Discovering Regularity in Point Clouds of Urban Scenes

Sam Friedman

Graduate Center, City University of New York

Recommended Citation
Friedman, Sam, "Discovering Regularity in Point Clouds of Urban Scenes" (2014). CUNY Academic Works.
http://academicworks.cuny.edu/gc_etds/40

This Dissertation is brought to you by CUNY Academic Works. It has been accepted for inclusion in All Graduate Works by Year: Dissertations, Theses, and Capstone Projects by an authorized administrator of CUNY Academic Works. For more information, please contact deposit@gc.cuny.edu.
Discovering Regularity in Point Clouds of Urban Scenes

by

Sam Friedman

Dissertation Submitted to the Graduate Faculty in Computer Science
in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy,
The City University of New York

2014
This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

Ioannis Stamos  
Chair of Examining Committee

Robert Haralick  
Executive Officer

Committee Members
Professor Zhigang Zhu, City College of New York  
Professor George Wolberg, City College of New York  
Professor Philippos Mordohai, Stevens Institute of Technology
Discovering Regularity in Point Clouds of Urban Scenes

By

Sam Friedman

Adviser: Professor Ioannis Stamos

Abstract

Despite the apparent chaos of the urban environment, cities are actually replete with regularity. From the grid of streets laid out over the earth, to the lattice of windows thrown up into the sky, periodic regularity abounds in the urban scene. Just as salient, though less uniform, are the self-similar branching patterns of trees and vegetation that line streets and fill parks. We propose novel methods for discovering these regularities in 3D range scans acquired by a time-of-flight laser sensor. The applications of this regularity information are broad, and we present two original algorithms. The first exploits the efficiency of the Fourier transform for the real-time detection of periodicity in building façades. Periodic regularity is discovered online by doing a plane sweep across the scene and analyzing the frequency space of each column in the sweep. The simplicity and online nature of this algorithm allow it to be embedded in scanner hardware, making periodicity detection a built-in feature of future 3D cameras. We demonstrate the usefulness of periodicity in view registration, compression, segmentation, and façade reconstruction. The second algorithm leverages the hierarchical decomposition and locality in space of the wavelet transform to find stochastic parameters for procedural models that succinctly describe vegetation. These procedural models facilitate the generation of virtual worlds for architecture, gaming, and augmented reality. The self-similarity of vegetation can be inferred using multi-resolution analysis to discover the underlying branching patterns. We present a unified framework of these tools, enabling the modeling, transmission, and compression of high-resolution, accurate, and immersive 3D images.
Acknowledgements

I would like to thank my advisor Ioannis Stamos, for his kind and constant help, tremendous understanding, and vast knowledge. He has been a wonderful guide for me throughout this process. I am especially grateful for the opportunities to travel to China, Rhode Island, and Seattle, for introducing me to computer vision, and of course, for all the macaroons.

My committee members, George Wolberg, Zhigang Zhu, and Philippos Mordohai, thanks so much for agreeing to serve and for the helpful comments and suggestions. The computer vision lab at Hunter College, Agis Mesolongitis, Tom Flynn, Juan Liu, and Allan Zelener, thanks for the mind-expanding lab meetings and for the input on the research project. Thanks to Baoquan Chen, and the Visual Computing Research Center at the Shenzhen Institute of Advance Technology, for generously sharing their vegetation scans and L-System library.

Endlessly thankful for my family– my dear parents: William and Lucy, and my bounty of siblings: Tanya, Denise, Ezra, Anna, Gideon, and Rachael, and my two little nieces Zora and Mira, so much love for you all– Happy Solstice and let’s spit water in Sunken Forest soon, ok? Galaxia, your relentless intelligence inspires me, thanks for electron orbitals, the mylenated transmissions, doing biz with the cuz, etc. Kate- miso milo- thanks for looking at these squiggly lines with me. Don’t tell Wavelette, but I ate all your bacon. Moontoes, thanks for seeing the rocks in Niagara, whorls and starfish in Mazatlan, and the buddha light– we so Zen in the gate– at Yellow Mountain. Help me teach the robots Tai Chi? Leastly, Zeno, thanks for showing me how much worse things could be, and for always being the evil clown on my shoulder.
Contents

1 Introduction .................................................. 1
  1.1 Data Acquisition ........................................... 2
  1.2 Problem Statement ......................................... 3
  1.3 Vision and Graphics Applications ......................... 4

2 Background and Related Work ............................... 7
  2.1 Symmetry Detection ......................................... 7
  2.2 Fourier Analysis ............................................ 13
  2.3 Wavelet Analysis ............................................ 20
  2.4 Procedural Modeling ........................................ 23

3 Detection of Periodic Structures ......................... 27
  3.1 Online Symmetry Detection ................................ 28
    3.1.1 Extraction of Repeated Features Through Fourier Analysis . . 29
    3.1.2 Finding The Major Planes ................................ 30
    3.1.3 The Column Function .................................... 31
    3.1.4 Frequency Space ......................................... 33
    3.1.5 Fenestration Centroids from Square Waves .............. 34
    3.1.6 Aggregation .............................................. 36
    3.1.7 Evaluating Regularity Detection ......................... 36
  3.2 Registration ............................................... 38
    3.2.1 Orthonormal Bases of Aggregated Columns .............. 39
    3.2.2 Computing Corresponding Pairs .......................... 42
  3.3 Compression ............................................... 45
    3.3.1 Segmentation and the Representative Region .......... 45
    3.3.2 Comparison of façade Segments ......................... 49
    3.3.3 Replacing Regions with Translations of the Representative .... 51
  3.4 Reconstruction ............................................ 53

4 Detection of Self-Similar Structures .................... 64
  4.1 Dendromorphology – the Growth and Form of Trees ........ 65
    4.1.1 Tree Parameters .......................................... 68
    4.1.2 Foliage .................................................. 73
  4.2 Automatic Inference of Procedural Models for Trees ....... 74
    4.2.1 Vegetation Extraction .................................... 75
    4.2.2 Symmetrization ........................................... 76
    4.2.3 Parameter Estimation from Hierarchical Decomposition ... 77
    4.2.4 Evaluation ............................................... 79
  4.3 Applications of Self Similarity Detection ................ 79
    4.3.1 Enhancement .............................................. 80
    4.3.2 Propagation .............................................. 80
    4.3.3 Compression .............................................. 84

5 Conclusion ................................................ 85

References .................................................... 87
List of Figures

1. A point cloud projected to the ground plane
2. A texture mapped range scan
3. Windows and basis vectors
4. Fourier Transform of a sinusoid
5. 2D Discrete Fourier Transform
6. Fourier Transform extracts major directions
7. Inverse relationship of DFT and a signal
8. Low pass filter in frequency space
9. Fourier Transform for text orientation
10. Fourier Transform is global
11. Frequency space reveals periodic structures
12. Wavelet Transform of four signals
13. Wavelet Transform four 2D signals
14. L-Systems with an angle parameter
15. Variations on the same L-System
16. Changing the angle parameter on the same L-System
17. Simple right triangles eludes L-System description
18. Varieties of column functions
19. The column function
20. Column functions before and after interpolation
21. Curvature as a local measure
22. The Fourier Transform of a periodic column function
23. The square wave fit to a column function.
24. Periodicity detection accuracy
25. Two scan views registered together.
26. Two synthetic scan views registered together.
27. Another angle of two scan views registered together.
28. Two scan views registered together.
29. Before and after ICP
30. Compression results on a large areas
31. Compression of buildings
32. Compression results
33. Chebyshev distance
34. Framework for scan compression
35. Compression threshold parameter
36. Compression results
37. Façade reconstruction
38. Occlusions are common issue in range scanning.
39. Different floors colored in alternating stripes of gray.
40. Occlusions caused by vegetation are filled.
41. Scan sampling density is maintained.
42. Façade reconstructed with the method of 3.4.
43. Façade reconstructed with the method of 3.4.
44. Synthetic shadows are generated and filled
45. Mystery planes
46. A coarse mesh
Several large meshes derived from point clouds using our method.
Missing points occluded by vegetation are filled.
Angiosperms and Gymnosperms.
Phototropism.
Apical control.
Conifers.
Palms.
Weeping tree forms.
Trees with different apical angle variances, $\alpha_{av}$
Trees with different branch angle means, $\beta_{\mu}$
Alternating phyllotaxis.
Foliage examples.
Textured branches.
Tree sampling is biased towards the scanner.
Minimum spanning tree.
Wavelet dyads.
Voxel grammar.
Four grammar generations of a tree.
Tree enhancement.
Propagation.
Interactive procedural modeler.
Propagation of Angiosperms.
Iterations on a palm like form.
List of Tables

1  Comparison of symmetry detecting algorithms .................. 12
2  The precision and recall of the regularity detection .......... 37
3  Algorithm runtime ............................................. 38
4  Compression results on different tree species ................. 84
1 Introduction

The central argument of this thesis is that the urban scene, which appears gritty and chaotic, actually abounds with mathematically elegant structures. We present efficient algorithms for discovering some of these structures in buildings and vegetation. This high-level information facilitates solutions to a host of vision and graphics challenges. In the process, we transform an unwieldy point cloud, containing millions of 3D points, into a concise and generative expression of the space and objects recorded by the laser camera.

Understanding 3D images is increasingly important in fields as diverse as medicine, architecture, and navigation. Each domain is replete with unique patterns that practitioners must uncover as they struggle to make sense of their data. In the urban environment, for example, the topology is characterized by the broad planes of façades, the slightly curving manifold of the street, and the fenestration of regular repeated architectural features like windows, balconies and cornices. At street-level we see the fractal branching patterns of vegetation, the Bezier curves of parked cars, and the high frequency spikes of objects in motion faster than the laser can catch them.

The first algorithm presented detects translational symmetry in building façades. Fo-
cussing on repeated architectural structures helps us understand the scene and leads naturally to algorithms for compressing, registering and reconstructing urban buildings in range scans. We demonstrate a method to discover these regular structures online—as the scanner scans. The method employs Fourier analysis to discover the architectural repetitions.

The second algorithm described infers parameters for a stochastic tree grammar. We use multi-resolution analysis to discover the self-similarity of the vegetation at different levels of resolution. Correspondences between different levels of detail are used to estimate the branching rules for different types trees.

Taken together these methods offer a novel toolset for transforming 3D point clouds into high-level descriptions that are simpler to analyze, transmit and store. It is important that they be concise so that they might work in mobile and networked applications. Simultaneously, it is important that these descriptions be generative as that makes them useful for geometry propagation, modeling and interactive enhancement.

1.1 Data Acquisition

The laser sensor, a time-of-flight camera, measures distance by shooting a laser beam at an object and recording the time it takes the laser beam to return. Knowing the laser travels at the speed of light, $c$, the distance to a sensed object is:

$$c = \frac{2d}{t} \Rightarrow d = \frac{c \cdot t}{2} \quad (1)$$

![Figure 2: A texture mapped range scan. The balconies, windows, and trees seen here can be detected and extracted on-the-fly using our techniques.](image)
The awesome velocity of light makes it impractical to use laser cameras over short ranges, because the reaction times of such sensors would need to be less than a few nanoseconds. Over the medium to long range (10-200 meters), however, laser cameras offer a high-resolution and accurate image. This makes time-of-flight range sensors an excellent choice for capturing the outdoor urban scene.

Our lab operates a Leica ScanStation2 laser scanner [Lei 2000]. The data retrieved is organized into a two dimensional array. The direction of the laser is controlled by delicately calibrated stepper motors. The laser repeatedly rotates a small consistent angle vertically and records a measurement. Once an entire column has been recorded another motor rotates the laser a small consistent angle horizontally and another column is recorded. In the pages that follow, we will refer to single arrays within the 2D array as both columns and scanlines. In the coordinate system returned by the scanner, the z-axis corresponds to the vertical direction.

This data is considered 2.5 dimensional, because the 2D array of 3D points contains valuable information that distinguishes the scan from an unstructured point cloud. For example, investigating neighborhoods in the 2D array we are able to fit normals in constant time without resorting to an exhaustive search of nearest neighbors, or the additional overhead of an approximate nearest neighbors library.

1.2 Problem Statement

Our first algorithm takes as input: a structured, vertically-oriented, 2D grid of 3D points exhibiting salient vertical periodicity. Through planarity detection the ground and façade points are segmented. We detect and extract the precise period with an efficient algorithm,
which analyses the frequency space of the columns of the scene. This data is then utilized in registering distinct 3D scans of overlapping buildings into the same coordinate space. Armed with the periodicity of a façade, we demonstrate new methods to compress, reconstruct, and model large urban buildings.

The second algorithm takes an unstructured point cloud of vegetation and transforms it with multi-resolution analysis into quantized wavelet bases or a kd-tree. Using these hierarchical objects, we infer parameters for procedural models, which succinctly capture the form of the vegetation input. These models are used for compression, scene generation and enhancement, as well as interactive editing.

1.3 Vision and Graphics Applications

We present solutions to a variety of vision and graphics challenges each built upon our regularity detection techniques. Specifically, from the vision domain we address 3D view registration, compression, façade reconstruction, and shape grammar inference. Additionally, we demonstrate the graphical applications of our method, in geometry enhancement and interactive editing of procedural models.

**Registration** – Each range scan only records a single vantage point of the surveyed scene. For this reason, multiple scans of the same general area are often acquired. To make use of this extra information we would like to integrate the images into a single image. This requires registering distinct 3D coordinate spaces into a single coordinate space, so that their axes align and the same points seen by different scanners are plotted together. We exploit the periodicity of architectural features as the foundation for high-level automatic registration algorithm which works by aligning the local frames of aggregated
vertical groups of windows or balconies. This novel registration technique is described in Section 3.2.

**Compression** – While entropy encoding compression techniques approach optimal lossless compression, semantic compressors with domain specific knowledge still offer compelling advantages. They can be excellent tools for more ambitious lossy compression (i.e. the wavelet-based JPEG standard). Additionally, since they are defined natively on the data they can be easier to decompress leading to more efficient 3D image processing pipeline. Our methods for semantic compression of repetitive architectural features is presented in Section 3.3, and our wavelet and grammar based compression techniques for vegetation are discussed in Section 4.3.3.

**Reconstruction** – The urban environment is a bustling, dynamic scene filled with obstacles. Occlusions make processing and visualizing virtual scenes difficult. The large discontinuities at the edges of the occlusion or shadow befuddle many algorithms. To make matters worse, the missing data and inconsistent sampling make image renderings seem incomplete and distorted. Periodicity in the façades can rectify these issues. By detecting periodicity and propagating periodic elements into the shadow regions surfaces can become completed. Details on this online method for reconstruction (or hole-filling as the process is often called) are presented in Section 3.4.

**Procedural Modeling** – Our reconstruction technique lays the seeds for an elegant and concise procedural model for the urban scene. We can store a compressed, but generative encoding for both buildings and trees. This encoding is also known as a shape grammar. In building façades, the shapes are the representative repetitive regions, and the grammar
dictates rules for propagating that shape to reconstitute the initial building or synthesize an artificial one. The grammar productions horizontally and vertically translate the representative, reconstructing façades automatically and potentially editing façades interactively. For trees, the grammar encodes the environmental and endogenous growth parameters which stochastically determine the tree’s structure. Because only a small number of parameters are used, they can be adjusted interactively with a user and/or inferred automatically by our method (Section 4.2.3).
2 Background and Related Work

Regularity detection in 3D data is a field of active research. Learning global properties of orthogonality, orientation, and shape [Li et al. 2011a] reconstructs engineered shapes from noisy point clouds. In [Mitra et al. 2006], the authors present a method for extracting approximate symmetries by voting in transformation space. Detecting lattices in the transformation space [Pauly et al. 2008] extends the transformation voting method to find smaller repeating symmetries. Harmonic decompositions for symmetry detection can be found in [Martinet et al. 2006]. Using spherical harmonic decompositions they develop a method that recognizes precise symmetries in 3D mesh data.

Online algorithms for processing Lidar data are gaining attention as 3D cameras increase in popularity. In [Hadjiliadis and Stamos 2010] the authors present an online method that uses Markov chains and detection of abrupt changes to classify vegetation, horizontal and vertical points. The authors of [Friedman and Stamos 2011] introduce the method of online detection of repeated features by using Fourier analysis of column functions.

2.1 Symmetry Detection

Symmetry is ubiquitous. In nature and in engineering, both precise and approximate symmetries abound. Life forms as varied as centipedes and orchids exhibit striking symmetries. These structures underlie the versatility and variety of practically all life on earth,
as observed in D’arcy Thompson’s classic text [1942]. Self-similar structures like trees, branching capillaries and coast lines are a matter of course in both physical and computational worlds as illustrated beautifully in Benoit Mandelbrot’s masterpiece [1983]. The human brain and eyes are keenly aware of symmetry for recognizing, navigating, and organizing our world. Computer vision systems can similarly benefit from symmetry detection although current state-of-the-art algorithms still pale in comparison to our visual cortex.

Broadly speaking, symmetry refers to regularity in space. This regularity can be precise as in the reflective symmetry of a chair or approximate as in the scaling and translational symmetry of a Nautilus shell. We consider both global symmetries, like the rotational symmetry of a sphere, and local symmetry, as in the repetition of windows throughout a larger building façade. Mathematically, two entities $x, y \in X$ are symmetric under some operation $O$ if $O^n(x) = y$ where $n$ is an integer representing the number of compositions of $O$. The set of all entities $y \in X$ where $O^n(x) = y$ makes up the symmetry group of $x$ under $O$.

Symmetry detection can also be viewed as discovery of antisymmetric components of an entity and the set of operations on those antisymmetric components necessary to reconstitute the entire object. This view illustrates the efficiencies that come with symmetry. Once an object’s regularity is known, the object can be decomposed into its constituent antisymmetric components. This can greatly reduce the storage necessary for the object and accelerate further processing. We sometimes refer to the antisymmetric constituent parts as generators. They are called generators because, by applying the correct operations in the correct order, these disparate parts can generate the entire starting object.
Modeling 3D urban scenes from range scan data is a field of active research [Allen et al. 2003; Stamos et al. 2008; Zhao and Shibasaki 2003]. There are a plethora of range image segmentation techniques from range data, including edge detection [Bellon and Silva 2002; Wami and Batchelor 1994], region growing [Besl and Jain 1988; Pulli and Pietikinen 1993], and polynomial surface fitting [Besl and Jain 1988; D. Marshall and Martin 2001]. Most of these methods provide edge maps or regions expressed as polynomial functions. [Yǔ et al. 2001] utilizes a graph-cut approach for segmentation. Our earlier work includes segmentation algorithms for the extraction of planar, smooth non-planar, and non-smooth connected components [Chao and Stamos 2007; Stamos and Allen 2002; Stamos et al. 2006].

A shape-grammar approach that uses ground-based images was presented in [Teboul et al. 2011; Teboul et al. 2010]. An earlier approach is the one of [Lee and Nevatia 2004]. The work of [Mayer and Reznik 2007] uses a sequence of images to produce a sparse 3D point cloud as input to the detection algorithms. Work on the same topic is presented in [Wu et al. 2010]. Finally [Park et al. 2009] detects regular patterns in images that are not constrained to be buildings.

Fewer approaches exist when the input is a 3D range scan. [Pauly et al. 2008] derives regularities of substructures from a 3D model or range scan of a scene. This works by detecting symmetries (or similarity transformations) of basic structures in a regular grid. This general approach can be used for extracting regularities. [Shen et al. 2011] presents a partitioning of urban façades which detects grids in an adaptive way. This extends the transformation voting technique of [Pauly et al. 2008] to façades that are not globally rectilinear.
In [Stamos and Allen 2002] window-like rectangular features were extracted by using 3D edge detection on high-resolution 3D data. Line features are again utilized in [Bokeloh et al. 2009] for symmetry detection. This symmetry detection of line features builds upon a graph based method for detecting symmetry presented in [Berner et al. 2008]. [Nan et al. 2010] presents an interactive interface which exploits regular structures in urban scenes to improve sparse point clouds. Employing a Markov Network approach [Triebel et al. 2006] labels points as windows, but requires training. The work of [Li et al. 2011b] provides a fusion of 2D and 3D segmentation and symmetry detection for decomposition of urban façades. The authors of [Martinet et al. 2006] use spherical harmonic decompositions to detect rotational and reflective symmetries in meshes. Our work also exploits harmonics, but we use them to detect translational symmetries per scanline.

This thesis provides algorithms that work online and purely on 3D-range input, without relying on 2D-images, or training data. We have presented work in the online detection of repeated patterns in urban scenes using 3D-information [Friedman and Stamos 2011]. This proposal extends the applicability of that work into the area of 3D-range registration, compression, and hole-filling. Table 1 compares and contrasts several of the algorithms discussed above.

In recent years, reconstructing images with missing data has received much attention. Given a 2D image [Drori et al. 2003] presents a method which treats the input image as a training set from which the missing data is inferred. Leveraging databases of millions of images [Hays and Efros 2007] completes scenes by finding semantically similar pictures. Fusing both 2D and 3D images [Li et al. 2011c] computes a layer decomposition from
a registration between the 2D and 3D data sets which allows for outlier removal and geometry propagation. [Sharf et al. 2004] introduces a context sensitive surface completion. Their algorithm iteratively aligns surface patches along the edge of holes until the surface is complete. A comprehensive survey of urban reconstruction techniques is presented in [Musialski et al. 2012].

Much attention has been focussed on detecting symmetry for reconstruction, registration, and compression applications. For example, in [Mitra et al. 2006] 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, [Berner et al. 2008], 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 [Mandelbrot 1982]. Self-similarity is both more general than symmetry, and more common in the natural world. A recent paper by [Huang and You 2012] used self-similarity features for 3D correspondence and registration of dense point clouds.
<table>
<thead>
<tr>
<th>Reference</th>
<th>Method</th>
<th>Input</th>
<th>Features</th>
<th>Symmetries</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Berner et al. 2008</td>
<td>Feature Graph Matching</td>
<td>Meshes Points</td>
<td>Slippage Features</td>
<td>Rigid</td>
<td>5-10 Minutes for 500K points</td>
</tr>
<tr>
<td>Berner et al. 2011</td>
<td>Dimension Reduction</td>
<td>Points</td>
<td>Subspace PCA</td>
<td>Self Similarities</td>
<td>5-10 Minutes size not given</td>
</tr>
<tr>
<td>Bokeloh et al. 2009</td>
<td>Feature Graph Matching</td>
<td>Meshes Lines</td>
<td>Rigid</td>
<td></td>
<td>18 Minutes 2.2 million points</td>
</tr>
<tr>
<td>Bokeloh et al. 2010</td>
<td>Grammar Inference</td>
<td>Meshes Docking Sites</td>
<td>Rigid</td>
<td></td>
<td>10 Minutes 1.2 million points</td>
</tr>
<tr>
<td>Friedman et al. 2011</td>
<td>Harmonics</td>
<td>Points</td>
<td>Eigenvalues Curvature</td>
<td>Translations</td>
<td>10 Seconds 1 million points</td>
</tr>
<tr>
<td>Kazdahan et al. 2004</td>
<td>Symmetry Descriptors</td>
<td>Voxels Spherical Autocorrelation</td>
<td>Reflections</td>
<td></td>
<td>0.72 seconds for 524 voxels</td>
</tr>
<tr>
<td>Martinet et al. 2006</td>
<td>Generalized Moment Functions</td>
<td>Meshes Moments Local Frame Approximate Isometries</td>
<td>Rotation Reflection</td>
<td></td>
<td>50 Minutes per shape</td>
</tr>
<tr>
<td>Mitra et al. 2006</td>
<td>Transformation Voting</td>
<td>Points</td>
<td>Local Frame</td>
<td>Approximate Isometries</td>
<td>2 Minute 200K points</td>
</tr>
<tr>
<td>Mitra et al. 2010</td>
<td>Grid Detection</td>
<td>Points</td>
<td>Autocorrelation Intrinsic Regularities</td>
<td></td>
<td>1 Minute size not given</td>
</tr>
<tr>
<td>Muller et al. 2007</td>
<td>Mutual Information</td>
<td>Rectified Images</td>
<td>Moments</td>
<td>Translation</td>
<td>30 - 90 Seconds 1600 x 1200 image</td>
</tr>
<tr>
<td>Pauly et al. 2008</td>
<td>Transformation Voting</td>
<td>Points</td>
<td>Curvature Similarity Transforms</td>
<td></td>
<td>3 Min 250K points 100K Transforms</td>
</tr>
<tr>
<td>Podolak et al. 2006</td>
<td>Symmetry Descriptors</td>
<td>Meshes</td>
<td>PRST See</td>
<td>Reflection</td>
<td>8 Seconds 64^3 grid resolution</td>
</tr>
<tr>
<td>Teboul et al. 2008</td>
<td>Grammar Fitting</td>
<td>Rectified Images</td>
<td>Randomized Forest Self Similarity</td>
<td></td>
<td>not given</td>
</tr>
<tr>
<td>Xie et al. 2007</td>
<td>Graph Matching</td>
<td>Points</td>
<td>Sequence Profile  Similarity</td>
<td></td>
<td>not given</td>
</tr>
</tbody>
</table>

**Table 1:** Comparison of symmetry detecting algorithms. Comparisons are across feature type, computational time, symmetry types, and the general method of discovery.
2.2 Fourier Analysis

Fourier analysis studies functions by decomposing them into their constituent sinusoids. The entire set of constituent sinusoids is called the Fourier Transform, and it equals:

$$F(\xi) = \int_{-\infty}^{\infty} f(x)e^{-i2\pi\xi x} \, dx$$ (2)

The transformation, $F(\xi)$ is an equivalent, (but not equal) representation of the input signal $f(x)$. The Fourier Transform, $F(\xi)$, stands in stark contrast to the input signal, which we presume to be spatially or temporally ordered. $F(\xi)$ is ordered by frequency. The spatial (or temporal) information is obscured, rendering the frequency information more salient.

The Fourier Transformation clearly exposes and quantifies the periodicity, major directions, and global orientations present in the input signal. This information was latent within the initial data, but may have been obscured by prominent local features that are suppressed by the global nature of the Fourier Transform. More specifically, each frequency is quantified at a specific location in the transform. For the input signal of a single sinusoid of a given frequency, the Fourier transform is a delta function at that frequency and 0 everywhere else. See Figure 4.

The Fourier Transform is defined for complex variable $x = a + ib$. This introduces some difficulty in visualizing the transform: should we show its real or imaginary part? Typically, the most informative view of the transform is its magnitude $= \sqrt{a^2 + b^2}$. For a real signal the Fourier Transform will be symmetric about the y-axis. All of the signals analyzed in this thesis are real.
This operation is invertible and energy preserving allowing us to compute the inverse of the Fourier transform as:

\[ f(x) = \int_{-\infty}^{\infty} F(\xi)e^{i2\pi\xi x} \, d\xi \quad (3) \]

This property allows us to do analysis and manipulations in the transformation and then restore the original signal (or an improved and modified signal). Because of the inverse relationship between frequency and wavelength (i.e. \( \frac{1}{\text{frequency}} = \text{wavelength} \)) a signal and its Fourier Transform often exhibit opposite or complementary features. See for example Figures 6.

We extend the Fourier transform to the discrete case with the equation:

\[ F[k] = \sum_{n=0}^{N-1} f[n] \cdot e^{-i2\pi kn/N} \quad (4) \]
And by exploiting the fact that the $n$th root of unity is given by $\omega_n = e^{2\pi i/n}$ we can express the discrete Fourier Transform on a 1D vector of length $N$ as:

$$F[\omega_n] = \sum_{n=0}^{N-1} f[n] \cdot \omega_n^{-k}$$

This formulation leads to the algorithm first discovered by Carl Friedrich Gauss in 1805 and later re-discovered and often attributed to Cooley and Tukey in 1965. The naïve implementation of the Fourier transform takes $O(n^2)$ operations, but using the Fast Fourier Transform (FFT) this running time is reduced to $O(n \log_2 n)$. The FFT is a divide and conquer algorithm, which works by recursively splitting the initial transform into two separate transforms, one transform of the odd indices and one of the even indices. This technique exploits the symmetry about the unit circle of the roots of unity. Often, the splitting is done in halves, which requires the input to be a power of two. This requirement, however, is not stringent and can be overcome by zero padding a signal up until the nearest power of two, as we do in Section 3.1.4.

We also define the 2-Dimensional discrete Fourier Transform on a discrete 2D array, such as a digital picture, $F[x, y]$ of dimension $W \times H$, where $x \in (0, 1, \ldots, W)$ and $y \in (0, 1, \ldots, H)$.

$$F[u, v] = \sum_{x=0}^{W-1} \sum_{y=0}^{H-1} f[x, y] \cdot e^{-i2\pi(ux/W + vy/H)}$$

The FFT can also process 2D signals by analyzing the rows and columns separately. The run time of the FFT on a 2D signal is thus $O(WH \log_2 W \log_2 H)$. Figure 5 shows
Figure 5: At left is a typical image and on the right is its 2D discrete Fourier Transform. To be precise, the right image displays the log of the magnitude of the complex variables returned by the 2D DFT algorithm. See Section 2.2.

a 2D Fourier transform on a typical digital image. To make the transform visually legible we display the log of the magnitude because the size of the zeroth frequency, which is the average of the signal, tends to overwhelm all the other frequencies present in the image.

The inverse relationship between an image and its transform is rendered quite clearly in Figure 6. While the initial image has a tiling pattern that seems to pull the eye inward, the transform seems to do the opposite pushing the eye out. Additionally, the areas with low energy in the input seem to have the highest energy in the transform.

One significant application of the Fourier transform is discovering the major directions of an input image. Lines in the transform run perpendicularly to the major directions in the input image. This is why transforms of many images display a salient cross along the x and y axes. Many natural images, especially those of built environments contain horizontal and vertical major directions. The global nature of the Fourier transform that renders
The Fourier transform is an excellent tool for extracting the major directions from an image. Figure 6: This transform illustrates the inverse relationship between a signal and its DFT.

Figure 7: The Fourier transform is an excellent tool for extracting the major directions from an image. the major directions so conspicuously.

This ability of the Fourier transform to detect major directions, can be used to orient text in images. In English, for example, text is written horizontally and arranged in lines vertically. However, many pictures of text deviate from this standard, because photographers may be unable to acquire a direct perpendicular view of the text in question. This presents a difficulty for image to text programs. The Fourier transform can ameliorate this problem by exposing the major directions of the text in the image. The angle that rotates the transform so that the major angles align with the x and y axes is also a good choice of an angle to rotate the text so that it is oriented correctly.

Another major application of Fourier transforms has been found in the noise reduction
of images. Most noise tends to be at high frequencies that don’t correspond to visually significant information. While it is difficult to extract high frequencies from within the spatial domain, the frequency domain relegates them to the edges of the image. By applying a low pass filter in the frequency domain and then transforming back to the spatial domain, high frequencies are efficiently removed from an image. See, for example, the low frequency image of a painted woman and its Fourier transform in Figure 9.

A major drawback of the Fourier transform is that it is global in time and local in frequency giving an exact picture of the various frequencies of a signal, but it gives little or no indication of when these frequencies reside within the duration of the signal. One major advance of wavelet analysis, presented in Section 2.3 is that is local in both frequency and time. Figure 10 illustrates some of the challenges arising from the
Because the Fourier transform is global in time the diagonal frequency of the rectangle in the center of the image at left perpetuates throughout the entire transform. This is challenging because once this frequency is detected, more work is still required in order to find out where in the initial image this frequency existed.

Lastly, another capability of Fourier analysis lies in the detection of repeated periodic patterns. For example, Figure 11 contains many repeated patterns stemming from the woven fabric. These repetitions present themselves as clusters in the transformation space. We use this capacity of Fourier Analysis in our periodicity detection algorithm, presented in Section 3.1.1.

Fourier analysis has been used extensively for the compression of images. This technique relies on the aforementioned visual importance of low frequencies and relatively large noise content of high frequencies in natural images, as shown in Figure 9. The compression scheme uses a windowed Fourier transform and com-
presses images by throwing away the high frequencies in each window. The input image is broken into tiles or windows, typically $8 \times 8$ pixels wide. The transform is then applied to each of the tiles. The high frequency content is removed from each tile and the compressed transforms are stored. To reconstruct the image the inverse transform is applied on each compressed tile and the tiles are arranged back into the original grid. This tiling process is very similar to the method of wavelet analysis presented below.

### 2.3 Wavelet Analysis

Wavelet analysis decomposes signals into waves of various shapes and durations. Wavelets detect local variation in a signal. This contrasts with Fourier analysis which measures the global variation across an entire signal. The simplest wavelet transform decomposes signals into a set of dilated and translated square waves. This is called the Haar wavelet. The algorithm can be understood as a filter bank consisting of a high-pass and a low-pass filter. The signal passes through the filters. The output of the high-pass filter becomes the fine level wavelet details. The output of the low pass filter is down-sampled and then recursively filtered again and again until the low-pass filter is simply the average of the signal and the entire signal has been decimated to one value by the down sampling. This interpretation of the wavelet transform makes defining a new wavelet basis as simple as specifying the coefficients for the high and low-pass filters.

The Haar wavelet is piece-wise constant, preventing effective compression non-constant functions. Ingrid Daubechies presented her celebrated wavelet basis to address this issue [Daubechies et al. 1992]. These famous numbers exhibit some striking properties. The
Daubechies 4-tap wavelet set is given by scaling numbers (low-pass filter coefficients):

\[
\frac{1 + \sqrt{3}}{4\sqrt{2}}, \frac{3 + \sqrt{3}}{4\sqrt{2}}, \frac{3 - \sqrt{3}}{4\sqrt{2}}, \frac{1 - \sqrt{3}}{4\sqrt{2}},
\]  

and the wavelet numbers or the high-pass filter coefficients are given by:

\[
\frac{1 - \sqrt{3}}{4\sqrt{2}}, \frac{-3 + \sqrt{3}}{4\sqrt{2}}, \frac{3 + \sqrt{3}}{4\sqrt{2}}, \frac{-1 - \sqrt{3}}{4\sqrt{2}}.
\]

As with the Haar wavelet, the scaling numbers mean is 0 allowing it to represent the local trend of a signal. On the other hand, the wavelet energy is 0, allowing the high-pass filter to find details in local areas of the signal. Unlike the Haar wavelet, Daubechies wavelet basis also satisfies two other equations. If we label the coefficients \(\beta_1\) through \(\beta_n\), then we have:

\[
\beta_1 + \beta_2 + \beta_3 + \beta_4 = 0 \quad (9)
\]
\[
0\beta_1 + 1\beta_2 + 2\beta_3 + 3\beta_4 = 0 \quad (10)
\]

Because of these equations, when a signal is approximately linear the wavelet coefficient over that region will be approximately 0. This means smooth signals which are locally linear can be compressed tremendously by transforming them into Daubechies wavelet basis [Walker 1999].
Figure 12: Four signals and their wavelet transforms. Notice how the square wave in the bottom right is in sync with Haar wavelet and all coefficients are zero except for one level of detail. The square wave to the left is out of phase with the Haar wavelet and requires non-zero coefficients from many levels of detail.

Figure 13: Four 2D signals and their wavelet transforms. Notice how the wide range of images all exhibit wavelet transforms with energy concentrated in coarse levels of detail.
2.4 Procedural Modeling

Developing procedural models for vegetation has been the subject of research for many years [Prusinkiewicz and Lindenmayer 1991]. Lindenmeyer systems, or L-systems, have tremendous expressive power for concisely describing many plant forms that appear in nature. Traditionally, deriving an L-system for a given model requires both botanical and and grammatical expertise. L-Systems were first introduced to model development at the cellular level in [Lindenmayer 1968]. They have since been extended to many other applications and to allow for environmental interaction with the model [Méch and Prusinkiewicz 1996].

The concept of shape grammars for buildings introduced by [Stiny 1982] as a formal approach to architectural design has been used for the image-based modeling of façades acquired through aerial photographs [Muller et al. 2007] and for the generation of synthetic cities [Muller et al. 2006].

In recent years efforts have been made to automatically infer a grammar from a given model. For example, the work of [Bokeloh et al. 2010] 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 pair-wise similarities, [Št'ava et al. 2010] automatically generate context-free L-systems from 2D images. Shape grammars are another approach to model façade regularity utilized in [Muller et al. 2007].

L-Systems are formal grammars that employ parallel rewriting of strings to model various phenomena. L-Systems have proven especially adept at capturing biological processes like plant development, and cellular growth in various species of algae. An L-System consists of an axiom or starting string, a set of rules for replacing symbols found within the
Axiom of any of its descendent. A simple, yet beautiful L-System is described below.

\[ \begin{align*}
Axiom & \Rightarrow X \quad (11) \\
F \Rightarrow FF \quad (12) \\
X \Rightarrow F[+X][-X]FX \quad (13)
\end{align*} \]

This L-System consists of only two rules. The first rule is trivial and simply elongates existing branches at each generation in derivation. The second rule is more interesting. We replace \( X \) with the string \( F[+X][-X]FX = rhs_1 \). So After one generation the string will simply read the right hand side or the first rule, \( rhs_1 \). After two generations we will replace the \( F \)s with \( FF \) and the \( X \)s are recursively replaced with \( rhs_1 \) from the first rule, yielding:

\[ rhs_2 = FF[+rhs_1][-rhs_1]FFrhs_1 = FF[+F[+X][-X]FX][-F[+X][-X]FX]FFF[+X][-X]FX \quad (14) \]

This unwieldy string has an elegant graphical interpretation as the movements, rotations, and memory operations of a turtle with a pen attached to his tail. The turtle maintains a location and an orientation, as well as a memory stack of previous locations and orientations. The \( F \) symbol indicates a move in the direction of the current heading some set amount. The \( [ \) symbol causes the current location and orientation to be stored on the memory stack and the \( ] \) symbol restores the last saved heading information, thus erasing the
Figure 14: The angle of an L-System can have a dramatic impact on the look and feel of the resulting productions. These tree-like structures are various generations of the L-System given in 13. All of the above structures have the angle, \( \alpha = 30 \) degrees.

current location and orientation. The + and – symbols require us to expand our L-System to allow its first parameter, the angle \( \alpha \). The plus and minus symbols are interpreted as positive or negative rotations in the 2D plane by the angle \( \alpha \). Other symbols like X are simply ignored by the turtle renderer.

Non-Parametric L-Systems are limited to integer multiple geometries. This precludes accurate description of many fundamental shapes, for example, the right triangle. Because the diagonal of a right triangle is the irrational number, \( \sqrt{2} \), there is no ratio of text replacement operations, where an integer number of characters are replaced by another integer number of characters, which precisely captures it. The rigidity of deterministic L-Systems motivated the invention of parametric L-Systems, and other procedural modeling methods,
Figure 15: Another example of the productions possible by the L-System given in 13. All of the above structures have the angle parameter, $\alpha = 90$ degrees.

Figure 16: Another example of the productions possible by the L-System given in 13. All of the above structures have the angle parameter, $\alpha = 115$ degrees.
including the stochastic parametric system we present in Section 4.1.1. In a parametric system, rules are applied stochastically according different internal and system-wide parameters, which allows for more flexible and dynamic shape productions.

Instead of the local features of [Mitra et al. 2006] or the graph-similarity employed by [Berner et al. 2008], in our approach, self-similarity is discovered by using the wavelet transform [Mallat 1999]. A filter bank system of high and low pass filters decomposes the input into various level of detail. 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 2. Each tile stretches from one power of two to a subsequent power of two. For this reason, we refer to these tiles as dyads.

3 Detection of Periodic Structures

Laser range scans of urban areas have a distinctive geometry dominated by façade and ground planes and repetitive regular fenestration. Detection of these ubiquitous features provides profound insights into the scene. We present a novel method for detecting major planes and repetitive architectural features. Armed with this knowledge we illustrate its application in compression and registration of range scans. Additionally, our algorithm operates online, processing the scan as it is retrieved by the scanner. This realtime approach opens up new possibilities in range data segmentation, compression and registration.
### 3.1 Online Symmetry Detection

Processing point clouds quickly is crucial in many scenarios. Analyzing scans column by column allows seamless integration with the 3D camera because the camera is likewise scanning the scene column by column. This synchronizes the execution of the algorithm with gathering the data, accelerating our work to the point of just-in-time scene understanding. While focusing on individual columns offers excellent local knowledge on the scene, it is important to keep sight of the big picture; as each column is just a sliver of a much larger tableau. Our algorithm does this by keeping an evolving data structure with macro features and greedily updating as the scan unfolds. In this way, we are not forced to choose between speed and global knowledge.

We can imagine this online processing as a line sweep algorithm. The scanline or column of measurements moves across the scene in discrete steps. At each step we search for periodicity and planes. We aggregate and update this data maintaining best estimates about higher level features in the scene. The processing occurs on the fly so that by the time the scanline reaches the edge of the scene our algorithm has done its work and requires no further manipulations of the data. Integration into the 3D camera hardware is a natural step forward for this method, because the algorithm explores the same way the sensor does.

Processing these large data sets is computationally expensive and generally beyond the capacities of embedded processors included in laser scanning hardware. For this reason data acquisition and processing are commonly performed separately in sequence. We present a novel algorithm that allows nearly simultaneous acquisition and processing.
3.1.1 Extraction of Repeated Features Through Fourier Analysis

Windows and balconies are intriguing data points in urban laser scans. They are ubiquitous: virtually every commercial and residential structure is perforated by some kind of fenestration. In general, they are regular, occurring at a defined period within the building. Yet despite this regularity they are challenging to detect because of their high variability of appearance.

Unobstructed windows often absorb the laser beam into an interior from which it cannot escape. In those cases, a façade is dotted with missing data. Curtains and blinds can also cause problems depending on their material. They may deflect the laser beam away from the scanner or because of ruffles and gaps provide inconsistent measurements. To compound the confusion many windowsills are crowded with house plants, children’s toys and other accouterments of apartment interiors. Despite the difficulty of interpreting data from a single window, a vertical line of windows is predictably unpredictable. The areas of missing and inconsistent data occur regularly, with a defined period.

Terraces, which protrude from a building, cast shadows on the façade. The shadows grow with the incident angle of the laser, so that the highest balconies on a tall building may be totally hidden. Leaving us with vastly different measurements for the same structure, depending on its altitude.

All these facts make detection of regular architectural features difficult. Nevertheless the periodicity of the features is salient. While each individual feature may be unpredictable when analyzed together they are predictably unpredictable. This regularity can be extracted through Fourier analysis and this knowledge is fertile ground for downstream processing.
3.1.2 Finding The Major Planes

We begin by approximating the major planes for each column. Most columns in the urban setting are dominated by points belonging to the ground and points from a façade. In general façades are extremely planar. The manifold of a city’s pavement is usually curved, but its curvature is in general low and a planar approximation is often sufficient. To extract these major planes we fit minor planes in small neighborhoods around each point in the scanline. Using Principal Component Analysis (PCA) we determine the normal to these minor planes as the eigenvector corresponding to the smallest eigenvalue of the covariance matrix of the neighborhood. The eigenvalue itself is also stored, as its magnitude is a measure of the goodness-of-fit of the local plane to the local data. Throughout the scan we maintain our best estimates of the façade and ground normals.

In general, buildings rise perpendicularly from the horizontal ground, so we may filter potential ground normals from potential façade normals by taking the dot product with the vertical vector. We assume the vertical vector is known as there a number of simple ways to determine it. We expect this product to be near 0 for ground normals and near 1 for façade normals. Now we have gathered two groups of small local planes from which we would like to deduce two global planes. The eigenvalues computed earlier help separate the noise from the signal by measuring the reliability of each local plane. However, simply selecting the normals corresponding to the smallest eigenvalues in the column would be a mistake. For example, it is common for a façade with a balcony to register two reliable vertical planes, or a scan that passes over a box truck or a ceiling from an interior may contain two or more horizontal planes with very low eigenvalues. To ensure against these
mis-detections, the major planes are chosen as those with low eigenvalues and agreement with the majority of other potential vertical or horizontal planes.

As the scan progresses and more columns are recorded we greedily update our major plane estimates. While searching for the most reliable façade normal, it is important that we remain vigilant against façade shifts that occur when the scanner passes over a corner of a building or two adjacent façades. With this in mind, before the running estimate of the façade is updated, the angle between the old normal and the new normal is measured. If it is insignificant ($\leq 0.1$ in our experiments) and more reliable than the previous, our estimate is updated. If this angle is near $\pi/2$ we note a corner. If this angle is not near zero and not near $\pi/2$ this may be a noisy column or modern architecture. The angle between façade normals does not detect façade translation which are common when adjacent buildings are offset from the street by different amounts. To ensure that the façade has not been translated we must check that:

\[
(f_1 - f_2) \cdot n_1 \approx (f_1 - f_2) \cdot n_2 \approx 0, \tag{15}
\]

where $f_1$ is a point on the old façade, $f_2$ is a point on the new façade and $n_1$ and $n_2$ are normals of the two façades. If this is not the case then we must note a façade shift and going forward greedily improve the new façade. In settings where online processing is unnecessary, it is possible to improve previous global estimates after one pass has been made over the data.

3.1.3 The Column Function
The global features of the major planes are complemented by the meso-level features of regular fenestration in windows and balconies. Fenestration tends to be regular, perforating each floor once. The first step in the detection of this regularity is construction of a column function. The estimates of the ground and façade planes allow us to distinguish all the points in the column on or near the façade. The rest of our analysis will focus on this subset.

The column function is derived from a local measure taken at each point in the column. It is conceivable that curvature, eigenvalues, or consecutive angles could be used. In our work we have chosen consecutive angles for the simplicity and speed of their computation. Two column functions are shown in Figure 18. Despite being a noisy and simple metric the regularity of repetitive features shines through.

**Figure 18:** The column function.
Before this regularity can be extracted the column function must be interpolated. As it stands the column function is distorted by an oversampling of close features and an under sampling of distant ones. Figure 21 illustrates the predicament. To rectify this, points are sorted by their height. The column function is scaled to reflect these heights. After the linear interpolation, the column function shows actual distances in the scene and not the concentration of measurements returned by the laser scanner.

3.1.4 Frequency Space

The signal after interpolation is truly periodic. Fourier analysis will indicate exactly which period is present in the function. The Discrete Fourier Transform maps our signal into the
frequency domain. To find the period in the column we take the Discrete Fourier Transform (DFT) of the column function. The frequency domain of a periodic column will register two secondary peaks corresponding to the prevalent frequency in the column. Plotting this frequency in the spatial domain shows the repetition of the regular features. Our Fast Fourier Transform implementation requires the size of the input function to be a power of 2. To accommodate this, for a column function with $s$ values we zero pad by finding the $k$ such that $2^{k-1} < s < 2^k$. If the index difference between the zeroth frequency and the dominant frequency is $d_i$, and the vertical height of the entire column is $h$, then the period is given by:

$$\text{period} = \frac{h}{d_i \left( \frac{s}{2^k} \right)} = \frac{2^k h}{d_i s}$$  \hspace{1cm} (16)$$

Typically the dominant period of a façade corresponds to the height of each floor.

### 3.1.5 Fenestration Centroids from Square Waves

While the DFT gives us the frequency of the period we still must compute the phase. We fit square waves to the column function. The phase is shifted to align with middle of the high values of the square wave. This corresponds with the center of the architectural features. Because the square wave measures the actual height of the architectural feature it allows us to extract the exact middle of the windows and balconies.
The previously gathered information about periodicity in the column allows us to uncover the best-fitting square wave quickly. Up to reasonably fine resolution, it is practical to exhaustively check all possible square waves, whose period coincides with the period the Fourier analysis uncovered in the column. The square wave that fits the column function best is used to estimate both the height and centers of the architectural features.

To fit a square wave the column function is turned into a binary function by taking all values above a threshold as 1 and all values below it as 0. We measure the difference between square waves and this binary thresholded version of the column function. We repeat this process for every possible square wave within the period of the column, up to some discrete resolution. The wave with the smallest difference is chosen as the square wave approximation of the column function.

The middle of the high edge of the square wave corresponds with the middle of the repeated fenestration. Square waves are generated for each column in each aggregated window group computed in Section 3.1.6. The center of the feature can then be estimated as the median of the vertical centers of each column and the mean of the horizontal distance between the first and last column.
3.1.6 Aggregation

The estimation of the period from a single column may be improved through aggregation. Since the algorithm is processing data online directly after uncovering the period latent in one column function another column function is ready for processing. With high probability this column will record data from the same repeated architectural features that were previously uncovered. Adjacent columns with similar periods are grouped providing higher level insight into the features present in the façade. The period estimates can be improved by detecting the secondary spikes in the sum of all the transforms of the neighboring columns. Before summing, the transforms must be centered about the zeroth frequency, because the columns are not necessarily the same size. The period in the aggregated transform is a more robust measure of the frequency of the repetitive features.

3.1.7 Evaluating Regularity Detection

We employ several methods to evaluate the quality of the detected translational symmetries. First, polygons of the repeated architectural features are superimposed on the scan data. This gives a visual verification of the detection of both the period and the phase of the features.

For a quantitative assessment we have annotated the scans with labels for the regular features. For example, points are labeled as windows by a human visually selecting them. This ground truth is compared against the automatically labelled windows. If 90% or more of the labeled points fall within the area detected as a window it is considered a true positive. If there is a detected window and no labeled points fall within it, then it is a false positive, and lastly, if there are labeled points but no detected window it is a false negative. In this
<table>
<thead>
<tr>
<th>Scan Name</th>
<th>True Positives</th>
<th>False Positives</th>
<th>False Negatives</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan 1</td>
<td>95</td>
<td>1</td>
<td>7</td>
<td>.9896</td>
<td>.9314</td>
</tr>
<tr>
<td>Scan 2</td>
<td>124</td>
<td>6</td>
<td>9</td>
<td>.9538</td>
<td>.9323</td>
</tr>
<tr>
<td>Scan 3</td>
<td>53</td>
<td>3</td>
<td>4</td>
<td>.9492</td>
<td>.9333</td>
</tr>
<tr>
<td>Scan 4</td>
<td>56</td>
<td>1</td>
<td>3</td>
<td>.9824</td>
<td>.9491</td>
</tr>
<tr>
<td>Scan 5</td>
<td>58</td>
<td>1</td>
<td>4</td>
<td>.9830</td>
<td>.9355</td>
</tr>
</tbody>
</table>

Table 2: The precision and recall of the regularity detection is presented here.

Figure 24: Superimposing the extracted mesh on the point cloud we can verify the accuracy of our results.

Evaluating a reconstruction of the type presented here is not straightforward. The algorithm synthesizes data we were unable to acquire. We must therefore evaluate our accuracy in hitting an unknown target. To begin we quantify both the precision and recall of our periodicity and phase detection by overlaying the polygonal windows on human-labelled input data. Figure 24 shows the windows in green superimposed upon the scan.

The algorithms presented here run in $O(n \log n)$ time where $n$ is the size of a single column. The sorting of the column function and the subsequent Fast Fourier transform both come with a cost of $O(n \log n)$. The aggregation of adjacent columns has the effect of multiplying $n$ by a constant but does not change our big-$O$ analysis. This is because windows and other periodic features do not scale up with the size of the scan. A larger
<table>
<thead>
<tr>
<th>Scan</th>
<th>Points</th>
<th>Column Size</th>
<th>Periodic Columns</th>
<th>Run Time (in milliseconds)</th>
<th>Time Per Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan 8</td>
<td>764154</td>
<td>7209</td>
<td>30</td>
<td>4859</td>
<td>.0064</td>
</tr>
<tr>
<td>Scan 2</td>
<td>546964</td>
<td>1364</td>
<td>232</td>
<td>6929</td>
<td>.0127</td>
</tr>
<tr>
<td>Scan 11</td>
<td>773376</td>
<td>2432</td>
<td>151</td>
<td>8390</td>
<td>.0012</td>
</tr>
<tr>
<td>Scan 14</td>
<td>1256904</td>
<td>4761</td>
<td>69</td>
<td>9415</td>
<td>.0075</td>
</tr>
<tr>
<td>Scan 9</td>
<td>1006740</td>
<td>1260</td>
<td>144</td>
<td>11419</td>
<td>.0113</td>
</tr>
<tr>
<td>Scan 6</td>
<td>1219106</td>
<td>3286</td>
<td>235</td>
<td>12705</td>
<td>.0104</td>
</tr>
<tr>
<td>Scan 3</td>
<td>1303359</td>
<td>2721</td>
<td>299</td>
<td>13951</td>
<td>.0030</td>
</tr>
<tr>
<td>Scan 1</td>
<td>805176</td>
<td>636</td>
<td>502</td>
<td>13316</td>
<td>.0165</td>
</tr>
<tr>
<td>Scan 4</td>
<td>1326117</td>
<td>2757</td>
<td>216</td>
<td>14943</td>
<td>.0112</td>
</tr>
</tbody>
</table>

Table 3: This table shows the algorithm’s running time on range scans of various sizes. Run time is not completely proportional to scan size. This is because the individual column size and the ratio of periodic columns also factor into the algorithm’s performance.

city does not have larger windows. Rather the windows have a more or less constant range of widths and heights regardless of the size of the scan in which they occur. By a similar argument our exhaustive search for a square wave approximation of the column function is limited by the size of the period.

We conducted time tests of our algorithm running a variety of scans. The results are presented in Table 3. The per point processing time varies from 3 to 16 microseconds. This is several orders of magnitude faster than the rate at which the scanner acquires a single point. Additionally, our code is implemented in Java and not optimized.

3.2 Registration

The 3D data retrieved by laser scanners is far more informative than conventional 2D images. However, like 2D images a single range scan provides only one viewpoint onto the scene and important information may be missing. For this reason many scans are often taken of the same scene and registered together.

Registration is a major topic of research in 3D computer vision community. Here
we present a realtime registration technique which builds upon the information acquired from the analysis of the façade periodicity. This strategy could be employed by two or more scanners, for simultaneous realtime multi-view scene acquisition. Our algorithm offers an extremely efficient and automatic solution for scenes with repetitive patterns. This technique was first suggested in [Friedman and Stamos 2011] and then implemented and described in [Friedman and Stamos 2013b].

3.2.1 Orthonormal Bases of Aggregated Columns

Solving for the rotation between two overlapping scans is essentially a problem of defining bases. We must find the orthonormal bases that describe the two sets to be registered. Let us call these two sets $S_1$ and $S_2$. For each point $x_1 \in S_1$ its corresponding point in $S_2$ called
Figure 27: Another angle of two scan views registered together.

Figure 28: Two scan views registered together.
$x_{rot}$ is given by:

$$x_{rot} = B_2 B_1^{-1} x_1,$$

where $B_1$ is the matrix composed of the orthonormal bases of $S_1$ and $B_2$ is a similar matrix for $S_2$. To find the orthonormal bases $B_1$ and $B_2$ we find the vertical vector $\mathbf{v}$ and the orthogonal façade vector $\mathbf{f}$. The façade vector is established by taking the vector between any two points within the façade, called $\mathbf{f}_{\text{start}}$ and projecting this vector onto $\mathbf{v}$. The façade vector is the vector between $\mathbf{f}_{\text{start}}$ and its projection onto $\mathbf{v}$. By subtracting out the component of $\mathbf{f}_{\text{start}}$ in the direction of $\mathbf{v}$ we ensure that $\mathbf{f}$ is orthogonal to $\mathbf{v}$.

$$\mathbf{f} = \mathbf{f}_{\text{start}} - ((\mathbf{f}_{\text{start}} \cdot \mathbf{v}) \mathbf{v}).$$

Both vectors $\mathbf{f}$ and $\mathbf{v}$ are normalized. Now their cross product yields the complete set of three basis vectors:

$$B_* = \begin{pmatrix} \mathbf{f} \\ \mathbf{v} \\ \mathbf{f} \times \mathbf{v} \end{pmatrix}.$$
3.2.2 Computing Corresponding Pairs

Once the rotation matrix $B_2B_1^{-1}$ is known, solving for the translation between the scans is reduced to finding a pair of corresponding points. More precisely, we wish to find two locations in each scan of a single real world point. The translation between scans $t$ is then:

$$ t = x_2 - B_2B_1^{-1}x_1, \quad (20) $$

where $x_1$ and $x_2$ are two data points that correspond to the same real world point.

There are several ways to compute $x_1$ and $x_2$. The probability that the laser will measure the exact same point in two different scans of the same scene is impossibly small. For that reason choosing a pair of points and simply calculating the translation vector between them is bound to introduce some error; it is preferable to rely on higher level knowledge about the scene. To this end, we propose two different methods for computing translations between disparate views. The window width sequence method is best when the areas of overlap between the views are large and the resolution is high. The corner vector method is not as reliable as the window sequence but it requires less resolution and overlap.

Registering two scans by a pair of architectural features should result in an alignment of all the overlapping features. We compute the rotation matrix based on two sets of these features, the window groups from Section 3.1.6. If the mapping between feature groups is correct it should ensure that all the adjacent feature groups are also aligned. It is easy to check this by relying on their width. Expanding from the pair of feature groups we have aligned, we should find similar feature groups in all the remaining sets of adjacent features.
until the edge of the overlap.

If the scan does not contain significant or high resolution overlap it may be difficult to distinguish this pattern of matching features because there might be only one set of features that we can align. In these circumstances we rely again on the macro data about major planes found in Section 3.1.2. Now we can use the ground plane estimate to compute the vertical coordinate of the translation vector.

For corresponding aggregated window column groups we try to match the first columns from each set of columns. This locks in two of the coordinates of the translation vector. To find the vertical translation we use our major plane detection from Section 3.1.2. The cross product of the façade and ground normals gives the direction of the line of intersection between the two planes (assuming they are not parallel):

\[ i_d = n_g \times n_f. \]

(21)

Now this directional vector must be translated to align with the intersection of the major planes. For this translation vector we combine the z-coordinate of the ground plane and x and y coordinates of a point on the façade plane in the current scan line. Essentially we are dropping a line from the façade plane until it hits the ground.

Since the number of feature groups is relatively small it is not impractical to compute many possible transformations and then evaluate them all and select the best to register the scans. As a result, many transformations are generated and one will be selected as the best. We eliminate spurious transformations in a number of ways. First we ensure that the feature
groups to be aligned have similar periods as detected in Section 3.1. Furthermore we only compute transformations between window groups of similar widths. We then evaluate the quality of the transformation by checking the distance between aligned columns once a transformation has been applied to one of them.

To further evaluate our registration algorithm we performed ICP on our results. Surprisingly, there was a notable degradation in the quality of the registration. After close analysis we concluded that the ICP algorithm tried to align regions like parked cars that did not actually correspond to the same real world structures. This experience illustrated another strength of our algorithm. Not only do we compute registration in realtime far faster than ICP, we base our registration on façade features which are lasting elements of the scene. ICP treats all close points equally, so scans performed on different days are susceptible to error when a new set of cars are parked on the street, for example. The average error returned by the ICP algorithm was 0.65 meters. ICP measures an error 0.914 meters before registration but much of this error comes from false correspondences and does not actually indicate a flaw in the registration. The scan before and after ICP is applied is shown in Figure 29, clearly the registration before ICP is more accurate.

The transformation matrix output by our algorithm was compared against the transformation matrix computed from an interactive registration program where a user selects several corresponding points in the two scans. The interactive program then uses a least squares technique to find the transformation matrix between the scans. In our tests the per entry difference between the rotation matrices from our automatic approach and the interactive program never exceeded 0.0084 and averaged 0.0037. The translation vectors
Figure 29: At left is the registered scene before ICP is applied and at right the same scene after ICP. ICP is sensitive to small variations in the scene and does not detect whether points come from a moving object like a pedestrian or a stationary object like a building. As a result it reduces the quality of the registration.

differed by an average of 0.12 meters.

3.3 Compression

Range scans of urban scenes are dense, and data-rich point clouds of millions of points are not practical for many applications. Navigational robots may have limited computational resources onboard and processing millions of points may exceed their capacity. Networked applications may not have the bandwidth necessary to transmit this huge volume of data. As previously noted, much of the urban scene is repetitive, especially fenestration. This redundancy can be reduced through compression.

3.3.1 Segmentation and the Representative Region

Compression begins by identifying a representative region which will be stored in full, and used as a replacement for the data we remove. The vertical period of the façade was discovered in Section 3.1.1. Moreover, in Section 3.1.6 horizontal sections of the façade
Figure 31: The top image is a segmented scan. The middle image is a compressed version of the scan above it. The bottom image shows the amount of points which have been removed. The compression ratio is 48% with 383735 points removed out of 805176. Even though some window groups contain multiple columns of windows compression is unaffected because the periodicity of all of them has been correctly identified.
Figure 32: The original scan with periodic elements segmented, the amount of data reduction, and the compressed scan. The left image is a segmented scan. The middle image is a compressed version of the scan to the left. The right image shows the amount of points which have been removed. The compression ratio is 55%, with 300628 points removed out of 546964 total points.

were grouped and classified as planar or periodic. We now wish to segment the façade into like regions. One of these regions ought to be able to replace all the others vertically aligned with it, without compromising the character of the façade. Furthermore, the planar sections of the façade, which are interspersed with the periodic sections of windows and balconies, are also ripe for compression.

Segmentation proceeds by creating a rectangle around each periodic feature. The height of the rectangle is determined by the period detected in Section 3.1.4. The width of the rectangle is determined by taking half the distance to the center of the neighboring periodic features, unless these features occur in the beginning or end of a façade in which case the rectangle is stretched to the façade’s edge. The Euclidean distance would extract the circle around each window center, so instead we use the Chebyshev distance to extract
Because windows typically have vertical and horizontal symmetry we wish to extract the feature’s surrounding rectangle. The Chebyshev distance between two points $\alpha$ and $\beta$ in $\mathbb{R}^3$ is defined as:

$$D_{\text{Chebyshev}}(\alpha, \beta) = \max_{i \in \{x, y, z\}}(|\alpha_i - \beta_i|).$$

We use this metric to determine the rectangular region with sides equal to the length of the dominant period. One of these regions is chosen as the representative, in our experiments we choose the median size feature from the sets aggregated in Section 3.1.6. At first glance it is tempting to choose the densest region as the representative. This strategy is often a mistake, because often the region that is densest has a unique feature that distinguishes it from the rest of regions, and this disqualifies it as fair representative. Instead the region with median density is chosen.

To complete the compression we translate the chosen region vertically by the distance of the period. In this way the size of the column can be reduced by the number of windows.
found in the group of columns.

For a more ambitious reduction we assume that the building façade is planar. The vertical distance that came from the Fourier analysis can be coupled with the horizontal distance to the next adjacent group of periodic columns. The Chebyshev metric is altered to handle two different distances one vertical, the other horizontal. In the coordinate system given by the scanner the vertical axis corresponds to the z-axis of the points. However, the horizontal direction along the façade may be a combination of both the x and y coordinates. So, to find the horizontal distance to the next group of columns we take the Chebyshev distance in both the x and y directions.

\[
D_{\text{Vertical}}(\alpha, \beta) : (|\alpha_z - \beta_z|) \tag{23}
\]
\[
D_{\text{Horizontal}}(\alpha, \beta) : \max(|\alpha_x - \beta_x|, |\alpha_y - \beta_y|) \tag{24}
\]

Now much larger swaths of the building are included in the compression, see Figure 33. The entire façade is compressed by a factor equal to the number of periodic elements it contains.

### 3.3.2 Comparison of façade Segments

Some regions may contain extra information that should not be discarded when we compress with the median region. To avoid compressing regions with something that is not truly representative, the regions in question should be compared. Comparing 3D point clouds can be computationally intensive. To maintain the realtime flavor of the algorithm
we choose a method based on [Osada et al. 2002]. Histograms are generated from a subset of points within each region. The histogram contains the Euclidean distance of randomly selected pairs of points from the subset. To prevent spurious matches, it is sensible to choose high curvature points (computed as in [Cazals and Pouget. 2005]), or points whose eigenvalues computed during the local PCA were high. These points contain more of the unique character of each region rather than the homogeneous planarity.

Comparing histograms is simple: one is subtracted from the other. Alternative histogram distance metrics exist, however the $L_1$ metric is simple and efficient for our purposes. A large absolute value in the difference between histograms indicates two distinct regions, both of which should be preserved in the compressed scan.

Here we introduce a parameter $\gamma$ as the threshold by which regions are declared similar. This parameter allows user’s discretion over the degree to which their scans are compressed and the amount of data loss they will tolerate. Figure 35 shows the same scan with different levels of compression achieved by altering this threshold. The graphs in Figure 36 show the compression rates and error at different histogram thresholds. We measure the

Figure 33: Scans with regions segmented using the Chebyshev distance centered at the repeated architectural features.
Figure 34: Our framework for scan compression. The compressed scan has reduced all the windows above while maintaining the ornamentation and irregularities.

error by summing the Hausdorff distance between the translated representative region and the region it replaces.

3.3.3 Replacing Regions with Translations of the Representative

Once the regions which are sufficiently similar to the representative are identified they can be removed. Working just with this reduced set it is still possible to recreate the look and feel of the initial scan. We are able to fill the missing data by translating the representative region vertically by multiples of the dominant period.
Figure 35: The same scan with different levels of compression applied. The amount of compression is determined by changing the threshold which separates the histogram shape signatures, as described in Section 3.3.2. From left to right the compression rates are 57%, 47% and 13%, and the histogram difference thresholds, $\gamma$ are 2000, 700, and 500.

Figure 36: At left the compression rates achieved at various histogram thresholds. At right the cumulative error at various histogram thresholds. The error is measured by averaging the Hausdorff distance (in meters) between the original regions and the representative that replaces them.
3.4 Reconstruction

We have shown how to uncover regularity in real-time by performing Fourier analysis on each scanline. This information is now combined with the planarity in the scene allowing us to fill holes, generate meshes, plan views and classify points all in lock step with scene acquisition. Holes caused by occlusions, interiors and missing data are filled. Both triangular and abstract polygonal meshes are constructed from the point clouds. Lastly we introduce methods for sensor planning and point classification. This reconstruction method was first introduced in [Friedman and Stamos 2012].

Occlusions are a major problem in processing Lidar data. Even small objects located near the scanner can cast tremendous shadows on the scene. These shadows interfere with many algorithms by leaving large discontinuities in the input data. A prototypical example is illustrated in Figure 38.

Occlusions are not the only cause of holes in laser scans. Highly specular surfaces like car exteriors, and glass can deflect the laser preventing the scanner from measuring the
time of flight. Additionally if the laser is pointed at the sky or objects beyond its range no data can be gathered. Our algorithm recognizes all of these types of holes and fills them by leveraging the planarity and periodicity of the façades in the scene. The synthetic data generated for the reconstruction conforms to architectural patterns of windows and balconies and the 2D structure of the point cloud acquired by the scanner.

![Occlusions are common issue in range scanning.](image)

Armed with the current estimates of the major planes we are ready to approximate the missing and occluded data. If the distance between a point and the scanner is less than the distance between the scanner and the plane we label the point as casting a shadow. To approximate the data inside the hole we extend the ray from the scanner until it intersects with one of the major planes. This is a reasonable approximation because the laser travels in a straight line. The point $p_1$ of intersection is obtained by scaling the ray to the obstructing object $p_{\text{occlusion}}$ by $d$:

$$
d = \frac{(p_0 - l_0) \cdot n}{p_{\text{occlusion}} \cdot n}$$  \hspace{1cm} (25)

where $p_0$ is a point on the plane, $n$ is the normal to the plane, and $l_0$ is a point on the ray from the laser to $p_{\text{occlusion}}$. The origin of the coordinate system is the location of the
scanner. Therefore all rays from the scanner pass through the origin, and \( \mathbf{l}_0 \) can be set to the zero vector. The planar approximation is:

\[
\mathbf{p}_i = (d)(\mathbf{p}_{\text{oocclusion}})
\]  

(26)

This planar approximation of the missing data may be satisfactory for some applications. It maintains the 2.5D structure of the scan since each shadow point can be replaced with its ray’s intersection with the major plane. This structure simplifies many algorithms so maintaining it is beneficial for downstream processing, however, without further refinement to this approximation, we risk replacing a detailed ornate piece of architecture with a blank flat wall.

To do better justice to the occluded data we may consider the periodicity of the column in question. Using the period extracted in Equation 16 we may fill this hole by extending a periodic feature into the shadow. To find the appropriate filling, the façade is segmented by its period. This separates the different floors in the building, as can be seen in the alternating colors in Figure 39. Of the segments without any occlusions we select the highest density floor as the representative. For each point in the planar filling of the façade we associate the closest point of the representative floor translated vertically to align with the floor of the planar point. This method ensures that this periodic filling maintains the 2.5D structure of the scan. Façades filled in this manner can be seen in Figures 37, 45.
The floor on which a point $p_i$ lives is given by:

$$floor = \left( \frac{P_i \cdot v}{v \cdot v} \right) \mod \text{period} \tag{27}$$

where $v$ is the vertical vector. For each planar point $p_i$ we must associate a periodic point $p_{\text{periodic}}$ extracted from the representative region, $\mathcal{R}$. If $f_r$ is the floor of the representative region and $f_i$ is the floor of the planar point the associated periodic point is given by:

$$p_{\text{periodic}} = \min_{p_k \in \mathcal{R}} ||p_i - (p_k - (f_r - f_i)(\text{period})v)|| \tag{28}$$

This definition suggests an exhaustive search of the entire representative region, but because the column of $p_i$ is known, we can safely limit our search to this column and a few of the neighboring columns for efficient computation.

Once the holes caused by occlusions have been filled we turn our attention to façade holes caused by missing data. This missing data is likely the result of specular or translucent surfaces like windows. On those surfaces we may be unable to obtain a measurement or we may record a point from the interior. Either way these points are inconsistent and disrupt the continuity of the façade. We reconstruct the continuous façade so that the synthesized data conforms to the periodicity of the building and the structured 2D array of the point cloud. Specifically, for each scanline the increment angle $\theta$ between readings and the axis of rotation $a$ are determined. $\theta$ is found by taking a sample of points in the scanline computing the angle between them and dividing this angle by the absolute value of index difference
between the points. For example if we have points \( p_i \) and \( p_j \) at indices \( i \) and \( j \) respectively then:

\[
\theta = \cos^{-1} \left( \frac{p_i \cdot p_j}{\|p_i\| \|p_j\|} \right)
\]  

(29)

Averaging over a small sample of point pairs from the scanline is sufficient to get a robust estimation of the vertical angle the scanner rotates between subsequent readings. Similarly the normalized axis of rotation for a scanline is given by:

\[
a = \frac{p_i \times p_j}{\|p_i \times p_j\|}
\]  

(30)

For numerical stability it is wise to choose \( i \) and \( j \) that are not close neighbors in the scanline. Now if an interior point or missing data is detected in a scanline we may construct an appropriate replacement that will maintain the continuity of the façade. If our last reliable reading was at point \( p_k \) and point \( p_l \) is determined to be a discontinuity we can replace \( p_l \) by rotating \( p_k \) by \((k - l)\theta\) about the axis \( a \) to obtain \( p_c \).

\[
p_c = R_a((k - l)\theta)p_k
\]  

(31)

Finally, this vector is intersected with the façade plane using Equations 25 and 26 with \( p_c \) replacing \( p_{\text{occlusion}} \). Points obtained this way are colored orange in Figure 45.

We define a “mystery plane” as the planar polygon connecting two detected façades. We compute the normals and area of all the “mystery planes” in the scene. The normal
Figure 39: Different floors colored in alternating stripes of gray.

Figure 40: Occlusions caused by vegetation are filled.
Figure 41: Scan sampling density is maintained.

Figure 42: Façade reconstructed with the method of 3.4.

Figure 43: Façade reconstructed with the method of 3.4.
Figure 44: Synthetic shadows are generated and filled. We take the Hausdorff distance between the real data and the filling to estimate the accuracy of our approach.

is calculated by taking the cross product of two spanning vectors, which are the vectors forming the quadrilateral between the edges of the two known planes. Now we extend this normal by the average length of all the vectors in current scene minus their projections in the vertical direction. This extension of the normal indicates a sensible place for locating the scanner, as it will have an excellent view of the missing data. If we have many “mystery” planes we may sort by their area and return an ordered list of subsequent scanner locations.

Some algorithms require a mesh as input, and cannot directly process point clouds. Using the regularity extracted in Section 3.1.1 we can create a hole-free polygonal mesh in realtime with minimal processing requirements. The major planes are perforated with intrusions or extrusions occurring at the frequency detected by the Fourier analysis. Each
feature is classified as an intrusion or an extrusion depending on where the majority of its points lie in relation to the façade plane. The amount of extrusion or intrusion is determined as the mean perpendicular projection to the façade plane of the intruding or extruding points. We may enforce global properties of orthogonality, adjacency and parallelism between façades in the scene when appropriate as described in [Li et al. 2011a].

A triangular mesh can be generated online from a structured point cloud. Since the data is organized into a 2D array we can construct triangles by connecting each 3D point to their nearest neighbors in the 2D array. Normally this naïve approach will generate deplorable meshes because the connectivity in the array only roughly corresponds to proximity in 3D space. However, since we have removed all occlusions and discontinuities, this simple algorithm can be employed to generate acceptable triangular meshes in a single pass.
Figure 47: A coarse mesh constructed from the major planes and perforated with the periodic fenestration detected in 3.1.1

Figure 48: Several large meshes derived from point clouds using our method.
over the data.
4 Detection of Self-Similar Structures

Trees are notoriously difficult to process in 3D range scans. They are riddled with self-occlusions, making it nearly impossible to recover the entire structure from a single vantage point. Furthermore, the 2D connectivity grid, useful for finding surface normals and neighbors, is full of discontinuities in scans of trees, giving little indication of the actual morphology. Nonetheless, trees exhibit a salient and self-similar branching pattern and we propose an algorithm to find it.

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 thereby elucidate the self-similarity signature of the tree. These recursive relations are encoded as rules and parameters of a stochastic shape grammar.

The inferred grammar can be used for compressing, propagating, 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, and low-resolution scans can be enriched in real-time, rendering immersive and compelling
4.1 Dendromorphology – the Growth and Form of Trees

Trees are perennial plants with a main stem or trunk that supports many smaller branches and leaves. Trees are often woody plants with long life spans that grow to great heights over many years. This makes them a significant feature of many landscapes and so they are important objects to be able to model quickly and accurately in computer graphics.

Trees belong to two main plant groups: the gymnosperms and the angiosperms. Gymnosperms (literally naked seed in Greek) are plants whose seeds are exposed without an enclosing ovary. The conifer group of trees (including pines, firs and cypresses) is the most bountiful example of gymnosperm. Angiosperms produce seeds enclosed in a fruit. Unlike gymnosperms, angiosperms flower and tend to produce nutrient rich endosperm tissue within their seeds. Angiosperm plants dominate terrestrial plant life with the exception of the coniferous forests.

Trees exhibit a breathtaking range of forms depending on a wide array of influences, both environmental, and endogenous. Amazingly, these myriad structures can be accurately captured in just a small number of formal parameters and few recursive rules for generating growth. The single most important environmental factor influencing tree growth is light. The availability and direction of light help determine both growth
Important endogenous factors include the branching and budding angles, as well as the relative growth rates of the trees apex and its lateral branches. Specifically, **Apical control** describes the degree to which growth at a tree's apex, or the apices of its branches, inhibits the growth rate of lateral branches [Wilson 2000]. High apical control results in trees with sharp conical crowns typified by the fir and pines families. A similar though distinct concept, is **apical dominance**. Apical dominance describes the likelihood that lateral buds flush relative to flushing activity of other buds. Plants with less apical dominance create more branches while plants with higher apical dominance devote more resources to a few main branches.

Figure 50 illustrates the varying affects of both apical dominance and apical control on tree forms. The exact biochemical pathways responsible for these phenomena remain unknown, but the plant hormone auxin is known to be a significant chemical signal for the plant cell differentiation responsible for bud formation and wood growth [Cline and Harrington 2006].

Trees are not grown in a vacuum and their form is a dynamic response to a host of exogenous influences and affronts that the tree must overcome or absorb in order to survive. Tropisms are environmental forces that attract tree branches. The most significant force is phototropism, or the tendency of tree branches to grow parallel to the light direction vec-
Apical control regulates the growth rate of lateral branches of trees. High apical control suppresses the growth of lateral branches and allows trees to quickly grow in one direction to compete for light.

down arrow, \( L_d \). The trees in Figure 51 displays exaggerated phototropism.

This dynamic intelligent growth is also regulated by the plant hormone auxin, which is known to concentrate on the side of branches not exposed to the light. Auxin concentrate encourages the pluripotent meristemic cells on the branches exterior to differentiate into woody cells causing that side of the branch to grow faster than the side of the branch facing the light. This results in the whole branch turning towards the light.

Gravitropism is the tendency of branches to grow away from gravity and roots to grow towards gravity. Gravitropism can be counteracted by gravity bending factors leading to the weeping form of tree often associated with the willow. The gravitropic tendency of tree branches can be overwhelmed by the weight of child branches and gravity’s pull on them leading branches to bend towards the ground. This results in the weeping tree shape, as in Figure 54.
Figure 52: Conifers are a type of soft wood trees (gymnosperms), typified by pines and firs. They tend to have high apical control and large branching angles. A Conifer with high apical dominance at left and decreasing apical dominance in the middle and rightmost trees.

Other environmental factors that dictate tree shape include competition with neighboring trees for light, water, and nutrients, as well as the interference from humans, animals, and tree pathogens, like the Basidiomycota fungus responsible for root rot. Humans often prune low branches or branches near overhead power and communication wires. Buildings and neighboring trees can cast shadows over a tree distorting its light field and causing it to attempt to grow out of the shadow.

4.1.1 Tree Parameters

A tree is essentially the realization of a stochastic process. Furthermore, this process obeys the Markov Property, each new growth of the tree depends solely on-- grows out of we might
Figure 53: The palm tree is a type of angiosperm known for extremely high apical control that is time and height dependent. (a) and (b) show weeping palm trees. (c) A palm tree with large branch angle mean and variance. (d) A palm tree with strong phototropism.

Figure 54: The weeping form of tree. The procedural system controls weeping with the $G_b$ parameter.
say– its current state. Amazingly, the vast array of geometric and environmental influences on tree morphology can be reduced to a relatively small set of parameters controlling this random process.

In each tree generation $L_b$ lateral buds are placed along the growing shoot. Each bud is endowed with a heading given by normally distributed random variable whose mean and variance are given by the branch angle parameters, $\sim \mathcal{N}(\beta_\mu, \beta_{av})$. The buds placement along the shoot is determined by the bud roll random variables, $\beta r_\mu, \beta r_v$, which are also normally distributed: $\sim \mathcal{N}(\beta r_\mu, \beta r_v)$. The apex of the shoot deviates from its heading according to the apical angle variance, which is the spherical angle given by:

$$
\theta \sim \mathcal{N}(0, \alpha_{av}) \\
\phi \sim \mathcal{U}(0, 2\pi)
$$

(32)

(33)

We simulate tree growth with separate budding and flushing processes to capture the interaction of the plant hormone auxin on tree development throughout the growing season. Following the premise established in [Cline and Harrington 2006], we separately implement apical dominance and apical control.

**Apical dominance** is given by a real number between 0 and 1 representing the probability that a given lateral bud will not flush. Higher apical dominance results in the growth of the apices of the trees at the expense of the lateral branches. This probability decreases with age as older trees try to spread to maximize their crown’s light exposure. The age
dependence of apical dominance is encoded in the $\alpha_{da}$ parameter.

The probability of a lateral bud flushing is also dependent on the number of neighboring buds that have already flushed. This is because each of these buds sends chemical signals which inhibit growth so that branches do not compete too fiercely with their sibling branches. This also helps minimize collisions or anastomosis of the branches (converging after diverging), which is uncommon in real trees.

The chemical signals (again involving the plant hormone Auxin) from a flushing bud are not uniformly distributed in the tree. Instead, they are concentrated near the growing bud and diffuse along a chemical gradient throughout its vascular system. In this way, neighboring bud growth is inhibited, while distant branches, which will not compete with the growing one, are unaffected. This property of apical dominance is captured in the $\alpha_{dd}$ parameter.

Here $\Lambda(b_i)$ is the set of all buds above $b_i$, $d(b_i, b_j)$ is the branch-wise distance between $b_i$ and $b_j$, and $I(b_i)$ is the illumination at the bud, $b_i$ calculated using the method of [Pirk et al. 2012].

**Apical control** determines the **growth rate** of lateral branches in relation to apical ones. High apical control encourages growth along the trees apex, resulting in a large trunk and a conical crown. Like apical dominance apical control decreases with age. When growth begins apical control may be high, allowing a tree to shoot up quickly and vertically. As the tree ages the benefit of height gives way to the desire for light. Reducing apical control in older branches allows small twigs and leaves to maximize their exposure by growing in all directions– both laterally and along the apex. The time dependence of apical
control is encoded in the $\alpha_{ca}$ parameter. $G_r(t+1)$ is the growth rate after $t$ generations, given by:

$$G_r(t+1) = \frac{G_r(t)}{\alpha_c t - Apex_d(b)}$$

$$\alpha_c(t+1) = \alpha_c(t)\alpha_{ca}$$

$Apex_d(b)$ measures the number of branch nodes between the root and the apex from which $b$ diverged. At the tip of the trees trunk, $Apex_d(tip) = t$, and in general $Apex_d(b) \leq t$ ensuring that the growth rate will decay or remain stable.

The last geometric parameter in the system is the internode length, $I_l$. This is essentially a scaling parameter which determines the distance between consecutive buds on a growing shoot.

Complementing the endogenous, geometric parameters are exogenous, environmental ones, whose influence can be profound. Chief among the environmental factors affecting tree morphology are the various tropisms. Tropisms exert a pull on tree branches in a given direction. **Gravitropism**, $G_t$, the force in the vertical direction, is generally negative in tree branches and positive in tree roots. While branches must oppose gravity, they cannot overcome it. Horizontal branches, and branches supporting a masses of child branches with oblique centroids are especially susceptible to gravity bending, $G_b$. The gravity bending parameter can be used to create the weeping tree forms.

**Phototropism**, the trees’ attraction to the light, is described by the weight $P_t$, and the
light direction vector $L_d$. In general, the Tropisms are applied to a branch with a normalized heading $h_{\text{old}}$ resulting in an altered heading, $h_{\text{new}}$ by:

$$h_{\text{new}} = (1 - w_{\text{total}})h_{\text{old}} + w_{\text{total}} \frac{\sum w_it_i}{\sum w_i} = (1 - w_{\text{total}})h_{\text{old}} + w_{\text{total}}t_{\text{all}}$$

where $t_{\text{all}}$ is the weighted combination of all the tropism vectors, $\sum \frac{w_it_i}{w_{\text{total}}}$.

4.1.2 Foliage

Phyllotaxy describes the distribution of leaves along a stem or a terminal branch. A branch with whorled phyllotaxis has leaves forming a sort of corona around the plant stem. Other common leaf arrangements are plants with spiral or opposing orientations about the stem.
We populate the terminal branches with leaves according to the leaf parameters, $L_d$ and $L_s$. $L_d$ describes the distribution of leaves around a stem,

$$L_d \sim [\text{whorled}, \text{spiral}, \text{opposite}]$$

$L_s$ is a polygon set (and optional texture) representing an individual leaf shape.

### 4.2 Automatic Inference of Procedural Models for Trees

Procedural models, like the one we presented in Section 4.1.1, do a wonderful job of expressing a wide range of tree forms. Unfortunately, these models are notoriously difficult to control. The parameter space is neither continuous nor uniform so small changes in parameters can lead to drastic changes in tree form. Equally distressing to tree modelers, some parameter changes can render other parameters useless making the final structure
Figure 58: More examples of foliage, and the variety of tree forms generated with our system.

seem totally uncontrollable. For these reasons, it is desirable to have an automatic scheme to generate parameters from an input tree. This method was first presented in [Friedman and Stamos 2013a] and is expanded and developed here.

4.2.1 Vegetation Extraction

A number of tree point classification and extraction methods for 3D images have been presented in the literature. In the work of [Livny et al. 2010] 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 for classifying vegetation points using Markov Random fields is presented in [Hadjilias and Stamos 2010].

Figure 59: Textures are mapped to the generalized cylinders of the branch structure graph. We include representative bark and leaf textures in the parameter set.
4.2.2 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, mitigates this bias. Specifically, if $r$ is the root of the tree and $s$ is the scanner position. This symmetrization technique produces a tree that appears full and complete. Unfortunately, when viewed from the plane of reflection, this approach gives a rigid symmetry. This rigidity can be overcome by symmetrizing the point cloud first, and then applying our algorithm.
4.2.3 Parameter Estimation from Hierarchical Decomposition

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. This stems from the dyadic tessellation of the time-frequency plane, which makes the relationships between different levels of detail explicit. We construct a pixel grammar by mapping a pixel in one dyad to the eight pixels in the subsequent dyad. We iterate across each level of detail generating rules that map pixels from coarse details to finer ones. See Figure 63 for a graphical depiction of this process. Applying the pixel 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.

We derive a super-resolution version of the input by augmenting the wavelet transform with synthetic wavelets derived by the pixel grammar. The reverse wavelet transform on the augmented set of wavelets yields and 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 pixel grammar. In this way, the super-resolution image lessens these distractions and leaves us with a richer, fuller picture, as shown in Figures 64 and 65.

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
species of vegetation, as well as algae, and other organisms [Prusinkiewicz and Lindenmayer 1991]. 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 Figure 63 mapped one pixel to eight pixels, 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. 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 62. 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.

The quantization errors associated with the voxelization of the point cloud can be avoided by comparing consecutive levels of a KD-tree. Comparing levels of the KD-tree to their neighbors we derive a set of branching and rolling angles. The variance and means of these sets are used to estimate the branching parameters $\beta_{\mu}$, $\beta_{av}$, the roll angle parameters $\beta r_{\mu}$, $\beta r_{v}$, and the apical angle variance $\alpha_{av}$. We determine apical control and dominance parameters by finding the best fitting (in the least squares sense) cone or hemisphere to the convex hull of the point cloud [Lukács et al. 1998].
4.2.4 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. This is because our code currently uses a brute force linear search to find nearest neighbors. 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 $128^3$.

4.3 Applications of Self Similarity Detection

The recursive rules responsible for the growth and form of vegetation can be extremely useful in a number of vision and graphics challenges. Because the rules, and the parameters controlling them, are much more concise than the final structures they exhibit on the street,
this self similarity knowledge can be used for compressing the vegetation point clouds, as we demonstrate in Section 4.3.3. Since the grammar extracted is a set of instructions for growing vegetation, trees can be enhanced to levels of detail not actually recorded by the scanner, or present in the scene. Section 4.3.1 illustrates this application. Lastly since the derived grammar is stochastic, multiple applications of it with different random number generator seeds will result in similar, but different structures. This can be useful if only one tree has been captured by the camera, but many trees of the same type are desired to be displayed. We discuss this application in Section 4.3.2.

4.3.1 Enhancement

Super-resolution reconstruction or enhancement is possible with our system. We derive a super-resolution version of the input by augmenting the wavelet transform with synthetic wavelets derived by the pixel grammar. The inverse 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.

4.3.2 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
Figure 64: Four growth generations of a tree grammar. From the spare simple form given as an axiom we develop a rich and intricate details. Keep in mind, the four growth generations pictured here do not signify 4 single years, in this example the developments has been exaggerated by increasing the growth rate parameter.

Figure 65: Four subsequent generations of tree enhancement on conifer type structure.
The ability to instantaneously generate tree forms is a welcome addition to any 3D modeling software package. We developed a graphical user interface for editing the grammar parameters themselves in real time. We can initialize the parameter set from our automatic inference in Section 4.2.3. The user is free to tweak the parameters to suit their needs, or blithely generate new tree models until the desired form is found.

While computer vision practitioners strive to make fully automatic systems, a fully automatic solution is often utterly useless in computer graphics, because applications almost always require models to be touched-up or checked by hand. The collaboration between the algorithm and the user yields workable tree models quicker than either the person or the machine could manage on their own.
Figure 68: More variations—three texture-mapped trees derived from the same grammar seeded with different random numbers.

Figure 69: Iterations on a palm like form.
<table>
<thead>
<tr>
<th>Species</th>
<th>Lossless</th>
<th>Lossy</th>
<th>[Livny et al. 2011]</th>
<th>Procedural Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bischofia Polycarpa</td>
<td>64%</td>
<td>94%</td>
<td>92.5%</td>
<td>99.5%</td>
</tr>
<tr>
<td>Delonix</td>
<td>68%</td>
<td>96%</td>
<td>93.5%</td>
<td>99.9%</td>
</tr>
<tr>
<td>Ficus Virens</td>
<td>68%</td>
<td>92%</td>
<td>98.4%</td>
<td>99.8%</td>
</tr>
<tr>
<td>Lagerstroemia</td>
<td>69%</td>
<td>95%</td>
<td>97.5%</td>
<td>99.9%</td>
</tr>
<tr>
<td>Mahogany</td>
<td>73%</td>
<td>95%</td>
<td>98.5%</td>
<td>99.9%</td>
</tr>
<tr>
<td>Palm</td>
<td>80%</td>
<td>95%</td>
<td>99.4%</td>
<td>99.2%</td>
</tr>
<tr>
<td>Pine</td>
<td>81%</td>
<td>96%</td>
<td>99.1%</td>
<td>99.4%</td>
</tr>
<tr>
<td>Terminalia</td>
<td>75%</td>
<td>95%</td>
<td>95.7%</td>
<td>99.9%</td>
</tr>
<tr>
<td>Willow</td>
<td>69%</td>
<td>95%</td>
<td>98.5%</td>
<td>99.9%</td>
</tr>
</tbody>
</table>

Table 4: Compression results on different tree species. 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 $\gamma$. In our experiments we set $\gamma = 0.2$, as we found this threshold yielded significant compression without introducing many artifacts. The compression results for [Livny et al. 2011] are computed from a representative tree of each species. The compression results for our procedural model assume there is no obstacle geometry supplied.

4.3.3 Compression

Wavelets are well known for their ability to compress images [Daubechies et al. 1992]. Both lossy and lossless compression were employed on our range images. When using lossy compression the user may set a threshold, $\gamma$ 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%.
5 Conclusion

We have shown an online algorithm for detecting regular features and we have exploited this regularity to compress and register urban range scans. The meso-level information of the regular architectural fenestration provides tremendous insight into the scan. It will be applicable to a host of other applications in urban scene processing.

We demonstrated that the regularity of the urban scene can be used to fill holes in point clouds and generate abstract meshes. Our algorithm works online allowing for near simultaneous processing and acquisition of 3D scenes. We foresee a slew of algorithms of this ilk, creating intelligent 3D cameras which generate excellent data out of the box.

One field ripe for research is integrating online algorithms of the kind described here into the hardware of the laser scanner. By processing the scan as it is acquired the offline computational resources are freed to pursue higher level recognition tasks.

Compression in particular could be helpful in an onboard application. A robot navigating a complex scene may simply not have the resources to store an entire point cloud of an urban scene. Integrating compression into the act of processing circumvents this need.

Another area of future work will employ the registration algorithm presented here to allow simultaneous realtime registering of scans from multiple scanners. This application could prove critical in range finding sensor networks. If registration can occur as the scans are recorded each sensor could leverage all the global information acquired as it becomes available. In this way each sensor could enjoy the speed and robustness that parallelism affords without sacrificing the global knowledge of a serial scan.
The method of multi resolution analysis presented here is especially well-suited to point clouds of trees because their 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. The concise procedural models for vegetation could also have a role in augmented reality applications where low-storage and immersive on-demand graphics are valuable computational assets.

We envision a vast database of stochastic parameters for tree models from different species, climates, and seasons. This data can be perused by 3D modelers and used as a benchmark for comparing different tree generation algorithms. In future work, we will allow the interactive procedural modeling program to be seeded with parameters inferred by the algorithm. Users will be able to further edit and manipulate the 3D mesh until they construct the tree they need.

It is often observed that computer graphics addresses the inverse problem of computer vision; computer vision is essentially reductive and analytic, whereas computer graphics is additive, synthetic. Computer vision attempts to break down a large amount of sensory input into a few simple decisions. Computer graphics, in contrast, hopes to render a rich and engaging visual display using as little data and computational power as possible. However, the fields are not as diametrically opposed as the observation implies. Rather, as our work in interactive modeling shows, the analysis done by computer vision algorithms can fuel the synthesis necessary in computer graphics.
References


