Polygons, Point-Clouds, and Voxels, a Comparison of High-Fidelity Terrain Representations

Scott Gebhardt
Eliezer Payzer
Leo Salemann
Lockheed Martin Simulation, Training & Support
Advanced Simulation Center (ASC)
13810 SE Eastgate Way, Suite 440
Bellevue, WA 98005
425-957-3262, 425-957-3246, 425-957-3209

Alan Fettinger
Lockheed Martin Missiles & Fire Control
12395 North Mead Way
Littleton, CO 80125
720-344-1037

Eduard Rotenberg
Lockheed Martin Simulation, Training & Support
Advanced Simulation Center (ASC)
35 Corporate Drive, Suite 250
Burlington, MA 01803
781-505-9528

Christopher Seher
Lockheed Martin Simulation, Training & Support
Virtual Worlds Labs
1140 Kildaire Farm Rd. Suite 200
Cary, NC 27511
919-469-9950

scott.gebhardt@lmco.com, eliezer.payzer@lmco.com, leo.salemann@lmco.com, alan.fettinger@lmco.com, eduard.rotenberg@lmco.com, chris.seher@lmco.com,

Keywords:
point clouds, voxels, polygons, storage, modeling, terrain

ABSTRACT: *Polygonal representations have proven to be an efficient means of representing terrain environments at current levels of fidelity. Regular Triangulated Networks (RTNs) are a computationally-efficient means of capturing all posts in an elevation raster, while Triangulated Irregular Networks (TINs) can provide a more natural terrain appearance with fewer polygons. Both of these representations have done well at modeling terrain from sensor data with 1-meter or larger ground separation distance, but what happens when such data are available at decimeter, centimeter, or even sub-centimeter resolution? Is there a "cross-over" point where non-polygonal representations can capture 3D gridded earth measurements with less storage, and/or in a manner conducive to faster queries or rendering? This paper will compare the effectiveness of polygons, point-clouds, and voxels at representing ultra-high resolution terrain environments. We will compare the storage footprint of a high-resolution terrain data set in point-cloud, polygonal, voxel-grid, sparse voxel and octree voxel forms. modeling and simulation.*

1 Introduction

Polygonal modeling has been the gold standard in 3D modeling and terrain representation for over twenty years. Its development heralded a level of fidelity and realism in computer graphics that was not previously possible in the representation of three dimensional objects. Polygonal modeling has enjoyed wide-spread popularity because of its visual realism and its efficiency in data storage and object rendering.

However, with newer sensor technology that may begin to change. The fidelity and availability of sensor data inputs used for modeling 3D terrain objects has increased several fold over the past decade. At the same time the corresponding cost of such inputs has decreased. This increased data fidelity has exposed some inherent limitations in polygonal modeling, and may present a very real challenge to the supremacy of this technique. The sub-meter and even sub-centimeter level of fidelity that is increasingly common in remotely sensed terrain data presents unique opportunities and challenges that traditional polygonal modeling techniques may be ill-suited to take advantage of.

While modeling terrain at one meter sample spacing may work well using polygonal modeling techniques, what happens when that post spacing is available at decimeter level? What about centimeter or even sub-centimeter level? At what point are so many polygons required to accurately model high-fidelity data that the traditional polygonal model falls apart due to the increased demands of file size and rendering?

In this paper we will look for a “cross-over” point in terrain modeling where the storage and rendering efficiencies that have been the hallmark of polygonal modeling no longer hold true. Furthermore we will argue that the use of voxels – a volumetric data model – is a viable technique that can pick up where the efficiency of polygonal modeling reaches the point of diminishing returns. The main advantages of voxels are that like most sensor data, they are raster based, and unlike height maps, they are truly 3D. In other words, voxels can model overhangs, caves, and porous structures – none of which can be modeled with 2.5 D height maps, and all of which present unique challenges to automated polygonal modeling techniques.

We will examine a number of common terrain data models in making this argument. Using a point cloud data set as a control, we will generate a number of different types of terrain data sets, comparing them based on the metric of file size. On the polygonal side we will examine a number of methods for terrain generation. We will then examine a number of volumetric representations of the same data, including gridded voxel, sparse voxel, and octree voxel.

Based on our research, we will present clear evidence that there are inherent advantages in using a volumetric data model to represent high fidelity terrain data sets. These benefits will be quantified in terms of file storage size on disk.

2 Background

The history of computer graphics is still rather young and still being written. We will not attempt to provide here a broad survey of the existing technology, but rather a brief explanation of the components which concern our research. We will start with a brief discussion of terminology, and then provide a description of the terrain models used for this study. It should be noted that while voxel technology is still somewhat of a rarity within the modeling and simulation community, it has been successfully employed within the medical imaging discipline for decades.

2.1 Terminology

Three common terrain representations of the earth’s surface are DEM, DSM, and point clouds. These three products differ in specific ways. DEM (Digital Elevation Model) and DSM (Digital Surface Model) both consist of a continuous surface of cells containing elevation values. A DEM describes only the “bare earth” elevation, while a DSM depicts the true intersection between surface features and the atmosphere, which may include above ground features such as buildings, bridges and vegetation, as shown in Figure 2.1 below.

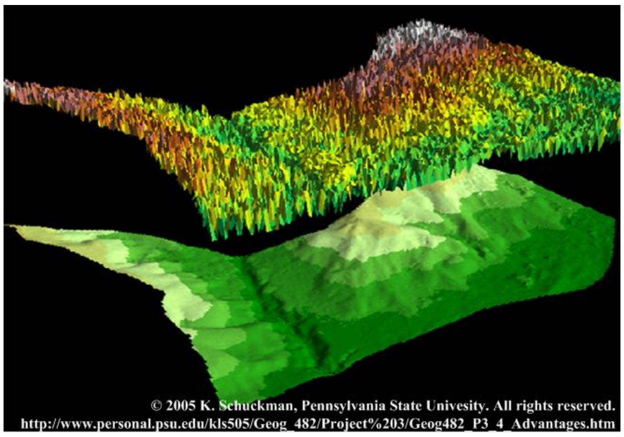
© 2005 K. Schuckman, Pennsylvania State Univesity. All rights reserved.
http://www.personal.psu.edu/kls505/Geog_482/Project%203/Geog482_P3_4_Advantages.htm

Figure 2.1 Digital Surface Model (DSM) above and Digital Elevation Model (DEM) below (Penn. State U., 2005).

A point cloud is a set of discrete points set within a 3 dimensional coordinate system. Unlike DEMMs and DSMs, which are properly referred to as 2.5D, point cloud can is true 3D. This is due to the fact that while a DEM and DSM can only contain a single elevation value at any specific XY location, a point cloud can contain any number of elevation values for any XY location, as shown in Figure 2.2. Multiple-return LiDAR is a perfect example of this, where multiple elevation samples may exist for a single XY location, describing both the bare earth as well as other above-surface features, such as a vegetation canopy or power line.

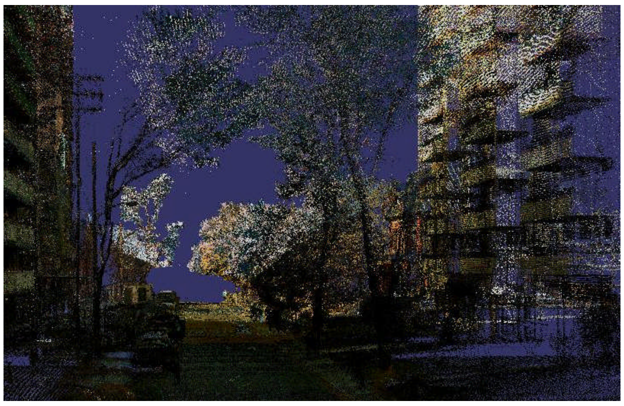
Figure 2.2 Point Cloud (Wright State Data Set)

The term "LiDAR data" is somewhat ambiguous as elevation data derived from LiDAR may come in any number of formats. For the purposes of this paper, when speaking of data products derived from LiDAR it is useful to distinguish between a LiDAR DEM (a collection of LiDAR elevation values expressed as a 2D surface of cells) and a LiDAR point cloud (a collection of LiDAR values expressed as discrete samples in 3D space).

2.2 History of Polygonal Modeling

The origins of polygonal modeling depend somewhat on who is telling the story. The technology has a number of origins and identifying a single "first source" is tricky. Generally speaking, the technique dates to the 1970's with the work of individuals such as David Evans, Ivan Sutherland, and James Blinn [1]. Among the first widespread commercial uses of the technology was a video game entitled The Sentinel, which debuted in 1986.

2.3 Triangulated Irregular Networks (TINs)

The TIN model first appeared in the 1970's as a simple way to build a surface from a set of irregularly spaced points. In the 1980's, commercial systems began using the TIN model, often as part of a Geographic Information System. TIN offers some advantages over a regular triangular network (RTN) in that it can present a more realistic representation of the earth surface, often with fewer sample points.

2.4 History of Point-Clouds

Point clouds are typically the product of a laser range scanner. In the arena of terrain generation this is most often done via LiDAR data collection, wherein thousands or even millions of discrete 3D points are collected as laser signal returns. Typically, each laser signal comes back as multiple returns, allowing the system to not only measure the surface of the physical environment but to penetrate porous objects such as tree canopies and measure what lies beneath.

Laser pulses were first used for atmospheric studies in the early 1960's. The first real use of LiDAR for the accurate survey of elevation values began in the late 1970's, but was of limited utility due to the fact that elevation values could only obtained for surfaces directly under the path of an aircraft. With the development of precise positioning instruments and inertial navigation systems in the 1980's and 1990's LiDAR truly came into its own as a highly precise and accurate means of collecting elevation samples over a wide geographic area [3].

2.5 History of Voxels

Voxels are the 3D analogue to pixels. Pixels (PICture ELements) are a way of dividing 2D space (e.g. an image or a height map) into uniform cells, typically squares. Similarly, voxels (VOLumetric PIXels, or VOLume ELements), divide 3D space into uniform 3D cells, typically cubes. Voxels share a desirable quality with pixels in that the location of a voxel is not explicitly stored as a set of XYZ coordinates, but rather is determined by its relative position to other voxels and the coordinate origin of the data set. This is very different from a Polygonal TIN, where the coordinate location of every triangle corner must be explicitly stored. The uniform size of both voxels and cell based pixels make this kind of efficient spatial referencing possible [4].

The use of voxels has its origins in medical imaging. In 1973 x-ray based computerized tomography (the CT scan) was introduced by Godfrey Hounsfield. This was among the first uses of volumetric modeling in medical imaging. Mr. Hounsfield received the Nobel Prize for Physiology in Medicine in 1979 (along with Allan Cormack) for his pioneering work in computerized tomography [5].

Voxel-based terrain rendering first entered the commercial video game market in 1992 with the introduction of Comanche: Maximum Overkill. This was the first commercial flight simulation game to apply voxel technology in this way, and provided a much more realistic and detailed terrain representation than what was possible with the vector graphics of the time.

A variety of voxel engines have been developed since these early applications and the state of the art continues to advance. Three of the more common voxel models are described below.

2.6 Gridded Voxel Models

Gridded voxel models treat a volumetric data set as a stack of individual voxel "slices" with each voxel slice, or grid, being one voxel deep. Processing and querying of this voxel data model can be performed slice by slice until all slices have been processed. This model introduces some efficiency in processing as only one slice is processed at a time. It does not, however, allow for the compression or aggregation of similar voxels into less dense representations, as other voxel data models do. Each and every location within the XYZ dimensions of the terrain will contain a voxel, including areas of no data, as shown in Figure 2.3 below.

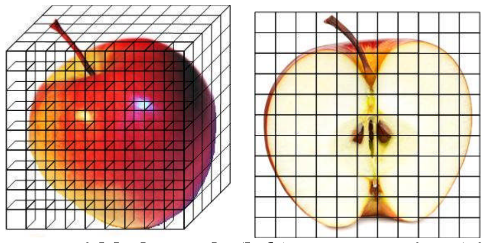
Figure 2.3 Gridded Voxels (left), cross-section (right).

2.7 Sparse Voxel Models

Sparse voxel models store only those voxels which actually contain information. Consider a voxel model in which each cell stores four values: red, green, blue, and opacity. Suppose we wish to represent an apple. A gridded voxel model would take the form of a cube of sufficient size to enclose the apple. Every voxel in the cube would contain a color/transparency value, even if it were (0,0,0,0), representing "air" or "vacuum." It is important to note that all voxels require the same amount of storage, whether they encode "air" or some portion of the apple. In contrast, a sparse voxel model would actually be shaped like an apple. The "air" voxels simply would not exist, as shown in Figure 2.4. The savings are even more dramatic for hollow and porous structures. If we were to pretend the inner flesh of the apple did not exist, we could save a significant amount of voxels by only modeling the skin, stem, and core.

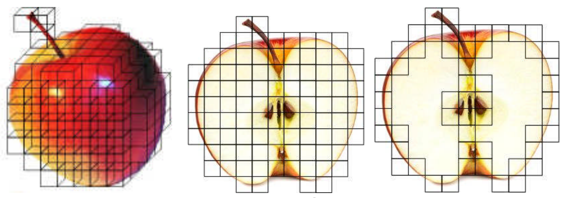
Figure 2.4 Sparse Voxels (left), cross-section (center) and "hollow" apple (right)

2.8 Octree Voxel Models

Octree voxel models consist of a data structure in which each internal node has up to eight children. It is essentially a way of compressing or partitioning volumetric space by combining collections of identical and adjacent voxels into larger aggregates [6]. Doing so allows for greater data compression, especially in areas that are homogenous or areas where no data is present. When modeling terrain, "empty" voxels can often account for over 95% of the voxels in a data set, so the space savings can be substantial. In our apple example, we would have an enclosing cube like the gridded voxel model, but much of the surrounding "air" would be modeled as large uniform blocks. Similarly, suppose that much of the interior of the apple is of uniform color. In both the grid and sparse voxel models, the interior of the apple would be filled with adjacent voxels of uniform size, all sharing the same color value. In an octree voxel model, this same information is captured in smaller numbers of larger Voxels.

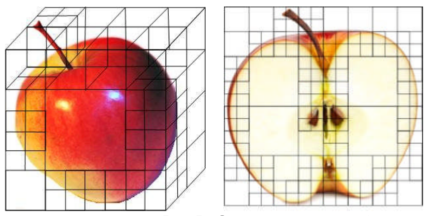
Figure 2.5 Octree Voxels (left), cross-section (right)

3 Data Source

The data source for this study is the Ohio Wright Center for Data's Wright State 100 Point Cloud, a 2x3 kilometer model of Ontario, Canada. The data is divided into 403 100x100 meter tiles, with a typical point spacing of 10 centimeters. For this study, we focused on three samples of this data: Tile 1, Tile 7 (shown in Figure 3.1), and Tiles 1-9, as shown in Figure 3.2.

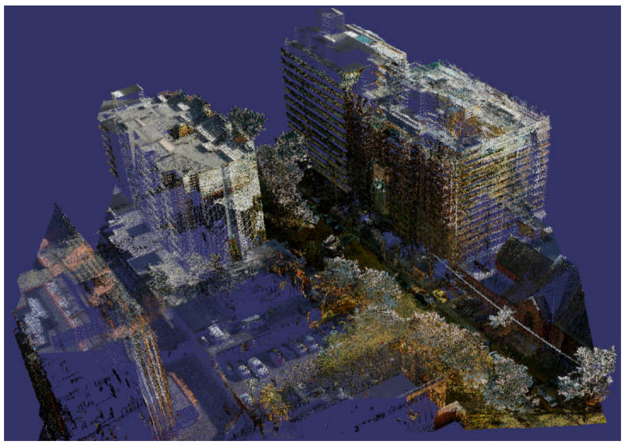
Figure 3.1 Wright State 100 Point Cloud (Tile 7)

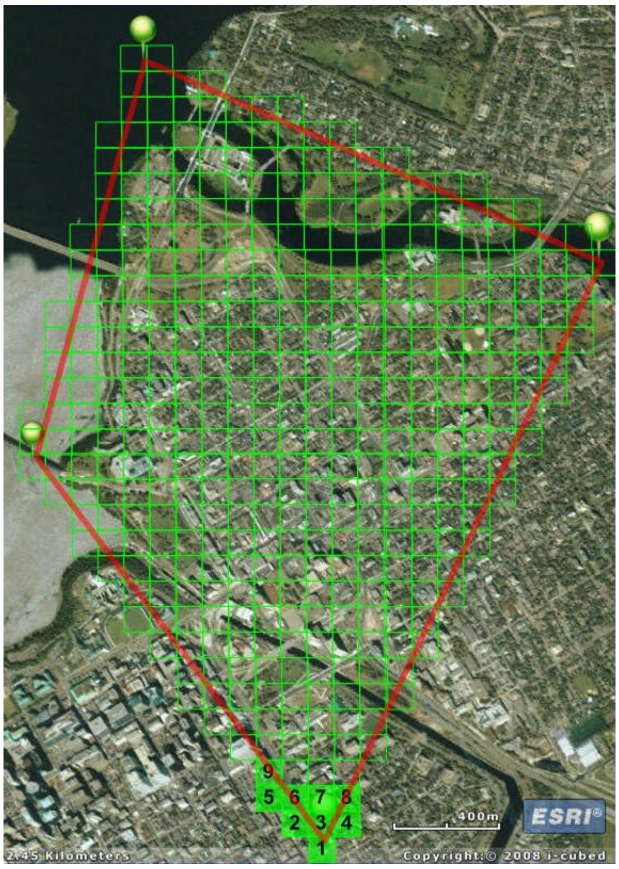

Figure 3.2 Wright State 100 Point Cloud Extent (red polygon) and Study Area (solid green tiles)

4 Procedure

The Wright State Point Cloud data was received in .upc format and subsequently converted to ASCII text files using the Extract utility provided with the upc data. The resulting files were of the form [time, X, Y, Z, R, G, B, laser #]. Laser # ranges from 0 to 4 and indicates which laser return a giving point is associated with. These converted ASCII files provided the input for each of the individual terrain products derived for this study. For each terrain product, Tile 1 was used as an initial test case due to its small size. Tile 7 was used as an intermediate test case, because it is the first tile in the data set that encompasses a full 100 by 100 meter area. Finally, tiles 1-9 were run in their entirety to get an upper bound. All available laser returns were used in each test case. A description of specific procedures for point clouds, polygons and various volumetric formats follows.

4.1 Point Clouds

The converted ASCII files contained a total of 32.4 million data points across all nine 100x100 meter tiles. The total storage size on disk for these files was 1.6 GB. The average sample spacing across all tiles is 10cm. A close up of the Tile 7 point cloud can be seen in Figure 3.1.

4.2 Polygonal Modeling, TINs

The goal of the polygonal modeling study was to produce a polygonal terrain representation using as many triangles as necessary to model the entire data set at the full spatial resolution present in the point cloud data. The task of converting the point cloud data to a polygonal model of irregular triangles (TIN) was not a simple one. A number of different approaches were taken, with varying degrees of success.

Two different COTS products were used in an attempt to create a triangulated terrain model from the Wright State point cloud data. The two software packages used, Geomagic and VRMeshStudio, are both products designed to ingest 3D laser-scan data and output 3D polygonal models. It is important to note that both tools were intended to ingest scans of relatively small and convex objects, such as tabletop sculptures or at most, full size vehicles. Neither tool was explicitly designed for the complexity and size inherent in full city LiDAR scans. Although initial progress was encouraging, our attempts at using the two tools to create a polygonal terrain from point clouds were inconclusive and incomplete. A number of major obstacles were encountered in this effort. Both tools in question were run on a 32bit machine which experienced severe system limitations (primarily memory-related) that caused the process to either derail and hang up completely, or create output that was incomplete and unusable. Considering the resource limitations that prohibited processing even a single tile to completion, one can imagine the difficulties that would be encountered if the density of sample points was doubled from the current resolution of 10cm. Finally, the interpolation parameters used during terrain generation can vary greatly, and developing the best approach for a given data set can be an iterative and time consuming process that does not lend itself well to automation. Though incomplete, single tile outputs yielded file sizes of around 55MB each (this is a figure for one tile worth of data, not all nine).

Ultimately, the Wright State points were successfully converted into a high-density polygonal terrain with MATLAB by using the 3D Delaunay triangulation function (Qhull algorithm). MATLAB could not build the entire nine tile data set into a single aggregated terrain data base. The performance requirements for doing so were too great, so the terrain was built as a series of individual terrain sets tile by tile. The cumulative storage requirements for the nine tiles were 779 MB, and contained a total of 2.45 million facets. A sample of the MATLAB polygonal output can be seen in Figure 4.1 below. As shown in Figure 4.1, the MATLAB process was allowed to run in a fully-automated fashion, with no manual post-processing or cleanup. As such, storage metrics derived from our results should be treated as an upper-bound for polygonal modeling, but not necessarily the results one would get when manually modeling an urban scene at this resolution.

Although building the polygonal terrain using MATLAB was successful, this technique also comes with some disadvantages that go beyond storage size. This technique was only successful when run on a 64bit machine. Build attempts on a 32bit machine consistently failed. Even with a more 64-bit machine terrain generation took multiple days to complete.

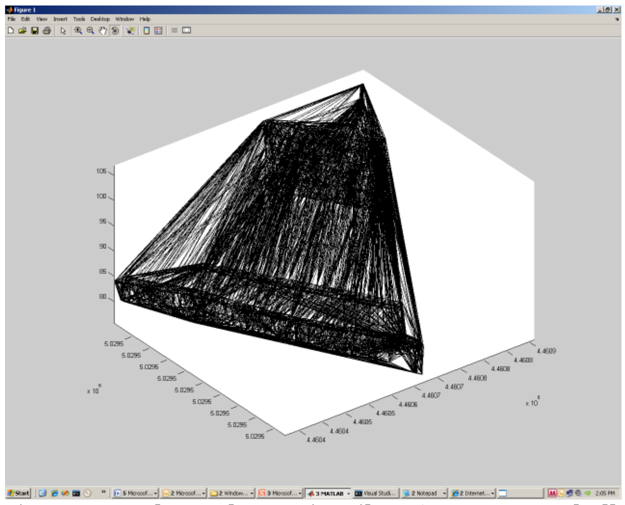
Figure 4.1 Polygonal Terrain Tile 1 (MATLAB, Qhull Delaunay triangulation function)

4.3 Voxel Grid

Creation of the gridded voxel terrain was performed using a software tool created in-house. This tool reads in ASCII files and outputs volumetric data in RAW format, which were inspected with the Drishti Volume Exploration and Presentation Tool [7]. Three RAW files where generated: Tile 1, Tile 7(Figure 4.3), and Tiles 1 through 9 together. The resulting gridded voxel terrain for all nine tiles was 7.83GB in size and contained approximately 8.4 billion voxels. Because this is a gridded data set, a voxel is stored for all locations regardless of whether the voxel represents solid material or empty space, which provides some explanation for the rather large file size.

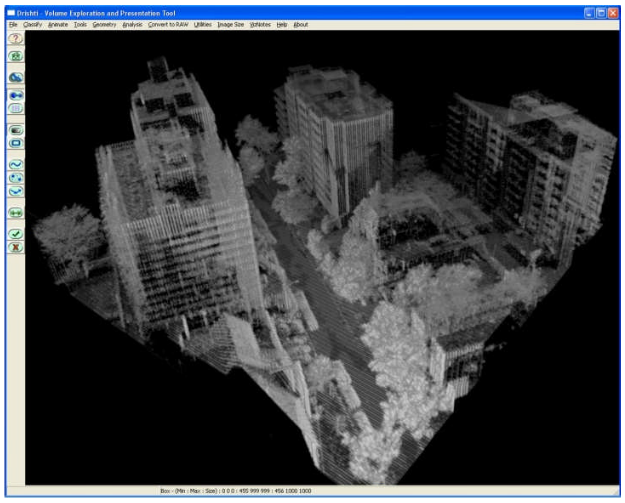
Figure 4.2 Gridded Voxel Terrain, Tile 7 (Drishti)

4.4 Sparse Voxel

For a representative sparse voxel format, we used NGW (NGRAIN World) format by NGRAIN®[8]. For each tile, Microsoft Office Access 2007 was used to gather the ASCII point data for all laser returns, and reformat it for consumption by the NGRAIN® Converter. This entailed dropping the time column, converting x, y, z into integer centimeters, and multiplying color values by 10 (and truncating to 255 where necessary) to increase visual intensity. The NGRAIN® PointDataToNGW.exe utility converted the point data into NGRAIN World (NGW) format, which consists of a sparse voxel model and a few xml files. Finally, the NGRAIN® Baseline Viewer was used to visually inspect each tile. An example appears in Figure 4.3 below.

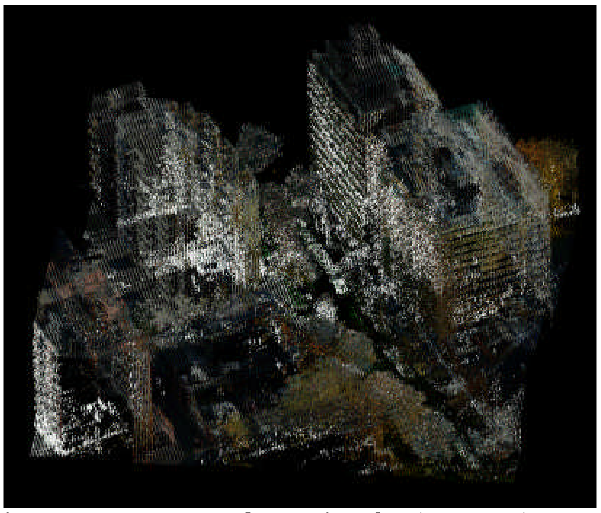
Figure 4.3 Sparse Voxel Terrain Tile 7(NGRAIN)

4.5 Octree Voxel

For the octree terrain, the ASCII point cloud data was converted to the Lockheed Martin Obstacle Representation Database (ORD) format, see Figure 4.4. The ORD model uses a hierarchical 3D cell-based data structure stored as an octree. Cells are only allocated for sensed regions of space, so the memory requirements are approximately proportional to the area of the model and grow with the inverse square of the model resolution. Moreover, regions of space that are homogeneous can be represented at a higher level in the tree to reduce the memory usage or to speed processing.

The octree is templated on a payload type, which is a data structure that is contained in each node of the tree. This payload is designed to be flexible, to contain any combination of payload components, such as color and opacity. ORD also supports efficient 3D queries (such as shape intersection and closest point) and other useful tools like height map generation. Programmers using ORD can define new payload components, as well as the methods for adding data to and for querying the new components.

The final voxel count for the nine tile aggregated ORD data set was 6.7 million voxels, with a storage size on disk of only 13MB.

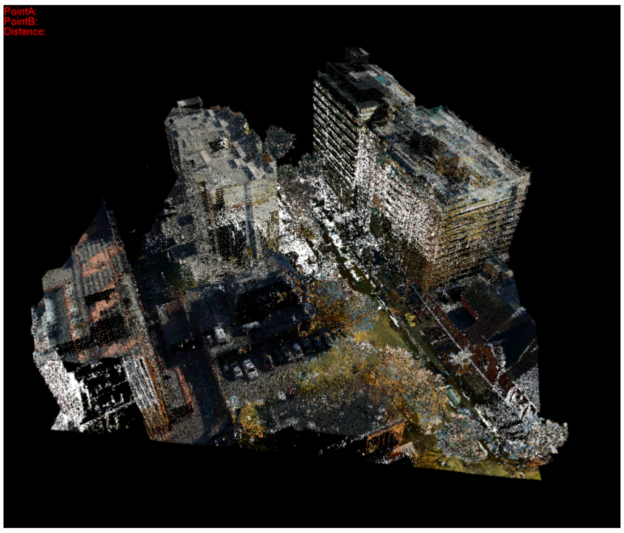

Figure 4.4 Octree Voxel Terrain (Lockheed Martin Obstacle Representation Database)

5 Results

The final storage footprint for each terrain model is summarized in Table 5.0. Metrics were gathered for Tile 1 (one of the smallest tiles), Tile 7 (the only tile to span a full 100 by 100 meters) and Tiles 1-9 (a wide area stress test).

Terrain Model	Tile 1 (MB)	Tile 7 (MB)	Tiles 1-9 (MB)
UPC Points	57	184	1,080
ASCII Points	82	267	1,610
Polygonal/TIN	39	126	779
Gridded Voxel	1,080	1,600	7,830
Sparse Voxel	0.634	3.0	13
Octree Voxel	1.1	6.8	13

Table 5.0 Summary of Results

As expected, the storage footprint of the gridded voxel model far exceeded any other format. Examining the data graphically suggests a few interesting trends as the input data set increases, as shown below in
Figure 5.1.

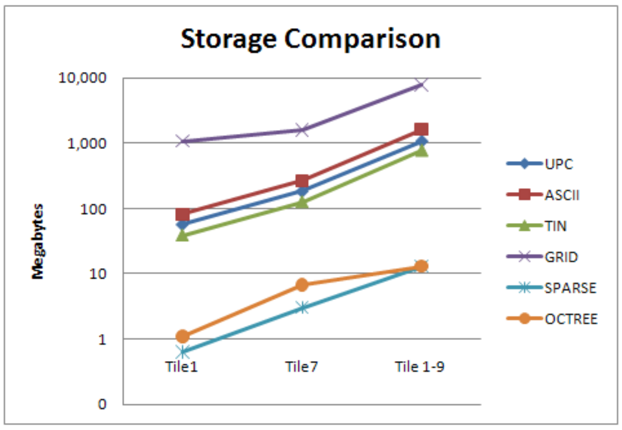

Figure 5.1 Storage Footprint Comparison.

The first thing to note is that a logarithmic scale was necessary to fit sparse and octree storage footprints on the same graph as the other formats. Although more data points would be necessary for a formal trend analysis, one can see that the storage footprint of our TIN representation seems to track with the original point data. This can be explained by our approach to use a fully automated method, whereby all points were triangulated in a brute-force manner. The data would need substantial post-processing to be usable in a typical visualization system, but it can be thought of as an upper bound. In practice, these values may have been lower and even followed a shallower growth curve, but would introduce manual re-work as a new variable. Could the storage footprints of the sparse and octree voxel models also be reduced by manual rework? Could the polygonal representation be lowered by two or three orders of magnitude? On the other hand, it should be noted that over 95% of the polygonal storage footprint comes from the vertices rather than the triangles. If your resolution or accuracy requirements are such that you cannot afford to drop a significant number of points from the original cloud, you could expect a storage footprint similar to our results.

Another potential trend is the relative performance of the sparse and octree models. As expected, the octree initially requires more storage. This can be explained by the need to model the entire cubic or rectilinear space, even if unoccupied or homogenous regions only require only a small number of large octree-voxels. However, as the terrain extent increases, the proportional overhead of the large octree-voxels decreases. Consider a "square" terrain model whose X- and Y-extent are the same. An octree voxel model would first attempt to model this space as a cube with the terrain at the bottom face, and to subdivide the cube into 8 leaf nodes, repeating until each leaf-node in the octree contains homogenous data. For larger terrain models, the Z-extent is proportionately small when compared to the X-Y extent. Thus, more of the cubic space can be modeled as a small number of large octree nodes. In the upper limit, for both octree and sparse voxel models, the voxel count and storage footprint should approach those of a 2D orthographic image of the same area.

Consider a hypothetical urban terrain model, where the highest buildings are 100 meters tall, and the scene is complex enough for us to pretend that the entire 100 meter tall 'slab' would need to be modeled as voxels. If you model a 100x100 meter terrain, your 'voxel-cube' will be fully populated. However, as the XY extent is increased, the percentage of solid voxels falls off significantly, to the point where a 100x100 kilometer terrain would only occupy a tenth of one percent of the enclosing cube.

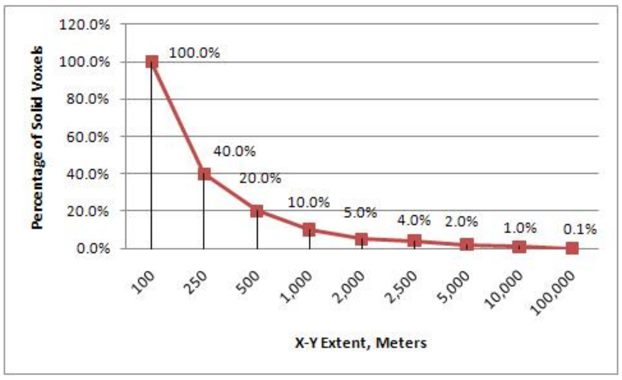

Figure 5.2 Relative Voxel Occupancy for a Hypothetical 100m Tall Urban Terrain

6 Conclusion

Looking at the results summarized in Table 5.0 it is clear that among the various polygonal and volumetric terrain data sets, octree and sparse voxel performed the best in regards to our primary metric of storage size on disk. While it is clear that sparse voxels outperform octrees for smaller/taller data sets, it is unknown how this trend plays out for larger XY extents.

Gridded voxels, while offering the advantages of a true volumetric data set, come with a large cost when it comes to file size. For most applications, 7.83 GB is probably not a reasonable amount of storage space to devote to an area no larger than 300 meters per side (our nine tile area). Again, with a gridded voxel data set every voxel location and value is stored, even if a voxel contains no data and is of little use analytically. This adds greatly to the storage cost of a gridded voxel terrain. The large footprint of this terrain format is measurable not only in disk size, but also in the number of voxels that are stored in the data set; at 8.4 billion the gridded voxel terrain stores more than 1,250 times the number of voxels that are stored in the octree voxel terrain for the same area.

In comparing the storage efficiency of the sparse and octree vs. gridded voxel model, we get a compression ratio of over 60,000 to 1. This is not surprising considering that more than 90% of the volume of space included in a typical 3D terrain data set derived from LiDAR may be simply empty space (atmosphere). This empty space is likely of little use in the majority of applications, and for those which require it; octrees present an elegant method of capturing low-resolution (atmosphere) and high-resolution (terrain) data in a single 3D space.

While a nine tile, terrain provides a useful study area with which to evaluate our different terrain data models, it is a bit small geographically to have broad utility in a modeling and simulation environment. If we assume our results for tile 7 are representative of all full tiles in the complete Wright State 100 Data set, we can estimate the storage required for the whole point cloud. Of the 403 Wright State tiles, 325 are 'complete,' or completely lie with the area of interest. The remaining 78 tiles are "partials' which straddle the AOI boundary as shown in Figure 3.2. Assuming these tiles are 50% "full" on average, we have equivalent of 325+39 = 364 full-tile equivalents. If the storage footprint of the sparse voxel model increases linearly, we would require 5.7GB to store the entire point cloud, which would contain over 297 million voxels.

Although the storage performance of our polygonal model is not representative of what can be achieved with careful interactive modeling, we believe it is a reasonable upper bound for what can be expected from automated techniques. Since vertices make up 95% of the polygon model's storage footprint, we believe that hand-modeled results would be within around 5% of our figures, if all vertices from the original point cloud are kept, to ensure maximum accuracy.

Although identifying an accurate cross-over point where the advantages of the voxel model exceed those of polygonal modeling would require more analysis performed at multiple scales with a variety of data inputs, we have been able to demonstrate that volumetric techniques can yield substantial space savings and are highly amenable to automation.

From the storage point of view, sparse and octree voxel models offer clear advantages to storing and modeling volumetric data. Accuracy, rendering, and modeling of larger-scale data sets are all topics worthy of further study.

References

[1] V.R. Auzenne: *"The Visualization Quest: A History of Computer Animation"*, Associated University Presses, 1994.

[2] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes: *"Computer Graphics: Principles and Practice"*, Addison-Wesley, 1990.

[3] T.M. Lillesand, R.W. Kiefer, J. W. Chipman: "*Remote Sensing and Image Interpretation*", Fifth Edition, Wiley, 2004.

[4] B. Lichtenbelt, R. Crane, S. Naqvi: "*Introduction to Volume Rendering*" (Hewlett-Packard Professional Books), Hewlett-Packard Company 1998.

[5] T. N. Raju: "*The Nobel chronicles. 1979: Allan MacLeod Cormack (b 1924); and Sir Godfrey Newbold Hounsfield (b 1919")*, Lancet, 1999.

[6] S.F. Frisken, R.N. Perry: "*Simple and Efficient Traversal Methods for Quadtrees and Octrees*", Mitsubishi Electric Research Laboratories, 2006.

[7] http://anusf.anu.edu.au/Vizlab/drishti/features.shtml

[8] www.ngrain.com

Author Biographies

SCOTT GEBHARDT is a Geospatial Analyst with over ten years of professional experience, as well as a Bachelors and Masters Degree in GIS and associated technologies. Scott has extensive experience working with a range of ESRI GIS products, as well as with satellite and photogrammetric image processing software. Scott has substantial experience managing spatial data and designing spatial databases, as well as developing GIS applications and automating GIS tasks in both desktop and web-based environments. Scott is well-versed in a broad spectrum of geospatial technologies and tradecraft, including remote sensing, environmental monitoring, vector processing, and server technologies.

LEO SALEMANN is the Software Engineering Manager for the LM STS ASC office in Bellevue, WA. Leo has been working for the LM STS Bellevue office since receiving his Bachelor's of Science in Computer Science & Engineering from the University of Washington in 1993. Leo specializes in geospatial technologies, urban modeling, and modeling and simulation for homeland defense and security. Leo helped develop various terrain database generation systems such as the SIMNET/S1000 Assembly Tool, the WARSIM Terrain Data Fusion System, (TDFS), and the Automated Building Generation System (ABGS). Leo has developed Environmental Data Models (EDMs, a form of geospatial ontology) for OneSAF, US-Army/TEC as well as NGA. In addition to software design and development, Leo has an extensive background in customer support and training. He has prepared and taught courses for 3 to 20 engineers, for customers including US Army Topographic Engineering Center (USA-TEC), Mitsubishi Precision Company, and the Swedish Defense Materiel Administration. Försvarets Materielverk (FMV). Leo now leads a team of research engineers in developing advanced technologies such as Dynamic Synthetic Environments.

EDUARD ROTENBERG graduated from St. Petersburg State University majoring in theoretical / math cybernetics. His PhD topic was devoted to multidimensional stabilization systems. Eduard has worked for multiple companies, including Bell Labs and Microsoft developing control and research system algorithms, simulations, visualizations, etc. He has 17 patents and few in a pipeline.

CHRISTOPHER SEHER graduated from The Art Institute of Pittsburgh with a Bachelor's in Game Art and Design. He worked in the field of computer gaming as an artist since 1997 and has been credited with working on an original PC strategy game and several ports of arcade games to home consoles. Christopher currently works as a lead artist at the Virtual World Labs division of Lockheed Martin and live in Raleigh, North Carolina.

ALAN FETTINGER is a Software Engineer at Lockheed Martin Autonomous Systems. He received his Master of Science in 2005 from the University of Illinois at Urbana-Champaign specializing in Computer Vision and Robotic Motion Planning. Alan has worked on various projects primarily involving modeling, tracking, and motion planning in 3D environments. He has worked with the Obstacle Representation Database (ORD) for over two years, and is proficient in advanced programming techniques necessary for the efficient storage and retrieval of 3D information.

ELIEZER PAYZER holds a Bachelor's in Computer Engineering and Computer Science from the University of Southern California. He has experience in computer graphics programming gained from working on school projects and interning with Lockheed Martin. He is currently pursuing a Masters in Computer Science, also from the University of Southern California.