Automatic Procedural Modeling of Tree Structures in Point Clouds Using Wavelets

3DV 2013 Conference, University of Washington, Seattle

Sam Friedman
Hunter College & Graduate Center of CUNY
695 Park Ave, New York NY, 10021
umpteee@yahoo.com

Ioannis Stamos
Hunter College & Graduate Center of CUNY
695 Park Ave, New York NY, 10021
istamos@hunter.cuny.edu

***Abstract*—We present a method for discovering the structure of trees in 3D point clouds by linking wavelets with shape grammars. Given a range scan of a tree we find a grammar that can reproduce that tree, and others like it, with sub-voxel accuracy. The grammar inferred is stochastic, allowing us to generate many permutations of related trees. The method of multi-resolution analysis, employed by the discrete wavelet transform, gives great insight into tree structure. Trees are self-similar and exhibit similar branching patterns at different resolutions. The wavelets make these patterns explicit by decomposing the tree into different levels of detail. The multi-resolution structure of the wavelet transform also allows us to infer an L-System grammar. The productions in the grammar are derived from the progressive levels of refinement in the wavelet transform. Each production maps a vector in the low resolution image to a set of vectors in the higher resolution image. Our method utilizes the Fast Wavelet Transform opening the door to real-time inference of procedural models. The grammar inferred is concise and generative, allowing for compression and graphics applications of our algorithm. We demonstrate novel applications of the grammar for shape completion, scan enhancement and geometry propagation.**

***Keywords*-inverse procedural modeling, tree modeling, wavelets, l-systems**

I. Introduction

Modern range sensors capture dense point clouds, containing millions of points from urban scenes. Points belonging to trees are notoriously difficult to process. Trees are replete with self-occlusions, making it nearly impossible to recover the entire structure from a single vantage point. Structured range scans organize points into a 2D connectivity grid which can be useful for finding surface normals and neighbors. However, in scans of trees this connectivity information is riddled with discontinuities and gives little indication of the actual tree structure.

Nonetheless, trees exhibit a salient and self-similar branching structure. We propose an algorithm to discover this self-similarity. A wavelet transform of the input model gives a multi-resolution view of the tree. In this transformation space, it is possible to recognize correspondences across various levels of detail, and thus elucidate the self-similarity present in the tree. The correspondences across different scales make up the rules of a shape grammar. The grammar gives a high-level description for the branching structure of the tree, but does not encode the foliage.

The inferred grammar can be used for compressing, propagating, symmetrizing, or enhancing the input model. The fast wavelet algorithm is extremely efficient. Grammars can be inferred almost instantaneously. We imagine a host of applications that exploit the high-level structural encoding provided by the shape grammars we infer. For example, natural-seeming virtual worlds of arbitrary size can be automatically generated from a single scan. Low-resolution scans can be enriched in real-time giving immersive and compelling pictures.

II. Related Work

Developing procedural models for vegetation has been the subject of research for many years [1]. Lindenmeyer systems, or L-systems, have tremendous expressive power for concisely describing many plant forms that appear in nature. L-Systems can be notoriously difficult to control. To give the user more control over the procedural methods the work of [2] uses the Metropolis algorithm to precisely regulate grammars by sampling from the production space. An interactive, tablet-based approach is presented in [3], where users can sketch outlines and detailed trees are generated following the outline by simulating competition for space and light amongst the branches. In [4] tree models are derived from video sequences. In [5] a library of L-Systems is combined with tree detection and classification techniques to find and compress tree structures. Traditionally, deriving an L-system for a given model requires both botanical and grammatical expertise. In recent years, however, efforts have been made to automatically infer a grammar from a given model. For example, the work of [6] uses the concept of docking sites and model symmetry to generate a context-free grammar for 3D models. This approach requires precise symmetry in the input. Using a 4D clustering of pairwise similarities, [7] automatically generate context-free L-systems from 2D images.

For scans from a laser sensor, a number of tree extraction methods have been presented in the literature. In the work of [8] all points are projected to the ground plane. Detecting trees is reduced to clustering in the ground plane. Trees rise vertically from their roots and when the point cloud is projected down to the ground plane the tree roots will be the centers of each cluster in this plane. An online method

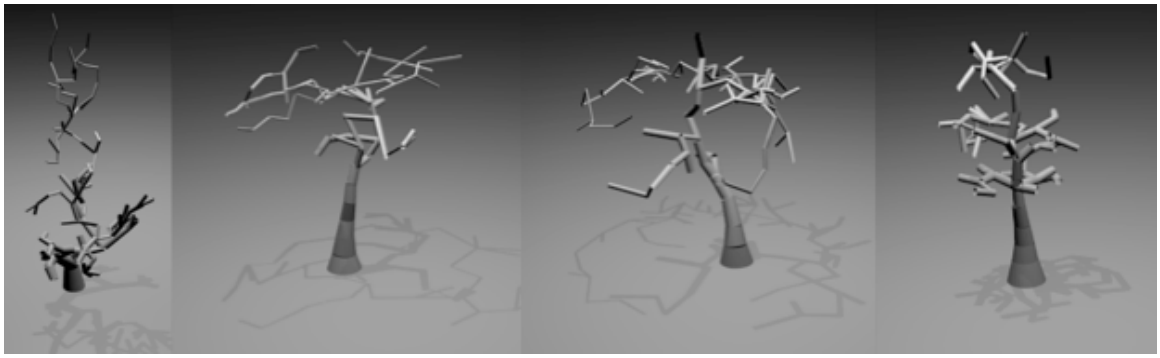

Figure 1. A variety of tree forms analyzed using the algorithm presented in this paper. From left to right are Pine, Delonix, Laegerstroemia and Terminalia tree species. See Section III-E for how we construct these trees from the input point clouds.

for classifying vegetation points using sequential algorithms and a hidden Markov model formulation is presented in [9].

Much attention has been focussed on detecting symmetry for reconstruction, registration, and compression applications. For example, in [10] clustering in a transformation space constructed from local features like curvature, allows the authors to discover both precise and partial symmetries. Taking a graph-based approach, [11], make graphs from the k-nearest neighbors of selected key points with low slippage features, and then search for graph similarities to find symmetries. Extending upon the recognition of symmetry is the search for self-similarity. Roughly speaking, self-similarity is resemblance of a model with itself at different scales or in different places [12]. Self-similarity is both more general than symmetry, and more common in the natural world. A recent paper by [13] used self-similarity features for 3D correspondence and registration of dense point clouds.

Instead of the local features of [10] or the graph-similarity employed by [11], in our approach, self-similarity is discovered by using the wavelet transform [14]. A filter bank system of high and low pass filters decomposes the input into various levels of detail. For a general reference on super resolution see [15]. The wavelet transform can be seen as tiling the time-frequency plane. Because downsampling is performed before each iteration of filtering the tiling corresponds to repetitive division by two. Each tile stretches from a power of two to a subsequent power of two. For this reason, we refer to these tiles as dyads.

The main contributions of our paper with respect to previous work include

(a) We infer both voxel and L-System productions by connecting grammar rules and levels of detail in the wavelet transform .

(b) The only input needed for our system is a single range scan of a tree.

(c) The inferred high-level grammar is employed in compression, transmission, enhancement, and symmetrization applications.

(d) All of the above is achieved without requiring user interaction, expert guidance, or training data.

III. Algorithm

Given a point cloud of a tree extracted using the methods presented in [8], [9], or manually, our algorithm begins by quantizing the point cloud. The wavelet transform of the voxel set is computed. Compression is achieved by removing small wavelet coefficients. The multi-scale structure of wavelet transform decomposes the input into various levels of detail. Mapping between different levels of detail in the wavelet transform yields shape grammars. We derive a simple shape grammar reminiscent of an octree, which we are calling a voxel grammar. An L-system grammar is also derived by running the minimum spanning tree algorithm in each wavelet dyad. The inferred grammars are applied for geometry propagation, enhancement and symmetrization. The following sections explain the algorithm in detail and Figure 2 depicts the pipeline graphically.

A. Quantization

To apply the wavelet transform we must quantize the point cloud into discrete voxels. We could quantize the points using the frame of reference of the scanner, essentially just discretizing the x,y, and z coordinates as they were captured. The problem with this approach is that the direction of the x and y axes will be dependent on the relative placement of the scanner and the tree being scanned. Because the scanner returns only a partial view, the same tree could yield vastly different quantizations when seen from different angles. So instead of quantizing the raw input points, we first transform them. The z-axis is the known vertical direction. The root is determined to be the point with the lowest z value. We then rotate the points, so that the y-axis is the projection of the vector from the scanner to the root, on the plane perpendicular to the z-axis passing through the root. Now the x direction has meaning: it is the horizontal span of the most complete view of the tree. Similarly, the y-axis is now perpendicular to the vertical axis, and aligned with the vector between the scanner and the tree root. Now different

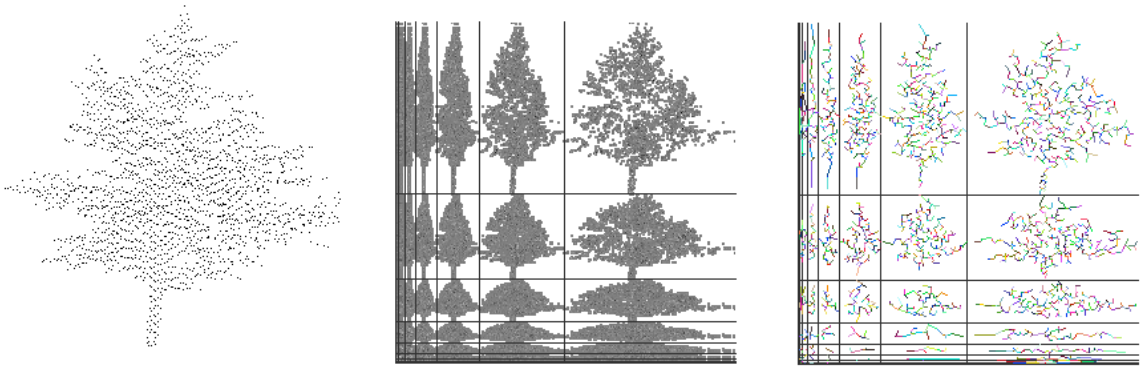

Figure 2. The algorithm begins by taking the wavelet transform on the voxelized point cloud (left image is a 2D projection of the input point cloud) yielding the middle image, shown in 2D here. Then, Prim's minimum spanning tree algorithm is executed in each wavelet dyad. The vectors that make up each tree are mapped to the subsequent dyads along the diagonal, thus automatically generating an L-System for the input tree.

y-coordinates represent different distances from the scanner. This allows us to easily extract 2D projections which respect the natural shape of the tree.

The size of the voxel grid must be selected with care. Obviously, larger grid sizes lead to long computation times. However, if the voxel grid is too small then artifacts of the grid will be present in the trees. These artifacts will manifest as 90 or 45 degree angles in the tree branches. An example of a poorly quantized tree is shown if Figure 3. In our experiments a grid of 64^3 was usually sufficient, but for larger inputs we used a 128^3 grid.

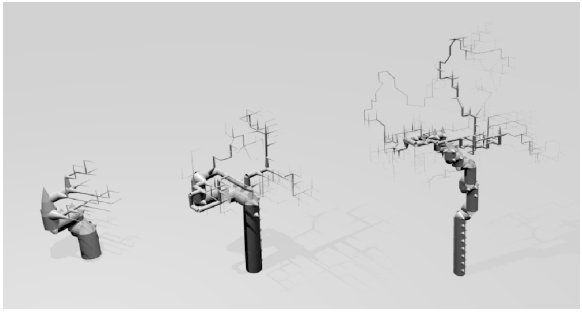

Figure 3. The quantization leads to right angle branching in the trees grown in the wavelet dyads. See Section III-E for full explanation.

B. Wavelet Transform

We apply the wavelet transform on the input. We compute the 3D transform on the whole voxel set. For visualization purposes we also compute the 2D wavelet transform on the projection to the plane perpendicular to the y-axis. 2D and 3D images can be transformed into standard or nonstandard wavelet decompositions, see page 5 in [16]. The nonstandard decomposition is quicker to compute, and in the standard transform the dyads off the diagonal are distorted. We use the standard transform in this paper.

C. Voxel Grammar

The wavelet transform decomposes the input into levels of detail. When the input is a voxel grid, each voxel in one level of detail corresponds to eight voxels in the subsequent higher resolution level. The dyadic tessellation of the time-frequency plane makes the relationships between different levels of detail explicit. The organization is similar to an oct-tree.

We construct a voxel grammar by mapping a voxel in one dyad to the eight voxels in the subsequent dyad. We iterate across each level of detail generating rules that map voxels from coarse details to finer ones. Each rule is a mapping from one voxel to eight voxels. See Figure 4 for a graphical depiction of this process. If we have many rules for a given voxel value we choose one of the applicable rules stochastically. Applying the voxel grammar to the finest level of detail gives a synthetic set of wavelets. These wavelets can be used for a super-resolution reconstruction of the input.

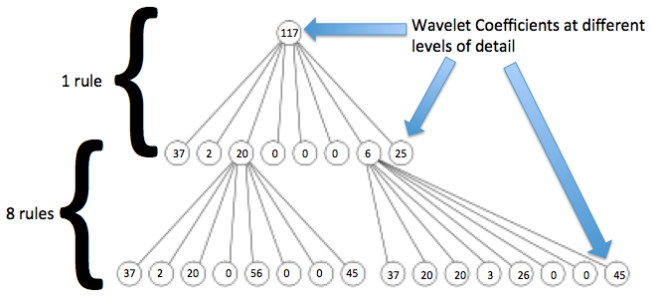

Figure 4. Graphical explanation of how a voxel in one wavelet dyad is mapped to 8 voxels in the subsequent wavelet dyad, forming a rule in the voxel grammar.

D. Super-resolution Reconstruction

We derive a super-resolution version of the input by augmenting the wavelet transform with synthetic wavelets

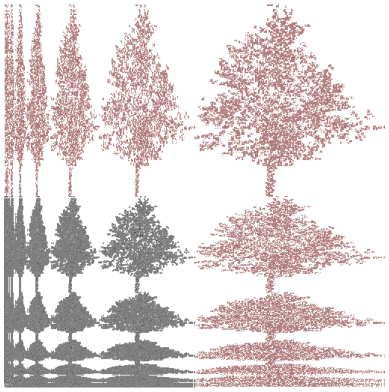

Figure 5. Grey points are the 2D wavelet transform of the input and synthetic wavelets generated using the voxel grammar, presented in Section III-C, are shown in maroon. For visualization purposes we are showing the 2D wavelet transform of the voxels.

derived by the voxel grammar. A wavelet transform of tr̂ee along with the synthetic wavelets generated by the voxel grammar is shown in Figure 5. The reverse wavelet transform on the augmented set of wavelets yields an image at twice the resolution of the original. Unlike a simple up-sampling the super-resolution image is detailed and smooth. Since trees are self-similar and the high-level details are derived from low-level ones the super-resolution image gives an accurate version of the real-world tree, quite plausibly it is more accurate than the input. Artifacts from the scanner like the grid pattern of laser samples only exist at one level of detail and are mostly ignored by the voxel grammar. In this way, the super-resolution image lessens these distractions and leaves us with a richer, fuller picture, as shown in Figures 6 and 7.

Figure 6. Synthetic wavelet super-resolution compared with a simple up-sampling. For visualization purposes we are showing the 2D projection perpendicular to the y-axis of the full 3D point clouds. The tree on the left is shown at super-resolution, and on the right the same tree is shown after up-sampling the voxels.

E. Minimum Spanning Tree in each Dyad

A voxel grammar is a useful abstraction, however, an L-System is a more expressive procedural framework for describing trees. L-Systems have been used to model many

Figure 7. Another example of synthetic wavelet super-resolution compared with a simple up-sampling.

species of vegetation, as well as algae, and other organisms [1]. Normally, L-systems are derived by experts familiar with both botany and shape grammars. We provide a method to automatically generate an L-System given only a 3D scan of the tree to be described. The voxel grammar presented in Section III-C mapped one voxel to eight voxels, but to derive the L-System we will map a vector to a set of vectors. To find these vectors we run Prim's algorithm to compute the minimum spanning tree (MST) in each dyad of the wavelet transform computed in Section III-B. Before the MST can be computed the wavelet space must be transformed into a weighted graph. To accomplish this, every voxel in the transform whose value is above some threshold, in our experiments we use 0.3, becomes a node in the graph. The weights are set to the Euclidean distances between the voxel grid coordinates above the threshold. Each tree is rooted at the lowest significant node in the dyad. The result of running the MST algorithm on these weighted graphs can be seen in Figure 8. We render these trees in 3D and set the cross-sectional area of the branches according to Leonardo Da Vinci's famous observation that the thickness of a branch is the sum of the thicknesses of the branching branches.

F. L-System Rules from Wavelet Progression

Armed with the spanning trees of each dyad it is now feasible to derive a proper L-System grammar using methods similar to those described in Section III-C. As before, each spanning tree is mapped to the neighboring dyad along the diagonal in the time frequency plane. Now, however, instead of associating a single voxel with a set of voxels, we associate a vector with a set of vectors. To map between vectors in neighboring dyads we must find the corresponding areas of each dyad. Figure 8 shows arrows between the dyads indicating the mappings. Specifically, given a vector from points α to β in the spanning tree of the j^{th} dyad we associate the vectors whose start or end points fall within the rectangle bounded by points r_1 and r_2. Where

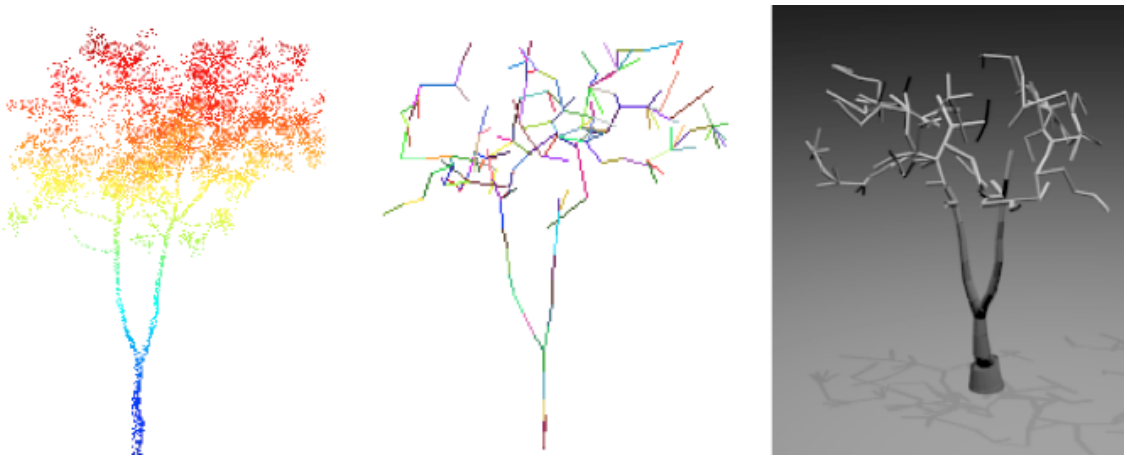

Figure 9. At left is the point cloud input to our algorithm, the hue is given by the height (z-value) of the points. The middle image shows the vectors of Minimum Spanning Tree (MST) each with a random color. The image on the right shows the 3D mesh generated from the MST.

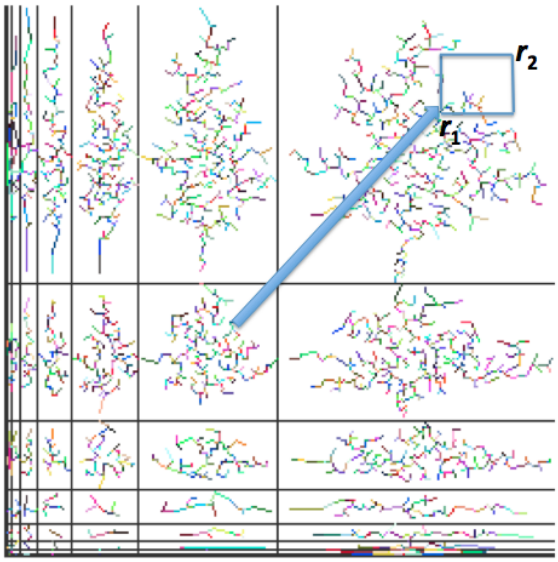

Figure 8. The results of running the minimum spanning tree algorithm in each wavelet dyad. For visualization purposes we are showing the results on the 2D projection of the point cloud, though the algorithm operates in 3D. The arrows show the mapping between vectors in subsequent wavelet dyads. The rectangle form by r_1 and r_2 is the area in from which the right hand side of the grammar production will be inferred. See section III-F for details.

$$d_{offset} = 2^{j+1} - 2^{j} \quad (1)$$

$$r_1 = (\min(\alpha_x, \beta_x) + d_{offset}, \min(\alpha_y, \beta_y) + d_{offset}) \quad (2)$$

$$r_2 = (\max(\alpha_x, \beta_x) + d_{offset}, \max(\alpha_y, \beta_y) + d_{offset}) \quad (3)$$

Two examples of rules obtained in this manner are shown in Figure 10. Each rule maps a single predecessor vector to one or more successor vectors.

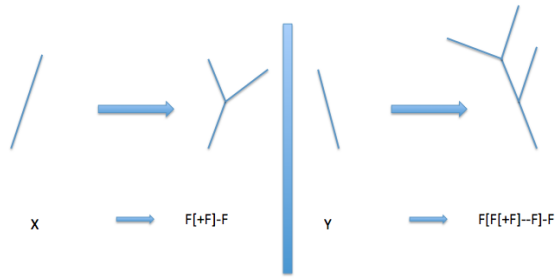

Figure 10. Visual representation of 2 L-rules derived from mapping neighboring trees in the wavelet dyads. The left columns show the predecessor vector and the right column shows the successor vector(s). Below the visual representations are the L-system productions. See Section III-F.

G. Rule Space

All the productions together with an axiom constitute the grammar. For most applications a dyad from the input can be chosen as an axiom. To have the most faithful productions one should choose the largest dyad. When more compression or variety is desired it makes sense to choose smaller dyad. When the grammar will be applied for geometry propagation then the dyad from the input should first be perturbed as explained in Section IV-C. To apply the grammar one must find a rule to match each vector in the axiom or the current production. There is no guarantee that each vector in the current production will match a predecessor vector in the rule set perfectly. To get the next production we must first search in the rule space for the most appropriate rule. The four dimensions of the rule space are given by the heading of the parent vector and its length. We use the standard Euclidean distance in this 4D space to find the best fitting rule from the inferred L-System. See Figure 10. The rule is applied by filling a synthetic wavelet with the right hand side vectors from the best fitting L-System production. We

Table I
COMPRESSION RESULTS ON DIFFERENT TREE SPECIES (WE USED INPUT RANGE SCANS FROM [5]). THE LOSSLESS COMPRESSION RATIO IS CALCULATED AS THE NUMBER WAVELET COEFFICIENTS EQUAL TO ZERO OUT OF THE TOTAL NUMBER OF COEFFICIENTS. THE LOSSLESS COMPRESSION RATE IS THE NUMBER OF WAVELET COEFFICIENTS BELOW A THRESHOLD γ. IN OUR EXPERIMENTS WE SET $\gamma = 0.2$, AS WE FOUND THIS THRESHOLD YIELDED SIGNIFICANT COMPRESSION WITHOUT INTRODUCING MANY ARTIFACTS.

Species	Lossless	Lossy
Bischofia Polycarpa	64%	94%
Delonix	68%	96%
Ficus Virens	68%	92%
Lagerstroemia	69%	95%
Mahogany	73%	95%
Palm	80%	95%
Pine	81%	96%
Terminalia	75%	95%
Willow	69%	95%

then apply the inverse wavelet transform and run the MST algorithm in the super resolution voxels. The process can be iterated to enhance trees to arbitrary size as described in Section IV-B.

IV. APPLICATIONS

A. Compression

Wavelets are well known for their ability to compress images [17]. Both lossy and lossless compression were employed on our range images. When using lossy compression the user may set a threshold, γ to determine the quality or the amount of compression desired. We set $\gamma = 0.2$, which yields significant compression, while not introducing serious artifacts into the reconstruction. Lossless compression rates on the tree images ranged from 40% - 60% and lossy compression rates ranges from 90% - 96%. All this compression occurs before we further process the wavelet transform to infer an L-System. The inferred L-System is itself another layer of data compression. Unless the axiom of the L-System is the entire input image, the L-System will represent a lossy compression. But not just lossy. The motivation of this paper was the fact that a grammar is high-level representation of a structure. The grammar is concise which gives us compression, but it is also generative. This generative capacity of the L-System gives a number of interesting applications in graphics. Figure 11 shows three successive generations of the same tree.

B. Enhancement

L-System productions can be applied iteratively each time yielding a more intricately detailed structure. In this way even a low-resolution scan can yield an impressive high-resolution image. This enhancement capability can be useful for mobile or networked applications with limited bandwidth. A server needs only to deliver the compressed L-System rule set and its axiom, and the client can apply the productions to give as many levels of detail as desired. It must be noted, that after many iterations the structure will no longer accurately reflect the real object. For example, while the tiny branches of a tree often present similar patterns to the larger branches, the plant cells that make up the branches are structured according to a different pattern.

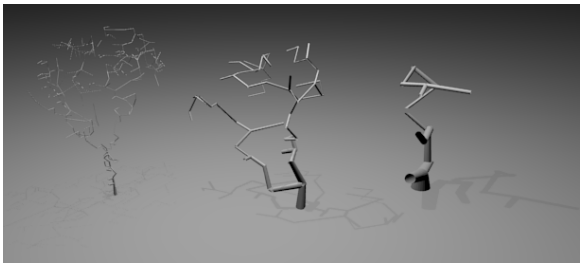

Figure 11. Three successive generations of a tree. The middle tree is created by applying the grammar to the rightmost tree, and the leftmost tree is created by applying the grammar to the middle one.

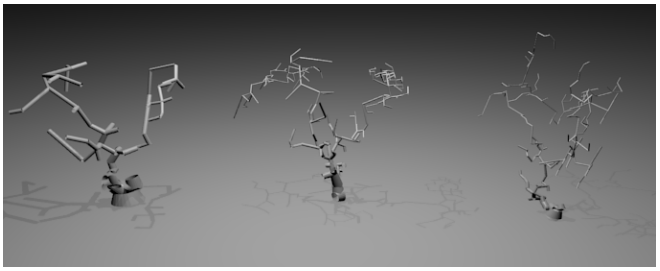

Figure 12. The rightmost tree is an enhanced version of the middle tree, computed using the L-System described in Section III-F. The L-System productions were derived by mapping between the leftmost tree and the middle tree. The process can be iterated generating arbitrarily large trees from relatively small input scans.

C. Propagation

Another advantage of the grammar representation is the ability to achieve similar yet distinct models. Trees can be tedious to construct by hand. Nonetheless, trees often occur in forests, or lining streets where the same species is repeated many times. Simply copying one tree model over and over again gives an artificial appearance. By staying in the abstract space of the shape grammar and varying the productions or the way they are applied, many different trees that all seem similar can be rendered with little effort. To ensure variety random perturbations can be applied to the starting axiom so that a different set of productions will be chosen. Different trees generated using the same L-System are shown in Figures 13 and 14.

D. Symmetrization

Each range scan presents only a single view of a tree. The sampling of points is biased towards those facing the scanner. Symmetrizing the tree, by reflecting about the plane perpendicular to the scanner and centered at the tree root,

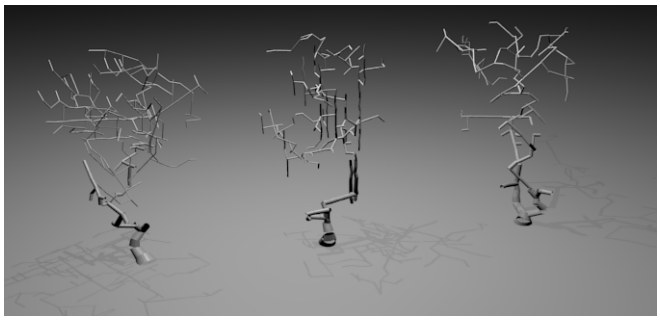

Figure 13. Three different trees generated using the same L-System. Geometry can be propagated by applying the same L-System on slightly altered versions of the axiom.

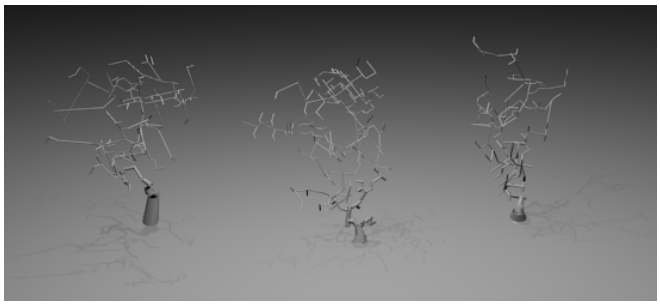

Figure 14. Another example of geometry propagation.

mitigates this bias. Specifically, if r is the root of the tree and s is the scanner position, This symmetrization technique gives a tree that appears full and complete. Unfortunately, when viewed from the plane of reflection this approach gives the rigid symmetry seen in the rightmost image of Figure 15 and 17. However, when symmetrization is performed as a pre-processing step to our algorithm, the resulting trees do not exhibit the oversampling bias in the direction of the scanner or the rigid symmetry of the reflected point clouds. For example see the three views of the tree in Figure 18.

E. Evaluation

The wavelet transform is efficient to compute, operating in $O(n)$ time. In our current implementation the most time-consuming step is the generation of the minimum spanning trees. Optimizing by integrating an approximate nearest neighbors algorithm would ameliorate this bottleneck. As it stands the algorithm runs in a few seconds on voxel grids of size 64^3, and about 30 seconds on grids of size 128^3.

The trees constructed by our algorithm are concise and generative, while retaining the overall character of the trees from which they were derived. For example, see Figure 19, to see a tree generated by an inferred grammar next to the tree from which the grammar was derived.

V. Conclusion and Future Work

The method of multi resolution analysis presented here is especially well-suited to point clouds of trees because their

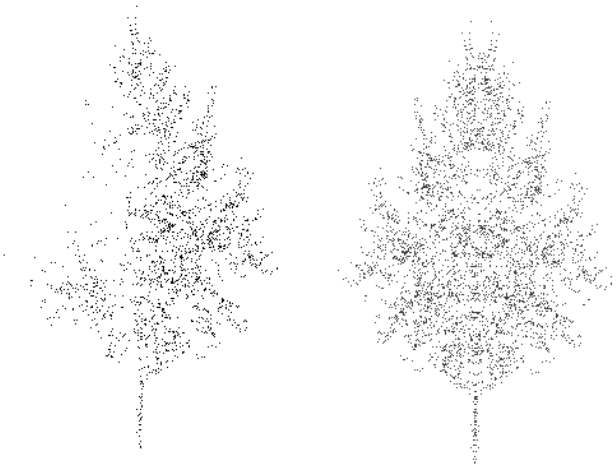

Figure 15. A tree point cloud with sampling biased towards scanner, and the symmetrized point clouds. When viewed from the plane of reflection the artifice of this naïve symmetrization technique is clear, as in the left image.

Figure 16. Another example of the naïve symmetrization technique.

hierarchical branching results in similar structures across many scales of resolution. However, trees are not the only such structures. Many terrains also exhibit self-similarity across many scales of resolution. The algorithm given above could be used to encode, enhance, compress or recreate point clouds from terrains captured by an aerial survey.

To avoid the issues with quantization discussed in Section III-A, the quantization step could be replaced with a surface reconstruction on the input point clouds. The wavelet transform could then be computed on the surface, rather than the voxels, using the method presented in [18]. The voxel grammar would become a vertex grammar, but the L-System derivation and the algorithm applications could remain practically unchanged.

Acknowledgements

We would like to thank Baoquan Chen and the Visual Computing Research Center for supplying many of the tree point clouds analyzed in this paper. This work has been supported in part by the following NSF grants: IIS-0915971, CCF-0916452 and MRI CNS-0821384.

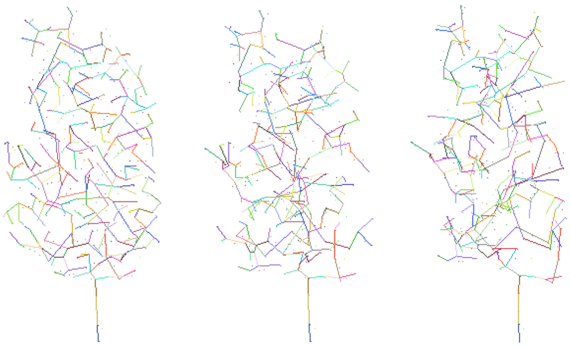

Figure 17. Three views of a tree before symmetrization. Notice the uneven distribution of branches when viewed from different angles. The vectors shown here were obtained by running the Minimum Spanning Tree algorithm as described in Section III-E.

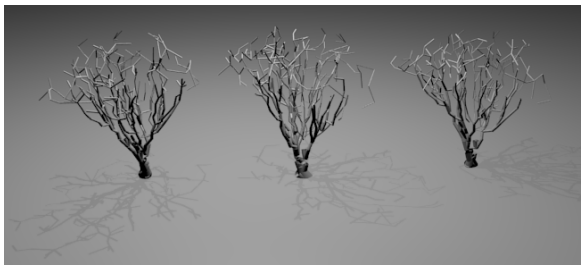

Figure 18. Three different views of the same tree. There is no viewing angle where the reflection will be apparent, because here the symmetrization is performed as a pre-processing step. The resulting trees do not have the rigid and unnatural look of the symmetrized point clouds shown in Figures 15 and 16

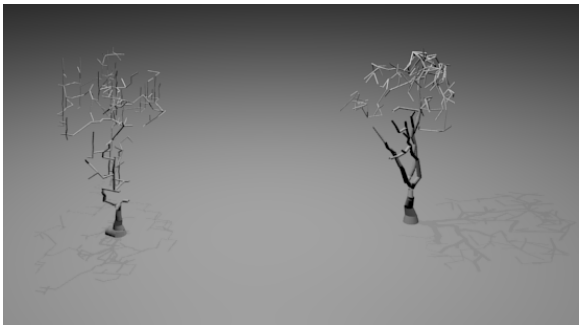

Figure 19. Visual comparison of a tree generated using our algorithm at left and the tree from which the grammar was inferred. The tree on the right is the MST of the input point cloud and the tree at the left is the reconstruction created from the grammar.

REFERENCES

[1] P. Prusinkiewicz and A. Lindenmayer, "The algorithmic beauty of plants (the virtual laboratory)," 1991.

[2] J. Talton, Y. Lou, S. Lesser, J. Duke, R. Měch, and V. Koltun, "Metropolis procedural modeling," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 2, p. 11, 2011.

[3] S. Longay, A. Runions, F. Boudon, and P. Prusinkiewicz, "Treesketch: interactive procedural modeling of trees on a tablet," in *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling*. Eurographics Association, 2012, pp. 107–120.

[4] C. Li, O. Deussen, Y.-Z. Song, P. Willis, and P. Hall, "Modeling and generating moving trees from video," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 6, p. 127, 2011.

[5] Y. Livnyl, S. Pirk, Z. Chengl, F. Yanl, O. Deussen, D. Cohen-Or, and B. Chen, "Texture-lobes for tree modelling," 2011.

[6] M. Bokeloh, M. Wand, and H.-P. Seidel, "A connection between partial symmetry and inverse procedural modeling," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 104:1–104:10, Jul. 2010. [Online]. Available: http://doi.acm.org/10.1145/1778765.1778841

[7] O. Šťava, B. Beneš, R. Měch, D. Aliaga, and P. Krištof, "Inverse procedural modeling by automatic generation of l-systems," in *Computer Graphics Forum*, vol. 29, no. 2. Wiley Online Library, 2010, pp. 665–674.

[8] Y. Livny, F. Yan, M. Olson, B. Chen, H. Zhang, and J. El-Sana, "Automatic reconstruction of tree skeletal structures from point clouds," in *ACM Transactions on Graphics (TOG)*, vol. 29, no. 6. ACM, 2010, p. 151.

[9] O. Hadjiliadis and I. Stamos, "Sequential classification in point clouds of urban scenes," in *Proc. 3DPVT*, 2010.

[10] N. Mitra, L. Guibas, and M. Pauly, "Partial and approximate symmetry detection for 3d geometry," *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3, pp. 560–568, 2006.

[11] A. Berner, M. Bokeloh, M. Wand, A. Schilling, and H. Seidel, "A graph-based approach to symmetry detection," in *Symposium on Volume and Point-Based Graphics*, 2008, pp. 1–8.

[12] B. Mandelbrot, *The fractal geometry of nature*. Times Books, 1982.

[13] J. Huang and S. You, "Point cloud matching based on 3D self-similarity," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*. IEEE, 2012, pp. 41–48.

[14] S. Mallat, *A wavelet tour of signal processing*. Academic press, 1999.

[15] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 349–356.

[16] E. Stollnitz, A. DeRose, and D. Salesin, "Wavelets for computer graphics: a primer. 1," *Computer Graphics and Applications, IEEE*, vol. 15, no. 3, pp. 76–84, 1995.

[17] I. Daubechies *et al.*, *Ten lectures on wavelets*. SIAM, 1992, vol. 61.

[18] M. H. Gross, O. G. Staadt, and R. Gatti, "Efficient triangular surface approximations using wavelets and quadtree data structures," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 2, no. 2, pp. 130–143, 1996.