Rendering Image-based Object Hierarchy

Tong-Yee Lee[1], Tai-Cheng Chen, Chao-Hung Lin

Visual System Laboratory
Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, TAIWAN, R.O.C
tonylee@mail.ncku.edu.tw
Phone: 886-6-2757575 ext. 62531
Fax: 886-6-2747076

Abstract

Image-based representations of objects are useful in computer game and virtual museum and art gallery applications. In this paper, we present a hierarchical image-based object (IBO) representation for a complex scene. In this hierarchy, we find a camera-sampling circle for each node and attempt to optimize the number of reference images that represent visible objects inside each node. While rendering this IBO hierarchy, we adaptively traverse this hierarchy to the levels with sufficient camera sampling rates for the output image. We experimentally evaluate the proposed schemes on a PC for complex scenes that consist of image-based objects. The proposed scheme achieves a near real-time rendering rate for such complex scenes with more than 1 million polygons.

Keywords: Image-based rendering, Image-based objects (IBO), 3-D warp, Hierarchical representation

1. Introduction

Interactive rendering extremely complex geometric environment has been an important research issue in computer graphics. Although the performance of graphics hardware has improved dramatically in recent years, the demand for higher performance continues to grow. Today, a virtual environment containing many millions of polygons becomes common. To rapidly visualize such complex scenes, the rendering algorithms must limit the number of polygons rendered in each frame. However, this limited number of polygons is still prohibitive to achieving a real-time rendering on most workstations. In past five years, image-based rendering (IBR) techniques have emerged as a potential solution to render such complex scenes. The basic concept of IBR methods is to employ a set of reference images to represent three-dimensional scenes instead of geometric primitives. With IBR methods, the computational complexity is independent of the number of geometric primitives. Therefore, IBR techniques are very attractive for interactive display of complex scenes.

Image-based object (IBO) is a compact, image-based representation for three-dimensional objects with complex shapes [1]. This representation can be very useful in games, virtual museum and art gallery applications. In some sense, the proposed method is similar to the concept of IBO, but different in many respects including reference image acquisition and representation, and rendering. Given a set of objects comprising a scene, we build a hierarchical bounding volume tree [2] by placing partitioning planes inside gaps between objects. In this tree, the geometry is stored only at the leaves of this hierarchy. For each node in this hierarchy, we find a camera sampling circle and then around the circle the optimized number of reference images are captured to represent the visible objects inside each node. During a walkthrough of the scenes, the proposed method adaptively traverses the hierarchy to the levels with sufficient camera sampling rates for the output image. Whenever a node sufficiently represents sampling of the output image, two of reference images stored on this node are selected to 3D warp. With the proposed method, a z-buffer is required to eliminate hidden surfaces. We implemented the proposed method and rendered scenes with more than one million polygons on a PC. The demonstrated rendering performance is very

[1] Corresponding author

promising near real-time with tolerable loss in rendering quality.

The remainder of this paper is organized as follows. The next section reviews related previous work. Section 3 presents schemes to construct an IBO hierarchy. We introduce the rendering of IBO hierarchy in Section 4. Experimental evaluation of the proposed schemes on a PC is given and discussed in Section 5. Finally, Section 6 closes with conclusions and future work.

2. Related Work

Shade et al. [3] and Schaufler et al. [4] employ the image caching or impostor technique to accelerate walkthroughs of complex scenes. Although the cached images can be reused in subsequent frames, the image impostors have to be regenerated whenever it is no longer suitable for the new view. Levoy et al. [5] and Gortler et al.[6] collect a large number of images in the databases to represent objects. At rendering time, the images in the databases are resampled to produce an interpolated view of the objects. McMillian and Bishop [7] present an efficient 3D image warping framework and a list-priority solution for the visibility problem. However, this algorithm cannot be used to warp multiple images acquired from different COPs simultaneously. Darsa et al. [8] and Pulli et al. [9] reconstruct meshes from a dense range maps to approximate a 3D image warp. To generate a new view, the meshes are blended on a per pixel basis and a z-buffer is used to eliminate hidden surfaces. Grossman et al. [10] use individual points or a set of points as samples in image-based rendering. These sampling points are rendered using a hierarchy of Z-buffer to detect tear artifacts. Shade et al. [11] use the Layered Depth Image (LDI) to deal with the disocclusion artifacts of image warping. Given a set of nearby reference images, the LDI is constructed by warping all these reference images to a carefully chosen camera setup. Later, Chang et al. [12] propose the LDI tree which combines a hierarchical space partitioning schemes with the concept of the LDI. This scheme preserves the sampling rates of the reference images by adaptively selecting an LDI in the LDI tree for each output pixel. Oliveira et al. [1] introduce the concept of image-based object (IBO) that is represented by six layered depth image sharing a single center of projection. The IBO can be used as a primitive to construct more complex scenes. In their work, a new list-priority algorithm for rendering such scenes and a back-face culling strategy is proposed. Rafferty et al. [13] attempt to improve upon a cells and portals framework by using images to replace geometry visible through portals in a 3D scene. The above techniques are related image-based rendering techniques to visualize complex objects and scenes.

On the other hand, several related non-IBR techniques for rendering large scenes are reviewed as follows. Clark [14] and Garlick et al. [15] investigate visibility culling algorithms to avoid drawing objects that not visible in the image. Aiery et al. [16] and Teller [17] consider the relationships of visibility within a complex scene. Only the potentially visible set of polygons is considered to render at each frame. Green et al. [18] propose a hierarchical z-buffer to achieve fast visibility culling. Another hierarchical method is to use Occlusion Maps [19] that does not use depth information to cull occluded geometry. Coorg and Teller [20] attempt to dynamically identify large occluders as the user moves. Then, these large occluders are used to perform culling using an octree structure.

3. Building IBO Hierarchy

In this section, we present the proposed schemes to create IBO primitives and to construct IBO hierarchy.

As a preprocessing step, we construct a BSP-tree like partitioning of a given scene. After this step, a compact bounding volume hierarchy [2] is constructed. Generally, the partition scheme must take the following criteria into account: (1) make the hierarchy as balanced as possible and (2) for each node in this hierarchy, make its bounding volume as compact as possible. The balanced hierarchy can help make the rendering time as constant as possible. Since we build an IBO for each node in the proposed hierarchy, the second criterion can influence image-sampling quality. However, these two criteria sometimes contradict each other. The compactness cannot always guarantee the perfectly balanced hierarchy.

Unlike original IBO [1], we build an IBO for each node in the proposed hierarchy. The proposed IBO in this paper can represent either a geometric object or a collection of geometric objects. For each IBO, we find an optimized number of reference images to represent objects by placing these selected cameras along a camera-sampling circle. The easiest way to acquire reference images is to place cameras along this circle in a uniform manner. There are several problems inherent in this approach: (1) fixed number of reference images potentially leads to too much redundancy (waste storage)

and (2) without considering geometric information, this method potentially misses some geometric details that lead to holes from image warp. Our method is described as follows. First, we uniformly place N cameras (i.e., one camera per $\frac{360}{N}$ degrees) along the camera-sampling circle and assume these N cameras can sufficiently cover all possible viewpoints. Let $P=\{P_i \mid i=1,2,...,N\}$, where P_i is the position of the ith camera on a given camera-sampling circle. C_i denotes the reference image captured at P_i and W_{mnk} represents the interpolated image at P_k by 3-D warping C_m and C_n, where $m<k<n$. Furthermore, we define an error function $\pounds_{mnk}$ for using W_{mnk} instead of C_k in equation (1).

$$\pounds_{mnk} = w_1 H + w_2 D \qquad (1)$$

,where H and D denotes the number of holes and the difference of depth between W_{mnk} and C_k, and w_1 and w_2 are weights for H and D.

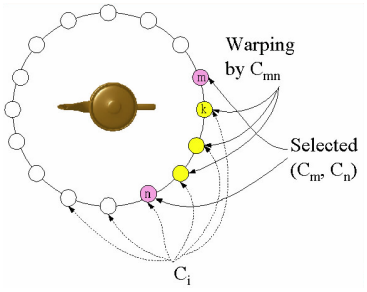


Figure 1. Optimal set of reference images

Considering Figure 1, let the sum of error functions for a selected (Cm, Cn) reference image pair to warp each P_k's image, k =m+1,m+2,…n-1, be denoted as $\pounds_{mn}$, $\pounds_{mn} = \sum_{k=m+1}^{n-1} \pounds_{mnk}$ .Our goal of camera placement is to find a optimal set of reference images on the camera-sampling circle from $|P|$ cameras that are placed uniformly. Assume we select $|S|$ reference images on this circle. Let $S=\{P_{s_1}, P_{s_2}, ..., P_{s_n} \mid S \subseteq P\}$, where each P_{s_i} is an optimally selected camera position from $|P|$ camera positions. Define error function for selecting these $|S|$ reference images by equation (2).

$$E = \sum_{\cup S} E_{P_{S_i} P_{S_{i+1}}} \qquad (2)$$

, where P_{S_i} is in clockwise order and $E_{P_{S_i} P_{S_{i+1}}}$ denotes the error sum of warping $C_{P_{S_i}}$ and $C_{P_{S_{i+1}}}$ for the other images captured (i.e., by uniform placement) between cameras P_{S_i} and $P_{S_{i+1}}$. Apparently, equation (3) is minimized as $|P| = |S|$. To find a minimal set of reference, we need to have a constraint on $|S|$ and to minimize equation (3).

$$\sum_{\cup S} E_{P_{S_i} P_{S_{i+1}}} + w_3 \times |S| \qquad (3)$$

, where w_3 is weight for $|S|$. In the course of minimizing equation (3), if the number of holes is over a threshold, we let $E_{P_{S_i} P_{S_{i+1}}} = \infty$ to ensure the quality of image warp. Currently, we employ ***dynamic programming*** technique [21] to solve the minimization of equation (3) and thus find these $|S|$ optimally selected reference images.

4. Rendering IBO Hierarchy

Once the IBO hierarchy has been built, the following pipeline (as shown in Figure 2) is used for display. In this pipeline, for each frame, visibility culling includes frustum culling, background pixel culling on each reference image and reference image selection per visible IBO node. Additionally, for each visible reference image, we compute on-line its multi-resolution representation to reduce number of warping pixels. Then, we compute splatting size of 3D warping and splats are drawn with Z-buffer enabled to resolve occlusion. To reduce number of holes, we simply find and fill these holes depending on their non-hole neighboring pixels. Once the position of a viewer is inside the bounding volume of leaf nodes in IBO hierarchy, we draw polygons instead of splats.

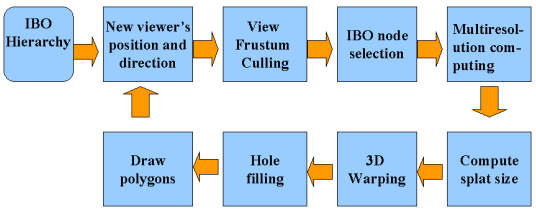


Figure 2. Pipeline of rendering IBO hierarchy

IBO_Hierachy_Render (node, viewpoint)
Begin
    if node is outside the view frustum
        return

```
    else if viewing position is outside this
    node
        select two reference images for
        this node
        using distance metric, compute
        multi-resolutions, if required
        compute splatting size
        3-D warping
    else if this node is a leaf
        draw polygons insdie this node
    else
        IBO_Hierachy_Render
        (node.leftchild, viewpoint)
        IBO_Hierachy_Render
        (node.rightchild, viewpoint)
End
```

The above algorithm is used for display and several stages of this algorithm are examined in detail in the later subsections. Considering situation in Figure 3, the original reference image (b) is captured on the camera-sampling circle. If we render the object from the center of projection of image (c) by warping the image (b), the possibly insufficient sampling rate of the image (b) (i.e., inside the circle) will generate many holes (i.e., zoom-in effect) in the image (c). In contrast, when we render the object from the center of projection of image (a) (i.e., outside the circle) by warping the image (b), the excessive sampling rate of the image (b) might not affect the quality of the output. However, it wastes time to process more pixels than necessary. In above rendering algorithm, if the viewer is at both (a) and (b) positions, we warp the image (b). When the viewer is at (c), we continue the next level in IBO hierarchy to find appropriate reference images with sufficient sampling condition. Although our approach is conservative, at least, we maintain sufficient sampling condition of warping. Later, we will attempt to reduce processing redundancy due to over-sampling as the viewer is at (a) position.

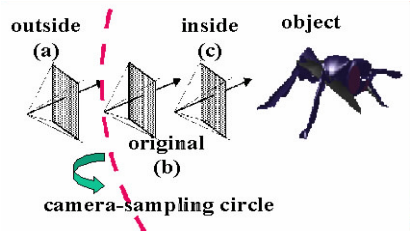


Figure 3. Finding sufficient sampling rate of reference image

4.1 Multiresolution

Considering Figure 4, there are two cameras with COP at V_1 and V_2 and an identical focal length d. P_1 and P_2 are two points in 3D space and the distance from V_1 and V_2 to $\overline{P_1P_2}$ are d_1 and d_2. Let l_1 and l_2 be projected length of $\overline{P_1P_2}$ on both image planes. We can have $\frac{L}{l_1}=\frac{d_1}{d}$ and $\frac{L}{l_2}=\frac{d_2}{d}$. Thus, $\frac{l_1}{l_2}=\frac{d_2}{d_1}$. Based on this simple metric, we compute lower resolution as the viewpoint is at far distance from objects. In this manner, we attempt to avoid warping more pixels than necessary.

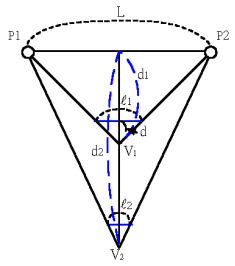


Figure 4. Distance metric of multiresolution;

4.2 Hole Filling

Our strategy for detecting holes is very naive but very fast. We simply detect holes by checking their non-hole neighboring pixels using a 3×3 mask. Considering Figure 5, if any of the following is true, the pixel at 0 is determined as a hole and it is filled with a color that is averaged from those of its non-hole neighbors.

$$1\equiv 8, 3\equiv 6, 2\equiv 7, 4\equiv 5 \qquad (5)$$

, where $a\equiv b$ returns true, if both a and b are similar in color and depth. This method only can fix one-pixel wide hole and result in blocky artifacts. To better treat hole problem, Grossman et al. [10] suggest a hierarchy Z-buffer. In future, we might include this option to treat holes.

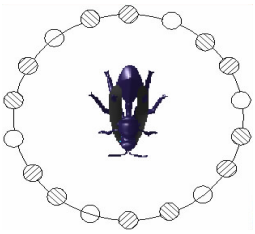
1	2	3
4	0	5
6	7	8

Figure 5. Hole detecting mask

5 Experimental Results

We implemented the proposed methods on a Pentium II 350 PC with a 224M main memory and a GeForce2 GTS/32M graphics card. We built a system prototype in Visual C++ and OpenGL. First, we evaluate the proposed camera placement using an example as shown in Figure 6. In this example, the threshold for the parameter H is 10% of the number of image pixels. We originally capture 18 reference

images in a uniform manner and from them we select 6 images after the optimization of the proposed method. To experimentally evaluate the proposed method, we also uniformly capture 6 reference images as indicated in Figure 6. In Figure 7, (a) is outputted by rendering polygons. From (b) and (c), the proposed method seems better than the uniform method. The uniform method generates more holes as marked by a circle.



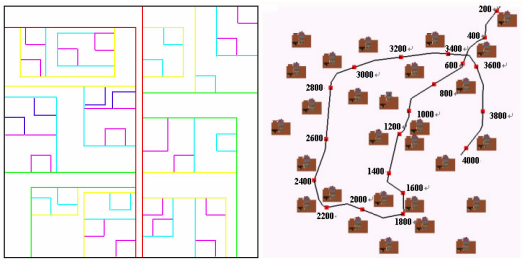
▧ : 6 captured positions of the proposed method
▨ : 6 captured positions of the uniform method

Figure 6. Captured positions of the proposed and uniform methods

(a) (b) (c)

Figure 7. Evaluating the proposed camera placement algorithm: (a) polygon drawing; (b) proposed method; (c) uniform method

Next, we evaluate the proposed method of rendering IBO hierarchy. In the test scene, there are many computer-tables on the ground plane. In total, there are 1,149,680 polygons in this test scene. Using the proposed method, the scene is partitioned, as shown in Figure 8 (a), and the resulting IBO hierarchy is built with 53 nodes and in total 492 reference images used. Before the optimization, we capture 18 reference images uniformly on each camera-sampling circle. Each reference image and output is rendered at 256×256 resolution with FOV = 90 degrees. Figure 8 (b) shows a bird's eye view of the test scene and each viewpoint on two 4000-frame walkthrough path. There are several viewpoints labeled frame number along the path. In our experiments, the viewer is currently limited to walk on the ground. The first walkthrough was performed using a hierarchical view frustum culling (using the same IBO hierarchy) but rendering all of the original polygons included in leaf nodes that are inside the view frustum. The second walkthrough was performed using the proposed method. Figure 9 shows several frames rendered in two walkthroughs. The images in the first row are very similar to those in the second row.



(a) (b)

Figure 8. (a) The partitioning of the scene; (b) A bird's eye view of the test scene.

(a)1500th frame (b)1800th frame (c)2100th frame

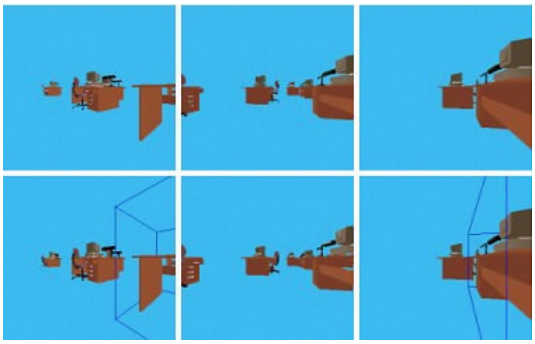


Figure 9.Frames from walkthroughs of the test scene. The first row shows four frames rendered using the original polygons. The second row shows the same frames rendered using the proposed method. In the second row, if the node is rendered using polygons, its bounding volume is also shown.

The plot in Figure 10 shows the frame rate performance for the two walkthroughs. On the average, the proposed method achieves about 23 frames/second and the other achieves about 2 frames/second. With our method, the peak frame rate is around 35 to 40 frames/second, but the slowest is about 8 to 10 frames/seconds. This performance difference is due to the number of drawn polygons. When the number of drawn polygons increases, the performance goes down such as frames around 1000. Figure 11 shows the number of drawn polygons using the proposed method in the 40000-frame walkthroughs. There is a peak with regard to the number of drawn polygons around 1000 frames, too. Next, Figure12 shows the number of warped images and IBO nodes rendered using polygons. The number of warping reference images is significantly more than that of number nodes rendered using polygons. Thus, the proposed method can save rendering cost.

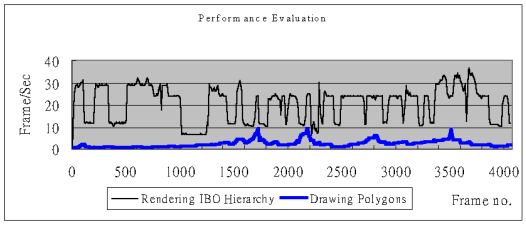


Figure 10. Frame rate performance comparison: rendering IBO hierarchy vs. drawing polygons

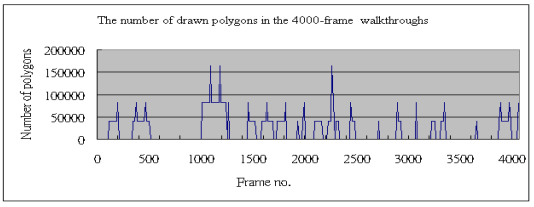


Figure 11. The number of drawn polygons using the proposed method

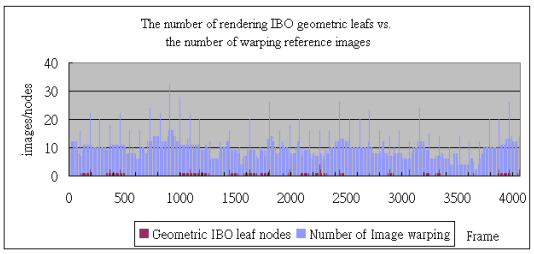


Figure 12. The number of rendered primitives: IBO leaf nodes vs. the number of reference images.

6 Conclusions and Future Work

The proposed rendering of IBO hierarchy has demonstrated a near real-time performance for a complex scene with more than one million polygons. The proposed scheme first partitioned the scene and built an IBO hierarchy. Each node in the IBO hierarchy is represented by a selected set of reference images. Depending on the viewpoint, we select IBO nodes with appropriate sampling condition. These selected nodes are rendered by two reference images. When the viewer is inside the leaf nodes, the polygons contained in these leaf nodes are rendered instead. Thus, the proposed scheme is a kind of hybrid technique: image-based and geometry-based. Although we achieve very promising results, there are several extensions to be done in near future as follows.

- Currently, the viewer is limited on the ground plane due to the fact that we captured the reference images per IBO node on a camera-sampling circle. All cameras are placed on the same plane. This restricts the movements of viewer as well as visual quality if viewer is allowed to move freely. It is easy to extend the camera-sampling circle to a sphere in 3D. For example, we uniformly place cameras on a sphere. Each location can be identified by (,). Then, we divide the (,) into square regions and employ dynamic programming technique to find an optimal set of (,) to represent an IBO node. During rendering time, the four corners of the region enclosing the desired viewpoint are used.
- The current hole-filling method is very naïve. This method can not fix holes with more than two-pixel width. Another limitation is that it is difficult to properly sample thin spoke-like structure. Additionally, it can not guarantee correct detection of holes. Actually, this is really a difficult problem. A better approach to treat above problems will be further investigated.
- In preprocessing, we statically built the IBO hierarchy. There are many potentials to miss some sampling details and thus to yield holes from arbitrary viewpoints. We can record the locations of uniform camera placement before optimization. When a viewpoint and its direction is very different from these recorded information, we can dynamically create a new reference image by rendering polygons. Then, this new image exists and is warped until it is replaced by the other new image. In this manner, we tradeoff the performance and visual quality.
- Finally, we should say the proposed method is kind of tradeoff between the performance and storage. Although the cost of memory storage continues to decline, we plan to employ image compression technique to reduce the storage of these reference images.

Acknowledgement

This paper is supported by National Science Council, Republic of China, Taiwan, under contract No. NSC 89-2218-E-006-028.

Reference

1. Oliveira, Manuel M. and Gary Bishop, “Image-based Objects,” Proceedings of 1999 ACM Symposium on Interactive 3D Graphics, April 26-28, 1999, pp. 191-198.
2. Timothy L. Kay and James T. Kajiya, “Ray tracing complex scenes,” Computer Graphics, 20(4):269-278, August 1986.
3. Jonathan Shade, Dani Lischinski, David H. Salesin, Tony DeRose and John Snyder, “Hierarchical Image Caching for Accelerated

Walkthrough of Complex Environments," In Proceedings of SIGGRAPH 1996, pages 75-82.
4. Gernot Schaufler and Wolfgang Sturzlinger, " A Three Dimensional Image Cache for Virtual Reality, " In Proceedings of Eurographics'96, pages 227-236. August 1996.
5. Marc Levoy and Pat Hanrahan, "Light Field Rendering," In Proceedings of SIGGRAPH 1996, pages 31-42.
6. Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski and Michael F. Cohen, "Lumigraph," In Proceedings of SIGGRAPH 1996, pages 43-54.
7. Leonard McMillan and Gary Bishop, "Plenoptic Modeling," In Proceedings of SIGGRAPH 1995, pages 39-46.
8. Darsa, Lucia, Bruno Costa Silva, and Amitabh Varshney, "Navigating Static Environments Using Image-Space Simplification and Morphing," Proceesings of 1997 Symposium on Interactive 3D Graphics, April 27-30, 1997, pp. 25-34.
9. Pulli, Kari, Michael Cohen, Tom Duchamp, Hugues Hoppe, Linda Shapiro and Werner Stuetzle, "View-based Rendering: Visualizing Real Objects from Scanned Range and Color Data," Proceedings of 8th Eurographics Workshop on Rendering, 1997, pp. 23-34.
10. Grossman, J., Dally, W., "Point Sample Rendering," Proceedings of the 9th Eurographics Workshop on Rendering, Vienna, Austria, June 1998, pp. 181-192.
11. Jonathan Shade, Steven Gortler, Li-wei He and Richard Szeliski," Layered Depth Images," In Proceedings of SIGGRAPH 1998, pages 231-242.
12. Chang, Chun-Fa, Gary Bishop, Aneslmo Lastra, "LDI Tree: A Hierarchical Representation for Image-based Rendering," In Proceedings of SIGGRAPH'99, pp.291-298.
13. Rafferty, Matthew M., Daniel G. Aliaga, Voicu Popescu, Anselmo A.Lastra," Images for Accelerating Architectural Walkthroughs," IEEE Computer Graphics and Applications, November/December, 1998.
14. James H. Clark, "Hierarchical geometric models for visible surface algorithms," Communications of the ACM, 19(10):547-554, October, 1976.
15. B. Garlick, D. Baum, and J. Winget, "Interactive viewing of large geometric databased using multiprocessor graphics workstations," SIGGRAPH'90 Course Notes: Parallel Algorithms and Architectures for 3D Image Generation, 1990.
16. John M. Aiery, John H. Rohlf, and Fredriick P. Brooks, Jr., "Towards image realism with interactive update rates in complex virtual building environments," Computer Graphics (1990 Symposium on Interactive 3D Graphics), 24(2):41-50,March 1990.
17. Seth J. Teller, "Visibility Computations in Densely Occluded Polyhedral Environments," Ph.D Thesis, Computer Science Division (EECS), UC Berkeley, Berkeley, California 94720, October 1992, Available as Report No. UCB/CSD-92-708.
18. Ned Greene, Michael Kass, and Gavin Miller, "Hierarchical z-buffer visibility," In Computer Graphics Proceedings, Annual Conference Series, pp. 231-238, August 1993.
19. H. Zhang et al.," Visibility Culling Using Hierarchical Occlusion Maps," Computer Graphics (Proceedings of SIGGRAPH'97), pp. 77-88, August 1997.
20. S. Coorg and S. Teller, "Temporally coherent conservative visibility," Proceedings of 12th Annual ACM Symposium on Computational Geometry, 1996.
21. Thmomas H. Cormen, Charles E.Leiserson and Ronald L. Rivest, "Introduction to Algorithm," The MIT Press, 1990, ISBN 0-07-013143-0 (McGraw-Hill).
22. X.Decoret, G. Schaufler, F. Sillion and J. Dorsey, "Multi-layered Impostors for Accelerated Rendering," Proceedings of Eurographics'99, C-61-72, 1999.
23. Szymon Rusinkiewicz, Marc Levoy, "Qsplat: A Multiresolution Point Rendering System for Large Meshes," to appear in the Proceedings of SIGGRAPH'2000.