Organization and Simplification of High-Resolution 3D City Facades

Mitchell Parry, William Ribarsky, Christopher Shaw, Justin Jang, and Nickolas Faust
GVU Center, College of Computing, Georgia Institute of Technology

ABSTRACT

This paper describes an approach for the organization and simplification of high-resolution geometry and imagery data for 3D buildings for interactive city navigation. At the highest level of organization, building data are inserted into a global hierarchy that supports the large-scale storage of cities around the world. This structure also provides fast access to the data suitable for interactive visualization. At this level the structure and simplification algorithms deal with city blocks. An associated latitude and longitude coordinate for each block is used to place it in the hierarchy. Each block is decomposed into building facades. A facade is a texture-mapped polygonal mesh representing one side of a city block. Therefore, a block typically contains four facades, but it may contain more. The facades are partitioned into relatively flat surfaces called faces. A texture-mapped polygonal mesh represents the building facades. By simplifying the faces first instead of the facades, the dominant characteristics of the building geometry are maintained. At the lowest level of detail, each face is simplified into a single texture-mapped polygon. An algorithm is presented for the simplification transition between the high- and low-detail representations of the faces. Other techniques for the simplification of entire blocks and even cities are discussed.

Keywords: interactive visualization, level of detail, LOD, hierarchical, city, urban, simplification, view-dependent

1 INTRODUCTION

We are now faced with the possibility and in some cases the results of obtaining accurate digital representations of our cities. But these large-scale city models will not be visible, in the sense of interactive visualization and navigation, unless we develop some fundamental capabilities. Already this is the case with the complex CAD models that have been developed for many major downtown areas (Manhattan, for example) both in the U.S. and abroad. These models are fundamentally too complex and ill-organized for interactive visualization. There is not even a clear procedure for simplification; this now involves a painstaking hands-on process. Indeed it is sometimes best to treat the CAD model the same as a physical model—that is, one collects digital images from the model and uses techniques such as close-range photogrammetry to produce a simplified, textured model.

Now various scanning technologies, such as ground-based laser range-finding or airborne techniques such as LIDAR, have been coupled with ground-based, airborne, and satellite imaging to produce high resolution, real-time views of urban areas that can be developed into dynamic models. Ultimately, these models can be fully exploited only if there is a way to organize them into scalable structures with multiple levels of detail for interactive visualization.

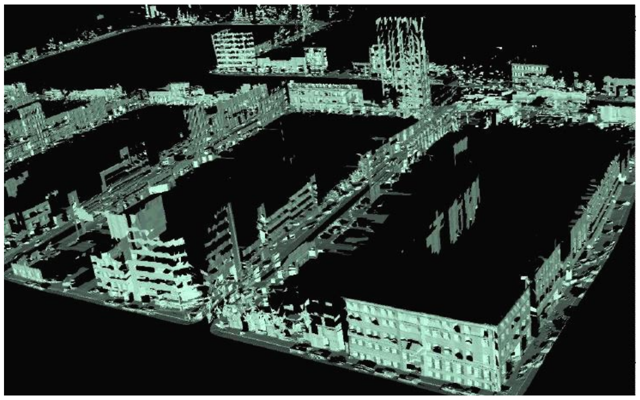


Fig. 1 Several blocks of 3D facades collected with an automated method. (Image courtesy of Avideh Zakhor [Fru01].)

Collaborators at UC Berkeley have developed a general method for fast acquisition of building facades [Fru01] along the length of a city block using ground level laser scans and calibrated imagery. The raw data does not represent a perfect representation of building facades. Artifacts such as trees, cars, and pedestrians occlude the building surface and must be taken into account. The procedure of accurately joining together and meshing can be automated so that several city blocks can be collected and a mesh produced in a period of hours. (See Fig. 1 for the result of collecting and automated modeling for the downtown area of Berkeley, CA.) In the future this process is likely to be even faster. The result is a highly detailed, though incomplete, model that is accurate both in the relative position of structures and in absolute positioning over the whole model. Such a model can be accurately inserted into a geospatial system such as our VGIS system [Fau00]. However, there is no method to interactively visualize these polygonal meshes, which could contain unbounded numbers of vertices, depending on the extent of the data acquisition phase. To provide interactive visualization we must have an automatic simplification procedure and a hierarchical organization. These are the topics of this paper. Although we focus on dense, urban 3D models and, in particular, on the acquired data depicted in Fig. 1, the simplification methods we describe here are general, mesh-independent, and applicable to many other types of extended 3D data. Indeed the organization and simplification of the disparate 3D CAD models mentioned above will be possible using these methods. In the future we expect to investigate this issue closely and to determine if we can produce visually navigable models. The hierarchical structure described here is designed for dense urban 3D data (such as that found in a city downtown) embedded in a global geospatial environment. Modifications of this hierarchy may be better for other types of data.

2 PREVIOUS WORK

There has not been much work that considers the interactive navigation of scalable, extended 3D environments. Sillion et. al. [Sil97] develop a hybrid approach for navigating dense downtown areas using textured building geometry in the foreground and imposter images in the background. An efficient hierarchical organization takes into account that the user is on ground level with a view that is greatly occluded by the foreground buildings. We wish to support also interactive navigation where the viewpoint is flying over the buildings, and the buildings are much more detailed than Sillion et. al. An alternative method has been developed that uses both impostors and geometry, with levels of detail (LODs) for each, and error metrics to quantify visual error [Ali99]. This shows that interactive walkthroughs of massive modeled interiors (such as a power plant) can be provided and that strategies for handling large-scale data such as prefetching and caching work well. However, this approach is only for walkthroughs of interior spaces with limited navigational freedom and preferred viewpoints.

There is also a significant amount of work on image-based rendering methods that is of relevance here [McM95, Cha00, Gor96]. Ultimately, our city databases will be extended enough that we will need an approach that fully integrates both image-based and geometric methods. This is because geometric modeling of far-field detail (especially in overview navigation) requires too much overhead to combine detail from collections of objects. Here image-based methods will win out, as shown recently by the analysis of Chai et. al. [Cha00]. Interestingly, one can infer from this work that hybrid polygonal and image-based methods are also best when one has highly detailed data for individual buildings. When the viewpoint is close to the facade geometry-based rendering is best and as the viewpoint recedes an image-based rendering of the facade is better. This is the approach we use in the work presented here.

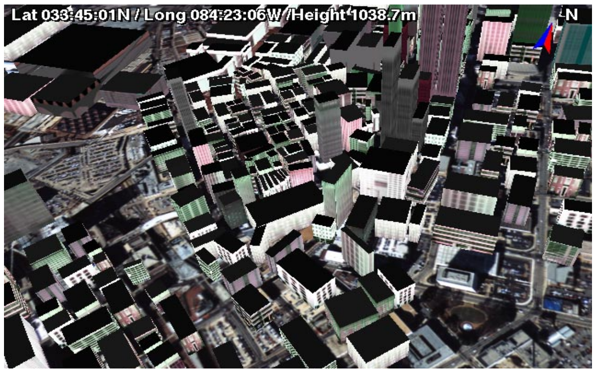


Fig. 2 View of 3D geometry in Atlanta downtown from VGIS global visualization system.

Recently a general and efficient hierarchical procedure for highly detailed 3D models has been developed {Rus00]. This uses a bounding sphere hierarchy and does not explicitly depend on any mesh connectivity. In fact connectivity can be ignored and the 3D cloud of points rendered using splats [Rus00], which lend themselves to a multiresolution structure and are often faster to render than polygons. However, splats do not give good visual quality for areas with smooth geometry or during zoom-in where the splats become larger than the screen pixels. Here textured polygons are superior, so that a hybrid approach that could switch between points and polygons would be best. Other research groups have recently begun to explore the possibility of hybrid point and polygon rendering [Coh01, Che01]. There are many unexplored issues for hybrid rendering, including maintaining texture information and scaling up to very large datasets. We begin exploring these issues here.

3 SCALABLE, HIERARCHICAL ORGANIZATION FOR BUILDINGS

The detailed 3D building facade data are integrated into a global geospatial hierarchy. We have used a similar approach for storing terrain height fields, simple buildings, and geospatial weather data [Dav99, Fau00, Jia01]. These are all part of our VGIS system which, among other things, provides visual navigation of global geospatial environments with the ability to zoom-in at will to areas of high detail [Fau00, Rib01]. The data object that we store in the global hierarchy is a city block. Each city block contains a latitude and longitude position and is inserted into our global quadtree based on this position. We choose the depth of the tree such that the size of the smallest quadnode is roughly the size of a city block and data retrieval is sufficiently fast. That way, in the typical case, up to four blocks overlap a quadnode.

The block subdivision makes sense because, especially in downtown areas, walls are shared and individual buildings are hard to distinguish. Since the blocks are accurately located in latitude and longitude, a GIS layer can be added that identifies sub-coordinates for individual buildings where desired. The global hierarchy described here supports this. Each block is composed of a set of simple facades to which textures and 3D details are attached. Facades will typically be flat, though care must be taken in handling smoothly curving building fronts. The simplest set of facades would be a box of the appropriate height extruded from a footprint of accurate location and area. This is basically what we have used in previous urban visualizations (Fig. 2). At lowest resolution the facade would support a texture representation of its details, which could be depth-layered imagery [McM95, Cha00]. The global quadtree hierarchy is used to organize the blocks for fast view culling and retrieval. We have previously used this structure for simpler block geometry [Dav99, Fau00] and have shown that it provides efficient view frustum culling for large or geographically distributed collections of buildings. The hierarchy also lends itself to efficient predictive traversal and a chunk size that supports effective prefetching and caching [Dav99]. The variable chunk size, closely aligned with the rendering properties of the geometry, has been shown to be a prime factor for efficient out-of-core visualization over a range of systems with different local compute power, disk I/O, and networking capabilities [Cox97].

Attached to the facade are not only details such as window frames or doorways but also separated geometry that may be in front of the building such as trees, lampposts, and sidewalks. All these items will have coordinates relative to the facade frame of reference. It appears most efficient to construct an object tree for the block with the facades as 1st level children and the separated objects as their children. (See Fig. 3.) Both the facade and the separated objects will also have sub-trees to handle levels of detail. The detailed geometry of the objects can be represented either by triangle meshes or by points, and we allow both geometric primitives in our models. The results presented here concentrate on the geometric details attached directly to the facade. The separated geometry (denoted by Object 1…Object M in Fig.3) will be considered in the future.

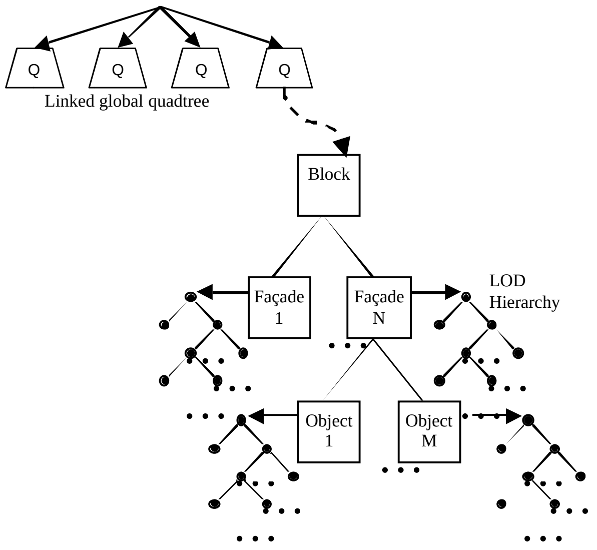


Fig. 3 Hierarchical structure of 3D urban block geometry.

A flexible approach to managing the relationships between the various geometric representations (facades and objects) in Fig. 3 is to use an appropriate hierarchical graph structure. A number of ways in which to create such a hierarchy are possible, and we believe that the bounding sphere hierarchy used for Qsplat [Rus00] and the multi-resolution graph (MRG) data structure used for hybrid simplification [Coh01] can provide good starting points for our application. The sphere hierarchy is set up by starting from the bounding box for the overall model and recursively splitting vertices along the longest axis to find the bounding boxes (and bounding spheres) of the children. In contrast, the MRG is a tree created by performing simplification operations such as edge collapse on the polygonal mesh and representing each such operation as a node in the hierarchy. Nodes in the MRG contain geometric primitives such as triangles and points. We seek a middle ground between the MRG, which produces too detailed a hierarchy, and the sphere hierarchy, which does not contain connectivity information. We perform simplification operations in order to produce our hierarchy, but instead of using fine-grain simplification operators such as those in [Coh01], we use vertex clustering techniques such as the one first proposed by Rossignac and Borrel [Ros93]. In the present work, we allow the bounding spheres to define the vertex clusters and the triangle connectivity at each LOD, as described further below. Each node of the completed tree will contain the sphere center and radius, a normal, the width of the normal cone, color properties, and, optionally, connectivity information. Without connectivity information, these node properties can be used for fast, multiresolution splat rendering [Rus00]. By using coordinates relative to the facade and a node-based quantization scheme appropriate to the original sampled data, a compact data structure can be generated. This can be a real advantage, for example, in incremental paging of out-of-core data when the in-core level of detail is insufficient for rendering the current scene.

Our hierarchical data structure has the significant advantage of supporting both splat- and polygon-based rendering. Trees, for example might be best rendered using splats. Other research groups have recently begun to explore the possibility of hybrid point and polygon rendering [Coh01, Che01]. One aspect of this research will be to determine the relative regimes for the two types of rendering. We will also compare with other types of mesh construction and simplification; modifications may result from this evaluation.

4 COMPLETING INCOMPLETE MODELS

Acquired models will typically be incomplete, as shown in Fig. 1. Using ground-based laser-range finding and imaging, for example, will produce models with the top floors of tall buildings missing. In addition there are many possibilities for occlusion, such as trees, cars, or even people blocking building geometry. On the other hand aerial data, such as LIDAR 3D data or aerial imagery, will often provide accurate location, height, and footprint information but will lack details from the sides of the building and other street-level detail. A complete solution will integrate results from all these modalities plus use modeling methods for details that may still be missing.

Model completion methods include geometric hole filling, photometry-assisted hole filling, geometry from symmetry, and texture synthesis. For holes where no photometric information is available, a purely geometric hole filling approach may be called for. The incomplete triangle mesh will have boundary edges of known triangles that surround a hole, and the problem is to triangulate the hole in a plausible manner. The first step is to find any triangulation that fills in the hole. Once an initial triangulation is found, then we can make incremental changes to the triangles in order to satisfy quality criteria such as triangle shape and surface smoothness. If a camera captures the appearance of a surface where we have missing geometry, we can use this information to improve the geometry given by hole filling. This is photometry-assisted hole filling. The idea is to augment the smoothness functional from above with a shape-from-shading or shape-from-stereo method, constrained by the positions of the boundary edges of the hole. Large backfacing portions of an object's geometry may also be missing due to the inability to observe each object from all vantage points. Back sides of trees, cars, signs, and buildings will inevitably be missed. Such objects will be completed by geometry from symmetry. Once an object is identified as a candidate for symmetry completion, it must be separated from the surrounding geometry and the axis of symmetry must be determined. Then we can mirror the existing geometry around this axis, giving two separate meshes that must now be merged. Finally, although photometric information will be available for some surfaces that have missing geometry, there will be instances where both geometric and photometric information is missing. This may be the case, for example, when a very wide object such as a car is obscuring the sidewalk. In this case, given a small sample image of a given pattern such as bricks or grass, a texture synthesis technique can create more of the same pattern.

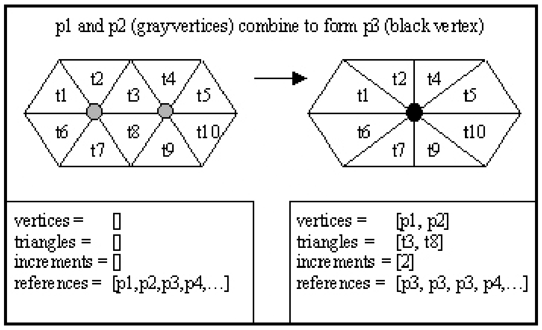
We have developed a semi-automated procedure that employs photometry-assisted hole filling, geometry from symmetry, and texture synthesis [Fau01, Was99]. This is the procedure we use here to complete incomplete models. However, our methods do not currently use all the data that is available to accurately complete the models. In addition the methods are semi-automatic, requiring significant input from an operator, though much less than with older, non-automated methods. In the future we will develop more fully automated and more complete methods.

5 VIEW-DEPENDENT SIMPLIFICATION FOR INTERACTIVE VISUALIZATION OF URBAN MODELS

To support interactive visualization of extended, detailed models, such as those acquired with the method depicted in Fig. 1 [Fru01], we must develop a view-dependent structure that chooses detail dynamically based on the user viewpoint at each frame. Parts of the scene closer to and facing the viewer are rendered in greater detail while parts farther away or at an oblique angle are rendered with less detail. A screen space error metric, with a geometric component and an image component, is used to determine the level of detail for each part. The screen space error is in terms of pixels and thus connects directly to the constraint of not drawing more details than there are pixels on the screen and not drawing detail at finer resolution than the pixel resolution. It has been shown that such a view-dependent metric greatly reduces the number of triangles and textures that must be rendered to visually navigate complex terrain or to visualize complex objects at full screen resolution and that appropriate adjustment of the screen space error can provide interactive visualization [Lin96, Hop97]. Our task is to provide a view-dependent structure not for continuous terrain height fields or for single, complex objects but rather for large, extended collections of detailed objects. Ultimately these collections of objects, in the case of buildings, will be combined with terrain height fields in an integrated view-dependent structure.

To develop the view-dependent structure we must first detail the simplification procedure. Here we concentrate on the bounding sphere version [Rus00] of the object hierarchy, which is embedded in the overall global structure, as described in Sec. 3. The bounding sphere hierarchy is a fast way to organize 3D vertices on each facade. At the highest LOD each vertex is represented by a unique sphere. This is a leaf node. Parent nodes hold the average position of their children and their bounding spheres encompass the child bounding spheres. For splat-based rendering, there is a splat associated with each node. For polygon-based rendering, a simplification process must be constructed that connects all the LODs. The progressive mesh procedure of Hoppe [Hop96, Hop97] requires too much time for our purposes. Most of the computation goes towards determining the “optimum” order of edge collapses to go from the full resolution mesh to the zero’th order mesh. Instead, we follow the bounding sphere hierarchy to create a simplification process. As discussed in Sec. 3, this approach is similar to the vertex clustering approaches [Ros93]. The optimality of our approach is not clear. Certainly it produces reasonable results for single, complex objects [Rus00, Ros93]. In the future we will look more closely at this issue.

To generate the detailed polygonal hierarchy, we use a parent node’s splat size as an approximate indicator of the error incurred by collapsing its children. In general smaller splat sizes indicate smaller adjacent polygons and therefore smaller changes to the geometry when those polygons are decimated. Since the splat is expressed in terms of a screen space projection, this provides a tie-in to the view-dependent metric we seek. The mesh simplification algorithm uses only references to the original vertices. (However, the actual vertex coordinates at lower LODs are averages of their child vertex coordinates.) A triangle is made up of three vertex references. As these vertices combine they are updated to reference higher levels in the hierarchy. Therefore as the mesh simplifies, triangles change shape and generally get bigger.



vertices = []
triangles = []
increments = []
references = [p1,p2,p3,p4,...]

vertices = [p1, p2]
triangles = [t3, t8]
increments = [2]
references = [p3, p3, p3, p4,...]

Fig. 4 The initial combination step.

The algorithm maintains an ordered array of combined child vertices, decimated triangles, triangle increments, and vertex references. Starting with the smallest splat size in the facade hierarchy (excluding leaf nodes), we combine child vertices into their parent and determine which if any polygons have been decimated. We update each child reference to point at its parent, add the child vertices to the vertex array, add any decimated triangles to the triangle array, and set the number of decimated triangles to the triangle increment array. We continue this process until all polygons have been removed and the simplification mesh is complete. Fig. 4 shows the first combination in the algorithm. Initially, the vertices, triangles, and increments arrays are empty, and the references array is initialized to point at the original points in the hierarchy.

Collapsing two vertices results in the decimation of 0, 1, or 2 polygons. The bounding sphere algorithm partitions the data without respect to the mesh connectivity. Thus two vertices can share a parent when they do not share an edge in the mesh. In this case, their combination does not remove a polygon. If the vertices are connected by an edge they decimate as many polygons as share that edge (1 or 2 polygons).

Once we have the resultant simplification arrays, we can render the model with any number of triangles less than the original mesh. One way to generate a simplification of the mesh is starting at the front of the arrays. We can initialize all array indices to zero and proceed by combining each pair of vertices as they appear in the vertices array. For the example shown in Fig. 4, we know that the first combination decimated 2 triangles (the first entry in the increments array). First we point the child vertices at their parent. Then increment the vertices array index by two, triangles array index by two, and increments index by one. Now we have a simplification of the mesh with 2 fewer triangles. If we want to generate a mesh with N triangles, we continue this process until only N triangles remain. Rendering only requires drawing the triangles from the current triangle index to the end of the array. Upon placing this procedure in the hierarchical structure as indicated in Fig. 3, we can institute a fast traversal process to provide either a target number of polygons or a target screen space error.

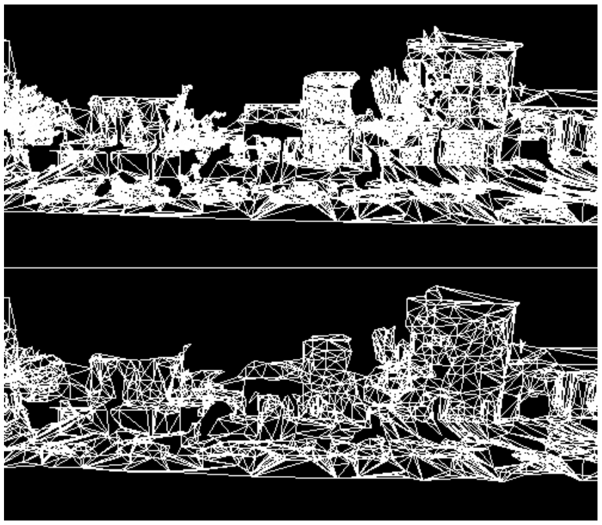


Fig. 5 Full resolution facade (top) and 75% reduction (bottom).

When extending this algorithm to the facade faces discussed previously, one of our requirements is that the shape of the face does not change during simplification. We accomplish this by assigning "silhouette" points to the original data set. These points represent the outline or shape of the face that we wish to preserve. During the construction of the facade hierarchy, silhouette vertices are not combined with other vertices. Simplification continues as described but does not affect the predefined silhouette. Automating silhouette vertex determination is a problem yet to be solved. However, we expect that the largely planar aspect of the faces will simplify the problem. Ultimately, at the highest level of the hierarchy (lowest LOD), we wish to retain the fewest points necessary to describe a facade face. In most cases the faces will be rectangular at this level, so four points will be retained. This makes the transition to the simplified building representations we have developed previously [Dav99].

The result of this simplification procedure is shown if Fig. 5, which is a segment of the data displayed in Fig. 1. A high resolution block facade [Fru01] is shown at the top. This mesh includes not only building fronts, but also details such as sidewalks and lampposts. The lower half of Fig. 5 show the mesh after 75% reduction in the number of polygons. We see that finer detailed polygons have been replaced by larger polygons. Further the overall silhouette shape of the original mesh has been retained.

6 CONCLUSIONS AND FUTURE WORK

We have described a scalable building organization structure appropriate for dense urban areas. It permits the handling of highly detailed building facades and the inclusion of other data collected at the same time, such as tree, sidewalk, or lamppost data. The hierarchy is embedded into a global hierarchical structure that has proven successful in the out-of-core visualization of large collections of simple buildings, which may be distributed geographically or clustered in one region. A simplification procedure at the level of the building facades is then presented. It is based on a bounding sphere hierarchy and permits both point-based and polygon-based representation of the data. For the polygon-based representation we define a simplification process that goes from the original high resolution mesh to the simplest representation of the facades. This is based on the multiresolution bounding sphere hierarchy. We show that this procedure gives reasonable results for highly detailed urban facade data. The simplification algorithm also takes into account constraints in the visual representation of the facade by permitting the insertion or selection of silhouette points that are not combined with other points.

This paper presents a first step in the application of these efficient organizational and simplification methods. There is much still to do. We will apply our method to a much larger collection of city data and will integrate the results into a fully view-dependent visualization structure (with terrain in VGIS). In addition we will develop a simplification procedure for facade textures and appearance information. We will look bringing in other objects associated with facades (trees, sidewalks, etc.) into the urban block hierarchy and the simplification structure. The latter may require a hybrid splat and polygon approach (with trees, for example). We will also look at the optimality of our simplification procedure with respect to other procedures. Finally we will look at the application of these methods to different data, such as the disparate and highly complex urban CAD models, which currently have no direct way for simplification and interactive visualization.

ACKNOWLEDGMENTS

This work was supported by the DoD Multidisciplinary University Research Initiative (MURI) program administered by the Office of Naval Research through the Army Research Office under Grant DAAD19-00-1-0352. This work was also supported by a grant from the NSF Large Scientific and Software Data Visualization program. In addition there was support from the Office of Naval Research and the Naval Research Labs under Award No. N00014-97-1-0898.

REFERENCES

Ali99 D. Aliaga, J. Cohen, A. Wilson, E. Baker, H. Zhang, C. Erikson, K. Hoff, T. Hudson, W. Stuerzlinger, R. Bastos, M. Whitton, F. Brooks, and D. Manocha. MMR: An Interactive Massive Model Rendering System Using Geometric and Image-Based Acceleration. *1999 Symp. On Interactive 3D Graphics*, pp. 199-237 (1999).

Cha00 J.X. Chai, X. Tong, S.C. Chan, and H.Y. Shum. Plenoptic Sampling. *Proc. SIGGRAPH 2000*, pp. 307-318 (2000).

Che01 Chen, Baoquan and Minh Xuan Nguyen. POP: A Hybrid Point and Polygon Rendering System for Large Data. *Proc. IEEE Visualization 2001*. San Diego, California. pp. 45-52 (Octoboer 2001).

Coh01 Cohen, Jonathan, Daniel Aliaga and Weiqiang Zhang. Hybrid Simplification: Combining Multi-Resolution Polygon and Point Rendering. *Proc. IEEE Visualization 2001*. San Diego, California. pp. 37-44 (October 2001).

Cox97 M. Cox and D. Ellsworth. Application-Controlled Demand Paging for Out-of-Core Visualization. Proceedings, *IEEE Visualization '97*, pp. 235-244 (1997).

Dav99 Douglass Davis, William Ribarsky, T.Y. Jiang, Nickolas Faust, and Sean Ho. "Real-Time Visualization of Scalably Large Collections of Heterogeneous Objects," Report GIT-GVU-99-14, pp. 437-440, *IEEE Visualization '99* (1999).

Fau00 N. Faust, W. Ribarsky, T.Y. Jiang, and T. Wasilewski. Real-Time Global Data Model for the Digital Earth. *Proc. INTERNATIONAL CONFERENCE ON DISCRETE GLOBAL GRIDS (2000).* An earlier version is in Report GIT-GVU-97-07.

Fau01 Nickolas Faust and William Ribarsky, “Development of Tools for Construction of Urban Databases and Their Efficient Visualization,” Modeling and Visualizing the Digital Earth, Mahdi Abdelguerfi, Editor (Kluwer, Amsterdam, 2001).

Fru01 C. Früh and A. Zakhor. Fast 3D model generation in urban environments. *International Conference on Multisensor Fusion and Integration for Intelligent Systems* 2001, Baden-Baden, Germany, p. 165-170 (2001).

Gor96 Gor96 Gortler, Steven J., Radek Grzeszczuk, Richard Szeliski and Michael F. Cohen. The Lumigraph. *Proc. SIGGRAPH 96*. pp. 43-54 (August 1996).

Hop96 Hugues Hoppe. Progressive Meshes. *Proc. SIGGRAPH 96*. pp. 99-108 (August 1996).

Hop97 Hugues Hoppe. View-Dependent Refinement of Progressive Meshes. *Proc. SIGGRAPH 97*. pp. 189-198 (August 1997).

Jia01 T.Y. Jiang, William Ribarsky, Tony Wasilewski, Nickolas Faust, Brendan Hannigan, and Mitchell Parry, “Acquisition and Display of Real-Time Atmospheric Data on Terrain,”, pp. 15-24, Proceedings of Eurographics-IEEE Visualization Symposium 2001.

Lin96 Peter Lindstrom, David Koller, William Ribarsky, Larry Hodges, Nick Faust, and Gregory Turner. Real-Time, Continuous Level of Detail Rendering of Height Fields. *Proc. SIGGRAPH 96*, pp. 109-118 (1996).

McM95 McMillan, Leonard and Gary Bishop. Plenoptic Modeling: An Image-Based Rendering System. *Proc. SIGGRAPH 95*. pp. 39-46 (August 1995).

Rib01 W. Ribarsky. Towards the Visual Earth. Invited paper, *National Research Council Workshop on Enhancing th Accessibility and Usability of Geospatial Information*, National Academies of Science, Washington, DC, October, 2001 (www4.nationalacademies.org/cpsma/cstb.nsf/web/project_geospatial_papers?OpenDocument)

Ros93 Rossignac, Jarek and Paul Borrel. Multi-Resolution 3D Approximations for Rendering Complex Scenes. In *Modeling in Computer Graphics*. B. Falcidieno and T. L. Kunii, editors. Springer-Verlag. Pp. 455-465 (1993).

Rus00 S. Rusinkiewicz and M. Levoy. Qsplat: A Multiresolution Point Rendering System for Large Meshes. *Proc. SIGGRAPH 2000*, pp. 343-352 (2000).

Sil97 F. Sillion, G. Drettakis, B. Bodelet. Efficient Impostor Manipulation for Real-Time Visualization of Urban Scenery. *Proc. EUROGRAPHICS'97*, pp. C-207-218 (1997).

Was99 Tony Wasilewski, Nickolas Faust, and William Ribarsky. "Semi-Automated and Interactive Construction of 3D Urban Terrains," *Proceedings of the SPIE Aerospace/Defense Sensing, Simulation & Controls Symposium*, Vol. 3694A, pp. 31-38 (1999).