Automatic Reconstruction of Tree Skeletal Structures from Point Clouds

Yotam Livny[1] Feilong Yan[1] Matt Olson[2] Baoquan Chen[1] Hao Zhang[2] Jihad El-Sana[3]

[1] Shenzhen Institutes of Advanced Technology (SIAT), China [2] Simon Fraser Univ. [3] Ben-Gurion Univ.

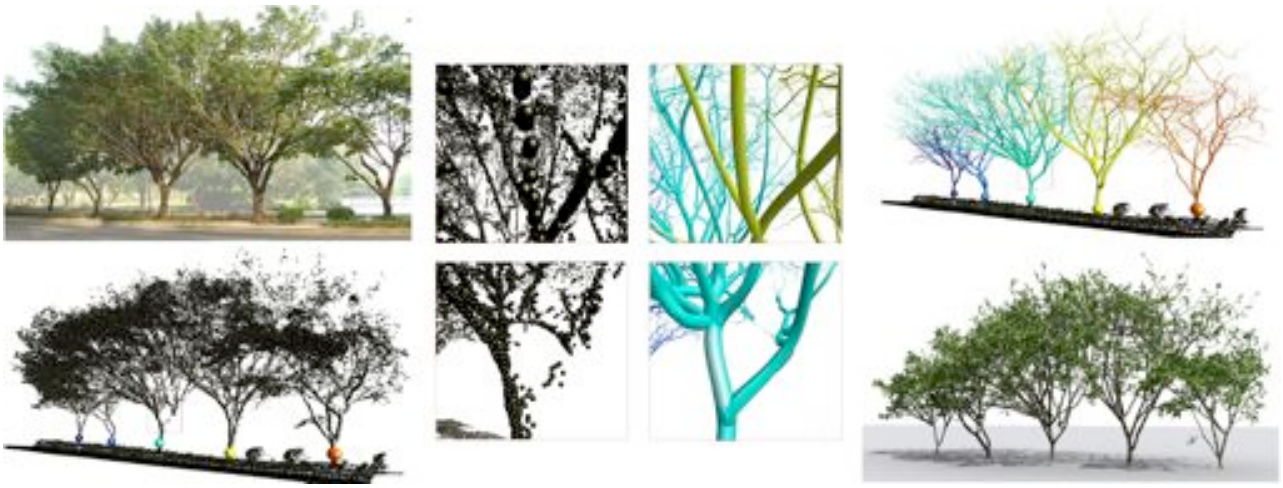


Figure 1: *A scene of five trees automatically reconstructed by our algorithm. The images show a photo of the scene, point cloud, reconstructed trees, and textured models with leaves. The insets show the ability of our method to handle overlapping crowns and missing data.*

Abstract

Trees, bushes, and other plants are ubiquitous in urban environments, and realistic models of trees can add a great deal of realism to a digital urban scene. There has been much research on *modeling* tree structures, but limited work on *reconstructing* the geometry of real-world trees – even then, most works have focused on reconstruction from photographs aided by significant user interaction. In this paper, we perform active laser scanning of real-world vegetation and present an automatic approach that robustly reconstructs skeletal structures of trees, from which full geometry can be generated. The core of our method is a series of *global optimizations* that fit skeletal structures to the often sparse, incomplete, and noisy point data. A significant benefit of our approach is its ability to reconstruct multiple overlapping trees simultaneously without segmentation. We demonstrate the effectiveness and robustness of our approach on many raw scans of different tree varieties.

1 Introduction

Wild and domesticated plants are commonly found throughout the world, and virtual worlds that lack vegetation often seem lifeless and artificial. There has been a great deal of research on modeling trees, predominantly using procedural approaches [Prusinkiewicz and Lindenmayer 1990]. Works on tree reconstruction have mostly been based on photographs [Reche-Martinez et al. 2004; Tan et al. 2007]. With recent advances in laser scanning, direct capture of 3D data of trees has become possible. However, such captures produce scattered points, and most applications require the reconstruction of complete tree geometry from the captured point cloud. In this paper, we are interested in reconstructing trees from point cloud data. The problem is challenging since the input point clouds are almost always incomplete and suffer from significant self-occlusions. The high variance in structure between individual trees, and the complex ways trees can overlap each other, both add to the challenge.

A natural approach to tree construction is to first reconstruct the skeleton of the captured tree, then apply further geometry completion through appropriate rules or heuristics. Skeletal structures are fundamental to tree geometry reconstruction. Existing methods take either an interactive [Quan et al. 2006; Tan et al. 2007] or an empirical approach, where the latter heavily relies on heuristics and manual adjustment of parameters [Xu et al. 2007]. Given the amount of time required in both approaches, an automatic alternative is desirable, especially when handling a large number of possibly overlapping trees. Indeed, state-of-the-art laser scanners often produce large amounts of nonsegmented point data representing hundreds of objects including trees that overlap. It is not only tedious but also difficult for the user to segment the data into individual trees [Tan et al. 2008]. This is necessary when current methods for tree reconstruction expect input points to come from a single tree [Tan et al. 2007; Xu et al. 2007]. These reconstructions also rely on local computations, leading to results that are heavily influenced by data quality over local areas. However, local data quality is rarely consistent in typical laser scans of real-world trees.

In this paper, we present an automatic algorithm for tree reconstruction from laser scans. Our method employs a series of global optimizations to consolidate a point cloud representing one or more tree objects into skeletal structures. This optimization aims to reconstruct the major skeletal branches of the captured tree(s), result-

ing in a graph structure which we call the *Branch-Structure Graph* or BSG for each tree; see Figure 2. Finer structures such as leaves are then synthesized from the BSGs with textures added at the end to complete the reconstruction pipeline.

The main component of the pipeline is the BSG construction step. Our optimization is driven by a set of weak assumptions on the length, thickness, smoothness, and density of tree branches that are naturally expected from biological growth properties of real-world trees [Honda 1971]. Our method can be applied to trees of various types, from large trees to small bushes and including those with diverse branching patterns, as long as they demonstrate significant skeletal structures. Multiple trees of different types, possibly overlapping, can be handled without pre-segmentation.

The key enabling feature of our algorithm and one that distinguishes our approach from previous methods is the use of global optimization, which not only makes the reconstruction robust to noisy and incomplete data, but also provides a better overall approximation of the tree branches. Furthermore, it relieves the user from having to wrestle with parameter tuning, as the algorithm adjusts the optimization settings based on the quality of the data being processed to arrive at the best approximation. We demonstrate the effectiveness and robustness of our reconstruction scheme on a number of raw scans which capture different tree varieties.

2 Related work

Existing literature on geometry reconstruction and urban scene modeling is vast. Here we focus only on works most closely related to ours, including those on tree modeling and extraction of skeletal structures from point clouds. Developments on generating tree models are largely classified into two categories: *modeling* virtual trees and *reconstructing* trees captured from the real world. The first predominantly uses procedural approaches, such as L-systems [Honda 1971; Rozenberg and Salomaa 1980] and their many variants. These approaches typically model trees by applying branching rules in sequence to generate complex structures. More recently, these methods have been guided by user sketches [Anastacio et al. 2006; Wither et al. 2009] and enhanced through examples [Okabe et al. 2006; Chen et al. 2008]. Reconstruction of existing trees can be obtained from either photographs or point clouds; the latter is becoming popular with the rapid development of 3D laser scanning technology.

Photographs can also be used to extract a volumetric representation of trees [Reche-Martinez et al. 2004]. Neubert *et al.* [2007] generate an approximate volumetric tree representation and further perform a 3D flow simulation to form twigs and branches. Photo images have also been used for extracting parameters for L-systems [Prusinkiewicz et al. 2001]. For trees with a dense crown area and little visibility within, Shlyakhter *et al.* [2001] first extract visual hulls from images, then use L-systems to synthesize branches within the crown. For trees with dominant leaf structures, Quan *et al.* [2006] use interactive sketching to generate leaf geometry, aided by sparsely reconstructed 3D points from images. For trees with relatively small leaves but more visible branch structures, Tan *et al.* [2007] automatically synthesize L-system rules from input images. In a follow-up, Tan *et al.* [2008] rely on user-drawn strokes to guide the synthesis, where the input is a single photograph. The reconstruction accuracy and efficiency from these image-guided methods is inherently limited by the quality and density of the 3D points that can be reconstructed from the images.

3D points obtained via laser scanning provide a more direct capture of tree geometry. Most methods focus on extracting skeletons representing tree branches, since points acquired for fine structures

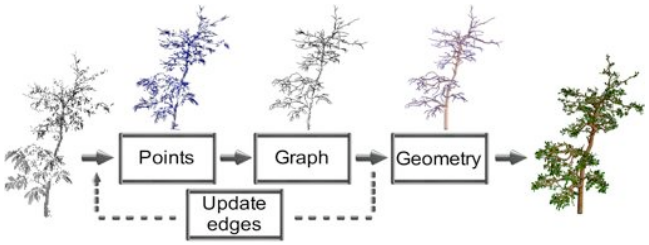


Figure 2: *Tree BSG reconstruction pipeline.*

such as leaves are generally too noisy to allow reconstruction with reasonable accuracy. Most curve skeleton extraction algorithms operate on complete mesh models [Cornea et al. 2007]. On point cloud data, Lazarus and Verroust [1999] use the length of edges in a spanning tree to cluster the points and reveal the skeleton. Runions *et al.* [2007] grow skeletal structures within tree envelopes by using points as local attractors. Bucksch *et al.* [2008; 2009] partition points into octree cells and form a curve skeleton by connecting local extractions in adjacent cells. To cope with significant missing data, Tagliasacchi *et al.* [2009] rely on a local cylindrical prior for curve skeleton extraction from point clouds but require normal information at each point. Recent work of Li *et al.* [2010] reconstructs shapes consisting of interleaved wires using *arterial snakes*, skeletal curves based on local attractors. Xu *et al.* [2007] propose a heuristic-based method to reconstruct major tree branches from 3D scans and then synthetically add small twigs and leaves to form the crown geometry. Côté *et al.* [2009] also synthesize minor tree and leaf geometry, but base their synthesis on light scattering properties obtained from scanned sample intensities.

The main difference between our method and those previously developed for tree skeletal structure reconstruction is that we use a global optimization, which is more robust to noise, nonuniform point density, and missing data than local computations. These artifacts often lead to unreliable estimated normals; however, our method does not require normals. Moreover, our algorithm adapts to the given point data, *e.g.* in clustering points during branch reconstruction and connection. In contrast, the tree reconstruction methods of [Xu et al. 2007; Bucksch et al. 2009] both use pre-defined clustering resolutions (0.2 meters in [Xu et al. 2007] and a preset partitioning resolution in [Bucksch et al. 2009]), which are unreliable when the branches have different densities in different parts of the tree. Finally, our method avoids heavy parameter tuning that is typical of local approaches, relieving the user from the tedious task of having to pre-segment large scans into individual trees.

3 Overview

Our reconstruction algorithm expects an input point scan of a set of trees. The point cloud may be noisy and incomplete, but is assumed to sample the major branch structure of the trees in the input. Our algorithm reconstructs skeletons that faithfully resemble the scanned trees. In order to resolve ambiguities arising from missing data, noise, and self-occlusion, we impose several weak constraints to guide global optimization. We aim to produce an optimized approximation while relieving the user from tedious parameter tuning, allowing for unified processing of multiple trees of different types without pre-segmentation.

Tree skeleton representation Each reconstructed tree is represented by a *Branch-Structure Graph* (BSG), defined as a spatially embedded and connected directed acyclic graph. The root node of the BSG corresponds to the base of the tree.

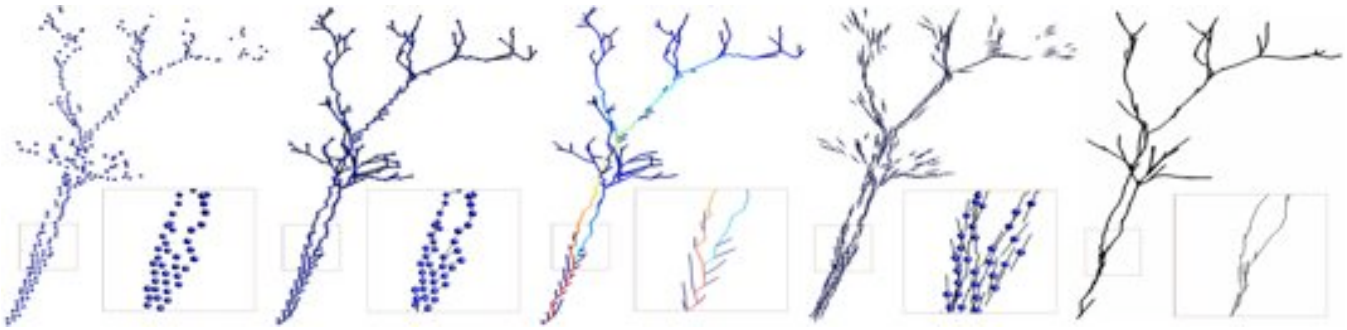
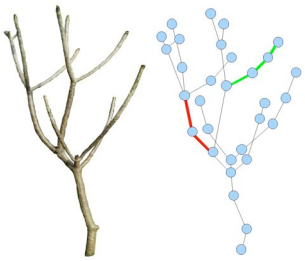


Figure 3: *The point processing and BSG refinement steps. From left to right: the input points; the initial BSG construction; the importance weights of vertices (shown by edge color); the smooth orientation field; and the smoothed BSG structure.*

BSG nodes are connected by straight edges and all lie spatially in the center of the tree branches. A branch is *simple* if it contains no branching points and is represented by a *Branch-Chain* (BC), which is a subgraph of the BSG given by a chain of connected nodes and edges; see the colored edges in the figure to the right. We assign an importance value to each BSG node equal to the size of its subtree (see Section 4.2), which assigns more weight in the global optimization steps to larger branches near the root of the tree.



BSG and two highlighted branch chains (red and green).

Optimization criteria In addition to requiring the reconstructed BSGs to be geometrically close to the input point cloud, we impose the following optimization objectives on the BCs that are derived from biological growth properties of typical trees [Honda 1971].

1. The BCs are smooth, as reflected by small bending angles between adjacent edges.
2. The BCs are longer and thicker near the root of the tree and shorter and thinner near the crown.
3. The density of the BCs is inversely proportional to their corresponding thickness.

Provided that the scanned trees obey these criteria, our reconstruction algorithm is independent of specific tree morphology.

Reconstruction pipeline and optimization The reconstruction pipeline for extracting the BSGs consists of three steps, where the first two are iterated, as shown in Figure 2.

1. **Initializing BSGs from points:** After detecting the ground surface and extracting the tree roots from the input scan, we run a multi-root Dijkstra's algorithm to extract the disjoint initial BSGs.
2. **Refining the BSG graphs:** We begin by assigning importance weights to each vertex based on the sizes of their subtrees. Then for each BSG, we generate a smooth orientation field by minimizing a sum of directional differences between adjacent edges, weighted by these importance values. The orientation field is used to optimize the spatial embedding of the BSGs, achieving a balance between tree smoothness and centered fitting to the point samples.
3. **Inflating the BSGs into tree geometry:** Finally, we compute the thickness or skeleton radius values along the edges of the BSGs as guided by the above optimization criteria.

Each optimization along the pipeline is formulated as a least-squares problem; hence, the solution is exact and efficiently obtained by solving a linear system. Given the input point scans containing one or more trees, execution of the BSG reconstruction pipeline is *fully automatic* and no user input is required.

4 Tree reconstruction

We begin by constructing a set of BSGs from the input point scan using a weighted spanning-tree method. Next, we refine the points to better approximate the geometry of the tree structures we seek, and iterate to compensate for noise or self-occlusion. Finally, we remove noise and build geometry around the refined BSGs.

4.1 Initializing BSGs

Given as input a laser scan of a scene with trees, we must first identify and approximate the skeletal geometry of the trees within the scene. We begin by identifying samples at the base of each tree, then build an initial approximation of the tree's BSG.

Tree base identification While we accept an arbitrary number of trees in our input point scan, we extract and refine tree geometry on a plant-by-plant basis. Therefore, we begin by determining the number of trees and the location of each tree in the input, identifying root nodes for the BSGs representing each tree.

Since the points sampled on a typical tree are centered around its trunk, projecting them onto a ground plane will form a cluster of points with high density. We utilize this characteristic of trees by projecting their samples onto the ground plane ($y = 0$) along the vertical axis and measuring the density of the projected points (see Figure 4). Then, we remove the low-density terrain points from the dataset. The remaining points form clusters, and we select a single point from each cluster and mark it as the root node of a BSG. We can select either the cluster centroid or the sample closest to the centroid as the root; we find that the smoothing process of Section 4.2 renders this choice moot.

Initial BSG construction We construct minimum-weight spanning trees over the input point set, which we call P, from the root nodes identified above. Over the majority of the scan, points that are close together are likely to belong to the same branch, and the tree structures we seek to create are likely to be those with minimum overall edge length. Thus, we construct a graph between

Figure 5: *A tree with thin and dense branches (left). This tree has been scanned from the right side, therefore point density is higher on the right than on the left (middle). Our reconstruction is data dependent – the right side of the output is more accurate than the left.*

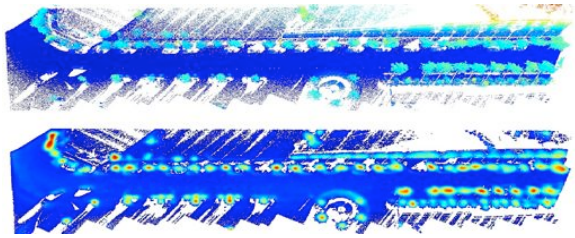


Figure 4: *A scan of trees colored by height (top) and the color-coded density of projected points (bottom). Low density points (blue) form the terrain surface, while high density points (yellow to red) indicate tree locations.*

input points with edges (u, v) weighted by the Euclidean distance $\|u-v\|^2$ and use Dijkstra's algorithm to extract a minimum-weight spanning tree from this graph. To ensure that all root nodes are contained in this spanning tree, we connect them with temporary zero-weight edges, which are removed from the spanning tree to create a forest $\{T_1, \ldots, T_n\}$ of initial BSGs. Once the zero-weight edges are removed, this results in an automatic segmentation of the input points into individual trees.

4.2 Refining BSGs

As explained in Section 3, natural trees tend to have smooth structures with relatively long branches and small angles between branches. Previous approaches insist that the user explicitly identify such structure in the initial graphs T_i [Neubert et al. 2007]. Instead, we define criteria to distinguish between representative vertices and those distorted by factors such as noise or occlusion. This process is complicated by the fact that correct structure differs within a given tree, for example between the trunk and the crown.

Weighting BSG vertices We assign *importance weights* to the vertices of the tree graphs to guide our optimization process. First and foremost, we want edges connecting heavily-weighted vertices to form long, smooth branches. Short branches with low-weight vertices near heavily-weighted samples indicate noise in the dataset. Furthermore, vertices on branches near the crown should have consistently low weights, resulting in the recovery of thin branches in the crown while similar branches at the trunk will be culled away.

To achieve these qualities, each vertex is assigned an importance weight given by the sum of edge lengths in its subtree. While apparently simple, this weight satisfies the critical properties listed above, as illustrated in Figure 3. In this scheme, adjacent vertices in T_i with similarly large subtrees will tend to have similar weights, and the weight of a given vertex will be the sum of the weights of its children. Further, as it depends on a Euclidean distance measure, this weighting scheme is not sensitive to the density of input points.

Building the orientation field We use global least-squares optimization to fit the scanned geometry of each tree to the biological constraints from Section 3. Critical to our method is the construction of an *orientation field* on the vertices of a BSG. Orientation fields have been used in other works [Neubert et al. 2007] to construct smooth tree graphs from point samples; we follow a similar approach, augmented by vertex importance weights.

Given a vertex $v \in T_i$ with parent v_p, we attempt to find an orientation o_v that minimizes the difference between (first) the orientations of both vertices and (second) between the orientation at v and the direction of the edge $e(v, v_p)$. We formulate the first as $\triangle O(T_i)$ in Equation 1 and the second as $\triangle E(T_i)$ in Equation 2.

$$\triangle O(T_i) = \sum_{v \in T_i} \left(\frac{c_{v_p} + c_v}{2} \left\| o_{v_p} - o_v \right\| \right)^2 \quad (1)$$

$$\triangle E(T_i) = \sum_{v \in T_i} \left(c_v \left\| o_v - \frac{e(v_p, v)}{\|e(v_p, v)\|} \right\| \right)^2 \quad (2)$$

Given these constraints, we construct a smooth orientation field that minimizes $\triangle E(T_i) + \triangle O(T_i)$. In both cases, we weight the orientation contributions by the importance weights c_v, c_{v_p} of the vertices involved. This ensures that short, noisy branches do not distort the orientation field near the trunk of the tree (see Figure 3).

Global BSG refinement Having constructed a smooth orientation field over the vertices in T_i, we update the positions of those vertices to reflect their orientations. Again, we perform this optimization by minimizing error functions. We smooth the graph by minimizing $\triangle A(T_i)$, the difference between the direction of an edge $e(u, v)$ and the orientations of incident vertices. To ensure that the original geometry is not lost, we constrain this optimization by $\triangle F(T_i)$, the difference between original and final edge centers.

$$\triangle A(T_i) = \sum_{e(u,v) \in T_i} \left(\frac{c_u + c_v}{2} \left\| (u' - v') - \frac{\|u - v\| \, (o_u + o_v)}{\|o_u + o_v\|} \right\| \right)^2 \quad (3)$$

$$\triangle F(T_i) = \sum_{e(u,v) \in T_i} \left(c_v \left\| \frac{u' + v'}{2} - \frac{u + v}{2} \right\| \right)^2 \quad (4)$$

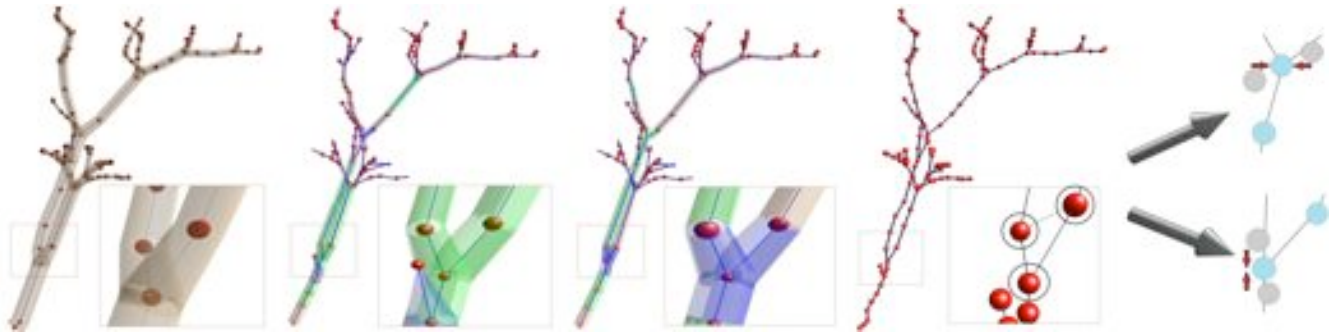


Figure 7: *Steps of the geometry construction process. From left to right: generalized cylinders represent the geometry of the BSG; neighboring cylinders (green) that overlap by more than* 50% *are clustered together (blue); clustering is performed using the edge-collapse operator (we show two possible edge collapses from the same starting configuration – grey points collapse to blue indicated by the red arrows).*

Figure 6: *Several iterations on the data from Figure 5. The images show the BSG generated after* 1*,* 2*, and* 3 *(convergence) iterations. Cluttered geometry in iteration* 1 *resolves to distinct skeletal structure in iteration* 3*. Improvements to the structure can be seen in the bottom of the tree.*

We find the updated vertex positions v' by minimizing $\triangle A(T_i) + \triangle F(T_i)$, and again we incorporate the importance weights described above. The resulting graph T_i' exhibits a smooth structure that satisfies our assumptions from Section 3, and minimizes distances between original and consolidated sample points. Figure 3 (right) shows the result of the consolidation step.

Iterated BSG construction Sparsity and occlusions in the data set may yield suboptimal results from a single pass, as in Figure 6 (left). After constructing and refining a tree's BSG, we may repeat the process and build a new initial BSG using information from the previous iteration. We use a method similar to the mean-shift belief propagation method [Minwoo et al. 2008] to update the squared-distance weights used by Dijkstra's algorithm (see Section 4.1). This allows us to incorporate information from the refined BSG T_i' into the next iteration.

For each edge $e = (u', v')$ in T_i', we find the edge $e_P = (P_{u'}, P_{v'})$ in the complete graph on P whose vertices are closest to those of e. We then scale the weight of e_P by $\frac{e}{\|e\|} \cdot \frac{e_P}{\|e_P\|}$. We proceed as before, applying Dijkstra's algorithm to the reweighted graph to construct another initial BSG. If this spanning tree differs from the tree from the previous iteration, we repeat the consolidation and refinement process on the new initial BSG. In practice, we find that this process converges in 2 or 3 iterations; see Figure 6.

4.3 Inflating BSGs

We have now constructed a BSG for each tree with vertices weighted by importance and optimized to produce long, smooth branches. Next, we reconstruct the tree geometry.

Geometry construction We begin by computing a radius for each vertex in T_i'. Using allometric theory [Tan et al. 2007; Xu et al. 2007], we assign to each vertex a radius proportional to its importance weight. The radii of adjacent vertices are constrained by the ratio expressed in Equation 5; this ensures continuity between the radii of parent and children vertices. Over the whole tree, this is represented by the constraint $\triangle R(T_i')$ in Equation 6.

$$r_{u'} = \left(\frac{c_u}{c_v}\right)^{2.5} r_{v'} \tag{5}$$

To ensure that neither occlusions nor the distribution of points in the input produce artifacts in the computed radii, we constrain the vertex radii with $\triangle D(T_i')$ (Equation 7). Here we insist that the average radius be close to d_{avg}, the average distance between the samples and the edges in T_i'.

$$\triangle R(T_i') = \sum_{u', v' \in T_i'} \left| r_{u'} - \left(\frac{c_u}{c_v}\right)^{2.5} r_{v'} \right|^2 \tag{6}$$

$$\triangle D(T_i') = \Big(\sum_{v' \in T_i'} c_v r_{v'}\Big)^2 - \Big(d_{\text{avg}} \sum_{v' \in T_i'} c_v\Big)^2 \tag{7}$$

As before, we assign radii by minimizing $\triangle R(T_i') + \triangle D(T_i')$, and again the contribution of each vertex is weighted by its importance weight. This results in small radii on spurious branches near the root of the tree; we use this property to cull such vertices.

Edge removal The geometry obtained by this step is represented by a set of generalized cylinders, shown in Figure 7. Once we obtain these cylinders, we remove edges with negligible contribution by computing the intersection of their generalized cylinders with those of other edges on the same parent vertex. If the volume of this intersection exceeds 50% of the volume of the edge's cylinder, we simply collapse the offending edge; see Figure 7 (second and third images).

Similarly, we remove vertices of degree 2 when their adjacent edges share similar orientations and radii. If v is a vertex with parent u and one child w, we collapse the edge (v, w) if the volume of a new edge $(u, v + (v - u))$ intersects more than half of (v, w). In either case, we collapse an edge by inserting a new vertex at the weighted midpoint $(c_u u + c_v v)/(c_u + c_v)$, as in Figure 7 (right). Clustering stops when no further edges can be collapsed; see Figure 8.

Note that when the radii of cylinders are small, as at the crown of the BSG, little overlap occurs even when the edges are dense. However, at the trunk where radii are large, trunk geometry intersects spurious edges in dense regions and only main branches survive clustering.

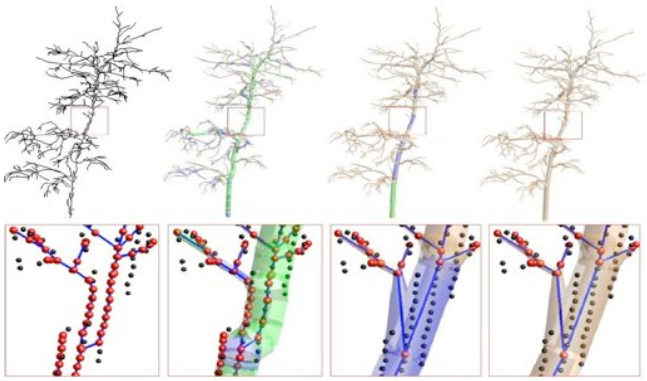


Figure 8: *Snapshots of the edge-removal process. Left to right: the smoothed BSG; generalized cylinders; edge removal; final result. Green cylinders represent candidates for orientation-based edge removal; blue cylinders indicate edges that may be redundant.*

Figure 9: *Robust tree reconstruction under varying point density. From left to right: the resulting BSG using* 100%*,* 50%*, and* 25% *of data points.*

Fine geometry synthesis Having reconstructed the major components of the tree, we now synthesize fine branches, populate the branches with leaves, and add textures. To grow fine branches, we extract and use L-system rules from the BSG as in [Tan et al. 2007]. Leaves are created next to each leaf node ℓ in the BSG. We generate a random number of leaves (between 20 and 50), with positions randomly chosen in a sphere centered on ℓ. At the end, we add textures to our geometry to enhance visual appeal.

5 Results

We employed Optech's Lynx mobile scanning system to acquire the data used in this paper. Scanning was performed atop a moving vehicle at normal driving speed (acquisition rate is 100K points per second); thus, each sample is part of a continuous scan and the example scans are obtained within a span of a few seconds. We perform reconstruction directly on these scans without preprocessing.

The first dataset we processed is a typical street scene with five trees in a row (Figure 1). The trees touch or overlap with each other at their crowns, making them difficult to segment manually. Our method automatically reconstructs the five trees with results that convincingly approximate the real scene, as visible in the photograph. We show the trees in different colors to demonstrate the effectiveness of tree separation during reconstruction. The accuracy of the reconstruction can be better appreciated by viewing the accompanying video where the scene is in motion and the reconstructed branches are overlaid with points.

Figure 10: *Reconstruction results for trees of different types. The tall tree has a large gap at its base due to occlusion caused by a passing vehicle. The bush data has low reconstruction quality since the plant is only half a meter tall, and therefore it is relatively sparsely sampled.*

Our approach is data-dependent yet robust. For particularly poor scans, the accuracy of the reconstruction is compromised; however, we will nevertheless reconstruct the most prominent features that are captured, *i.e.* branch lengths and branching angles. In Figure 5, we show a tree with thin but dense branches. This type of tree is difficult to reconstruct due to the noisiness and ambiguity of the captured point clouds. The left part of the tree is especially poorly scanned as vehicle access is limited to the right side. The reconstructed branches on the left are less than ideal, but still capture the basic properties shown in the photo. The right half, with more samples, is reconstructed with high fidelity. In Figure 9, we show the results of a stress test conducted by randomly removing points from a tree scan. Note that high-level structure is preserved even when most of the points are discarded.

Our method can process trees of different types and sizes. Figure 10 shows our results for challenging inputs. For each tree we present a photo, point set, reconstructed branches, and textured result from two view points. Note that the bush in this figure is rather small (only half a meter tall) and has few samples. Moreover, the visible bias in the scan pattern does not affect the reconstructed branches. Figure 11 shows reconstructed trees of various types within a large scene containing buildings, roads, and several occluders. However, in this experiment the user marks a few rough regions containing the trees, viewing the scene from above, as in Figure 4.

In the final experiment, we compare our reconstruction results to those obtained from the method of [Xu et al. 2007], which is the closest related to ours. Their method is unable to generate high-quality results on most of the datasets shown in this paper. This has two main causes: First, to generate tree skeletons, the method sets a threshold for edge length which can lead to disconnected graphs where points are sparse. Second, a predetermined clustering resolution is used, which cannot adapt to branch density and results in few coarse branches in the crowns of the trees. In Figure 12 we provide a side-by-side comparison of the generated tree skeletons.

Our current unoptimized implementation takes a matter of seconds to reconstruct a single tree and a few minutes for multiple trees from a large scan; see Table 1. The times are recorded on an AMD Athlon 7750 2.71GHz machine with 2GB RAM.

Limitations Our approach is predominantly data-driven. While it is able to properly connect branches that are otherwise disconnected in the data due to small-scale occlusions, it is not designed to reconstruct or synthesize skeletal structures over large regions of missing data. For trees that have very dense crowns, leading to large-scale occlusion of the interior branches, our method is obvi-

Figure 11: *An urban scene with more than* 10 *million points, about* $200,000$ *of which capture* 20 *trees. The trees are of various species, some with low-quality captures. The top two images show a photo and the point scans of the trees, colored by height. Note that the trees are* not *pre-segmented from the rest of the data. Our method automatically extracts the structures of the trees from the rest of the data (bottom) in about 30 minutes.*

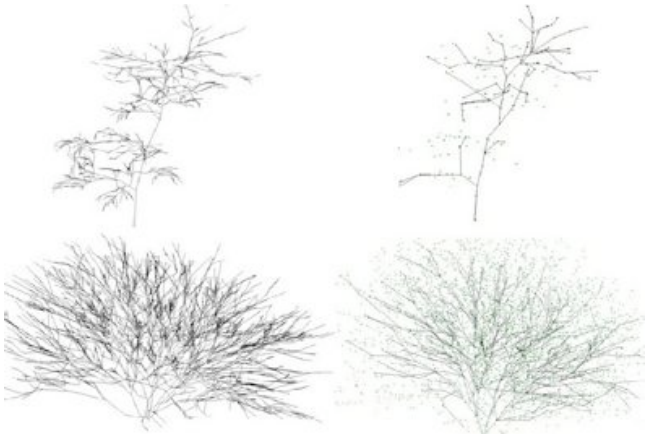


Figure 12: *The result of our method (left) and that of [Xu et al. 2007] (right) on two typical scans. The overlaid points (right) show the differences in the quality of the results and the advantages of our global approach.*

ously unable to provide a faithful reconstruction, as in Figure 13. In these cases, generative methods will be inevitably required.

While our method is able to reconstruct a wide variety of tree *structures* with compelling results, it does not attempt to reconstruct the *appearance* of different types of trees. For example, leaves are placed at random rather than from input data (samples on leaves are particularly susceptible to noise), and textures are chosen to produce credible visual results rather than based on photographs of the actual trees.

The branch structures of natural trees manifest clear self-similarity patterns. These patterns usually include small sets of branching rules (at branch splits) and nearly constant ratios in the lengths

Figure	#points	#Trees	Time
11	10,201,034	20	30 minutes
1	301,287	5	2 minutes
5	65,419	1	30 seconds
10(upper)	14,008	1	7 seconds
10(lower)	917	1	1 seconds

Table 1: *Point cloud size and tree reconstruction times for the different results shown in the figures. The majority of the time for Figure 11 consists of user interaction.*

of parent and child branch chains. Our method makes no direct attempt to preserve or enforce these self-similarities. This is a promising direction for future research.

6 Conclusion and future work

The key feature of our tree reconstruction method is its ability to *automatically* reconstruct skeletal structures representing multiple trees directly from laser scan data; no pre-segmentation is necessary. The approach makes few assumptions on the properties of the captured objects and is widely applicable to trees of various types and sizes, as long as they demonstrate significant branching structures. The quality, versatility, and robustness of our method can be largely attributed to the use of global optimization, in contrast to previous works which rely on local heuristics. While global, the optimization problems we formulate are of the least-squares type and efficient to solve. Experiments on many representative scenes in the paper show that our approach is accurate, scalable, and robust.

One direction for future work is to further generalize our approach so that it can handle a wider range of plants with equally high quality. We are also interested in handling large-scale occlusions, allowing generative methods to reproduce skeletal structures over large missing or occluded regions while maintaining the characteristics of the well-sampled data.

Figure 13: *Trees with significant leaf cover are difficult to reconstruct. In particular, the lack of sufficient sample density on the left side of the tree leads to sparse branching structure.*

It may be possible to evaluate how closely the BSG of a tree matches the self-similarity properties of natural trees by extracting the BSG's pattern (branching rules) and estimating its quality based on its preservation of this pattern. Small variations between the pattern and the values in different parts of the BSG indicate good reconstruction quality. Côté *et al.* [2009] has presented a method for comparing reconstructed trees to scanned and procedurally-synthesized trees via simulated scanning; we may be able to incorporate a similar approach.

Acknowledgments

We thank the anonymous reviewers for their valuable suggestions. This work was supported in part by National Natural Science Foundation of China (NSFC) for Distinguished Young Scholar, National Natural Science Foundation of China (60902104), National High-tech R&D Program of China (2009AA01Z302), CAS Visiting Professorship for Senior International Scientists, CAS Fellowship for Young International Scientists, Shenzhen Science and Technology Foundation (GJ200807210013A),Lynn and William Frankel Center for Computer Sciences and the Tuman Fund, and the Natural Sciences and Engineering Research Council of Canada (No. 611370). Finally, we acknowledge the scanning team of Shenzhen Institute of Advanced Technology (SIAT) for their effort during data acquisition and processing.

References

ANASTACIO, F., SOUSA, M. C., SAMAVATI, F., AND JORGE, J. A. 2006. Modeling plant structures using concept sketches. In *Proceedings of NPAR 2006*, 105–113.

ANONYMOUS. 2010. Analysis, reconstruction and manipulation using arterial snakes. In *Proceedings of SIGGRAPH ASIA 2010*, ??

BUCKSCH, A., AND LINDENBERGH, R. 2008. Campino – a skeletonization method for point cloud processing. *ISPRS journal of photogrammetry and remote sensing 63*, 1, 115–127.

BUCKSCH, A., LINDENBERGH, R., AND MENENTI, M. 2009. Skeltre – robust skeleton extraction from imperfect point clouds. In *Proceedings of Eurographics Workshop on 3D Object Retrieval*, 13–20.

CHEN, X., NEUBERT, B., XU, Y.-Q., DEUSSEN, O., AND KANG, S. B. 2008. Sketch-based tree modeling using markov random field. *ACM Trans. on Graphics 27*, 5, 1–9.

CORNEA, N. D., SILVER, D., AND MIN, P. 2007. Curve-skeleton properties, applications, and algorithms. *IEEE Trans. on Visualization and Computer Graphics 13*, 3, 530–548.

CÔTÉ, J.-F., WIDLOWSKI, J.-L., FOURNIER, R. A., AND VERSTRAETE, M. M. 2009. The structural and radiative consistency of three-dimensional tree reconstructions from terrestrial lidar. *Remote Sensing of Environment 113*, 5, 1067 – 1081.

HONDA, H. 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. *Theoretical Biology 31*, 331–338.

LAZARUS, F., AND VERROUST, A. 1999. Extracting skeletal curves from 3D scattered data. In *Proceedings of IEEE Conf. on Shape Modeling and Applications*, 194–201.

MINWOO, P., YANXI, L., AND ROBERT, C. 2008. Efficient mean shift belief propagation for vision tracking. In *Proceedings of IEEE Conf. on CVPR*, 1–8.

NEUBERT, B., FRANKEN, T., AND DEUSSEN, O. 2007. Approximate image-based tree-modeling using particle flows. *ACM Trans. on Graphics 26*, 3, 1–8.

OKABE, M., OWADA, S., AND IGARASHI, T. 2006. Interactive design of botanical trees using freehand sketches and example-based editing. *Comput. Graph. Forum 24*, 3, 487–496.

PRUSINKIEWICZ, P., AND LINDENMAYER, A. 1990. *The algorithmic beauty of plants*. Springer-Verlag New York, Inc.

PRUSINKIEWICZ, P., MÜNDERMANN, L., KARWOWSKI, R., AND LANE, B. 2001. The use of positional information in the modeling of plants. In *SIGGRAPH 2001: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 289–300.

QUAN, L., TAN, P., ZENG, G., YUAN, L., WANG, J., AND KANG, S. B. 2006. Image-based plant modeling. *ACM Trans. on Graphics 25*, 3, 599–604.

RECHE-MARTINEZ, A., MARTIN, I., AND DRETTAKIS, G. 2004. Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Trans. on Graphics 23*, 3, 720–727.

ROZENBERG, G., AND SALOMAA, A. 1980. *Mathematical Theory of L-Systems*. Academic Press, Inc.

RUNIONS, A., LANE, B., AND PRUSINKIEWICZ, P. 2007. Modeling trees with a space colonization algorithm. In *Proceedings of Eurographics Workshop on Natural Phenomena 2007*, 63–70.

SHLYAKHTER, I., ROZENOER, M., DORSEY, J., AND TELLER, S. 2001. Reconstructing 3d tree models from instrumented photographs. *IEEE Computer Graphics Applicalion 21*, 3, 53–61.

TAGLIASACCHI, A., ZHANG, H., AND COHEN-OR, D. 2009. Curve skeleton extraction from incomplete point cloud. *ACM Trans. on Graphics 28*, 3, 1–9.

TAN, P., ZENG, G., WANG, J., KANG, S. B., AND QUAN, L. 2007. Image-based tree modeling. In *Proceedings of SIGGRAPH 2007*, 87.

TAN, P., FANG, T., XIAO, J., ZHAO, P., AND QUAN, L. 2008. Single image tree modeling. *ACM Trans. on Graphics 27*, 5, 1–7.

WITHER, J., BOUDON, F., CANI, M.-P., AND GODIN, C. 2009. Structure from silhouettes: a new paradigm for fast sketch-based design of trees. *Comput. Graph. Forum 28*, 2, 541–550.

XU, H., GOSSETT, N., AND CHEN, B. 2007. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Trans. on Graphics 26*, 4, 19.