

Generation of Decomposition Hierarchies for Efficient Occlusion Culling of Large Polygonal Models

Michael Meißner*, Dirk Bartz*, Gordon Müller†, Tobias Hüttner*, and Jens Einighammer*

Abstract

Efficient handling of large polygonal scenes has always been a challenging task and in recent years, view-frustum and occlusion culling have drawn a lot of attention for accomplishing this goal. The problem of how to efficiently organize such scenes for fast image synthesis is widely neglected, although the answer heavily affects the overall performance. In this paper, we present three adapted algorithms for efficient scene decomposition and compare those with another already available algorithm for decomposing general polygonal models into a hierarchy of sub-models for an occlusion culling application. While the latter is available as a commercial product, the three other approaches introduce new algorithms for scene decomposition which achieve significant better results.

Keywords: Large Polygonal Models, Hierarchical Scene Organization, Occlusion Culling.

1 Introduction

Hierarchical methods are crucial for the efficient handling of large computer graphics scenes. Several approaches address the problem of generating scene hierarchies, such as subdivision surfaces [28], multi-resolution frameworks [14], scene databases [27], or regular decomposition schemes such as BSP-trees [8] or octrees [21]. An important application field of scene organization is view-frustum and occlusion culling, which specifically requires appropriate decomposition schemes. However, most available approaches lack either flexibility or efficient handling.

*University of Tübingen, WSI/GRIS, Email: {meissner,bartz}@gris.uni-tuebingen.de

†TU Braunschweig, Computer Graphics Group, Email: gordon.mueller@tu-bs.de

1.1 Related Work

Several previous papers on visibility and occlusion culling touch the topic of scene organization. Generally, we observed that decomposition schemes for arbitrary polygonal models are difficult to derive. Hong et al. [13] used a technique which decomposes a CT-based colon volume dataset along its skeleton. The size of the different decomposition entities depends on how many voxels belong to this entity. While this scheme produced good results for a tube-like colon model, it is not efficient for general models.

Snyder and Lengyel [24] proposed that the designer of the scene needs to provide the scene organization. Similarly, Zhang et al. used a pre-defined scene database [27, 15]. Models built in large Computer-Aided-Design (CAD) systems might already include appropriate hierarchical organization information, due to the design process which uses hierarchical notions like grouping and replication. Besides approaches which use a building floor plan for this purpose [1, 2, 26, 18], no other methods for deriving decomposition hierarchies from CAD models are known to us.

A more general approach is to organize a polygonal model into regular spatial decomposition schemes, such as BSP-trees [8, 20, 11] or Octrees [10, 12, 6, 7, 25]. While these decomposition schemes produce good results on polygonal models extracted by the Marching Cubes algorithm from uniform grid volume datasets — which provide a “natural” decomposition on Marching Cubes cell base —, these schemes run into numerous problems on general models. If a polygon of the model lies across a decomposition boundary, it must be either split into several parts, in order to produce a disjunct representation of the bounding entities, or handled in another special way. Splitting polygons however, can increase the number of small and narrow polygons tremendously.

Significant work on model organization has been

published in the field of collision detection. Methods based on oriented bounding boxes (OBB) were explored by Gottschalk et al. [9]. A bottom-up approach for the construction of a model hierarchy is suggested in [3] in which nodes representing small parts of the geometry are “merged” into higher hierarchy nodes. A similar approach is used in [17]. Related model organization strategies are also known from global illumination.

In [4], the spatialization functionality of SGI’s OpenGL Optimizer package [23] was used to generate scene hierarchies automatically. However, these decomposition hierarchies needed to be tuned manually in order to get sufficient performance and motivated the work described in this paper.

2 Occlusion Culling

Generally, a polygonal scene can be decomposed into a hierarchical or non-hierarchical decomposition, denoted *scene tree* or *scene graph*. Each part of it is a *scene entity* which can be either be a *scene node* or a *geometry node*. The latter contains geometry of the actual scene.

The decomposition quality of the four presented approaches is evaluated with a standard hierarchical occlusion culling algorithm [5] based on the HP-Occlusion-Culling-Flag [22] and a basic view-frustum culling algorithm [4]. A given scene tree hierarchy is processed in top-down, left-right order. Initially, the actual polygons of the geometry node closest to the eye is rendered into the empty framebuffer. Then, each scene entity of the tree is tested for intersection with the view-frustum (view-frustum culling). Thereafter, all the remaining scene entities are depth-sorted according to their associated bounding boxes and tested for occlusion using the HP flag. All further rendering is performed in an interleaved fashion. First, bounding box of the next closest scene entity is rendered in a special occlusion mode which does not alter the framebuffer. If this bounding box would not have any contribution to the framebuffer, the HP Occlusion Culling Flag is set FALSE by the graphics hardware and consequently this scene entity is culled. Otherwise (flag is set TRUE), the child scene entities are processed unless the entity itself is a geometry node; in this case, the polygons are rendered in standard render mode.

3 Generating Decomposition Hierarchies

Generating hierarchical decomposition of very large models can be done in numerous ways. In this section, we will present three research algorithms and one commercial tool which generate a decomposition hierarchy starting from given models: *D-BVD*, *p-HBVO*, *ORSD*, and *SGI*.

The latter is part of SGI’s OpenGL Optimizer toolkit (ooptimize), while the other three novel algorithms have been developed and implemented by the authors. All approaches decompose general polygonal models, whereas the octree-based ORSD only decomposes regular grid datasets.

3.1 Dimension-oriented Bounding Volume Decomposition (D-BVD)

The goal of the dimension oriented D-BVD decomposition algorithm is to generate evenly-sized, cube-shaped bounding boxes, hence minimizing the area of the screen projection of these bounding boxes. This goal is approached by splitting the bounding boxes multiple times in the largest dimension. The size of the bounding boxes and the associated sub-models is controlled by user-specified parameters, such as the minimal number of polygons.

Starting with the root scene entity – which contains the whole model – the associated bounding volume (bv) is split n times along its largest dimension such that each fraction is approximately of the same size as the second largest dimension.

$$n = \frac{\text{largest bv dimension}}{\text{second largest bv dimension}} \quad (1)$$

This process continues recursively until the termination criteria are met. These criteria give lower bounds for the number of polygons of the scene entities or the size of the dimensions of the associated bounding boxes, in order to avoid undersized decomposition entities which increase the occlusion culling overhead without improving the cull rate sufficiently.

Occasionally, the split-operation of the decomposition process splits a polygon which lies across the decomposition boundary into two new polygons (this is usually not the case for uniform grid datasets). This can tremendously increase the number of polygons and frequently, these polygons are

small and narrow, thus resulting in numerical problems. To compensate this, we apply two techniques. First, the decomposition plane is moved along the decomposition dimension to reduce the number of additional polygons. The direction and value of the movement is controlled by user-specified parameters. Nevertheless, very large polygons cannot be handled by this technique, because they cover several high-level scene entities (i.e., a single polygon representing the floor of a factory). To account for this, we use a second technique, where those polygons are *pulled up* into a geometry node located at the appropriate level in the scene tree.

In general, this algorithm can handle all types of polygonal scenes without producing significantly more polygons. It optimizes shape and size of the scene entities, hence their bounding boxes. However, the polygon load is not evenly distributed to the scene entities, possibly resulting in a less balanced scene tree.

3.2 Polygon-based Hierarchical Bounding Volume Optimization (p-HBVO)

The polygon-based Hierarchical-Bounding-Volume-Optimization (*p-HBVO*) method decomposes recursively a set of polygons into two sets. Instead of arbitrarily selecting possible decomposition planes, these planes are given by the barycenter of each polygon (triangle). By evaluating a cost-function, an optimal decomposition plane is established. At each decomposition level, the individual polygons are assigned to exactly one scene entity of that level resulting in possibly partially overlapping bounding boxes but in no additional polygons.

Starting from the root node, at each decomposition step, we sort the polygons along all coordinate axes, where the barycenter of each polygon serves as sorting key. Based on these three ordered lists, we evaluate the potential decomposition planes along each axis for each entry in the respective list by splitting the sorted list of polygons into a *left* and *right* part. In contrast to pre-defined decomposition planes of the median cut scheme [16], we evaluate for each possible decomposition plane – defined by the entries in the lists – a cost function which approximates the costs of rendering the polygons of one of the two scene entities, generated by the respective decom-

position plane. By minimizing this cost-function over all possible decomposition planes, we obtain an optimal decomposition plane which generates two new scene entities; one contains all *left* polygons, the other one contains all *right* polygons. The decomposition process terminates when either the number of polygons, or the decomposition depth exceeds one of the two pre-defined parameters: *Max_Triangles_Per_Decomposition_Entity* or *Max_Decomposition_Depth*. These parameters are specified by the user and supplied at the start of the decomposition process.

In most cases, our cost function is identical to one which has already been successfully applied in ray tracing environments [19]. Adopting this cost function is possible since the objective is the same; both algorithms traverse the scene graph in a similar way to determine visibility. The costs of a scene entity H , with left H_l and right H_r children, is given by:

$$C_H(\text{axis}) = \frac{S(H_l)}{S(H)} \cdot |H_l| + \frac{S(H_r)}{S(H)} \cdot |H_r|$$

where

- $|H|$ is the number of polygons within hierarchy H ,
- $S(H)$ the surface area of the bounding box associated to sub-scene H , and
- $\text{axis} \in \{X, Y, Z\}$.

Overall, this algorithm generates well balanced scene trees with respect to their polygon load. Furthermore, polygons of individual objects are detected and clustered together (see Section 4, city dataset). The highest culling performance was achieved with finer decompositions, which usually require more occlusion culling tests, resulting in higher culling costs. If these culling costs are not compensated by lower rendering costs, it results in an overall lower rendering rate (see ventricular system in Table 1).

3.3 Octree-based Regular Space Decomposition (ORS)

As mentioned earlier, regular decompositions are well suited for uniform grid datasets, such as generated by MRI or CT scanners (i.e., ventricle dataset). Uniform grid datasets consist of a set of sample values (voxels), arranged on an uniform grid. A cube of eight neighboring voxels is called a cell, where cells with a non-empty intersection with the selected isosurface are called relevant cells.

The Octree-based Regular Space Decomposition method (ORSD) exploits these natural decomposition borders to generate a non-polygon splitting model decomposition. In contrast to the other approaches, ORSD uses a cell-based (voxel-based) evaluation criterion, where the number of relevant cells (relevant cell load or RCL) controls the decomposition process. This criterion is only a rough approximation of the actual number of extracted polygons, considering that each relevant cell represents between one and five triangles. In our experiences however, RCL turned out to be sufficiently accurate.

After the construction of the entire octree, the RCL of each octant is already calculated. Subsequently, the octree is traversed recursively, starting with the superblock. If the RCL is above a user-specified threshold, the block is classified as a scene node, thus being further decomposed into its child blocks. Otherwise, a geometry node is instantiated with the corresponding polygons (associated relevant cells).

Overall, ORSD is a simple but efficient decomposition scheme which generates an adequate polygon load balance and bounding box sizes. As shown in the results (see Section 4), the indirect evaluation method (RCL instead of number of polygons) does not adversely affect the occlusion culling performance. However, the implemented version of ORSD is limited to regular grids.

3.4 SGI's opoptimize (SGI)

SGI's OpenGL Optimizer is a C++ toolkit for CAD applications that provides scene graph functionality for handling and visualizing large polygonal scenes. It includes mechanisms for the decomposition of model databases as well as for tessellation, simplification, and others.

Opoptimize (depicted in the following sections as "SGI"), which is part of the toolkit, provides functionality for the decomposition of model databases. The decomposition method realized in SGI is similar to the construction of an octree; each geometry set is split into eight equally sized sets. This process is repeated recursively, until a certain threshold criteria for the iterated decompositions is reached.

Octree-based spatial decomposition is a simple and efficient decomposition scheme. However, the SGI decomposition mechanism decomposes space not by simply bisecting edges of a cube, as in an

octree, but by choosing decomposition planes such that the rendering loads of the resulting parts are similar. As a result, the amount of geometry after each decomposition on each side of the decomposition plane is approximately the same. Polygons which are split due to the decomposition are distributed to the respective geometry nodes. The main parameters that can be used to control the decomposition are hints for the lowest and highest amount of triangles (tri_{min}, tri_{max}) in each geometry node at the leaf-level of the decomposition hierarchy. However, the decomposition algorithm only tries to meet these criteria but is not bound to it.

In general, SGI generates decomposition hierarchies with a well balanced polygon load. However, the bounding boxes of the scene nodes are less suited for occlusion culling applications, because the cost function determining the subdivision is obviously not optimized with respect to the volume of the bounding boxes. We observed that the right-most branch of the scene tree frequently contained large subsets (bounding box volume size) of the model, even in the lower tree levels.

4 Results

In this section, we discuss the efficiency of our three novel algorithms and optimize of SGI's OpenGL Optimizer (SGI) with respect to their occlusion culling-based render performance. All measurements are performed on a HP B180/fx4 graphics workstation. The three different polygonal datasets (see Table 2) represent typical scenarios of different application areas. During our evaluation, we measured the time spent for view-frustum culling, occlusion culling, and rendering of the not occluded geometry of different decomposition granularities. For the evaluation, the decomposition with the best culling-based render performance was selected at the averaged results are shown in Table 1. In addition, frame rates of walk-throughs of the datasets and the respective render rate, which mirrors the percentage of the geometry of the scene which was rendered (this is reciprocal to the cull rate) are presented, see Figure 2.

Ventricle Dataset

The first dataset is a polygonal model of the ventricular system of the human brain extracted from

Ventricle				Cathedral dataset			City dataset		
pHBVO	D-BVD	ORSD	SGI	pHBVO	D-BVD	SGI	pHBVO	D-BVD	SGI
80	41	6	29	9	59	51	2722	266	420
81	41	28	36	10	67	52	2723	495	420
0.008	0.007	0.003	0.004	0.002	0.008	0.01	0.02	0.01	0.01
0.03	0.02	0.01	0.016	0.004	0.032	0.03	0.05	0.03	0.03
0.05	0.06	0.06	0.06	0.136	0.124	0.14	0.01	0.08	0.08
16.0	19.7	22.4	19.7	30.0	25.9	30.1	0.1	4.1	3.6
12.3	13.4	15.3	13.6	12.4	8.8	7.8	14.0	10.9	11.8
29.4	21.3	17.0	21.1	33.1	25.9	22.2	47.0	50.7	50.4
5.5	5.3	5.3	5.3	12.4	10.8	9.5	1.0	1.4	1.4

Table 1: Walk-through Measurements: rows from top to bottom: #Scene nodes. #Leaf nodes. View-frustum culling time [s]. Occlusion culling time [s]. Rendering time [s]. Rendering rate [% of original polygon count]. Frame rate [fps]. Rendering rate [%], view-frustum culling only. Frame rate [%], view-frustum culling only.

Dataset	Grid Type/ Source	#Triangles	FR
Ventricular System	Uniform/ MRI	270,882	4.6
Cathedral	Unstructured/ CAD	416,763	3.8
City	Unstructured/ Modeler	1,408,152	0.9

Table 2: Models; as gold standard for the speed-up due to occlusion culling, we show the frame rate for the datasets without any culling mechanism (FR).

a MRI scan, see Figure 3(a,b) and Table 2. We explore the dataset by moving through the lower part (Cisterna Magna) of the polygonal model. Most of the model structures through-out the walk-through are located within the view-frustum, while the structures with the largest number of polygons (located in the upper part or lateral ventricles) were not visible due to occlusion. All polygons of this model are aligned on the uniform cell grid and are approximately of the same size. All three adapted algorithms were able to detect this “natural” decomposition boundaries; only SGI generated approximately 15% additional polygons due to splitting operation between the grid points.

Figure 2 show frame rate (a) and render rate (b) of the four evaluated algorithms and Table 1 shows the averaged time measurements. The most inter-

esting detail is the low amount of time consumed by view-frustum and occlusion culling by ORSD, due to its coarse decomposition. The render rate of p-HBVO and D-BVD was approximately 25% better than the render rate of the ORSD approach. However, the finer decomposition introduced additional culling costs twice as much as for ORSD, resulting in a lower frame rate.

Cathedral Dataset

This dataset represents the interior of a gothic cathedral, designed with a CAD system (see Fig. 3(c,d) and Table 2). Occlusion is limited to small parts of the model, because a large share of the polygons are visible from most view points within the model. Figure 1 shows three fine decompositions of the cathedral model. Especially the p-HBVO approach (a) adapts very nicely to the structures of the model, such as pillars and arcs. In contrast, the decomposition generated by SGI (c) introduces very large bounding boxes, which do not adapt properly to the actual geometry.

The p-HBVO approach performed best on this dataset (see Figure 2(c,d)) which is due to the low culling costs, compared to SGI and D-BVD (see Table 1). The bounding boxes of D-BVD and SGI are not really suited for occlusion culling; especially their occlusion culling time is significantly higher compared to p-HBVO, thus severely reducing the frame rate. Consequently, view-frustum culling only is faster than occlusion culling for those ap-

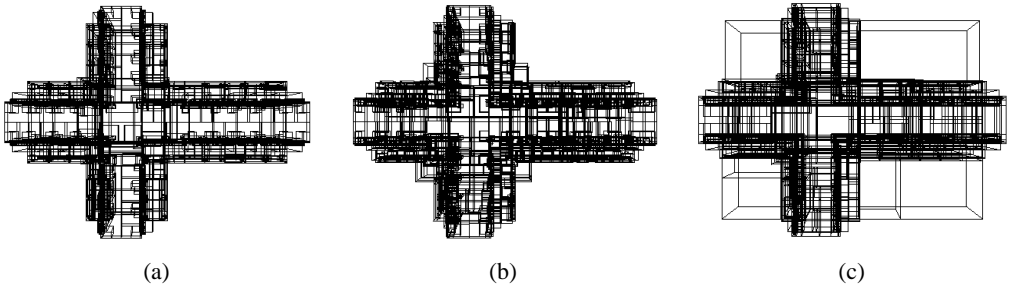


Figure 1: Cathedral dataset – decomposed by (a) p-HBVO, (b) D-BVD, and (c) SGI; the arts and pillars of the cathedral are well detected by p-HBVO and D-BVD. SGI only used a regular spatial decomposition.

proaches, while it is basically as fast as occlusion culling based on the p-HBVO hierarchy (see Table 1).

City Dataset

The city dataset is an artificial model of 400 basic building models with some interior, which contains most of the polygonal complexity, see Fig. 3(e,f) and Table 2. Consequently, most of the polygons of this model are occluded. However, only the p-HBVO approach was able to decompose all the interior into individual scene entities, hence resulting in a large number of nodes, a very low render rates, and high occlusion culling costs which were more than compensated by the small amount of potentially visible geometry (see Table 1 and Figure 2(e,f)). In contrast, the D-BVD approach did not detect the interior objects. The optimal computed decomposition was of coarser granularity (10% of the scene entities of p-HBVO), resulting in less time spent for culling (vfc and occ), but a higher render rate of 41 times larger than the render rate of p-HBVO (see Table 1). Similar, the SGI approach did also not detect the interior objects. It used a coarse granular decomposition, which consequently increased the rendering time in comparison to p-HBVO.

Summary

Overall, two of our three adapted model organization approaches were able to generate decompositions with faster rendering due to higher cull performance. This was achieved by reducing culling

costs or by reducing the render rate of the dataset. On uniform grid datasets, the basic ORSD approach produced a model decomposition which performed best, mostly due to the low time spent to establish occlusion or non-occlusion.

Generally, we observed that models with high occlusion do not require very fine decomposition (ventricle dataset, D-BVD vs. ORSD). On the other hand, a fine decomposition pays off if interior (thus completely occluded) objects are clustered in a scene entity (city dataset, p-HBVO vs. SGI). In contrast, models with low occlusion (cathedral dataset) can benefit from finer decompositions if the culling costs loss is only a fraction of the rendering costs gain (see city dataset, p-HBVO). However, this was not true of the cathedral dataset (D-BVD).

Note that the p-HBVO approach builds a binary scene tree. This usually results in deeper trees, hence more intermediate scene entities. This increases the time spent for occlusion culling significantly. Once this binary tree was re-build into a quad tree representation, we accomplished a frame rate increase by approximately 20%.

To summarize, we always achieved a speed-up due to occlusion culling-based rendering. Especially with the city dataset, we accomplished a speed-up of 15.6 after culling of 99.9% of the model geometry with the p-HBVO algorithm. The other approaches obtained a similar speed-up, while culling less polygons.

On the ventricle dataset, the ORSD approach accomplished the best results; 77.6% of the geometry were culled, due to view-frustum and occlusion culling. This culling performance resulted in a

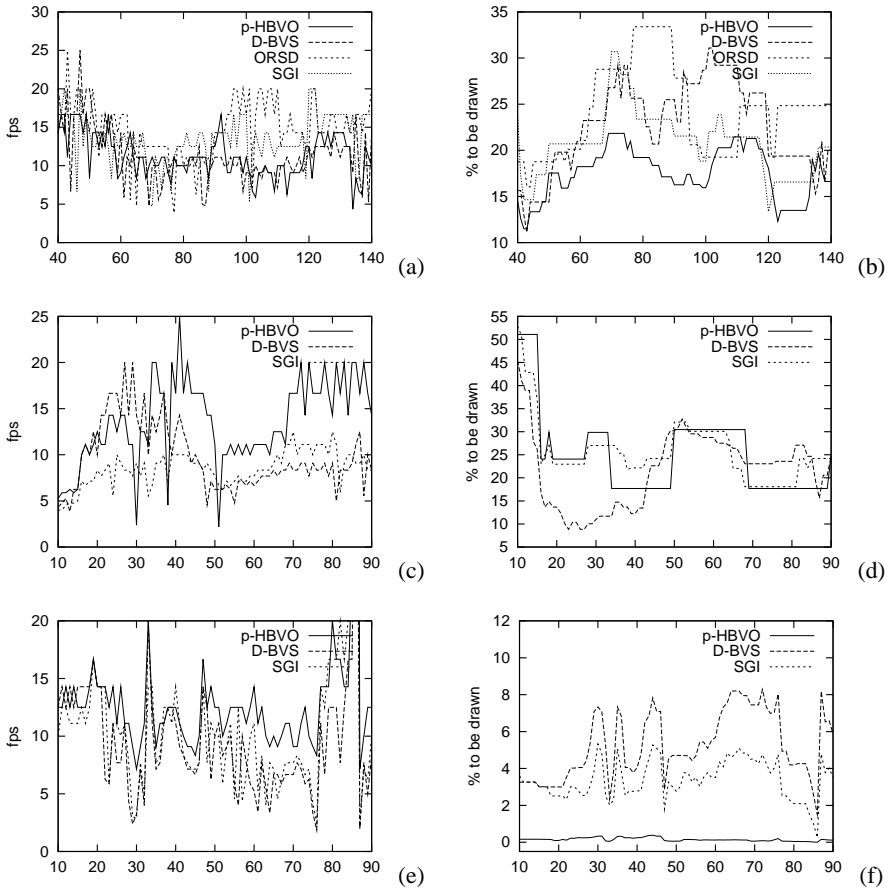


Figure 2: Decomposition results: Left column shows frame-rate and right column shows final geometry drawn. (a,b) Ventricle dataset; window of frames 40 to 140. (c,d) Cathedral dataset; all frames of the path are shown. (e,f) City dataset; window of frames 10 to 90 (out of 200).

frame rate speed-up of 3.3.

5 Conclusion and Future Work

Four different approaches for hierarchical decompositions of large polygonal models were presented and discussed. One of them is a commercially available product (SGI’s OpenGL Optimizer), while the other approaches are still research projects. Overall, the former approaches achieved similar (D-BVD) or better performance — evaluated with an occlusion culling application — than the tool of SGI’s OpenGL Optimizer. It turned out that a simple

octree-based scheme (ORSD) suits very well to uniform grid datasets; bounding box size as much as polygonal load balance were handled well. However, this scheme does not work on unstructured grid datasets because of the missing “natural” decomposition on cell-base. In particular the p-HBVO approach performed well on all datasets. Whereas the D-BVD approach optimized the size and shape of the bounding boxes, the p-HBVO approach was able to generate good bounding boxes, while also balancing the polygon load.

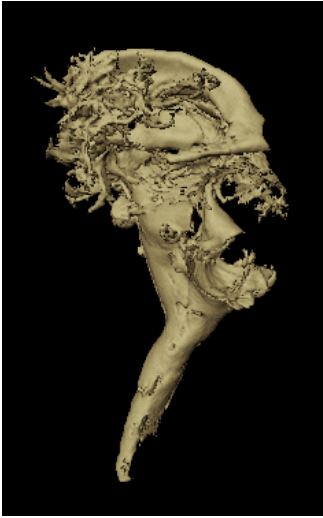
So far, only spatial coherence information is exploited in order to combine raw triangles into scene

entities. If neighborhood connectivity information –, i.e., semantic information which describes what kind of object should be combined (like the roof of a house in the city dataset) – is used, we expect decompositions which perform better than the current ones. This will be a major future research focus.

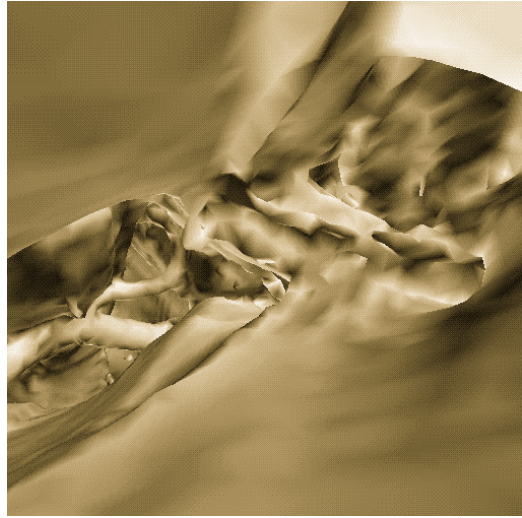
References

- [1] J. Airey, J. Rohlf, and F. Brooks. Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments. In *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 41–5, 1990.
- [2] J.M. Airey. *Increasing Update Rates in the Building Walkthrough System with Automatic Model-Space Subdivision and Potentially Visible Set Calculations* PhD thesis, Department of Computer Science, University of North Carolina, Chapel-Hill, 1990.
- [3] G. Barequet, B. Chazelle, L. Guibas, J. Mitchell, and A. Tal. BOXTREE: A Hierarchical Representation for Surfaces in 3D. In *Proc. of Eurographics*, pages 387–396, 1996.
- [4] D. Bartz, M. Meißner, and T. Hüttner. OpenGL-assisted Occlusion Culling of Large Polygonal Models. *Computers & Graphics*, 23(5):667–679, 1999.
- [5] D. Bartz and M. Skalej. VIVENDI - A Virtual Ventricle Endoscopy System for Virtual Medicine. In *Proc. of Symposium on Visualization*, pages 155–166, 1999.
- [6] S. Coorg and S. Teller. Temporally Coherent Conservative Visibility. In *Proc. of ACM Symposium on Computational Geometry*, pages 78–87, 1996.
- [7] S. Coorg and S. Teller. Real-Time Occlusion Culling for Models with Large Occluders. In *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 83–90, 1997.
- [8] H. Fuchs, Z. Kedem, and B. Naylor. On Visible Surface Generation by a Priori Tree Structures. In *Proc. of ACM SIGGRAPH*, pages 124–133, 1980.
- [9] S. Gottschalk, M. Lin, and D. Manocha. OBBTree: A Hierarchical Structure for Rapid Interference Detection. In *Proc. of ACM SIGGRAPH*, pages 171–180, 1996.
- [10] N. Greene. *Hierarchical Rendering of Complex Environments*. PhD thesis, Computer and Information Science, University of California, Santa Cruz, 1995.
- [11] N. Greene. Hierarchical Polygon Tiling with Coverage Masks. In *Proc. of ACM SIGGRAPH*, pages 65–74, 1996.
- [12] N. Greene, M. Kass, and G. Miller. Hierarchical Z-Buffer Visibility. In *Proc. of ACM SIGGRAPH*, pages 231–238, 1993.
- [13] L. Hong, S. Muraki, A. Kaufman, D. Bartz, and T. He. Virtual Voyage: Interactive Navigation in the Human Colon. In *Proc. of ACM SIGGRAPH*, pages 27–34, 1997.
- [14] H. Hoppe. Progressive Meshes. In *Proc. of ACM SIGGRAPH*, pages 99–108, 1996.
- [15] T. Hudson, D. Manocha, J. Cohen, M. Lin, Kenneth E. Hoff, and H. Zhang. Accelerated Occlusion Culling Using Shadow Frusta. In *Proc. of ACM Symposium on Computational Geometry*, pages 2–10, 1997.
- [16] T. L. Kay and J. T. Kajiya. Ray Tracing Complex Scenes. In *Proc. of ACM SIGGRAPH*, pages 269–278, 1986.
- [17] J. Klosowski, M.Held, J. Mitchell, H. Sowizral, and K. Zikan. Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics*, 4(1):21–36, 1998.
- [18] D. Luebke and C. Georges. Portals and Mirrors: Simple, Fast Evaluation of Potentially Visible Sets. In *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 105–106, 1995.
- [19] G. Müller and D. Fellner. Hybrid Scene Structuring with Application to Ray Tracing. In *Proc. of ICVC*, pages 19–26, 1999.
- [20] B. Naylor. Partitioning Tree Image Representation and Generation From 3D Geometric Models. In *Proc. of Graphics Interface*, pages 201–212, 1992.
- [21] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, 1994.
- [22] N. Scott, D. Olsen, and E. Gannett. An Overview of the VISUALIZE fx Graphics Accelerator Hardware. *The Hewlett-Packard Journal*, (May):28–34, 1998.
- [23] SGI. *OpenGL Optimizer Manual*. Silicon Graphics Inc., Mountain View, 1997.
- [24] J. Snyder and J. Lengyel. Visibility Sorting and Compositing without Splitting for Image Layer Decompositions. In *Proc. of ACM SIGGRAPH*, pages 219–231, 1998.
- [25] O. Sudarsky and C. Gotsman. Output-Sensitive Visibility Algorithms for Dynamic Scenes with Applications to Virtual Reality. In *Proc. of Eurographics*, pages 249–258, 1996.
- [26] S. Teller and C.H. Sequin. Visibility Pre-processing for Interactive Walkthroughs. In *Proc. of ACM SIGGRAPH*, pages 61–69, 1991.
- [27] H. Zhang, D. Manocha, T. Hudson, and Kenneth E. Hoff. Visibility Culling Using Hierarchical Occlusion Maps. In *Proc. of ACM SIGGRAPH*, pages 77–88, 1997.
- [28] D. Zorin, P. Schröder, T. DeRose, L. Kobbelt, A. Levin, and W. Sweldens. Subdivision for Modeling and Animation. In *ACM SIGGRAPH Course 23*, 2000.

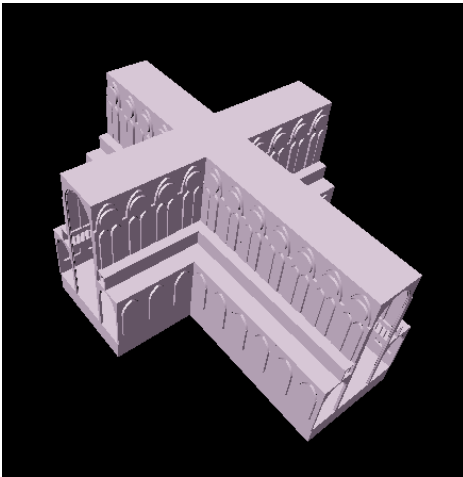




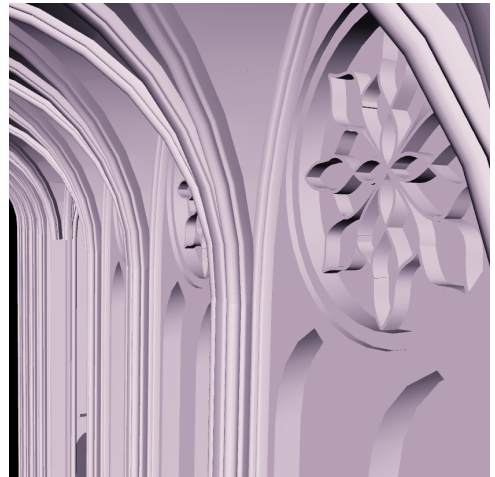
(a)



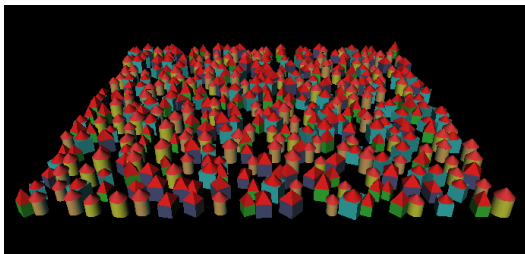
(b)



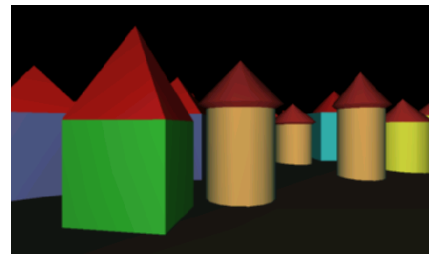
(c)



(d)



(e)



(f)

Figure 3: Ventricle dataset: (a) Overview, (b) Inside view. Cathedral dataset: (c) Overview, (d) Inside view. City dataset: (e) Overview, (f) Inside view.