Literature Synthesis on Dynamic Viewing of Large 3D Models

Timothy Trewartha
Supervisor: Hussein Suleman

May 14, 2012

Abstract

This literature synthesis examines existing solutions for real-time viewing of large 3D models using multiresolution data-structures. Various hierarchical structures such as octrees, bounding sphere hierarchies and hierarchies of tetrahedra are discussed and examined. Experimental results indicate that using these techniques results in a significant speedup and enables us to view large models at interactive frame rates. The R-tree is also presented as an efficient way to index 3D data. Additional techniques such as visibility culling and out-of-core storage are also discussed.

Introduction

The Zamani project, started by the UCT Department of Geomatics, aims to preserve African cultural heritage by documenting heritage sites and producing laser scanned models. Some of the models are very detailed, containing over 8-billion points. Given this vast scale of data, traditional viewing methods and the current hardware and software systems are not able to cope. In particular, commonly used GIS systems such as ArcGIS cannot handle the volume of data. As a result, before viewing or manipulating the data, one must go through a process of decimating the original data by a factor of 10, 100 or more. This can be achieved with 3D point cloud processing software such as Leica Cyclone. However, this solution is inadequate in many cases as one may lose too much detail. Ideally, one would like to be able to view the model at a low-resolution initially, and have the software dynamically increase the resolution as one zooms in, up to the point where the full original detail is available. This level of detail is often necessary for cultural heritage sites in order to view details such as cracks and flaws, with a view to preserving the site and preventing damage.

This literature review examines existing solutions for dynamic viewing of 3D data. Common methods for structuring 3D data include octrees [Wand et al., 2007], R-trees [Zhu et al., 2007], bounding sphere hierarchies [Rusinkiewicz and Levoy, 2000], and Hilbert Space Filling Curves [Wang and Shan, 2005], each with their own advantages and disadvantages. For this project it will be important to review the literature on these methods and consider their implications for developing a more efficient real-time rendering system for the Zamani models.

Hierarchical Datastructures

As early as 2000, systems were being developed for handling models with hundreds of millions of polygons, perhaps the most notable of which is QSplat [Rusinkiewicz and Levoy, 2000]. This system made use of a bounding sphere hierarchy to achieve a significant speedup. For example, they were able to render models containing over 8 million points interactively using a multiresolution scheme. In order to gain an additional speedup they also implemented visibility culling (that is, discarding of primitives that are not visible from the current view point). Although the QSplat system met with a significant degree of success, the models considered are not as large as those scanned by the Zamani project. Most of their models were sourced from The Digital Michelangelo Project [Levoy et al., 2000] and contained only millions of points, not billions.

Around the same time, an alternative approach was suggested for rendering complex geometric objects. This is described in the paper 'Surfels: Surface Elements as Rendering Primitives' [Pfister et al., 2000]. Surfel is short for surface element; surfels are point primitives but without any explicit connectivity as is common in mesh based schemes. To enable interactive rendering the authors chose to use an octree as a multiresolution hierarchical structure. Although the results they achieve allow for interactive frame-rates, the sizes of the models are still small compared with some of the sites documented by the Zamani project. However, these approaches laid the basis for the work to follow.

Cigoni et al. present yet another spatial partitioning multiresolution data structure in their paper [Cignoni et al., 2008]. The authors use a regular conformal hierarchy of tetrahedra to spatially partition the model. The resulting technique is fully adaptive and is able to retain all the original topological and geometrical detail, even for massive datasets. Additionally, it is not limited to meshes with a particular subdivision connectivity and is strongly GPU bound. This means that it is over one order of magnitude faster than

previously existing adaptive tessellation solutions since the data structure is able to exploit on-board caching, out-of-core representation and prefetching for efficient, real-time rendering.

In addition to viewing models, more recent research in the area of handling massive point clouds has also included the ability to edit and manipulate the data. There have been a number of papers on this topic, and while editing of the Zamani models is not explicitly required, the data structures presented are still important. Wand et. al. describe a new out-of-core multiresolution data structure for real-time visualization and interactive editing of large point clouds [Wand et al., 2007]. Their chosen data structure consists of a dynamic octree with a grid-quantization-based dynamic multiresolution representation in each inner node. The octree is able to easily handle dynamic operations due to its regular structure. Using this data structure the authors were able to achieve real-time walkthroughs and interactive modifications of a data set containing 2.2 billion points and totaling 63.5GB, although it took over 14 hours to build the data structure.

Gobetti and Marton also present a simple and efficient data structure for rendering of large point sampled models such as those that are common in GIS [Gobbetti and Marton, 2004]. The resulting system is capable of rendering over 60 million points/second. The multiresolution approach creates a hierarchy over the samples of the datasets, simply by reordering and clustering them into point clouds of approximately constant size arranged in a binary tree. It is thus possible to obtain the required density by accumulating point clouds as the hierarchy is traversed top-down. The root node is thus the coarsest available model.

A more mathematical approach to partitioning the 3D data is based on Hilbert Space Filling Curves. These can be used to partition the dataset, which can then be stored in a spatially indexed relational database [Wang and Shan, 2005]. In 3-Dimensions one can think of a Hilbert Curve as being a curve that passes through every point in the specified 3-Dimensional space. Although this seems counterintuitive, it is possible since the two spaces have the same cardinality. This provides a way to partition the space, as well as a mapping between a 1D space (for example the hard disk) and the 3D space in which the points are located. Additionally, Hilbert Curves have the advantage that points that are close together on the curve, are also close together in the 3D space. It is possible to consider other Space Filling Curves (there are many) but both mathematical analysis and practical applications suggest that the Hilbert curve has the best clustering ability and performance in data retrieval and response time [Faloutsos and Roseman, 1989]. In the paper by Wang J. and Shan J, this technique is described and used to gain better query performance on large 3D data consisting of millions of points [Wang and Shan, 2005]. It is not certain if this method will, however, scale to billions of points, as is required by the Zamani models.

Indexing 3D Data

As well as developing efficient level-of-detail hierarchies to allow for real-time rendering of large models, it is also important to consider the indexing of the 3D data. Although there are many different spatial indices such as kd-trees and cell-trees, they are not however well suited to 3D applications [Zhu et al., 2007]. Based on the idea of B-trees, Guttman presented the R-tree as a new way of indexing multi-dimensional information [Guttman, 1984]. Subsequent modifications to the R-tree include the R^+-tree, which allows one object to exist in multiple nodes [Sellis et al., 1987] and the R^*-tree, which has better clustering [Beckmann et al., 1990]. More recently, Zhu et al. have introduced a new spatial cluster grouping algorithm (k-means clustering) that uses 3D overlap and coverage volume as well as the minimum bounding box shape as the integrative grouping criteria

[Zhu et al., 2007]. Using these methods, the authors were able to gain significantly better performance when querying the spatial data.

There has also been some effort to find a way of integrating the level-of-detail hierarchies discussed in the previous section with different indexing methods. Of particular interest is the paper by Kofler that attempted to combine the R-tree with LOD (level-of-detail), and presented the LOD-R-tree method in which the level of the R-tree represents the required level-of-detail representation [Kofler et al., 2000]. Zlatnova also tried to find a similar way of uniting R-trees with LOD and put forward various grouping methods that take into account location, shape and altitude [Zlatanova, 2000]. It seems, however, that this area has not been investigated thoroughly and there is still more work to be done as noted by Zhu et al. [Zhu et al., 2007].

Out-of-core Storage

A common problem encountered when dealing with large amounts of data such as with these vast point clouds is the inability to store all the data locally in random access memory [Cignoni et al., 2008]. Consequently, all algorithms dealing with interactive rendering must take into account the location of the data that they are accessing, as access to data on hard-disk presents a significant bottleneck. Such algorithms are termed out-of-core algorithms, meaning that a portion of the working dataset must be stored on the hard-disk. The octree data structure previously discussed is fairly efficient even when some data is out-of-core [Wand et al., 2007]. Standard virtualisation techniques are applied to the given data structure. This works and is efficient because only those nodes that are needed for rendering must be loaded into main memory. Given a certain required level-of-detail, only a small number of the nodes needs to be accessed while a large fraction of the data structure remains unused. The authors also built in two methods to support specific out-of-core operations: *fetch* and *access*. *Fetch* indicates that a node is to be used in the near future and should hence be moved to main memory (this is handled by a separate thread to hide disk-access latencies). *Access* asserts that the data is already in main memory and hence readily accessible. The authors opted for a Least-Recently-Used policy to swap out unused nodes to disk when the memory cache is full. Determining the correct block size is also an important parameter for gaining optimal efficiency.

Visibility Culling

Visibility culling is another important technique for achieving real-time rendering of large data sets. Visibility culling refers to the fast exclusion of portions of the data that are not visible from the current point of view. Greene et. al. describe a general algorithm to discard primitives that are blocked by closer geometry using a hierarchical Z-buffer [Greene et al., 1993]. This is a type of visibility culling known as occlusion culling. The methods presented in this paper performed well regarding the two key criteria for an ideal visibility algorithm, namely that it should quickly reject most of the hidden geometry in the model and, secondly, it should exploit the spatial and temporal coherence of the images being generated. This was a considerable step forward given that previous methods had only been able to satisfy either one or other of these criteria but not both. The key insight on their part was to use two hierarchical data structures: an object-space octree and an image-space Z-pyramid, thus making it possible to reject hidden geometry very rapidly.

In addition to occlusion culling, two other important types of visibility culling are frustrum and backface culling. Frustrum culling is the removal of objects that lie outside the view frustrum. As previously mentioned, the QSplat system implements both these tech-

niques [Rusinkiewicz and Levoy, 2000]. The authors note that backface culling of primitives is commonly implemented in hardware, and Kumar and Manocha have presented an algorithm for hierarchical backface culling based on cones of normals [Kumar et al., 1996]. Finally, most of the speed benefit from frustrum culling come as a natural consequence of implementing the data structure correctly.

Conclusion

In conclusion, there has been a substantial amount of work towards efficient real-time rendering of large models. This is important as the amount of data generated by 3D laser scanners is indeed enormous and is likely to increase as one seeks to examine larger models in greater detail. Having examined the literature and current solutions, it will be necessary to consider future directions and the necessary steps to implement an efficient rendering system for the Department of Geomatics, which can handle data models containing billions of points without extensive decimation of the original data.

References

[Beckmann et al., 1990] Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. (1990). The r*-tree: an efficient and robust access method for points and rectangles. *SIGMOD Rec.*, 19(2):322–331.

[Cignoni et al., 2008] Cignoni, P., Ganovelli, F., Gobbetti, E., Marton, F., Ponchio, F., and Scopigno, R. (2008). Adaptive tetrapuzzles: efficient out-of-core construction and visualization of gigantic multiresolution polygonal models. In *ACM SIGGRAPH ASIA 2008 courses*, SIGGRAPH Asia '08, pages 33:1–33:8, New York, NY, USA. ACM.

[Faloutsos and Roseman, 1989] Faloutsos, C. and Roseman, S. (1989). Fractals for secondary key retrieval. In *Proceedings of the eighth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, PODS '89, pages 247–252, New York, NY, USA. ACM.

[Gobbetti and Marton, 2004] Gobbetti, E. and Marton, F. (2004). Layered point clouds: a simple and efficient multiresolution structure for distributing and rendering gigantic point-sampled models. *Computers & Graphics*, 28(6):815 – 826.

[Greene et al., 1993] Greene, N., Kass, M., and Miller, G. (1993). Hierarchical z-buffer visibility. In *In Computer Graphics (SIGGRAPH '93 Proceedings*, pages 231–240.

[Guttman, 1984] Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA*, pages 47–57. ACM.

[Kofler et al., 2000] Kofler, M., Gervautz, M., and Gruber, M. (2000). R-trees for organizing and visualizing 3d gis databases. *Journal of Visualization and Computer Animation*, 11(3):129–143.

[Kumar et al., 1996] Kumar, S., Manocha, D., Garrett, W., and Lin, M. (1996). Hierarchical back-face computation. In *Proceedings of the eurographics workshop on Rendering techniques '96*, pages 235–253., London, UK, UK. Springer-Verlag.

[Levoy et al., 2000] Levoy, M., Pulli, K., Curless, B., Rusinkiewicz, S., Koller, D., Pereira, L., Ginzton, M., Anderson, S., Davis, J., Ginsberg, J., Shade, J., and Fulk, D. (2000). The digital michelangelo project: 3d scanning of large statues. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 131–144, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Pfister et al., 2000] Pfister, H., Zwicker, M., van Baar, J., and Gross, M. (2000). Surfels: surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 335–342, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Rusinkiewicz and Levoy, 2000] Rusinkiewicz, S. and Levoy, M. (2000). QSplat: A multiresolution point rendering system for large meshes. In *Proceedings of ACM SIGGRAPH 2000*, pages 343–352.

[Sellis et al., 1987] Sellis, T. K., Roussopoulos, N., and Faloutsos, C. (1987). The r+-tree: A dynamic index for multi-dimensional objects. In *Proceedings of the 13th International Conference on Very Large Data Bases*, VLDB '87, pages 507–518, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Wand et al., 2007] Wand, M., Berner, A., Bokeloh, M., Fleck, A., Hoffmann, M., Jenke, P., Maier, B., Staneker, D., and Schilling, A. (2007). Interactive editing of large point clouds. In Chen, B., Zwicker, M., Botsch, M., and Pajarola, R., editors, *Symposium on Point-Based Graphics 2007 : Eurographics / IEEE VGTC Symposium Proceedings*, pages 37–46, Prague, Czech Republik. Eurographics Association.

[Wang and Shan, 2005] Wang, J. and Shan, J. (2005). Space-filling curve based point clouds index. *Geocomputation*.

[Zhu et al., 2007] Zhu, Q., Gong, J., and Zhang, Y. (2007). An efficient 3d r-tree spatial index method for virtual geographic environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, 62(3):217 – 224.

[Zlatanova, 2000] Zlatanova, S. (2000). *3D GIS for Urban Development*. PhD thesis, International Institute for Geo-Information Science and Earth Observation, Netherlands.