virtual forests. Although our LOD method is designed for tree branches, it can be combined with foliage LOD algorithms, such as provided in [DZYJ10].

3. An Overview of the Approach

The overall work of this paper is a LOD technique for tree branches, described as *branch simplification*. The framework of the proposed approach breaks down into seven points, where the six steps (1) to (6) are illustrated in Figure 2.

(1) *Definition of branch skeletons and Frenet frames.* The skeleton of a branch is a list of points with their corresponding radii, and the Frenet frames are a list coordinate systems, built from tangents, normals and bi-normal vectors.



(a) Internode sequences **(b)** Internode merging

Figure 4: *Internodes in branch equations. (a) Internode sequences drawn on all branch equations. (b) Five internode sequences merge on a given branch. The merging process is based on the branch equation.*

(2) *Branch modelling.* Branch segments are considered as cylindrical primitives and modelled by corresponding equations.
(3) *Internode merging.* Internode merging constructs a new internode to replace consecutive internodes. Internode merging is controlled by the maximum permitted error.
(4) *Simplification of topology structures.* Topology simplification decimates some child branches while keeping the parent under error control.
(5) *Construction of a multi-resolution branch model.* Our multi-resolution branch model step polygonizes branch segments or converts them to lines in a sequence of branch LOD models under error control.
(6) *GPU instancing of multi-resolution models.* This step sends the appropriate 3D multi-resolution geometrical model to the GPU and instances it on request. It also includes semi-transparent line models for thin branches.
(7) *Application to forest navigation.* Forest navigation combines multi-resolution sequences and multi-resolution model instances, where line models with different transparencies are instanced by merging multiple tree samples into a single model, thereby reducing the number of draw calls and thus increase virtual forest rendering performance.

The permitted spatial error is often defined in object space, we adopt it to control the degree of model simplification. A metric formulation in screen space, however, is more convenient in many practical applications.

The *permitted pixel error*, labelled as δ, is specified as the number of pixels allowed for representing the quality of shape details in the synthesized image, adopted from [DZYJ10]. δ may be chosen as 0.5 pixels for the model with original details, since a pixel error less than 0.5 is not visible.

If we fix δ and the camera parameters, the distance of the camera to the tree will be closely related to the permitted spatial error ϵ.

Control errors in branch simplification drive the processes of internode merging within a single branch, branch segment discretization and topology simplification. Four independent spatial error metrics, E_1, E_2, E_3 and E_4 are involved in these three tasks.

Internode merging is controlled by two error metrics E_1 and E_2 where E_1 controls the skeleton polyline geometry, i.e. the internode merging in latitudinal direction, and E_2 controls the simplification of the skeleton for radius deviation and rotational aspects, considering branch torsion in latitudinal direction.

Multi-resolution branch modelling is controlled by the error metric E_3, defining the number of mesh polygons and lines on all branch segments.

Topology simplification, i.e. branch decimation, is controlled by the error metric E_4. Topology simplification is performed before the multi-resolution branch models are constructed, thereby we avoid processing of branches that are further decimated. Internode merging is processed first, then we do a topological simplification, followed by the multi-resolution branch construction, and lastly by the instancing of the multi-resolution sequences for navigation.

The organization of the paper is as follows: Section 4 presents details about the branch equations and internode merging. Section 5 concerns multi-resolution branching models, i.e. error-controlled branch digitalisation for LODs, Section 6 concerns forest navigation using LOD models with semi-transparent line rendering and instancing. This is followed by a discussion of the results and ideas about future works.

4. Branch Definition and Internode Merging

The branch skeleton of an idealized tree is a smooth curve, and the branch geometry is usually modelled as a generalized smooth cylinder [Blo85, Blo95]. A globally smooth branch surface, however, requires high computational costs and considerable memory and the visual results are not optimal for navigation in large forests. As an alternative, we investigated the use of connected straight cylinders, described as *branch equation*, to represent the geometry of a single branch of the tree, similar to [dRDJ90]. We used the concept of Frenet frames along the skeleton [Blo85] and performed a multi-resolution analysis of the generalized cylinders inspired by the work of [Blo95].

4.1. Internodes for defining branches

We use *internodes* to represent the geometry of a branch element in a manner related to the definition of such internodes in botany. Similarly, the term *node* represents one of the two internode endings. So any single branch is composed by a sequence of consecutive internodes.

An internode is modelled as a circular cylinder around a segment of the skeletal poly-line that defines a branch, and two circles at the two endings orthogonal to the tangent vectors at the end nodes (Figure 3). Consecutive internodes share a common circle at their intersection. We define the k-th internode as I_k.

4.2. Branch skeleton and Frenet frames

A branch skeleton $\mathcal{B}$ of K internodes is a list of node points $\{\vec{P}_k\}$ and corresponding radii $\{R_k\}$, $\mathcal{B} = \{(\vec{P}_k, R_k); 1 \le k \le K\}$, with the arc length of the k-th internode $\lambda_k > 0$. Let $\vec{L}_k$ represent the direction

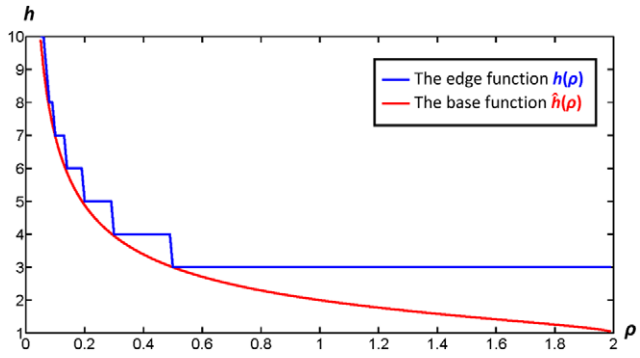


Figure 5: *A graph of the edge number function $h(\rho)$ (blue) and its continuous basis function $\hat{h}(\rho)$ (red), where the variable of two functions is ρ, the error on internode polygonization.*

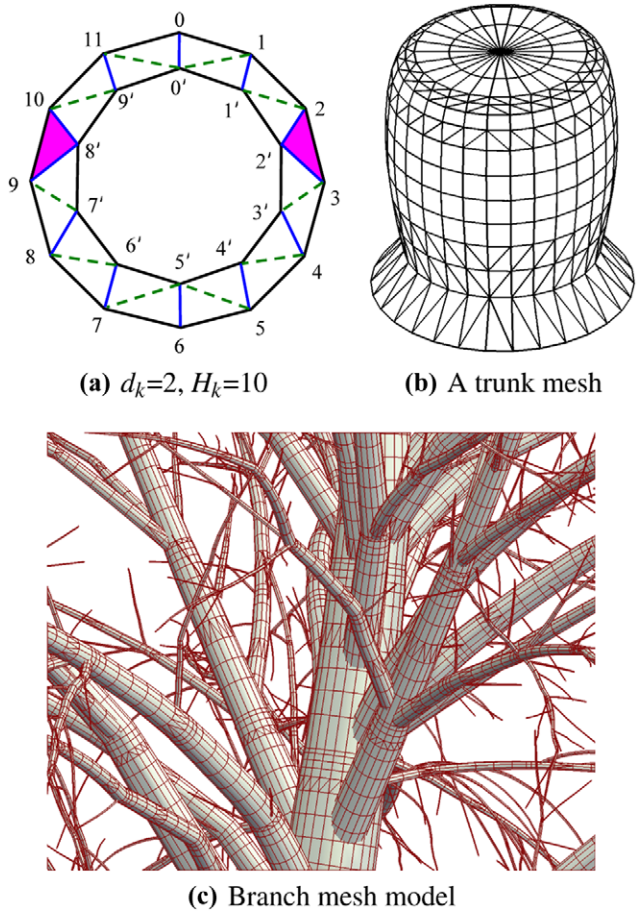


Figure 6: *Construction of a branch mesh. Figure 6(a) is from a top view of an internode I_k for vertex connection, Figure 6(b) is an upper side view of a short trunk, and Figure 6(c) is the side view of Figure 4(a) with all branch meshes.*

vector of each skeleton segment , shown as the up arrow of $\vec{L}_k$ in Figure 3; the half angle, $\beta_k = \arccos(\vec{L}_k \cdot \vec{L}_{k-1})/2$, is then called the *node angle*.

Frenet frames $(\vec{T}_k, \vec{N}_k, \vec{B}_k)$ are constructed from the nodes similar to [Blo90], where $\vec{T}_k$, $\vec{N}_k$, and $\vec{B}_k$ are the tangent vector, the normal vector and the bi-normal vector respectively. We call $\vec{G}_k = (\vec{P}_k, R_k; \vec{T}_k, \vec{N}_k, \vec{B}_k)$ a *skeletal vector* , composed of a node point and its corresponding radius and corresponding Frenet frame, or four 3D vectors and a scalar component. It is used to specify the shape of a branch from its skeleton.

The *torsion angle* is the angle between two bi-normal vectors $\vec{G}_k$ and $\vec{G}_{k+1}$, characterizing a twist of a Frenet frame around the skeleton in differential geometry. As seen in Figure 7(c), the torsion can be represented by lines of longitudinal twisting along a cylinder.

As proven in Equation (A2) in the Appendix, the torsion angle between $\vec{G}_k$ and $\vec{G}_{k+1}$, is not greater than $\beta_k + \beta_{k+1}$, therefore β_k is a relevant measure for the consecutive connection of skeletal parts. We define Γ to be the *torsion threshold*, the maximum allowed torsion angle.

4.3. Branch definition

The cylindrical geometry of internode I_k is now constructed through linear interpolation between two end circles of the skeletal vectors $\vec{G}_k$ and $\vec{G}_{k+1}$, i.e. a ruled surface linearly interpolates these two circles using $\vec{G}_k$ and $\vec{G}_{k+1}$. Thus, the overall branch is a piecewise continuous parametric surface:

$$\begin{aligned}\vec{V}_k(t,\theta) = {} & (1-u)\cdot\vec{P}_k + u\cdot\vec{P}_{k+1} \\ & + [(1-u)\cdot R_k\cdot\vec{N}_k + u\cdot R_{k+1}\cdot\vec{N}_{k+1}]\cdot\cos\theta \\ & + [(1-u)\cdot R_k\cdot\vec{B}_k + u\cdot R_{k+1}\cdot\vec{B}_{k+1}]\cdot\sin\theta,\end{aligned} \quad (1)$$

where $(t,\theta) \in [t_k, t_{k+1}] \times [0, 2\pi]$, $t_0 = 0$, $t_k = \lambda_1 + \cdots + \lambda_k$, and $u = (t - t_k)/(t_{k+1} - t_k)$. The normal vector $\vec{D}_k(t,\theta)$ of Equation (1) is specified in Equation (A3) in the Appendix. Therefore, the integral equation of a branch is:

$$\vec{V}(t,\theta) = \vec{V}_k(t,\theta), (t,\theta) \in [0, t_K] \times [0, 2\pi],$$

where $k = k(t) = \max\{l \in Z; t_l \le t\}$. Figure 4(a) shows all branch equations of a 30-year-old virtual white poplar tree (*Populus alba*).

4.4. Internode merging

Internode merging reduces branch complexity by replacing sets of consecutive internodes by sets of longer ones. It is the basic process for branch simplification and concerns all branch skeletons.

We use $I_{k,q}$ to represent the long internode spanned by $\vec{G}_k$ and $\vec{G}_{k+q}$ with $0 \le k \le K - q - 1; 1 \le q \le K - 1$, and we call the list of consecutive internodes represented by $U_{k,q} = \{I_k, \ldots, I_{k+q-1}\}(q \ge 1)$ an *internode group*. Thus $I_{k,q+2}$ with $q \ge 0$ replaces all internodes in group $U_{k,q+2}$ during internode merging. Figure 4(b) shows an internode merging sequence for a branch.

Figure 7: *Multiresolution of a helix cylinder with quad counts.*

We constrain the error $\psi_{k,q+2}$ due to internode merging of $U_{k,q+2}$ from $I_{k,q+2}$, $(q \geq 0)$, as follows

$$\psi_{k,q+2} \leq \phi_{k,q+2} + \varphi_{k,q+2}, \tag{2}$$

where $\phi_{k,q+2}$ is the maximum distance between skeletal vertex positions before and after merging, and $\varphi_{k,q+2}$ is the maximum radius deviation, specified here-under:

$$\begin{aligned} \phi_{k,q+2} &= \max\{||\vec{X}_{k,j}||; 0 \leq j \leq q\} \\ \varphi_{k,q+2} &= \sqrt{2}\max\{||\vec{Y}_{k,j}||, ||\vec{Z}_{k,j}||; 0 \leq j \leq q\}, \end{aligned} \tag{3}$$

where

$$\begin{aligned} \vec{X}_{k,j} &= (1-\tau_{k,j}) \cdot \vec{P}_k + \tau_{k,j} \cdot \vec{P}_{k+q+2} - \vec{P}_{k+j+1} \\ \vec{Y}_{k,j} &= (1-\tau_{k,j}) \cdot R_k \cdot \vec{N}_k + \tau_{k,j} \cdot R_{k+q+2} \cdot \vec{N}_{k+q+2} \\ &\quad - R_{k+j+1} \cdot \vec{N}_{k+j+1} \\ \vec{Z}_{k,j} &= (1-\tau_{k,j}) \cdot R_k \cdot \vec{B}_k + \tau_{k,j} \cdot R_{k+q+2} \cdot \vec{B}_{k+q+2} \\ &\quad - R_{k+j+1} \cdot \vec{B}_{k+j+1} \end{aligned} \tag{4}$$

and

$$\tau_{k,j} = (t_{k+j+1} - t_k)/(t_{k+q+2} - t_k). \tag{5}$$

The proof of Equation (2) can be checked in the Appendix. Errors in Equation (3) are directly related to q, the number of iterations in internode merging, so that the amount of merging can be predicted explicitly, avoiding repeated calculation of intermediate internodes in the merging process.

Internode merging $\mathcal{S} : I_{k,q+2} \rightarrow U_{k,q+2}$ is performed when $\phi_{k,q+2} \leq E_1$, $\varphi_{k,q+2} \leq E_2$ and the corresponding angle β_k of $I_{k,q+2}$ is less than the torsion limit Γ. Internode merging has the strongest effect on mesh decimation with respect to the number of reduced internodes. In practice, errors $\phi_{k,q+2}$ and $\varphi_{k,q+2}$ are precomputed and are used to guide mesh construction on-the-fly during rendering.

5. Multi-Resolution Branch Models

In this section, we describe how we build the corresponding surface model of meshes and lines after internode merging.

5.1. LOD representation with meshes

Prisms are often used to represent branches in traditional approaches [Blo85, Blo95, dREF*88]. The drawback of this representation is that mesh density is not adaptive, i.e. the mesh may be too sparse for the thick end, and too dense for the thin end. We adapted mesh density along the branches using a two-stage process: first computing the optimal discretization ratio for each internode, then performing vertex connections from one internode to another.

Polygonization of internode ends. For each internode, we convert the two circles at their ends into two equilateral polygons. For each circle of radius R, we take an error metric ε as the distance from an edge of the equilateral polygon to the surrounding circle, so that the number of polygon edges can be predicted directly. ε represents the control error E_3.

Let H be the maximum number of edges of all equilateral polygons considered for all tree branches, we set constants $P_i = 1 - \cos(\pi/i)$ with $(i = 3, \ldots, H)$ for our analysis, where P_i is a monotonously decreasing list. Considering the h-sided polygon $(3 \leq h \leq H)$, the approximation error becomes $\varepsilon = R[1 - \cos(\pi/h)]$. The ratio of the spatial error over the radius, called *error on internode polygonization*, would then be

$$\rho = \varepsilon/R = 1 - \cos(\pi/h) \in [P_H, 0.5]. \tag{6}$$

Let us define ε_0 as a pre-specified permitted error for cylinder discretization, with $\varepsilon \leq \varepsilon_0$. If $\varepsilon_0 \in [P_H R, 0.5R]$, then $\rho \in [P_H, 0.5]$, so that the edge number can be computed as an integer function

$$h(\rho) = \lfloor \pi / \arccos(1-\rho) \rfloor + 1. \tag{7}$$

Its base function is described as Equation (8)

$$\hat{h}(\rho) = \pi / \arccos(1-\rho). \tag{8}$$

If we extend ρ so that $\rho \in (0.5, 2)$, Equation (7) is still correct for internode polygonization, and the integer $h(\rho) \in (1, 3)$ or $h(\rho) = 2$, so that $h(\rho)$ has to be clamped to value 3 standing for a prism.

This way we construct an analytical and fast way of defining the discretization ratio, using the piecewise $h(\rho)$ function Equation (9), which is called the *edge function*

$$h(\rho) = \begin{cases} 3; & 0.5 \leq \rho < 2; \\ i+1; & P_{i+1} \leq \rho < P_i, 3 < i < H-1; \\ H; & \rho < P_H; \end{cases} \tag{9}$$

Figure 5 is a graph showing the relation of the edge number function $h(\rho)$ (blue) and its continuous basis function $\hat{h}(\rho)$ (red).

Polygonization of an internode. In order to simplify branch geometry, each internode is polygonized into a prism if the numbers of vertices of two ends are the same, or into a *prismoid* if they are different. Let H_k be the *number of edges* of a prismoid approximating

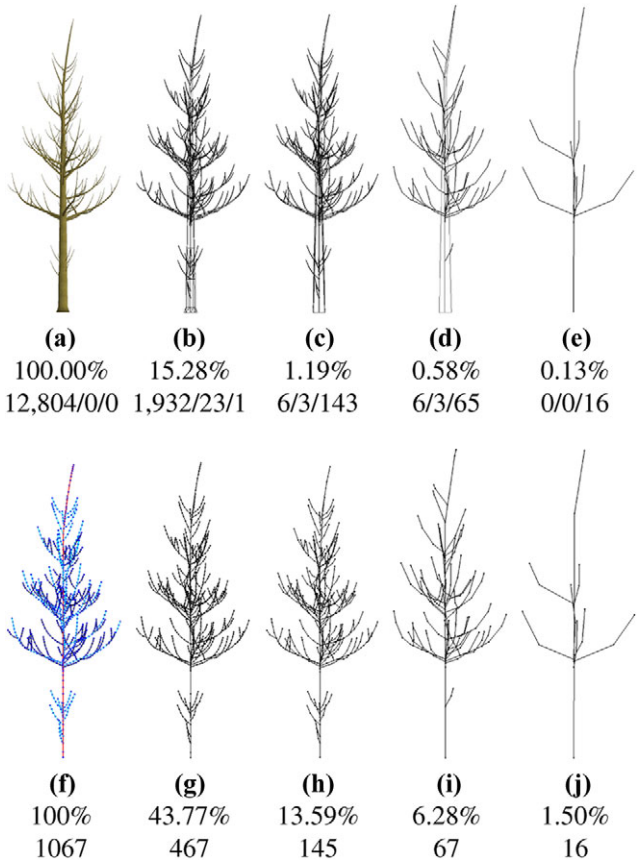


Figure 8: *Tree branch simplification and its corresponding topology simplification (after the internode merging process) of a virtual tree, 2.2 metres tall, and viewed from 36 meters away. (a) to (e) are the simplification of a holly tree model from a detailed mesh to a few lines based on different error thresholds. The simplification ratio of graphical primitives and primitive counts (quadrilaterals/triangles/lines) are shown below the top row. (f) to (j) show topology simplification, where the simplification ratios of the skeleton line counts are shown below the bottom row. The skeletons in (f) are coloured based on their topological hierarchy: skeleton line counts: high level skeleton with red, medium with cyan), and low with blue.*

the internode cylinder at node $\vec{G}_k$ with E_3 as the error, H_k is derived by

$$H_k = h(E_3/R_k). \tag{10}$$

For simplicity, the approximation of a prismoid to an internode cylinder is estimated at the two end circles only.

For the vertex connection at internode I_k, we consider the difference of vertex numbers at its two ends, $d_k = \|H_{k+1} - H_k\|$, and $f_k = min(H_k, H_{k+1})$. There are generally three cases for vertex connection based on the values of H_k and f_k. In Figure 6(a), a branch is shown from its top section view; the inner circle and outer circle represent the k-th and $k+1$-th circle of the branch.

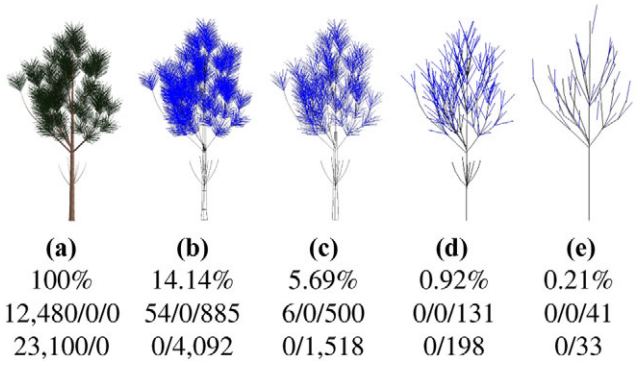


Figure 9: *A 5-year-old Scots pine tree, 2.2 metres tall and viewed from 36 metres away, with both its branches (black) and conifer leaves (blue), which are simplified at various ratios. The primitive counts (branch quads/branch triangles/branch lines) and the primitive counts (needle polygons/needle lines) are shown below each figure.*

(i) If $f_k = 0$ and $d_k \geq 3$, like $H_k \geq 3$ and $H_{k+1} = 0$ for example, we have a cover defined by a triangle fan of d_k triangles as shown in the top section of Figure 6(b). Thus, the following cases concern $f_k > 0$.

(ii) If $d_k = 0$, the internode is a prism of H_k rectangles (quads) as shown in the middle section of Figure 6(b).

(iii) If $d_k \geq 1$, the internode is a prismoid made of d_k triangles and f_k rectangles, as shown in Figure 6(a) when $d_k = 2$. We avoid thin triangles by this construction to ensure mesh quality. In practice, because there is a limited number of cases, the connection of the mesh is pre-generated and looked-up during run-time mesh construction.

Figure 6(c) shows a mesh model of a virtual white poplar tree (*Populus alba*), directly constructed from the branch equations shown in Figure 4(a), therefore rendered from the same camera position. The construction is based on polygonization of internode ends and polygonization of an internode. It should be noted that the meshes of each branch are independent from each other. At the branch node points, no intersection is explicitly calculated in order to maintain a single mesh for the entire branch.

In order to illustrate LOD representation of branches with the mesh model, we choose to sample node points from a circular helix, a typical curve with constant curvature and constant torsion. Figure 7 shows multiresolution meshes of the cylinder from high multiresolution to low, with the number of quads used beneath the figure. The number of edges of each prism is 12, 6, 4, 3 and 3, respectively. The lines of longitude of Figure 7(c) show the torsion of the bi-normal vectors.

Although this combination is helpful in ensuring a smooth surface, in our case we construct mesh LODs from skeletons on-the-fly for each branch and thus discrete elements are required. On the other hand, the decrease in visual quality resulting from this is minimal for forest navigation since, most of the time, individual trees are viewed from a significant distance.

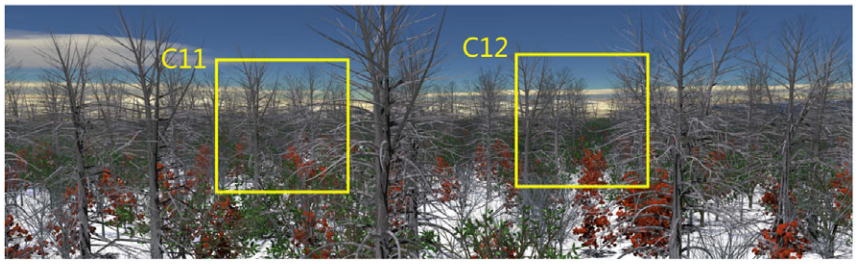


Figure 10: *Rendering result of the semi-transparent line models (Ground truth).*

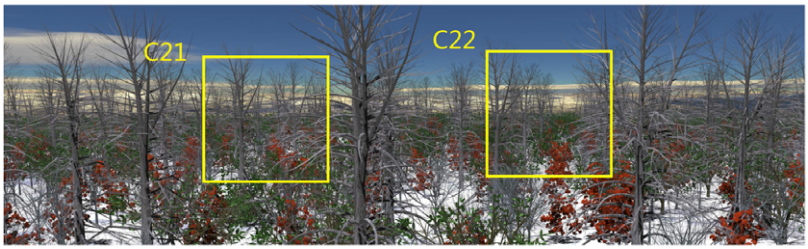


Figure 11: *The instancing method presented by [BLZD12] gives a sparse looking image*

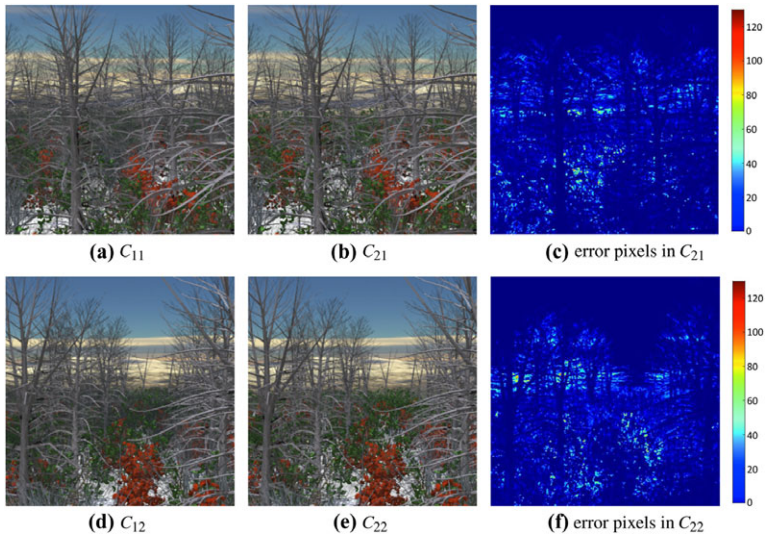


Figure 12: *Differences between detail windows in Figures 10 and 11. (a) and (b) are detailed views of windows C_{11} and C_{21}, respectively. (c) shows pixel differences in C_{21} compared to C_{11}, with false colours, where the value of each pixel of (c) is the absolute difference of the intensity of that pixel in C_{11} and that in C_{21}, ranging from 0 to 255. (d) and (e) are detailed views of C_{12} and C_{22}. (f) shows pixel differences in C_{22} compared to C_{12}, with false colours.*

5.2. LOD representation with lines

Switching from polygons to semi-transparent lines is meaningful for representations of distant branch geometry. This simplifies geometry and warrants higher quality anti-aliasing during rendering [DCSD02]. We will consider two aspects: linearization for internodes and topology simplification.

Linearization for internodes. We can extend Equation (9) to include line geometry, considering $\rho \geq 2$. In fact, h can be any integer satisfying $\rho \geq 1 - \cos(\pi/h)$, so $h = 1$ is chosen for the linearization of a branch in form of a polyline; the width of each segment satisfies $2R = 2E_3/\rho < 1$. For this purpose, Equation (9) is extended to Equation (11)

$$h(\rho) = \begin{cases} 1; & 2 \leq \rho; \\ 3; & 0.5 \leq \rho < 2; \\ i+1; & P_{i+1} \leq \rho < P_i, 3 < i < H-1; \\ H; & \rho < P_H. \end{cases} \quad (11)$$

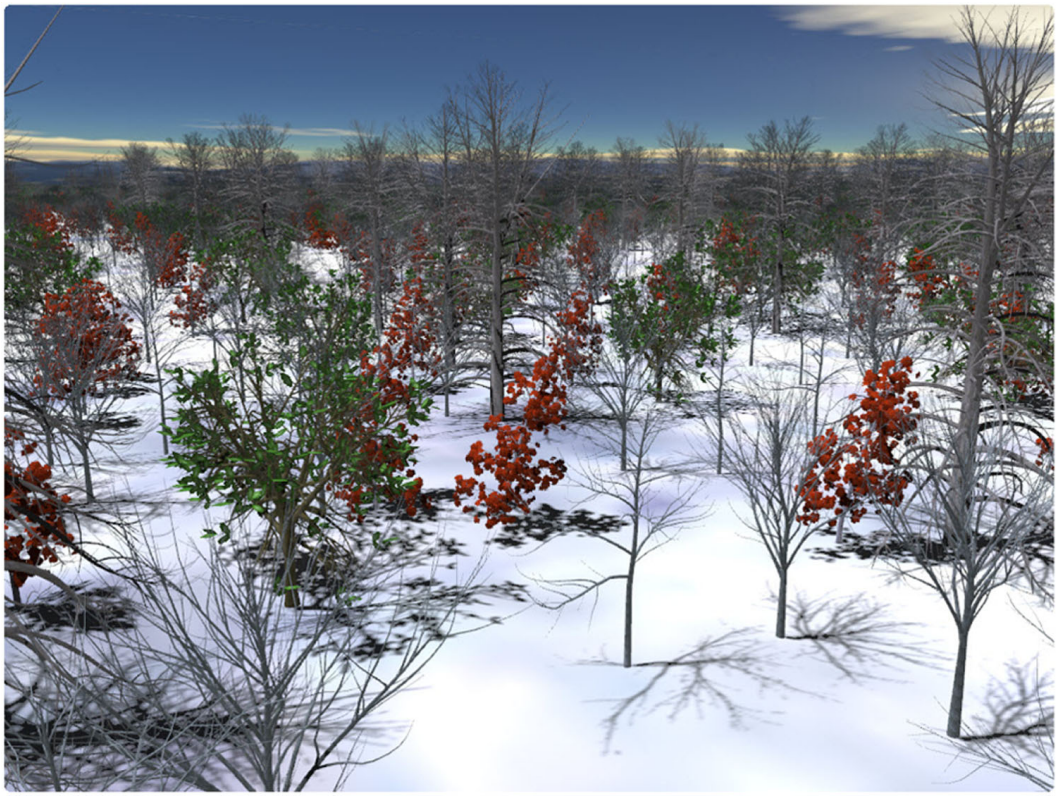


Figure 13: *Our instancing result. Error pixels are dramatically reduced by our new instancing method.*

Equation (11) provides a consistent LOD mechanism for creating line representations and meshes which is efficient for direct shape representations in graphics memory. As line discretization is usually not applied at a sub-pixel level, we compute its transparency (alpha-value α) with a penalty for improper line thickness or darkness:

$$\alpha(\rho) = \begin{cases} 2/\rho, & \rho \geq 2; \\ 1, & \rho < 2; \end{cases} \tag{12}$$

5.3. Topology simplification

While replacing meshes with lines significantly reduces the number of graphical primitives, more lines than necessary might still remain when a tree is viewed from far. Therefore, further line simplification is highly desirable.

Simplifying lines entails not only replacing connected lines with a few line segments, but also removes small branch segments. We call this operation *topology simplification*, or *branch decimation*.

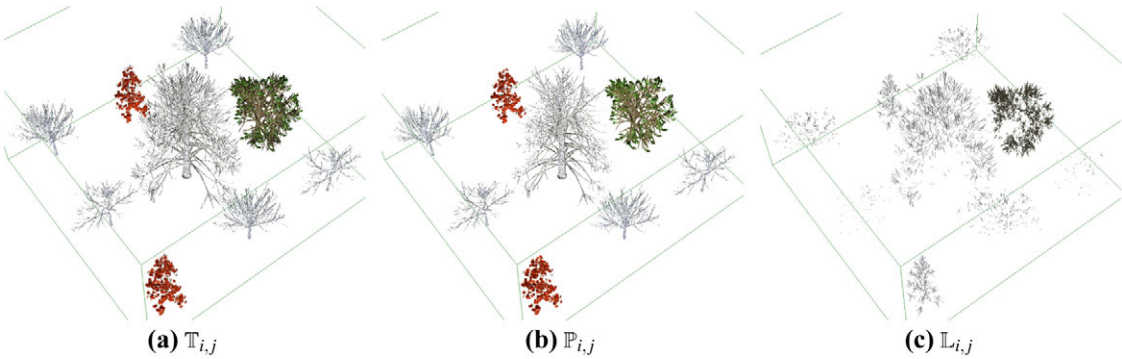


Figure 14: *The decomposition of a tree patch $\mathbb{T}_{i,j}$ in (a), into a mesh patch $\mathbb{P}_{i,j}$ in (b) and a semi-transparent line patch $\mathbb{L}_{i,j}$ in (c). Each patch is made up of 5 tree samples.*

We guide the removal of small branch by the Hausdorff distance. The distance is computed between each branch and its parent branch (its bearer). The accumulated distance of the branch to its parent and all its child branches defines when the branch can be deleted. E_4 is the error threshold that controls this operation.

The effect of topology simplification represented by skeleton lines is shown in Figures 8(i) and (j).

5.4. LOD sequence of tree branches

Lastly, the complete LOD sequence of a tree is created by combining branch internode merging, branch multi-resolution models using meshes, branch multi-resolution models using lines, and topology simplification.

Figure 8 shows a multi-resolution representation of a 6-year-old holly tree, (*Ilex*) with a height of 2.2 metres based on branch simplification seen from a distance of 36 metres. Figures 8(a) to (e) show the tree model simplification from a detailed mesh to a few lines based on different error thresholds, where the permitted spatial errors, assuming $E_1 = E_2 = E_3 = E_4$, are given as 0.092, 3.87, 10.8, 18.8 and 39 centimetres, respectively. The spatial error of 0.092 centimetres represents the finest model corresponding to an error of 0.5 pixel. These operations result in a significant reduction of the number of geometrical primitives.

Figures 8(f) to (j) show the effect of skeleton simplification corresponding to Figures 8(a) to (e), where the effect of internode merging can be seen. The drastic reduction is made possible by internode merging and topology simplification. Note that even the extremely simplified representations in Figure 8(d) still maintain the main structure and shape of the tree.

These pixel errors, or permitted spatial error, can be set to other values based on user's requests. Setting high error metrics will lead to over-simplified models, with straight branches in form of multiple lines and a limited numbers of edges for a prismatic geometry. Such a request may be made to increase the rendering speed; corresponding models are shown in Figures 8(d) and (e). However, setting low error values is of less interest; the effects would reverse. Using the model in Figure 8(a) for the case of subfigure (b), unnecessary details would be kept for a very fine quality and rendering speed would slow down without any gains in visual quality.

5.5. LOD of a conifer tree

Conifer needles are simpler in shape, on the other hand the number of conifer needles is very large, so we only concentrate on them. Such needles are organised as bundles along twigs. The structure of conifer trees is very similar to that of all branches of a tree, so our approach can also be used for the simplification of the whole conifer tree.

The shape of conifer needles and small twigs is much simpler than that of branches. The diameter is nearly constant for all needles and needle textures can be omitted. We have developed a simplified way of dealing with conifer needles that saves GPU memory and makes the process of leaf simplification easier.

Our simplification is performed by iteratively merging two closeby needles into one. This simplification stops when only one needle line is left. The distance between two needles, defined from their extremes (points and radii), is used to measure the error for this simplification. It is a linear combination of three metrics: the distance between the lines, the angle between the lines, and the merging rank of each needle. The merging rank of a needle is defined as the number of needles that it represents. The rank of each needle is one at start, and the coefficients of the three error metrics are empirically set to 0.25, 0.25 and 0.5. This iterative merging is a time consuming process. An Octree data structure is used to improve finding the most similar needles. This merging calculation can also be performed in pre-processing and is then stored for an on-the-fly usage. An error threshold E_5 is specified for controlling this operation.

Figure 9 shows a 2.2-metre-tall conifer tree, a Scots pine, viewed from a distance of 36 metres with different LODs for branches and needles, where the permitted spatial errors ($E_1 = E_2 = E_3 = E_4 = E_5$) are set to values of 0.047, 0.45, 1.82, 8.75, 22 centimetres, respectively. The spatial error of 0.047 centimetres represents the finest model that corresponds to an error of 0.5 pixel.

To enable proper illumination of lines from different viewing directions, the normals of the line 'surfaces' need to be approximated. To do this we use three lines with evenly sampled normal directions to represent one internode.

6. Forest Navigation using LOD Models

To make effective use of new hardware instancing techniques for fast forest navigation, a series of discrete LOD models are extracted first. We use so-called Large-Scale Instancing (LSI) rendering of trees in [BLZD12] for mesh models and have developed a new technique for line models. The LOD models are selected based on their associated spatial errors. Let us assume that we have M tree samples in the forest scene, and each has N user-specified LOD levels.

To achieve a good visual result while keeping as few polygons as possible, the permitted pixel error δ of the finest model of a tree is set to 0.5 pixels, and other LOD models of the tree are adjusted according to their relative distance from the camera. The permitted spatial error ε of the lowest LOD model is heuristically set to 22 centimetres. Experiments carried on various tree structures led to this value. As an example on broad-leaf trees, in Figures 8(d) and 8(e), their respective spatial errors are 18.8 and 39 centimetres respectively. In Figure 8(e), significant parts of the tree structure are lost. Another example using a conifer is given in Figure 9(e) here, the spatial error set to 22 centimetres. In this case the overall tree structure is kept, despite the disappearance of the small branch vertices on the trunk.

Let N LOD levels of M tree samples be represented as $\{\mathbb{T}_{i,j}; 1 \leq i \leq M, 1 \leq j \leq N\}$. The geometry $\mathbb{T}_{i,j}$ is separated into two parts: $\mathbb{P}_{i,j}$ and $\mathbb{L}_{i,j}$

$$\{\mathbb{T}_{i,j}\} = \{\mathbb{P}_{i,j}\} \cup \{\mathbb{L}_{i,j}\}, \tag{13}$$

Table 1: *Comparison with eleven state-of-the-art forest rendering techniques. The symbol '+' means that the proposed technique is efficient for the concerned criterion; conversely, the symbol '-' means either inefficient, or inappropriate.*

	Branch details	Single Tree Branch LOD continuity	Memory cost	Scene scale	Forest Real-time rendering	Real-time shadow
Realistic lighting (RRL) model[BN12]	Low	–	High	Large	+	+
Leaf distribution model (LDM)[ACV*14]	Low	–	Low	Large	+	–
Grouping branches (CGB)[BK04]	Low	–	Middle	Very Small	–	–
Billboard clouds (BC)[BCF*05]	Middle	–	Middle	Large	+	–
View-dependent pruning (VDP)[GCRR11]	Middle	–	Middle	Small	+	–
Multi-resolution foliage (MRF)[DZYJ10]	Middle	poor	Middle	Small	+	–
Large-scale instancing (LSI)[BLZD12]	Middle	–	Middle	Large	+	+
Generalized cylinder (GC)[Blo85, BL99]	Middle	–	Middle	Very Small	–	–
Texture-lobe (TL)[LPC*11]	Middle	–	Low	Middle	+	+
Generic tree model (GTM)[PMKL01]	Middle	–	High	Small	–	–
Bud fate control (BFC)[PHL*09]	High	–	High	Middle	–	–
Adaptive Billboard Clouds (ABC)[KCD*14]	Middle	–	High	Large	+	–
Hybrid Representation (HR)[DLX*15]	Middle	–	Middle	Small	+	–
Our method	High	+	Middle	Large	+	+

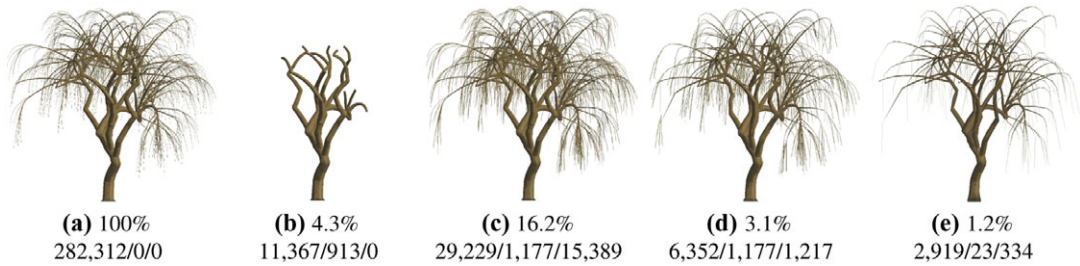


Figure 15: *A comparison of tree model simplification methods for a 25-year-old weeping willow model (Salix babylonica), 7.9 metres tall and viewed from 129 metres away. (a) is the original model. (b) is simplified by Llunch et al. [LCV03]. (c) to (e) are simplified models with our method. Primitive counts (quadrilaterals/triangles/lines) are shown below each figure.*

where $\mathbb{P}_{i,j}$ consists of the set of opaque polygon meshes, and $\mathbb{L}_{i,j}$ is the set of semi-transparent lines with the N different transparencies. The transparency value of each LOD level is the same. N is restricted to 8 due to hardware constraints. Figure 14 shows the decomposition of a tree patch $\mathbb{T}_{i,j}$ into an opaque polygon mesh patch $\mathbb{P}_{i,j}$ and semi-transparent line patch $\mathbb{L}_{i,j}$. We fill in the frame buffer, render opaque mesh models, including broad leaves, the terrain, the sky, and all $\mathbb{P}_{i,j}$ first. $\mathbb{L}_{i,j}$ is rendered with our approach as specified below.

6.1. Rendering semi-transparent line models

Order-independent transparency techniques such as alpha-to-coverage and depth peeling cannot efficiently deal with scenes having many complex objects. The alpha-to-coverage technique employs multisampling techniques to render semi-transparent objects. However, its rendering result is inferior to alpha blending, and larger sample masks would reduce the rendering speed. Depth peeling is not suitable for rendering scenes containing a large quantity of objects as too many render passes would be needed. To achieve a higher quality of the synthesized images, we render opaque objects first and then do a depth-sorted rendering of transparent surfaces, with appropriate alpha blending. Semi-transparent lines and their instances are rendered after the opaque polygons.

We organize the $\mathbb{L}_{i,j}$ sets into a vertex buffer object. After rendering $\mathbb{P}_{i,j}$, including the leaf sets, we render the $\mathbb{L}_{i,j}$, sorted far-to-near, according to the viewing distance. In the fragment shader, we employ Phong shading and set the alpha value of each fragments according to Equation (12). Figure 10 shows a rendering result of models with semi-transparent lines, where hardware instancing is not applied.

6.2. Instancing semi-transparent line models

To reduce CPU overhead with small batch sizes, we employ hardware instancing [BLZD12] to instance $\mathbb{P}_{i,j}$ both in the light frustum for computing shadows and in the view-frustum for forest navigation. When walking through or flying over a forest, shadows have a stronger impact for close objects than those in medium and far distances. Shadow changes caused by LOD models switching

between two adjacent LOD levels are not significant. Semi-transparent lines, which require a lot of rendering resources, contribute little to shadows, so in our implementation, $\mathbb{L}_{i,j}$ is not rendered in the light frustum.

Tree culling and LOD specification are entirely implemented on the GPU to avoid breaking drawing batches by complex and expensive CPU-based methods. This method, however, generates erroneous pixels for instancing semi-transparent models. It instances one tree sample after another, so the rendering order is not correct. For example, $\mathbb{L}_{i+1,j}$ is farther away than $\mathbb{L}_{i,j}$ and it is occluded by $\mathbb{L}_{i,j}$ in some positions, fragments from $\mathbb{L}_{i+1,j}$ will not be rendered due to their larger depth values. This may result in an image with sparse backgrounds as shown in Figure 11.

Figure 12 shows the differences between two pairs of snapshots from Figure 10 and in Figure 11, where Figure 10 is rendered by considering the sequence of semi-transparent lines given above, and Figures 11 is rendered with no such consideration. Two false colour diagrams in Figures 12(c) and (f) show the differences.

We have implemented three improvements above [BLZD12] to reduce the rendering errors from semi-transparent lines. Figure 13 shows the rendering results of our new instancing approach. The improvements are:

(1) We pack all M tree samples into one single tree patch represented as $\{\mathbb{T}_j; 1 \leq j \leq N\}$ for each LOD level, where N is the number of LOD levels. This reduces both the rendering effort and the pixel error mentioned above. Each $\mathbb{T}_j$ is made up of a mesh model $\mathbb{P}_j$ and a semi-transparent line model $\mathbb{L}_j$. Figure 14 shows a tree patch consisting of five tree samples. We instance the $\mathbb{T}_j$ set instead of rendering one tree sample after another.
(2) In each tree culling pass, the method of [BLZD12] selects four LOD levels each time. In order to obtain a correct rendering order, we select only one LOD level in the geometry shader instead of multiple levels for $\mathbb{L}_j$. The order of LOD levels is determined according to their distances from the viewer, from far to near.
(3) We use a buffer object to capture the LOD level positions from the culling pass in (2) and sort them from far to near based on their distances from the viewer on the CPU. Here we obtain the right rendering order. Lastly, the sorted positions are sent back to the GPU for rendering in each frame.

7. Discussion

We compared our method with thirteen state-of-the-art forest rendering techniques mentioned in Section 2. Table 1 summarizes aspects of this comparison.

All thirteen methods adopt polygons or volumes to represent trees. Trees are rendered without geometry in RRL[BN12]. Branch details are well preserved in two methods only, BFC[PHL*09] and our method. We use detail primitives to represent tree branches and thereby achieve the highest geometry fidelity of all techniques. Only continuous branch LODs are extracted in our method.

The model files in LSI[BLZD12] range from approximately 100 KB to 700 KB. The GPU memory complexity of a forest is O(M) where M is the number of tree samples used in the forest. This is the same memory cost as in MRF[DZYJ10] and our method. The memory cost of TL[LPC*11] is the smallest of all methods but this is only an approximate method. RRL[BN12] costs more than 45 MB per tree sample, so the number of tree samples that can be used simultaneously in a given view is very limited. LDM[ACV*14] requires 5 MB per tree sample. Trees in HR[DLX*15] contain 2000 to 8000 triangles for computer rendering, and 100 to 600 triangles for rendering on mobile devices, so the memory cost is intermediate. Our method falls within a medium range in memory cost; however, it has to extract more LODs to achieve continuous model transition.

BC[BCF*05] extremely simplifies the overall geometrical complexity using billboards, thus allowing large-scale virtual forest rendering. Although Adaptive Billboard Clouds [KCD*14] only comprise a single billboard representation for a given model, this method can be naturally extended to generate a hierarchical representation of billboards for large forests. HR[DLX*15] renders only 30 to 100 trees in the view frustum and cannot handle large-scale forests. LSI[BLZD12] and our method benefit from hardware instancing to render large scale forests with highly detailed trees. RRL[BN12] allows rendering of vast forest, too. LDM[ACV*14] uses OpenGL 4 tessellation and geometry shaders to refine leaves; however, simplification of branch geometry is not considered.

Geometry modifications are made frequently in MRF[DZYJ10] and VDP[GCRR11] when the view position changes. These two methods do not benefit from hardware instancing and thus the sizes of possible forests are limited. For the same reason, MRF[DZYJ10], VDP[GCRR11], ABC[KCD*14] and HR[DLX*15] cannot handle real-time shadowing properly, as they must recalculate the model geometry in the light frustums, leading to considerable overload. Among them, RRL[BN12] provides the most realistic lighting effect of forests.

All in all, although our method does not perfectly and simultaneously solve all problems, it surpasses the other methods in most aspects, especially in geometrical fidelity (thin branches in navigations), scene scale and GPU performance usage.

8. Results

Our system was developed in C++ using OpenGL 4.4 and runs on a Windows 7 system. The hardware is a desktop computer with Core i7 CPU 920 2.67 GHz/Intel, 3G RAM, and an NVIDIA GeForce GTX 770 video card.

All tree models were given as polylines obtained from the AMAP-Genesis software. Figure 8 demonstrates the application of our LOD techniques to a conventional tree. To demonstrate the benefits of internode merging and topology simplification, we applied our techniques to a Scots pine tree (*Pinus sylvestris*) with long, thin, and dense leaves (Figure 9). We have further applied our techniques to a weeping willow (*Salix babylonica*) (Figure 15). Figure 15(b) shows that the procedural multi-resolution method by Llunch *et al.* [LCV03] completely removed the long and thin branches that are a prominent feature of willow trees. The models generated by our technique, however, maintain the long and thin branches, and

Figure 16: *A bird's-eye view of a dense forest without leaves (with 11 820 trees and 4 429 236 primitives in the view, where the FPS is 22).*

Figure 17: *A bird's-eye view of a dense forest (with 497 trees and 5 715 347 primitives in the view, where FPS is 24).*

the overall shape of the tree, even when simplified significantly (cf. Figures 15 c–e).

The advantage of our LOD technique is a drastic simplification of plant geometry combined with high fidelity and continuity. Figure 16 shows a bird's-eye view of a winter forest consisting of various tree species, including birch (*Betula*), chestnut (*Castanea*), holly, Aleppo pine (*Pinus halepensis*), and white poplar. It consists of 11,820 trees and 4,429,236 primitives in view, rendered at 22 FPS.

Figure 17 shows a bird's-eye view of another virtual forest which contains older and more complex trees than the forest in Figure

16. In this view 497 trees are visible, and 5.7 million primitives remain after simplification, 4.5% of the full detailed primitive count, corresponding to a compression ratio of 95.5%. 24 FPS are reached with a screen resolution of 1920×945.

9. Conclusions and Future Work

We propose a new multi-resolution representation method for interactive virtual forest rendering. Based on tree skeletons we provide mathematical formulations that describe the geometry of such ramified objects with a wide range of fidelity – from very detailed meshes to simple line approximations. The basic mesh generation is based on Frenet frames of discrete skeletons with torsion control. We define error metrics to guide the generation of geometrical primitives. These error metrics ensure smooth transitions between the various data detail levels.

By merging multiple tree samples, choosing one single LOD level in the geometry shader, as well as sorting the positions, line-models with different degrees of transparency are instanced on the GPU. In this case the number of graphic driver invocations is reduced to a large extent. Our tests show that large-scale forest scenes can be rendered with shadows in real-time.

Keeping skeletons plus Frenet frames as the starting representation to build the LODs on-the-fly is more effective than starting from a fully fledged mesh. This approach consumes less memory and computation for simplification and also guarantees smooth transitions during topology simplification. This strategy can also be useful for facilitating remote visualization of forest data over the internet because only skeletons and Frenet frames need to be transferred while geometrical primitives for branch surfaces can be generated on-the-fly.

Rendering efficiency can be further improved by incorporating other classical acceleration strategies such as occlusion culling. Since meshes are generated on-the-fly based on skeletons, culling away invisible skeletons can avoid unnecessary mesh generation. This will be even more useful when rendering a fully foliated forest. While our method accommodates viewing forests at close or mid-range distances, it is not designed for long-ranges. We plan to integrate our method with existing techniques devoted to long-distance forest representation (e.g. [BN12]) to allow real-time forest rendering at any distance. Texture-lobes [LPC*11] and sub-structures [YdRPH03] could also be used to organize all branches into several groups for more compact representations of the models.

Acknowledgements

This work is supported in part by National Natural Science Foundation of China with Nos. 61331018, 61571439, 61571400 and 61561003, in part by the National High Technology Research and Development Program of China (2015AA016402), in part by the National Foreign 1000 Talent Plan (WQ201344000169), and in part by Leading Talents of Guangdong Program (00201509).

A Appendix

Here are some supplementary materials for detailed mathematical deduction of the main equations of branch shapes and some descriptions of continuous simplification of branches in the body of this paper. The purpose of this section is to avoid considering tedious mathematical deductions.

The description of branch surfaces is based on Frenet frames, which are constructed from their skeleton curves. Continuous simplification of branches is accomplished by repeated internode merging.

A.1 *Frenet frames for branch skeletons*

The Frenet frame consists of the tangent vector $\vec{T}$, normal vector $\vec{N}$, and binormal vector $\vec{B}$, which collectively form an orthonormal coordinate frame along the curve defined by $K+1$ nodes. We calculate the tangent vector $\vec{T}_k$ at each node as the bi-sector of segment angle at first. Similar to [Blo90], the principal normal $\vec{B}$ and bi-normals $\vec{N}$ are defined recursively as

$$\begin{cases} \vec{B}_{k+1} = \dfrac{(\vec{T}_{k+1} \times \vec{N}_k)}{\|\vec{T}_{k+1} \times \vec{N}_k\|}; \\ \vec{N}_{k+1} = \vec{B}_{k+1} \times \vec{T}_{k+1}; \end{cases} \quad k = 0, \ldots, K-1. \tag{A1}$$

As proved in Equation (A2) of Theorem 1, we see that the torsion of binormal vectors of Frenet frame at $\vec{P}_k$ with respect to its neighbour frame at $\vec{P}_{k+1}$ is not bigger than $\beta_k + \beta_{k+1}$, which is the basis of our metrics for branch geometry simplification.

Theorem 1 Torsion constraints**.** *The tangent, principal normal and the bi-normal vectors defined in Equation (A1) satisfy the following inequality (A2)*

$$\begin{cases} \vec{T}_k \cdot \vec{T}_{k+1} \geq \cos(\beta_k + \beta_{k+1}) \\ \vec{N}_k \cdot \vec{N}_{k+1} \geq \cos(\beta_k + \beta_{k+1}) \\ \vec{B}_{k+1} \cdot \vec{B}_k \geq \cos(\beta_k + \beta_{k+1}) \end{cases} \tag{A2}$$

Proof. The first inequality is an obvious deduction from the definition of tangent vectors T_k.

When B_{k+1} in the first equation of Equation (A1) is applied to the second, we have

$$\vec{N}_{k+1} = [\vec{N}_k - (\vec{T}_{k+1} \cdot \vec{N}_k)\vec{T}_{k+1}]/||\vec{T}_{k+1} \times \vec{N}_k||$$

Then we have

$$\begin{aligned} \vec{N}_{k+1} \cdot \vec{N}_k &= [1 - (\vec{T}_{k+1} \cdot \vec{N}_k)^2]/||\vec{T}_{k+1} \times \vec{N}_k|| \\ &= ||\vec{T}_{k+1} \times \vec{N}_k|| \end{aligned}$$

Since

$$\begin{aligned} (\vec{T}_{k+1} \cdot \vec{N}_k)^2 &\leq 1 - (\vec{T}_{k+1} \cdot \vec{T}_k)^2 \\ &= 1 - [\cos(\beta_k + \beta_{k+1})]^2 \end{aligned}$$

Then

$$\vec{N}_k \cdot \vec{N}_{k+1} \geq \cos(\beta_k + \beta_{k+1})$$

Similarly

$$\begin{aligned}\vec{B}_{k+1}\cdot\vec{B}_k &= [\vec{T}_{k+1}\cdot(\vec{N}_k\times\vec{B}_k)]/||\vec{T}_{k+1}\times\vec{N}_k||\\ &\geq \vec{T}_k\cdot\vec{T}_{k+1}\\ &\geq \cos(\beta_k+\beta_{k+1}.)\end{aligned}$$ □

A.2 *Branch equation and internode merging*

This subsection explains details of the Branch Equation, Equation (1). We modelled the geometry of internode I_k by a simple ruled surface. This surface, constructed from $\vec{G}_k$ and $\vec{G}_{k+1}(0\leq k\leq K-1)$ was described in Equation (1) where $u=(t-t_k)/(t_{k+1}-t_k)$, the unifies parameter of t at $[t_k,t_{k+1}]$. The corresponding normal vector equation of (1) is approximated by

$$\begin{aligned}\vec{D}_k(t,\theta) = (1-u)\cdot[\cos\theta\cdot\vec{N}_k+\sin\theta\cdot\vec{B}_k]\\ +u\cdot[\cos\theta\cdot\vec{N}_{k+1}+\sin\theta\cdot\vec{B}_{k+1}]\end{aligned}\tag{A3}$$

which is a linear interpolation of the normal vectors on two end circles of the internode.

Polygonization of a branch involves using a prismoid for each internode cylinder Therefore, vertices at node $\vec{G}_k$ $(k=0,\ldots,K)$ are specified as:

$$\begin{aligned}\vec{V}_{j,k} &= \vec{V}(t_k, j\cdot 2\pi/(2H_k))\\ &= \vec{P}_k + R_k\cdot\vec{N}_k\cdot\cos(j\cdot\pi/H_k)\\ &\quad + R_k\cdot\vec{B}_k\cdot\sin(j\cdot\pi/H_k)\end{aligned}\tag{A4}$$

deduced from (1) ($j=0,\cdots,H_k$, $k=0,\ldots,K$). H_k defines the degree of prismoid disretization step in (10). The normal vectors at vertex $\vec{V}_{j,k}$ are

$$\vec{D}_{j,k}=\vec{N}_k\cdot\cos(j\cdot\pi/H_k)+\vec{B}_k\cdot\sin(j\cdot\pi/H_k)\tag{A5}$$

A.3 *Error analysis on internode merging*

This subsection explains details of Equation (2).

Theorem 2 Error on internode merging**.** *Given an internode group $U_{k,q+2}(q\geq 0)$, the error $\psi_{k,q+2}$ between $I_{k,q+2}$ and $U_{k,q+2}$ satisfies*

$$\psi_{k,q+2}\leq\phi_{k,q+2}+\varphi_{k,q+2},\tag{A6}$$

where $\phi_{k,q+2}$ and $\varphi_{k,q+2}$ are specified in Equation (3).

Proof. The function of $I_{k,q+2}$ is

$$\begin{aligned}\vec{V}_{k,q+2}(t,\theta) = (1-t)\vec{P}_k + t\cdot\vec{P}_{k+q+2}\\ +[(1-t)R_k\vec{N}_k + tR_{k+q+2}\vec{N}_{k+q+2}]\cos\theta\\ +[(1-t)R_k\vec{B}_k + tR_{k+q+2}\vec{B}_{k+q+2}]\sin\theta\end{aligned}$$

Since $I_{k,q+2}$ defines a ruled surface, and $U_{k,q}$ is a surface generated by polylines, the maximum difference is between a series of polylines and corresponding single line segments. Therefore $F_{k,q+2}$ comes from the maximum value of

$$\begin{aligned}&\vec{V}_{k,q+2}(\tau_{k,j},\theta)-\vec{V}_{k+j+1}(\tau_{k,j},\theta)\\ &\quad=(1-\tau_{k,j})\vec{P}_k+\tau_{k,j}\vec{P}_{k+q+2}-\vec{P}_{k+j+1}\\ &\quad+\cos\theta[(1-\tau_{k,j})R_k\times\vec{N}_k+(\tau_{k,j})R_{k+q+2}\times\vec{N}_{k+q+2}\\ &\quad-R_{k+j+1}\times\vec{N}_{k+j+1}]\\ &\quad+\sin\theta[(1-\tau_{k,j})R_k\times\vec{B}_k+\tau_{k,j}R_{k+q+2}\times\vec{B}_{k+q+2}\\ &\quad-R_{k+j+1}\times\vec{B}_{k+j+1}]\end{aligned}$$

for all $0\leq j\leq q$, because $\tau_{k,j}\in[0,1]$. Thus

$$\psi_{k,q+2}\leq||X_{k,j}||+||Y_{k,j}||\cos\theta+||Z_{k,j}||\sin\theta$$

Therefore,

$$\psi_{k,q+2}\leq\phi_{k,q+2}+\varphi_{k,q+2}.$$ □

References

[ACV*14] Andújar C., Chica A., Vico M. A., Moya S., Brunet P.: Inexpensive reconstruction and rendering of realistic roadside landscapes. *Computer Graphics Forum 33*, 6 (2014), 101–117.

[BCF*05] Behrendt S., Colditz C., Franzke O., Kopf J., Deussen O.: Realistic real-time rendering of landscapes using billboard clouds. *Computer Graphics Forum 24*, 3 (2005), 507–516.

[BK04] Beaudoin J., Keyser J.: Simulation levels of detail for plant motion. In *Proceedings of the 2004 ACM SIGGRAPH/ Eurographics Symposium on Computer Animation* (2004), pp. 297–304.

[BL99] Bloomenthal J., Lim C.: Skeletal methods of shape manipulation. In *SMI '99: Proceedings of the International Conference on Shape Modeling and Applications* (Washington, DC, USA, 1999), IEEE Computer Society, p. 44.

[Blo85] Bloomenthal J.: Modeling the mighty maple. In *SIGGRAPH '85* (1985), pp. 305–311.

[Blo90] Bloomenthal J.: Calculation of reference frames along a space curve. In *Graphics Gems*. A. Glassner (Ed.). Academic Press, Boston (1990), pp. 567–571.

[Blo95] Bloomenthal J.: *Skeletal Design of Natural Forms*. PhD thesis, Calgary, Alta., Canada, 1995.

[BLZD12] Bao G., Li H., Zhang X., Dong W.: Large-scale forest rendering: Real-time, realistic, and progressive. *Computers & Graphics 36*, 3 (2012), 140–151. Novel Applications of VR.

[BN12] Bruneton E., Neyret F.: Real-time realistic rendering and lighting of forests. *Computer Graphics Forum 31*, 2pt1 (2012), 373–382.

[CHPR07] Cook R. L., Halstead J., Planck M., Ryu D.: Stochastic simplification of aggregate detail. *ACM Transactions on Graphics 26*, 3 (July 2007), 79:1–79:8.

[DCSD02] Deussen O., Colditz C., Stamminger M., Drettakis G.: Interactive visualization of complex plant ecosystems. In *Proceedings of the Conference on Visualization '02* (Washington, DC, USA, 2002), VIS '02, IEEE Computer Society, pp. 219–226.

[DHL*98] Deussen O., Hanrahan P., Lintermann B., Mech R., Pharr M., Prusinkiewicz P.: Realistic modeling and rendering of plant ecosystems. In *SIGGRAPH '98* (1998), pp. 275–286.

[DLX*15] Dong T., Liu S., Xia J., Fan J., Zhang L.: A time-critical adaptive approach for visualizing natural scenes on different devices. *PLoS ONE 10*, 2 (2015), 1–26.

[DN04] Decaudin P., Neyret F.: Rendering forest scenes in real-time. *Computer Graphics Forum* (2004), 93–102. Proc. Symposium on Eurographics.

[dRDJ90] de Reffye P., Dinouard P., Jaeger M.: Basic concepts of computer plants growth simulation. *NICOGRAPH'90 6*, 1 (1990), 219–233.

[dREF*88] de Reffye P., Edelin C., Françon J., Jaeger M., Puech C.: Plant models faithful to botanical structure and developmentr. In *SIGGRAPH '88* (1988), vol. 22, pp. 151–158.

[DZYJ10] Deng Q., Zhang X., Yang G., Jaeger M.: Multiresolution foliage for forest rendering. *Computer Animation and Virtual Worlds 21*, 1 (2010), 1–23.

[GCRR11] Gumbau J., Chover M., Remolar I., Rebollo C.: View-dependent pruning for real-time rendering of trees. *Computers & Graphics 35*, 2 (2011), 364–374.

[GMN05] Gilet G., Meyer A., Neyret F.: Point-based rendering of trees. In *Eurographics Workshop on Natural Phenomena* (2005).

[GS02] Garland M., Shaffer E.: A multiphase approach to efficient surface simplification. In *Proceedings of Visualization '02* (2002), pp. 117–124.

[HB05] Hernandez E., Benes B.: Robin hood's algorithm for time-critical level of detail. In *Graphicon'2005 Russia* (2005).

[JT03] Jaeger M., Teng J.: Tree and plant volume imaging - an introductive study towards voxelized functional landscapes. In *Proceedings on Plant Growth Modeling and Applications* (2003), pp. 169–181.

[KCD*14] Kratt J., Coconu L., Dapper T., Schliep J. W., Paar P., Deussen O.: Adaptive billboard clouds for botanical tree models. *Digital Landscape Architecture* (2014), 274–282.

[LCV03] Lluch J., Camahort E., Vivó R.: Procedural multiresolution for plant and tree rendering. In *AFRIGRAPH '03* (2003), pp. 31–38.

[LD99] Lintermann B., Deussen O.: Interactive modeling of plants. *IEEE Computer Graphics and Applications 19*, 1 (1999), 56–65.

[LP02] Lane B., Prusinkiewicz P.: Generating spatial distributions for multilevel models of plant communities. In *Graphics Interface* (Calgary, Alberta, Canada, 27–29 May 2002), Canadian Human-Computer Communications Society, pp. 69–80.

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

[LVM04] Lluch J., Vivó R., Monserrat C.: Modelling tree structures using a single polygonal mesh. *Graphical Models 66*, 2 (2004), 89–101.

[Max90] Max N.: Cone-spheres. In *SIGGRAPH '90 Conference Proceedings* (New York, NY, USA, 1990), ACM Press, pp. 59–62.

[MNP01] Meyer A., Neyret F., Poulin P.: Interactive rendering of trees with shading and shadows. In *Eurographics Workshop on Rendering* (2001), pp. 183–196.

[MP96] Mech R., Prusinkiewicz P.: Visual models of plants interacting with their environment. In *SIGGRAPH '96* (1996), pp. 397–410.

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

[ony] Onyx-tree. In http://www.onyxtree.com/, Onyx Computing, Inc. Accessed on 30 June 2014.

[PHL*09] Palubicki W., Horel K., Longay S., Runions A., Lane B., Měch R., Prusinkiewicz P.: Self-organizing tree models for image synthesis. *ACM Transactions on Graphics 28*, 3 (July 2009), 58:1–58:10.

[PL90] Prusinkiewicz P., Lindenmayer A.: *The Algorithmic Beauty of Plants*. Springer-Verlag, New York, 1990.

[PMKL01] Prusinkiewicz P., Mündermann L., Karwowski R., Lane B.: The use of positional information in the modeling of plants. In *SIGGRAPH '01* (2001), pp. 289–300.

[SG97] Schmalstieg D., Gervautz M.: Modeling and rendering of outdoor scenes for distributed virtual environments. In *Proceedings of ACM VRST* (1997), pp. 209–215.

[YdRPH03] Yan H., de Reffye P., Pan C., Hu B.: Fast construction of plant architectural models based on substructure decomposition. *Journal of Computer Science and Technology 18*, 6 (2003), 780–787.

[ZBDS12] Zinsmaier M., Brandes U., Deussen O., Strobelt H.: Interactive level-of-detail rendering of large graphs. *IEEE Transactions on Visualization and Computer Graphics 18*, 12 (Dec 2012), 2486–2495.

Supporting Information

Additional Supporting Information may be found in the online version of this article at the publisher's web site:

Movie S1.