Proceedings of DETC'97
1997 ASME Design Engineering Technical Conferences
September 14-17, 1997, Sacramento, California

DETC97/DFM-4371

ENABLING VIRTUAL REALITY FOR LARGE-SCALE MECHANICAL CAD DATASETS

Amitabh Varshney *
Department of Computer Science
State University of New York
Stony Brook, NY 11794-4400
USA

Jihad El-Sana
Department of Computer Science
State University of New York
Stony Brook, NY 11794-4400
USA

Francine Evans
Department of Computer Science
State University of New York
Stony Brook, NY 11794-4400
USA

Lucia Darsa
Department of Computer Science
State University of New York
Stony Brook, NY 11794-4400
USA

Bruno Costa
Microsoft Corporation,
One Microsoft Way,
Redmond, WA 98052-6399
USA

Steven Skiena
Department of Computer Science
State University of New York
Stony Brook, NY 11794-4400
USA

ABSTRACT

Reconciling scene realism with interactivity has emerged as one of the most important areas in making virtual reality feasible for large-scale mechanical CAD datasets consisting of several millions of primitives. This paper surveys our research and related work for achieving interactivity without sacrificing realism in virtual reality walkthroughs and flythroughs of polygonal CAD datasets. We outline our recent work on efficient generation of triangle strips from polygonal models that takes advantage of compression of connectivity information. This results in substantial savings in rendering, transmission, and storage. We outline our work on genus-reducing simplifications as well as real-time view-dependent simplifications that allow on-the-fly selection amongst multiple levels of detail, based upon lighting and viewing parameters. Our method allows multiple levels of detail to coexist on the same object at different regions and to merge seamlessly without any cracks or shading artifacts. We also present an overview of our work on hardware-assisted image-based rendering that allows interactive exploration of computer-generated scenes.

INTRODUCTION

Virtual Reality (VR) provides a powerful enabling technology for visualization, evaluation, and communication by immersing one or more users within a computer-generated environment which contains the dataset being analyzed. Compared with traditional geometric visualization systems, VR-based systems provide users a superior interaction environment for exploring CAD datasets in real time. The last decade has witnessed the emergence of mega models in the manufacturing industry. Such full-scale digital mock-ups of large mechanical structures such as submarines, aircraft, and automobiles consist of tens of millions of polygons distributed amongst tens of thousands of objects. Visualization of such large models in real time offers new promises and challenges. The promises include dramatically reduced design cycle times, an integrated approach to manufacturing, and real-time geographically distributed collaborative design. The challenges lie in large-scale model engineering – novel paradigms for three-dimensional immersive design, extensions of databases to be able to manage and query three-dimensional geometric objects, hierarchical design with version control, association of parts to manufacturers, robust handling of real-life model degeneracies, and interactive three-dimensional visualization. This paper focuses on the challenge of interactive three-dimensional visualization for large-scale mechanical CAD models.

The two most crucial requirements of any VR system are – (a) real-time update rates (i.e. at least 10 and prefer-

*Email: varshney@cs.sunysb.edu

ably 25 frames/sec) of the stereo scene being viewed with low latency in location and orientation tracking, and (b) scene realism. These two are mutually conflicting goals and yet have to be simultaneously achieved to maintain the illusion of a virtual world. These pose substantial processing constraints on the rendering and tracking hardware and software. For large-scale CAD visualization the situation is worse since it demands extensive computational resources, often several orders of magnitude greater than what the hardware can deliver. Further, the increase in complexity of such virtual environments has outpaced the hardware advancements in computer graphics. This paper outlines our ongoing research for achieving interactivity without sacrificing realism in virtual reality walkthroughs and flythroughs of large-scale polygonal CAD datasets.

Recent research on graphics acceleration for the navigation of large-scale three-dimensional environments has been motivated by attempts to bridge the gap between the desired and the actual hardware performance, through algorithmic and software techniques. This research has involved reducing the geometric and rendering complexities of the scene by using

- compression strategies (Deering 95; Evans *et al* 96)
- level-of-detail hierarchies (Turk 92; Schroeder *et al* 92; Rossignac and Borrel 93; Cohen *et al* 96; Hoppe 96; DeRose *et al* 93; He *et al* 96; Guéziec 95; Hamann 94; Xia *et al* 97),
- visibility-based culling (Airey 90; Teller and Séquin 91; Luebke and Georges 95; Greene 96; Zhang *et al* 97),
- various levels of complexity in shading and illumination models(Bergman *et al* 86),
- texture mapping (Blinn and Newell 76), and
- image-based rendering (Chen and Williams 93; Chen 95; McMillan and Bishop 95; Shade *et al* 96; Darsa *et al* 97).

Our research thus far has included compression strategies, static and dynamic level-of-detail hierarchies, and image-based rendering.

CONNECTIVITY COMPRESSION: TRIANGLE STRIPS

Interactive display rates are crucial to virtual reality. The speed of high-performance rendering engines on triangular meshes in computer graphics can be bounded by the rate at which triangulation data is sent into the machine. Obviously, each triangle can be specified by three vertices, but to maximize the use of the available data bandwidth, it is desirable to order the triangles so that consecutive triangles share an edge. Using such an ordering, only the incremental change of one vertex per triangle need be specified, potentially reducing the rendering time by a factor of three

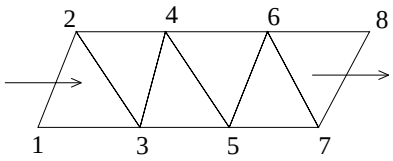


Figure 1. A Triangle Strip

by avoiding redundant clipping and transformation computations. Besides, such an approach also has obvious benefits in compression for storing and transmitting models.

Consider the triangulation in Figure 1. Without using triangle strips, we would have to specify the five triangles with three vertices each. By using triangle strips, as supported by the OpenGL graphics library (Open GL Architecture Review Board 93; Open GL Architecture Review Board *et al* 93), we can describe the triangulation using the strip $(1,2,3,4,5,6,7,8)$, and assuming the convention that the ith triangle is described by the ith, $(i+1)$st, and $(i+2)$nd vertices of the *sequential* strip. Such a sequential strip can reduce the cost to transmit n triangles from $3n$ to $n+2$ vertices.

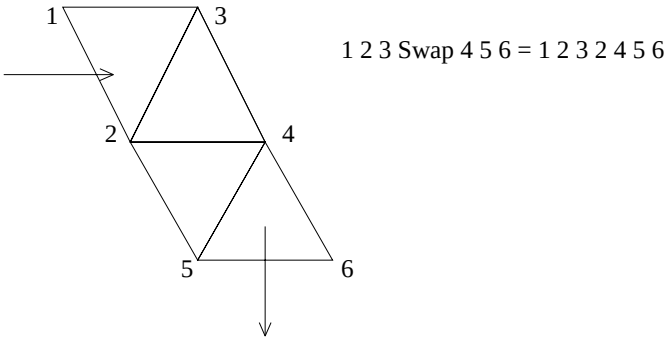


Figure 2. Replacing a swap requires an extra vertex.

To allow greater freedom in the creation of triangle strips, a "swap" command permits one to alter the first-in-first-out queuing discipline in a triangle strip (Silicon Graphics, Inc. 91). A swap command swaps the order of the two latest vertices in the buffer so that the instead of vertex i replacing the vertex $(i-2)$ in a buffer of size 2, vertex i replaces the vertex $(i-1)$. This allows for a single triangle strip representation of the collection of triangles shown in Figure 2, as $(1,2,3,SWAP,4,5,6)$. This form of a triangle strip that includes swap commands is referred to as a *generalized triangle strip*.

Although the swap command is supported in the GL graphics library (Silicon Graphics, Inc. 91), keeping portability considerations in mind it was decided to not sup-

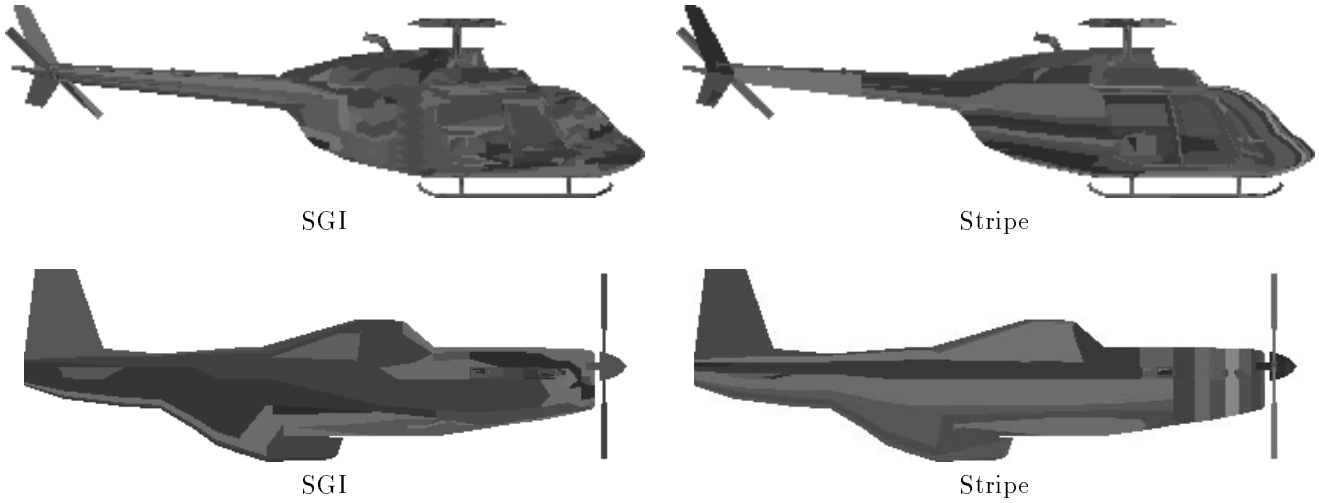


Figure 3. Visual Comparison of Triangle Strips Generated by SGI and Stripe

port it in OpenGL. With OpenGL gaining rapid acceptance in the graphics software community, the one-bit-per-vertex cost model that was appropriate for a swap command in GL is now outdated. A more appropriate cost for such a swap command under the OpenGL model is a penalty of one vertex as explained next. One can simulate a swap command in OpenGL by re-transmitting the vertex that had to be swapped. This results in an empty triangle two of whose vertices are the same. This is illustrated in Figure 2, where we simulate $(1, 2, 3, SWAP, 4, 5, 6)$ by $(1, 2, 3, 2, 4, 5, 6)$. Note that, even though a swap costs one vertex in the OpenGL model, it is still cheaper than starting a new triangle strip that costs two vertices.

We have considered the problem of constructing good triangle strips from polygonal models. Often such models are not fully triangulated, and contain quadrilaterals and other non-triangular faces, which must be triangulated prior to rendering. The choice of triangulation can significantly impact the cost of the resulting strips. We have recently proved that the problem of triangulating a polygonal model for optimal strips is NP-complete (Evans *et al* 97). Our work on efficient triangle strips therefore explores several heuristics that we have empirically observed to produce good triangle strips on real-life models (Evans *et al* 96). Our linear-time algorithm manages to achieve this by exploiting both the local and the global structure of the model. Our analysis of the global structure of a geometric model is done via a non-geometric technique we term *patchification*, which we believe is of general interest as an efficient tool for logically partitioning polygonal models.

The best previous code for constructing triangle strips which we are aware of is (Akeley *et al* 90), implementing what we will call the SGI algorithm. The SGI algorithm seeks to create strips that tend to minimize leaving isolated triangles. It is a greedy algorithm, which always chooses as the next triangle in a strip the triangle that is adjacent to the least number of neighbors.

We have experimented with several variants of local and global algorithms; the details are available in (Evans *et al* 96). For our local approaches there were ten different options for each data file that we ran our experiments on. Similarly, for our global approaches there were ten different options for each data file that we ran our experiments on. After comparing the results we had from the above-mentioned 20 different approaches on over 200 datasets, we found that the best option was to use the the global row or column strips with a patch cutoff size of 5. In this approach we first partition the model into regions that have collections of $m \times n$ quadrilaterals arranged in m rows and n columns, which we refer to as a *patch*. Each patch whose number of quadrilaterals, mn, is greater than a specified cutoff, in our case 5, is converted into one strip, at a cost of three swaps per turn. Further, every such strip is extended backwards from the starting quadrilateral and forwards from the ending quadrilateral of the patch to the extent possible. For extension, we use an algorithm similar to the SGI algorithm. However, we triangulate our faces "on the fly", which gives us more freedom in producing triangle strips. We have implemented this option in our tool, *Stripe*. This utility is available from our web-site `http://www.cs.sunysb.edu/~evans/stripe.html` and is free for non-commercial use.

The times for execution of our algorithms behaved linearly with respect the input size. The timings for our local algorithms were about a factor of 2 slower than those generated by SGI. When rendering the models with the triangle

strips that were produced by each algorithm, the savings in transmission time to the renderer did prove to be a significant savings in rendering time. The triangle strips produced by our code were on average 10–20% faster to draw than those produced by the SGI algorithm, and were about 60% faster to draw than without using triangle strips at all. Figure 3 provides visual comparison of the results obtained by our tool Stripe and those obtained by the earlier algorithm being used by SGI.

LEVEL-OF-DETAIL HIERARCHIES

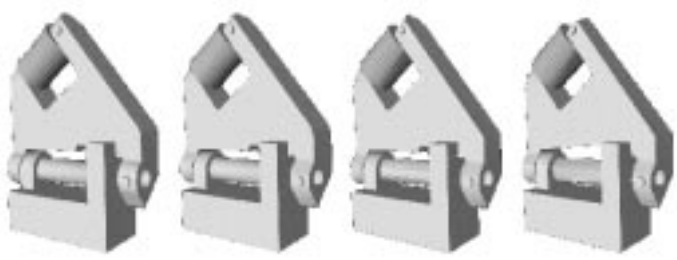
The basic idea in a level-of-detail-based rendering scheme is to use the perceptual importance of a given object in the scene to select its appropriate level-of-detail representation. Thus, higher detail representations are used when the object is perceptually more important and lower detail representations are used when the object is perceptually less significant. This method allows one to achieve higher frame update rates while maintaining good visual realism. This is illustrated in figure 4 where four levels of details of a torpedo roller in a notional submarine are substituted at increasing distances in (a) and are shown side-by-side in (b). The four levels of detail of the torpedo rollers in Figure 4(b) consist of 2346, 1180, 676, and 514 triangles from left to right.

Automatic generation of multi-resolution object hierarchies has become a crucial process in reconciling scene realism with interactivity through level-of-detail-based rendering (Clark 76; Crow 82; Funkhouser and Séquin 93; Rohlf and Helman 94; Maciel and Shirley 95). The hierarchy of level-of-detail approximations for the torpedo rollers shown above was generated using the approach of *simplification envelopes* (Varshney 94; Cohen *et al* 96). Our approach guarantees that all points of an approximation are within a user-specifiable distance ϵ from the original model and all points of the original model are within a distance ϵ from the approximation. Simplification envelopes provide a general framework within which a large collection of existing simplification algorithms can run. The key advantages of our approach are that it provides (a) a general technique providing guaranteed error bounds for genus-preserving simplification, (b) automation of both simplification process and selection of appropriate viewing distances, (c) prevention of self-intersection, (d) preservation of sharp features, and (e) allows variation of approximation distance across different portions of a model.

Genus-Reducing Simplifications

A constraint common to most existing work on automatic generation of multi-resolution object hierarchies has been the genus preservation criterion. Genus is related to the number of holes in an object – a sphere has genus 0,

(a) Receding



(b) Upclose

Figure 4. Visual Comparison of Four Levels-of-Detail of a Torpedo Roller

a torus has genus 1, and digit 8 has genus 2. We have developed methods to simplify the genus of an object in a *controlled* fashion. Preservation of topology (or the genus) is crucial for certain applications such as study of tolerances in mechanical CAD. Clearly, if the target application demands topology preservation, then the simplification algorithm should adhere to it. However, if the goal is fast and realistic rendering, such as for virtual reality CAD visualization, the topology preservation criterion could stand in the way of efficient simplification. Let us consider a flythrough in a virtual reality CAD model. A tiny hole on the surface of a mechanical part in this model will gradually disappear as the observer moves away from the part. However, genus preserving simplification of this object will retain such features, thereby reducing frame rates (due to limits on the amount of geometry-simplification that one can achieve while preserving topology) and increasing image-space aliasing (due to undersampling, especially in perspective viewing).

We view the simplification of an object of arbitrary topological type as a two-stage process – simplification of the topology (i.e. reduction of the genus) and simplification of the geometry (i.e. reduction of the number of vertices, edges, and faces). One can mix the execution of these two stages in any desired order. We have observed that genus reductions by small amounts can lead to large

overall simplifications. This observation together with the fact that genus-reducing simplifications are usually faster than genus-preserving simplifications, makes such an approach quite attractive for generating multiresolution hierarchies. Our research extends the underlying concepts of α-hulls(Edelsbrunner and Mücke 94) to polygonal datasets and allows one to perform genus-reducing simplifications. The primary targets of our research are the interactive three-dimensional graphics and visualization applications where topology can be sacrificed if (a) it does not directly impact the application underlying the visualization and (b) produces no visual artifacts. Both of these goals are easier to achieve if the simplification of the topology is finely *controlled* and has a sound mathematical basis.

We first developed a method to perform genus simplifications in the volumetric domain (He *et al* 96). Our results from that approach appear in figure 5 where a notional mechanical gear is simplified all the way to an octahedron.

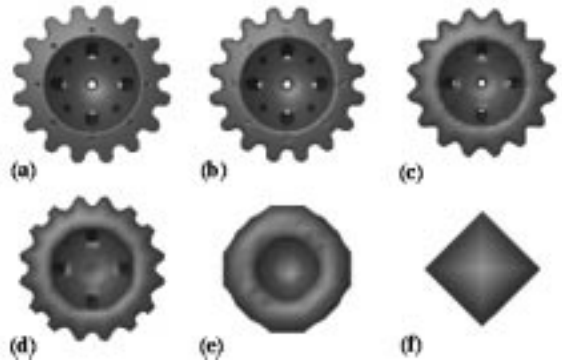


Figure 5. Volume domain genus simplification of a notional gear

We have recently also developed an algorithm that can directly perform genus-reducing simplifications in the polygonal domain. The intuitive idea underlying our approach is to simplify the genus of a polygonal object by rolling a sphere of radius α over it and filling up all the regions that are not accessible to the sphere. This also happens to be the underlying idea behind α-hulls over point-sets. The problem of planning the motion of a sphere amidst polyhedral obstacles in three-dimensions has been very nicely worked out (Bajaj and Kim 88). We use these ideas in our approach. Let us first assume that our polygonal dataset consists of only triangles; if not, we can triangulate the individual polygons (Seidel 91; Narkhede and Manocha 95). Planning the motion of a sphere of radius α, say $S(\alpha)$, amongst triangles is equivalent to planning the motion of a point amongst "grown" triangles. Mathematically, a grown triangle $T_i(\alpha)$ is the Minkowski sum of the original triangle t_i with the sphere $S(\alpha)$. Formally, $T_i(\alpha) = t_i \oplus S(\alpha)$, where $\oplus$ denotes the Minkowski sum which is equivalent to the convolution operation. Thus, our problem reduces to efficiently and robustly computing the union of the grown triangles $T_i(\alpha)$. The boundary of this union, $\partial \bigcup_{i=1}^{n} T_i(\alpha)$, where n is the number of triangles in the dataset, will represent the locus of the center of the sphere as it is rolled in contact with one or more triangles and can be used to guide the genus simplification stage.

We have tested our approach on several real-life datasets and have achieved encouraging results. Results of our approach on a couple of polygonal mechanical CAD objects are shown in figures 6 and 7.

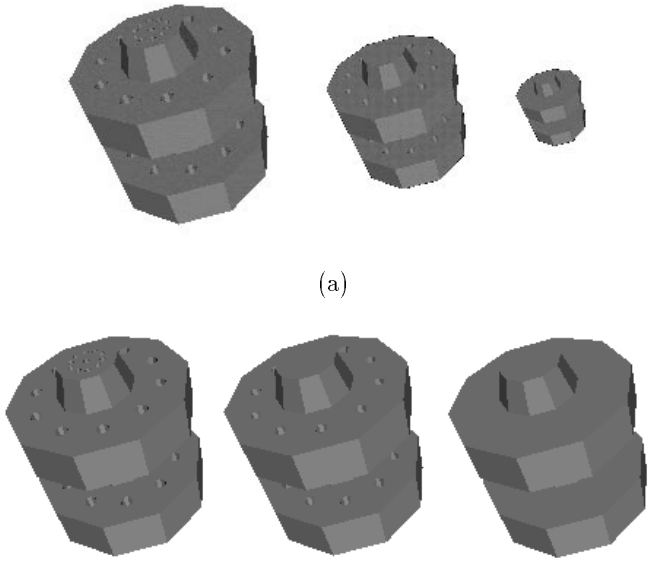
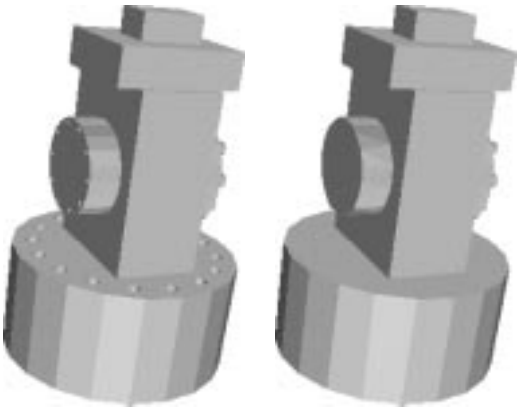


Figure 6. Hierarchical simplifications of the genus

In addition to two-manifold polygonal objects our approach also works in presence of some limited cases of non-manifold polygonal objects including those that have T-junctions and T-edges (shown in Figure 8). Such degeneracies are quite common in mechanical CAD datasets from the manufacturing industry due to numerical inaccuracies in computing surface-surface intersections and in conversion from B-reps to polygonal representations.

View-Dependent Simplifications

Level-of-detail-based rendering has thus far emphasized object-space simplifications with minimal feedback from the image space. The feedback from the image space has been



(a) (b)

Figure 7. Genus-reducing simplifications for an industrial CAD part

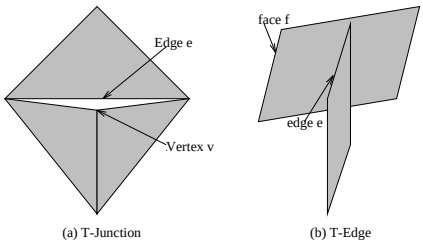
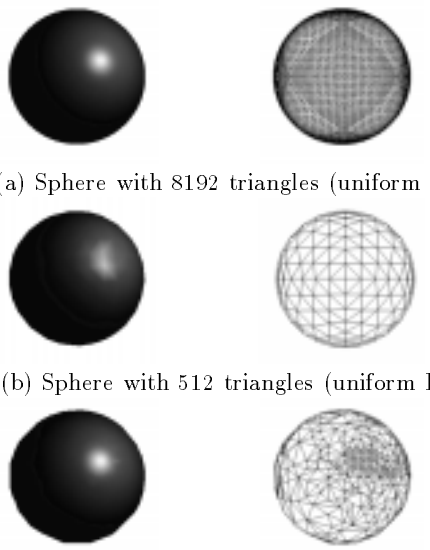


Figure 8. Permissible non-manifold degeneracies

in the form of very crude heuristics such as the ratio of the screen-space area of the bounding box of the object to the distance of the object from the viewer. As a result, one witnesses coarse image-space artifacts such as the distracting "popping" effect when the object representation changes from one level of detail to the next (Helman 95). Attempts such as alpha-blending between the old and the new levels of detail during such transitions serve to minimize the distraction at the cost of rendering two representations. However alpha blending is not the solution to this problem since it does not address the real cause – lack of sufficient image-space feedback to select the appropriate local level of detail in the object space; it merely tries to cover-up the distracting artifacts.

In contrast to the traditional approaches of precomputing a fixed number of level-of-detail representations for a given object we have developed an approach that involves statically generating a continuous level-of-detail representation for the object (Xia *et al* 97; Xia and Varshney 96). This representation is then used at run-time to guide the selection of appropriate triangles for display. The list of displayed triangles is updated incrementally from one frame to the next. Our scheme can construct seamless and adaptive level-of-detail representations on-the-fly for polygonal objects. Since these representations are view-dependent, they take advantage of view-dependent illumination, visibility, and frame-to-frame coherence to maximize visual realism and minimize the time taken to construct and draw such objects. Our approach shows how one can adaptively define such levels of detail based on (a) scalar attributes such as distance from the viewpoint and (b) vector attributes such as the direction of vertex normals. A simple example illustrating our approach is shown in Figure 9. Another example is shown in Figure 10.



(a) Sphere with 8192 triangles (uniform LOD)

(b) Sphere with 512 triangles (uniform LOD)

(c) Sphere with 537 triangles (adaptive LOD)

Figure 9. Uniform and adaptive levels of detail

IMAGE-BASED RENDERING

As the complexity of the three-dimensional object-space has increased beyond the bounded image-space resolution, image-based rendering has begun to emerge as a viable alternative to the conventional three-dimensional geometric modeling and rendering, in specific application domains. Image-space-based rendering has been used to navigate (although with limited freedom of movement) in environments modeled from real-world digitized images (Chen 95; Szeliski 96; McMillan and Bishop 95). More recent approaches (Gortler *et al* 96; Levoy and Hanrahan 96) generalize the idea of plenoptic modeling by characterizing the complete flow of light in a given region of space.

We next present an overview of our system that performs image-based rendering using image-space simplification and morphing. Given a collection of z-buffered images representing an environment from fixed viewpoints and view directions, our approach first constructs an image-

Highest detail (293K tris) 4.6 sec to render

Simplified (68K tris) 1.2 sec to determine and render

Figure 10. Dynamic adaptive simplification for the Buddha

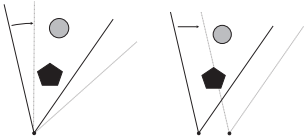
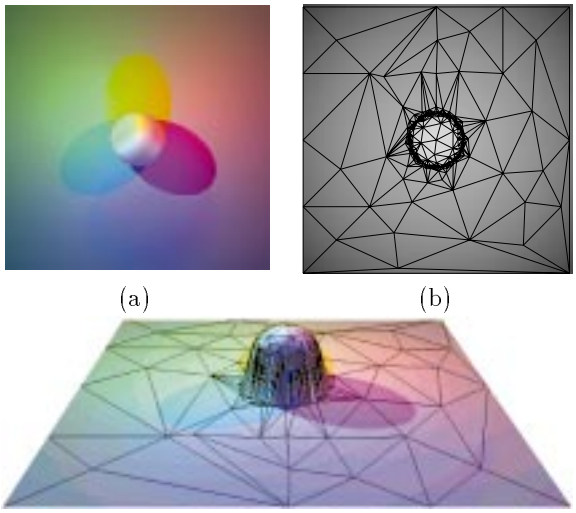


Figure 11. Visibility: (a) Rotation (b) Translation.

space simplification of the scene as a pre-process, and then reconstructs a view of this scene for arbitrary viewpoints and directions in real-time. We use the commonly available texture-mapping hardware for speed, and partially rectify the visibility gaps ("tears") pointed out in previous work on image-based rendering (Chen and Williams 93; Chen 95) through morphing.

Changes of visibility that occur as an observer moves freely in an environment can be simulated by using precomputed views of the scene at selected viewpoints. A *node* is associated with every selected viewpoint and consists of an environment map with depth and color information for every direction. This is essentially an extension to the plenoptic function, that also associates depth information to each direction, in addition to color. Each node provides limited information about the world, which is not sufficient to determine the view from an arbitrary viewpoint and direction. If the desired visualization parameters differ only in the direction, there is no parallax and no new areas can become visible (see Figure 11). If they differ in the position, however, parallax is introduced and the restricted information provided by a single node becomes evident.

This problem can be overcome by combining the information from neighboring nodes—through node morphing—to create an image for any viewpoint and direction. Morphing two nodes involves two warpings to register the information, followed by their combination (Gomes *et al* 95). The particular case of image morphing is extensively discussed in (Wolberg 90). The depth information and the visualization parameters allow the determination of the mapping functions between the original views and the new arbitrary view. After applying these mappings, the warped information can be combined with local control, used to determine the predominant information at each region. To generate environment maps for viewpoints intermediate to the previously selected nodes, we morph neighboring environment maps into an intermediate one.



(a) (b)

(c)

Figure 12. Range image triangulation: (a) Input image; (b) 2D texture coordinates; (c) 3D triangulation textured by input image

Our solution to the image-space-based rendering problem simplifies the environment by linearly approximating it with polygons, as seen from a given viewpoint. This polygonal mesh is created by triangulating the depth information associated with the environment map, as shown in the example in Figure 12(b). Each triangle in this mesh represents an object (or part of an object) at a certain depth. The parallax effect can then be correctly simulated by warping each of these triangles appropriately. Since image warping can be efficiently performed with hardware assistance through texture mapping, we determine the appropriate projective transformation which is then applied to this mesh textured by the environment map colors. The hardware z-buffer is used to resolve occlusions, or mesh *foldovers*. Multiple nodes are used to fill in the gaps resulting from mesh *tears* by combining z-buffered images from various nodes using

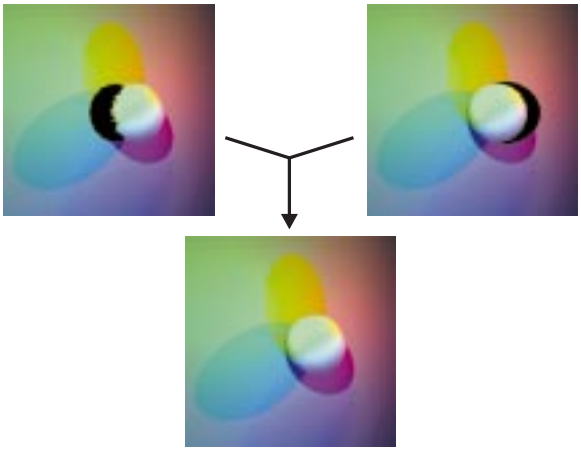


Figure 13. Blending from two nodes (visibility gaps are in black).

alpha blending and the stencil or the accumulation buffer (Open GL Architecture Review Board *et al* 93).

The polygonal mesh derived from the depth information is in fact a 3D triangulation that, when viewed from the original viewpoint, will look exactly like the flat image. The triangulation can be reprojected to any other arbitrary viewpoint in space by using standard viewing transformations, such as in the side view shown in Figure 12(c).

By applying the warping process described above to each node individually, we get two different z-buffered images $I_1^{z'}$ and $I_2^{z'}$—from nodes P_1 and P_2, respectively. What remains is the combination of $I_1^{z'}$ and $I_2^{z'}$, a range transformation B which, for each point (x, y), depends solely on the values of the z-buffered images at that position, resulting in the image $I_f^z = B(I_1^{z'}, I_2^{z'})$ as illustrated in figure 13. Different forms of this blending function are discussed and compared by us (Darsa *et al* 97).

The results of our system on a model of the Stonehenge generated by us appear in Figure 14. The initial model was ray-traced and two cubical environment maps, each consisting of six 256×256 images (with depth) were generated. From these $786K$ data points, we obtained a simplified representation consisting of a total of $30K$ texture-mapped triangles using a top-down approach to generate a Delaunay triangulation. We have tested our system on a single SGI Challenge R10000 processor with – one Raster Manager, Infinite Reality with 64 MB of texture memory, and 2 MB of secondary cache and have achieved frame rates of roughly 4 frames per second. After some further optimizations, our system runs on a Pentium Pro 200 MHz PC with the graphics card Permedia NT at 7.75 frames per second for 512×512 images.

CONCLUSIONS

Virtual reality is emerging as an enabling technology that has the potential to significantly aid the planning and design phases of CAD for large-scale mechanical structures. Real-time frame update rates are crucial for making virtual reality feasible for such a challenge. We have outlined several approaches for reconciling scene realism with interactivity for large datasets, particularly mechanical CAD datasets. These approaches are mutually complementary and support each other in achieving faster frame update rates without sacrificing scene realism. We have overviewed our approaches that compress the connectivity information for polygonal datasets and help in faster rendering. We have also outlined genus-preserving and genus-reducing simplifications as well as real-time view-dependent simplifications that are clearly useful in the visualization of large-scale mechanical CAD datasets. Image-based rendering is an emerging area within three-dimensional interactive graphics that has a significant potential for use in visualization and collaborative design. We have presented an overview of our system that helps in combining the missing visibility information from two or more neighboring nodes in such image-based rendering approaches.

ACKNOWLEDGMENT

This work has been supported in part by the National Science Foundation CAREER award CCR-9502239 and a Department of Defense DURIP award. Jihad El-Sana has been supported by a Fulbright/Arab-Israeli Scholarship; Francine Evans by fellowships from NSF and Northrop-Grumman; Lucia Darsa and Bruno Costa by CNPq (Brazilian Council of Scientific and Technological Development); Steven Skiena by ONR award 400x116yip01. The objects that appear in Figures 4 and 7 are parts of the dataset of a notional submarine provided to us by the Electric Boat Division of General Dynamics. The objects in Figure 3 have been provided to us by Viewpoint DataLabs. The happy Buddha in figure 10 has been provided by the Stanford Computer Graphics Laboratory.

REFERENCES

J. M. Airey. *Increasing Update Rates in the Building Walkthrough System with Automatic Model-Space Subdivision and Potentially Visible Set Calculations.* PhD thesis, University of North Carolina at Chapel Hill, Department of Computer Science, Chapel Hill, NC 27599-3175, 1990.

K. Akeley, P. Haeberli, and D. Burns. tomesh.c : C Program on SGI Developer's Toolbox CD, 1990.

C. Bajaj and M.-S. Kim. Generation of configuration space obstacles: the case of a moving sphere. *IEEE Journal of Robotics and Automation*, 4, No. 1:94–99, February 1988.

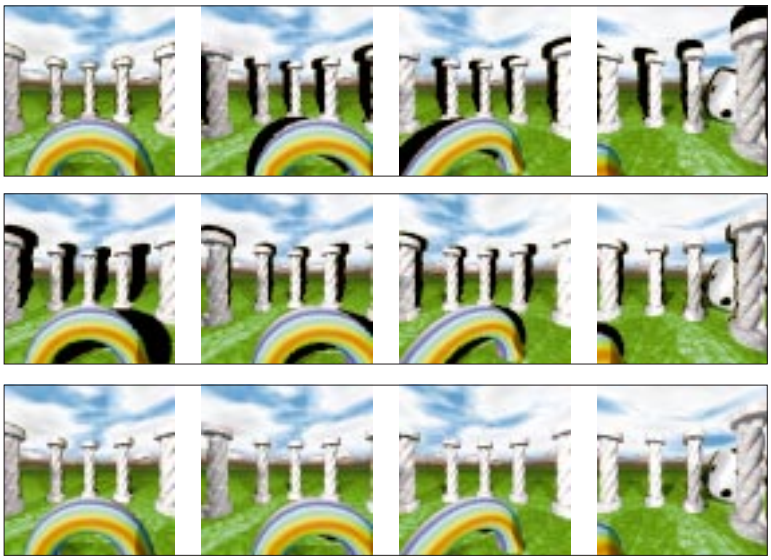


Figure 14. Navigating in Image-Space: Frames from Node 1 are in top row, Frames from Node 2 are in the middle row, blended frames are in the bottom row.

L. Bergman, H. Fuchs, E. Grant, and S. Spach. Image rendering by adaptive refinement. In *Computer Graphics: Proceedings of SIGGRAPH'86*, volume 20, No. 4, pages 29–37. ACM SIGGRAPH, 1986.

J. F. Blinn and M. E. Newell. Texture and reflection in computer generated images. *CACM*, 19(10):542–547, October 1976.

S. Chen. Quicktime VR – an image-based approach to virtual environment navigation. In *Computer Graphics Annual Conference Series (SIGGRAPH '95)*, pages 29–38. ACM, 1995.

S. Chen and L. Williams. View interpolation for image synthesis. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 279–288, August 1993.

J. Clark. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, 19(10):547–554, 1976.

J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. P. Brooks, Jr., and W. V. Wright. Simplification envelopes. In *Proceedings of SIGGRAPH '96 (New Orleans, LA, August 4–9, 1996)*, Computer Graphics Proceedings, Annual Conference Series, pages 119 – 128. ACM SIGGRAPH, ACM Press, August 1996.

F. C. Crow. A more flexible image generation environment. In *Computer Graphics: Proceedings of SIGGRAPH'82*, volume 16, No. 3, pages 9–18. ACM SIGGRAPH, 1982.

L. Darsa, B. Costa, and A. Varshney. Navigating static environments using image-space simplification and morphing. In *Proceedings, 1997 Symposium on Interactive 3D Graphics*, 1997.

M. F. Deering. Geometry compression. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 13–20. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

T. D. DeRose, M. Lounsbery, and J. Warren. Multiresolution analysis for surface of arbitrary topological type. Report 93-10-05, Department of Computer Science, University of Washington, Seattle, WA, 1993.

H. Edelsbrunner and E. P. Mücke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, January 1994.

F. Evans, S. Skiena, and A. Varshney. Optimizing triangle strips for fast rendering. In *IEEE Visualization '96 Proceedings*, pages 319 – 326. ACM/SIGGRAPH Press, October 1996.

F. Evans, S. Skiena, and A. Varshney. Efficient triangle strips for fast rendering. *ACM Transactions on Graphics*, 1997. (submitted).

T. A. Funkhouser and C. H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *Proceedings of SIGGRAPH 93 (Anaheim, California, August 1–6, 1993)*, Computer Graphics Proceedings, Annual Conference Series, pages 247–254. ACM SIGGRAPH, August 1993.

J. Gomes, B. Costa, L. Darsa, L. Velho, G. Wolberg, and J. Berton. *Warping and Morphing of Graphical Objects*. SIGGRAPH '95 Course Notes #3, 1995.

S. J. Gortler, R. Grzeszczuk, R. Szelinski, and M. F. Cohen. The lumigraph. In *Proceedings of SIGGRAPH '96 (New Orleans, LA, August 4–9, 1996)*, Computer Graphics

Proceedings, Annual Conference Series, pages 43–54. ACM SIGGRAPH, ACM Press, August 1996.

N. Greene. Hierarchical polygon tiling with coverage masks. In *Proceedings of SIGGRAPH '96 (New Orleans, LA, August 4–9, 1996)*, Computer Graphics Proceedings, Annual Conference Series, pages 65 – 74. ACM Siggraph, ACM Press, August 1996.

A. Guéziec. Surface simplification with variable tolerance. In *Proceedings of the Second International Symposium on Medical Robotics and Computer Assisted Surgery, MRCAS '95*, 1995.

B. Hamann. A data reduction scheme for triangulated surfaces. *Computer Aided Geometric Design*, 11:197–214, 1994.

T. He, L. Hong, A. Varshney, and S. Wang. Controlled topology simplification. *IEEE Transactions on Visualization and Computer Graphics*, 2(2):171–184, June 1996.

J. Helman. Graphics techniques for walkthrough applications. In *Interactive Walkthrough of Large Geometric Databases, Course Notes 32, SIGGRAPH '95*, pages B1–B25, 1995.

H. Hoppe. Progressive meshes. In *Proceedings of SIGGRAPH '96 (New Orleans, LA, August 4–9, 1996)*, Computer Graphics Proceedings, Annual Conference Series, pages 99 – 108. ACM SIGGRAPH, ACM Press, August 1996.

M. Levoy and P. Hanrahan. Light field rendering. In *Proceedings of SIGGRAPH '96 (New Orleans, LA, August 4–9, 1996)*, Computer Graphics Proceedings, Annual Conference Series, pages 31–42. ACM SIGGRAPH, ACM Press, August 1996.

D. Luebke and C. Georges. Portals and mirrors: Simple, fast evaluation of potentially visible sets. In *Proceedings, 1995 Symposium on Interactive 3D Graphics*, pages 105 – 106, 1995.

P. W. C. Maciel and P. Shirley. Visual navigation of large environments using textured clusters. In *Proceedings of the 1995 Symposium on Interactive 3D Computer Graphics*, pages 95–102, 1995.

L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *Computer Graphics Annual Conference Series (SIGGRAPH '95)*, pages 39–46. ACM, 1995.

A. Narkhede and D. Manocha. Fast polygon triangulation based on seidel's algorithm. *Graphics Gems 5*, pages 394–397, 1995.

Open GL Architecture Review Board. *OpenGL Reference Manual*. Addison-Wesley Publishing Company, Reading, MA, 1993.

Open GL Architecture Review Board, J. Neider, T. Davis, and M. Woo. *OpenGL Programming Guide*. Addison-Wesley Publishing Company, Reading, MA, 1993.

J. Rohlf and J. Helman. IRIS performer: A high performance multiprocessing toolkit for real–Time 3D graphics. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 381–395. ACM SIGGRAPH, July 1994.

J. Rossignac and P. Borrel. Multi-resolution 3D approximations for rendering. In *Modeling in Computer Graphics*, pages 455–465. Springer-Verlag, June–July 1993.

W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. In *Computer Graphics: Proceedings SIGGRAPH '92*, volume 26, No. 2, pages 65–70. ACM SIGGRAPH, 1992.

R. Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Comput. Geom. Theory Appl.*, 1:51–64, 1991.

J. Shade, D. Lischinski, D. Salesin, T. DeRose, and J. Snyder. Hierarchical image caching for accelerated walkthroughs of complex environments. In *Proceedings of SIGGRAPH '96 (New Orleans, LA, August 4–9, 1996)*, Computer Graphics Proceedings, Annual Conference Series, pages 75–82. ACM SIGGRAPH, ACM Press, August 1996.

Silicon Graphics, Inc. Graphics Library Programming Guide, 1991.

R. Szeliski. Video mosaics for virtual environments. *IEEE CG&A*, pages 22–30, March 1996.

S. Teller and C. H. Séquin. Visibility preprocessing for interactive walkthroughs. *Computer Graphics: Proceedings of SIGGRAPH'91*, 25, No. 4:61–69, 1991.

G. Turk. Re-tiling polygonal surfaces. In *Computer Graphics: Proceedings SIGGRAPH '92*, volume 26, No. 2, pages 55–64. ACM SIGGRAPH, 1992.

A. Varshney. Hierarchical geometric approximations. Ph.D. Thesis TR-050-1994, Department of Computer Science, University of North Carolina, Chapel Hill, NC 27599-3175, 1994.

G. Wolberg. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA, 1990.

J. Xia and A. Varshney. Dynamic view-dependent simplification for polygonal models. In *IEEE Visualization '96 Proceedings*, pages 327 – 334. ACM/SIGGRAPH Press, October 1996.

J. Xia, J. El-Sana, and A. Varshney. Adaptive real-time level-of-detail-based rendering for polygonal models. *IEEE Transactions on Visualization and Computer Graphics*, June 1997. (to appear).

H. Zhang, D. Manocha, T. Hudson, and K. Hoff. Visibility culling using hierarchical occlusion maps. In *To appear in Proceedings of SIGGRAPH '97 (Los Angeles, CA)*, Computer Graphics Proceedings, Annual Conference Series. ACM SIGGRAPH, ACM Press, August 1997.