PROGRESSIVE COMPRESSION OF VISIBILITY DATA FOR VIEW–DEPENDENT MULTIRESOLUTION MESHES

Christopher Zach Konrad Karner

VRVis Research Center
Inffeldgasse 16
A–8010 Graz, Austria

{zach}@vrvis.at

ABSTRACT

In this paper we present a lossless and optionally lossy compression method for precomputed visibility data for view-dependent multiresolution meshes, which supports out-of-core rendering and progressive transmission through slow connections. Our approach has the feature, that visibility information can be stored directly in the nodes of the multiresolution structure and only necessary parts of visibility data need to be transmitted. Nevertheless our lossless compression method performs better than well-known general purpose solid compressors which inhibit selective access of required data. Even higher compression can be achieved when lossy preprocessing is performed. We evaluate our method on several artificial and real datasets.

Keywords
Compression, visibility preprocessing

1 INTRODUCTION

In this work we address the efficient encoding of precomputed visibility data for hierarchical representations of geometry, especially for view-dependent multiresolution meshes. Nevertheless our encoding strategy can be applied to any other hierarchy based on nested bounding volumes. Our encoding allows partial transmission of visibility information as required by the rendering application.

Determining visibility of objects can be done either at runtime using recent hardware extensions to test rapidly, whether some part of the scene is occluded, or in a preprocessing stage. In the later case the set of (potentially) visible objects for a region is calculated and stored for use at runtime. These information is often represented as binary matrix with a one in row i and column j if the object with ID j is potentially visible in view cell i. Our method transforms this visiblity matrix into a more compact representation.

We aim on visualisation of very large 3D models, which only fit partially into the main memory of the rendering computer. Therefore we incorporate visibility preprocessing into our rendering framework to enable selective retrieval of appropriate parts of the scene geometry from external memory [Zach02]. Additionally the visibility information can be used for run-time occlusion culling on hardware without accelerated visibility checking capabilities.

2 RELATED WORK

2.1 View–dependent Multiresolution Mesh Generation and Runtime Management

The seminal papers by Hoppe [Hoppe93], [Hoppe96] opened the field of continuous level of detail generation by successive local mesh modifications (for a survey see e.g [Puppo97]). Various authors proposed a hierarchical intermediate data structure based on these local mesh updates to obtain a suitable level of detail for runtime visualization [Xia96], [Hoppe97], [Flori97], [Luebk97]. The extraction of the displayed mesh is guided by a refinement criterion, which depends at least on the current viewing position and direction, but may utilize e.g. silhouette information as well [Luebk97].

A bounding volume is associated with every node in the hierarchical multiresolution structure to test whether the appropriate part of the mesh is currently within the viewing frustum. We use these bounding volumes (bounding boxes in particular) as occludees in the visibility preprocessing phase.

Currently we employ mesh simplification using quadric error metrics [Garla97] to obtain continuous levels of detail, and our runtime visualization is similar to the approach proposed in [Luebk97].

El-Sana et al. [El-Sa01] combine viewing parameters with approximate visibility information for selective mesh refinement. Visibility information is estimated at runtime using a solidity of regions approach [Kloso99], [Kloso00].

2.2 Visibility Preprocessing

Precalculating visibility requires determination of visible objects from predefined regions (view cells). The result of the calculation is a mapping from view cells to the set of potentially visible objects (PVS). The essential (and challenging) task is solving the *visibility from region* problem. A survey of methods for visibility computations is given in [Duran00] and [Cohen00]. We utilized software by P. Wonka [Wonka99], [Wonka00] to calculate the PVS for each view cell.

2.3 Compression of Visibility Data

The size of the precomputed visibility data can be in the order of the geometric model size, nevertheless there exists rather little literature about efficient representation of visibility data.

Panne and Stewart [Panne99] proposed a lossless and optionally lossy encoding of the visibility matrix based on clustering equal (or similar) regions of the visibility matrix. During the process of clustering the true entries in the matrix become sparser. Finally the matrix is represented by the lengths of the false runs. In their article they reported striking compression ratios: up to 7:1 for lossless compression and 150:1 for lossy compression. Their encoding is not suitable for progressive transmission of visibility information. Additionally their method is very time consuming, since the clustering step explores a very large state space. For these reasons direct application of this approach to our data sets is not feasible, but we apply the idea of lossy view cell clustering to enhance the compression ratio.

Gotsman et al. [Gotsm99] organize the viewing space into a BSP like hierarchy and propose a suitable method of encoding visibility information. We employ a similar approach (but with a hierarchy on the 3D model) as the basis for our progressive coding.

3 OVERVIEW

In this section we give a brief sketch of the ideas related to our compression scheme. We assume, that the structure of the hierarchy is known and needs not to be encoded further. Additionally we assume, that the geometry of the view cells is already known to the rendering client. Since the view cells comprise a triangulated 2.5D heightfield, the corresponding mesh can be efficiently compressed.

We exploit the following simple observations and ideas:

- *Optional lossy preprocessing.* View cells with similar sets of (potentially) visible objects can be merged into one new logical view cell. The idea was already used by Panne and Stewart [Panne99] and it requires encoding metainformation describing the clustered view cells.
- *Transposed visibility matrix.* Usually the visibility matrix is read in row order: for a given view cell the visible objects are enumerated. We encode the set of view cells, in which the considered object is visible. This set is represented as a bit vector, the *visibility bitsets.*

- *Visibility data coupled with nodes.* For every node in the multiresolution hierarchy we store the visibility information of its children (if any). Whenever a node is loaded from external memory, only a small, but relevant fraction of the visibility matrix is retrieved as well. We remark that this feature is very different to most approaches to data compression, which compress large fractions of the source data at once. Our approach has the side-effect, that preloading strategies [Teler01], [Zach02] can utilize visibility information to guide retrieval of subsequent nodes from external memory.

- *Monotony of visible bitsets.* If a node (resp. its bounding volume) is not visible from a certain view cell, this is true for all children. Therefore it is sufficient to encode shortened visibility bitsets for child nodes [Gotsm99].

- *Run length coding.* The visibility bitsets of child nodes comprise a binary matrix, which is run length coded. Slightly better compression ratios because of longer runs can be achieved, if the view cells are suitably reordered.

The next section describes the compression of the visibility matrix in more detail.

4 COMPRESSED VISIBILITY CODING

4.1 Lossy Preprocessing

The size of the visibility information can be substantially reduced, if view cells with similar sets of visible objects are merged and one conservative set of visible objects is generated for these cells. This set is just the union of the PVS of the individual cells.

We choose randomly a set of seed view cells and select the most similar cell for each of the seed cells. More formally, if $pvs(c_i)$ and $pvs(c_j)$ are the visible sets for view cell c_i resp. c_j, a score function is defined as

$$w_i\ |pvs(c_j) \setminus pvs(c_i)| + w_j\ |pvs(c_i) \setminus pvs(c_j)|\,.$$

The weight w_i for view cell c_i is the number of cells already accumulated in c_i. For instance, if c_i represents 5 original view cells, that were merged lossy, losing one occlusion in c_i means 5 lost occlusion in the original dataset. The initial weights for original view cells is set to 1.

The pair of view cells, which has the lowest score, is replaced by one new logical view cell and the procedure is repeated. Finding the best view cell for a given seed cell requires a linear search through the set of cells. We accelerate this search using an additional index, which is a sorted data structure mapping the size of the PVS for each view cell to the corresponding view cell. If the current seed is c and the best pair at present has a score equal to s, the candidate cells with a potential lower score have a PVS size in the range $[|pvs(c)| - s, |pvs(c)| + s]$. The secondary index can be used to limit these candidate cells efficiently.

This loop is continued, until the number of lost occlusions is larger than a given threshold. We allowed 5% of lost occlusions in our experiments, which means, if e.g. 80% of objects are hidden on average in the original visibility dataset, after lossy preprocessing at least 76% of objects are occluded. Performing this procedure is optional.

4.2 Reordering of View Cells

For the purpose of determining visibility of objects from a given view cell the actual numbering of view cells is of no importance. Since run length coding in general is sensitive to the order items, it is rational to permute the columns of the visibility matrix, such that longer runs are more likely.

We use a greedy approach to perform the reordering: the view cell, which is most similar (maximizing the number of objects with the same visibility status) to the currently last cell is appended next. This process requires a quadratic number of operations, but we use a secondary index (similar to the speed-up technique used for lossy preprocessing) to accelerate this procedure.

Since in many cases the view cell most similar to a given view cell is some adjacent cell, it can be sufficient to search only the neighbouring cells for the best match. Thus, the reordering can be achieved in nearly linear time.

This reordering procedure is again optional.

4.3 Visibility Bitsets

Essentially we encode the sets $v(A)$ of view cells, from which the given object A (a node in the hierarchy) is visible. The set $v(A)$ is represented as a bit vector. If node A has 2 children, B and C, obviously $v(B) \subseteq v(A)$ and $v(C) \subseteq v(A)$ (since the bounding volume of A encloses the bounding volumes of B and C). Therefore it is sufficient to represent $v(B)$ and $v(C)$ as bit vectors with length $|v(A)|$ [Gotsm99]. Figure 1 illustrates this observation.

4.4 Run-length Coding

The sets $v(B_i)$ (resp. the corresponding bit vectors) of children B_i of the parent node A are stored in conjunction with the multiresolution data of A. The concatenated bit vectors are run-length coded and a histogram with the lengths is updated. Entropy coding of the lengths completes the compression pipeline.

A schematic layout of the node structure is shown in Figure 2. The node data structure contains geometric information related to the multiresolution mesh structure and its layout is similar to the one proposed in [Luebk97].

Since the bounding volume of the root node contains the complete scene, it is visible from every view cell and encoding this visibility bitset is not required.

5 TESTED COMPRESSION METHODS

We have implemented some variations of the method described in the previous section. Method 0 omits the run-length coding step and stores the visibility bit vectors directly. Method 1 initially stores the intersection of the bitsets $\cap_i B_i$ and encodes $v(B_i)$ using $|v(A)| - |\cap_i B_i|$ bits. This method assumes, that successor nodes share many view cells which have the same visibility information. These two methods were proposed in [Gotsm99] to encode visibility information scenes with a hierarchical view cell structure. Method 2 is similar to method 1, but the intersection is now restricted to a suitable subset of child nodes. The subset S is chosen such that the total number of encoded bits

$$|\cap_{i \in S} B_i| + |S| \, (|v(A)| - |\cap_{i\in S} B_i|) + |S^C| \, |v(A)|$$

is minimized. Subnodes that belong to S are encoded similar to method 1, whereas method 0 is used for nodes not belonging to S. Additional bits are required to decide, whether a node belongs to S. All these methods do not depend on the numbering of view cells.

Method 3 is the approach described in Section 4 without lossy preprocessing, but with the initial reordering of view cells. Method 4 replaces run-length coding in method 3 by a sparse matrix coding technique similar to the encoding described in [Panne99]. Obviously the ones are more likely in the visibility matrix, therefore we eliminate zeros and store the run-lengths of ones (instead of the other way round as proposed by Panne and Stewart). Additionally the lengths are entropy coded.

We use `gzip` and `bzip2` as general purpose compressors for comparison. These compressors are applied to the binary visibility matrix in row order. Since these coders work on a byte level, every new row of the matrix is aligned on a byte boundary, resulting in significantly improved compression ratios compared to tightly packed matrices.

6 TEST DATASETS

We run the compression methods outlined in Section 5 on two artificial and two real datasets. The important properties of these datasets are summarized in Table 1. The two artifical models were generated by simulating a dense environment by placing boxes on a ground plane. The boxes are located on a jittered regular grid. The bounding volume hierarchies for the artificial datasets were obtained using quadtrees to cluster individual objects and subsequent calculation of tight bounding boxes for each node. The artificial datasets differ only in their scene complexity.

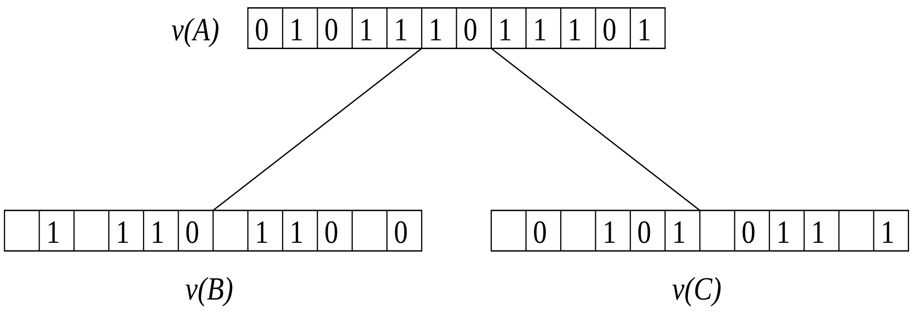

Figure 1: Exploiting the monotony of visibility in the hierarchy. The sets $v(B)$ and $v(C)$ are encoded using vectors with 8 bits, which is the number of ones in $v(A)$. The empty entries in $v(B)$ and $v(C)$ are implicitly zero and need not to be stored.

The real models comprise parts of an urban dataset covering a large fraction of the town of Graz. The *inner city* dataset (shown in Figure 3) covers the historic center of Graz, whereas the *complete* dataset consists of the full urban city model available to us. Nevertheless the complete dataset has some parts missing, which (in combination with the smaller number of view cells) results in higher average visibility of objects.

These real-world datasets are preprocessed using quadric error metric mesh simplification [Garla97] to generate the multiresolution hierarchy. To trade off granularity and tree complexity, the resulting binary vertex trees are transformed into trees, in which every node has 4 children, by collapsing two consecutive levels of the original vertex trees.

The view cells were obtained by cutting out the occluders (buildings) from the ground plane (for the artificial dataset) resp. the digital terrain model (for the real datasets) and subsequent triangulation.

7 RESULTS

We applied several methods to compress the visibility information for our test datasets. The widely used `gzip` and `bzip2` compression software was used to evaluate our lossless method. We remark that these encoders perform compression of the entire data and therefore cannot provide random access capabilities of the other tested methods.

Table 2 shows the results of lossless compression for our datasets. The figures represent only the size of the information stored in the visibility matrix itself and do not include view cell geometry and other possibly required metainformation. Our proposed technique (method 3) is uniformly better than the other evaluated methods. In our opinion the inner city dataset (3rd column) is the most representative one, since it resembles our final 3D model of the town of Graz at best. Further it can be observed that the sparse matrix coding proposed in [Panne99] (method 4) is inferior to simple run-length coding used in method 3.

In general datasets with smaller average visibility yield to better compression, which is clearly observable from the numbers for methods 0 to 2. If we compare the figures for method 0 with those of method 3, run-length coding combined with entropy coding increases the compression ratio by a factor of 2 to 4. From the length histogramm we calculated the entropy and noticed, that arithmetic coding would reduce the numbers for method 3 by 13–15%.

The results for lossy compression are presented in Table 3. Interestingly, the achieved compression ratio (wrt. to the original table size) is approximately the lossless compression ratio times half the reduction in the number of view cells. This implies that during lossy preprocessing some redundancy is lost and the two compression stages are not completely independent.

We finish this section by noting that about 20

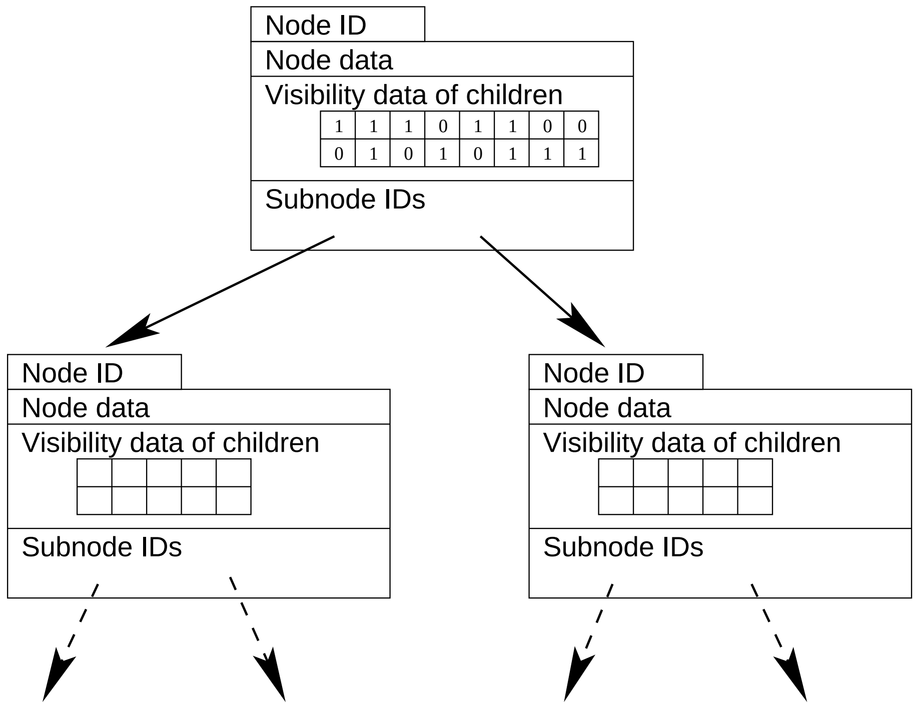

Figure 2: The logical layout of the external node structure. The physical layout may be different due to quantization and compression of several fields.

Scene properties	artificial 1	artificial 2	inner city	complete
# view cells	5766	23814	7259	2890
# occludees	1365	5461	36604	72159
average visibility	18.4%	8%	16.6%	35.1%

Table 1: Scene characteristics for the datasets.

bytes per nodes are required for the real datasets to store (lossy) visibility information, which is only a fraction of the size of geometric data for each node, which is about 120 bytes uncompressed.

8 CONCLUSION AND FUTURE WORK

We presented a novel lossless and optionally lossy compression method to encode visibility data for hierarchically organized scenes. Our method is suitable especially in combination with out-of-core rendering of view-dependent multiresolution meshes, since the compressed visibility information can be stored within the multiresolution data structure and only required parts of the visibility table need to be loaded at runtime.

Future version of our compression software will incorporate the view cell geometry to enable faster lossy preprocessing and reordering as outlined in Section 4.2.

9 ACKNOWLEDGEMENTS

We would like to thank Peter Wonka and Michael Wimmer for providing their YARE software to compute visibility information. Additionally we are grateful to Gerald Schrcker for implementing the necessary software to employ YARE.

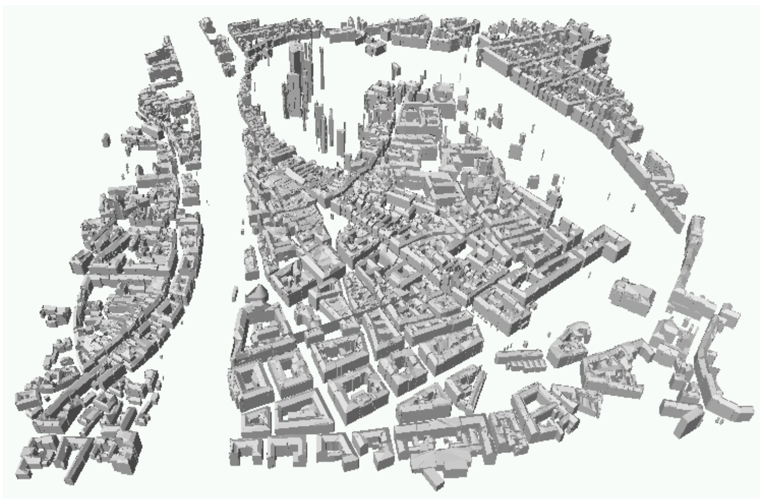
Figure 3: The inner city dataset.

	Artificial 1	Artificial 2	Inner city	Complete
Table size	983824	16264962	33217184	26067800
Gzipped table size	188018 (5.2x)	1545482 (10.5x)	4186757 (7.9x)	4834404 (5.4x)
Bzipped table size	176768 (5.6x)	1127423 (14.4x)	2825137 (11.8x)	3550038 (7.3x)
Method 0	300845	2314438	6650227	10387055
Method 1	305901	2497702	4587842	6126083
Method 2	301083	2409445	4476382	5813139
Method 3	140484	1061190	2111627	2624425
Compression ratio	7.0x	15.3x	15.7x	9.9x
Avg. run length	9.57	9.86	12.7	12.51
Method 4	226009	1771864	2729580	3181445
Compression ratio	4.4x	9.2x	12.2x	8.2x
Avg. run length	7.7	7.18	9.07	11.46

Table 2: Compression results for lossless compression.

This work has been done in the VRVis research center, Graz and Vienna/Austria (http://www.vrvis.at), which is partly funded by the Austrian government research program Kplus.

References

[Cohen00] D. Cohen-Or, Y. Chrysanthou, and C. Silva. A survey of visibility for walkthrough applications. In *Proc. of EUROGRAPHICS'00, course notes*, 2000.

[Duran00] F. Durand. A multidisciplinary survey of visibility. ACM SIGGRAPH Course Notes, 2000.

[El-Sa01] J. El-Sana, N. Sokolovsky, and C. T. Silva. Integrating occlusion culling with view-dependent rendeing. In *IEEE Visualization 2001*, pages 371–378, 2001.

[Flori97] L. De Floriani, P. Magillo, and E. Puppo. Building and traversing a surface at variable resolution. In *IEEE Visualization'97*, pages 103–110, 1997.

[Garla97] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH '97*, pages 209–216, 1997.

[Gotsm99] C. Gotsman, O. Sudarsky, and J. Fayman. Optimized occlusion culling using five-

	Artificial 1	Artificial 2	Inner city	Complete
# remaining view cells	1329	1921	1330	934
Lossy table size	227267	1312051	6085415	8424564
Method 3	53819	188916	715207	1543002
Compression ratio	18.3x	86.1x	46.4x	16.9x
Avg. run length	5.22	4.35	7.46	8.95
Method 4	72137	243086	823346	1698149
Compression ratio	13.6x	66.9x	40.3x	15.4x
Avg. run length	8.25	8.18	8.19	10.93

Table 3: Results of lossy compression. Lossy preprocessing was performed such that at most 5% of original occlusions were lost.

dimensional subdivision. *Computer & Graphics*, 23(5):645–654, 1999.

[Hoppe93] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. *Computer Graphics*, 27(Annual Conference Series):19–26, 1993.

[Hoppe96] H. Hoppe. Progressive meshes. In *Proceedings of SIGGRAPH '96*, pages 99–108, 1996.

[Hoppe97] H. Hoppe. View-dependent refinement of progressive meshes. In *Proceedings of SIGGRAPH '97*, pages 189–198, 1997.

[Kloso99] J. T. Klosowski and C. T. Silva. Rendering on a budget: A framework for time-critical rendering. In *IEEE Visualization '99*, pages 115–122, 1999.

[Kloso00] J. T. Klosowski and C. T. Silva. The prioritized-layered projection algorithm for visible set estimation. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):108–123, 2000.

[Luebk97] D. Luebke and C. Erikson. View–dependent simplification of arbitrary polygonal environments. In *Proceedings of SIGGRAPH '97*, pages 199–208, 1997.

[Panne99] M. van de Panne and A. J. Stewart. Effective compression techniques for precomputed visibility. In *Rendering Techniques 1999 (Proceedings of the Eurographics Workshop 1999)*, pages 305–316, 1999.

[Puppo97] E. Puppo and R. Scopigno. Simplification, LOD and multiresolution - principles and applications. Eurographics '97 Tutorial Notes, 1997.

[Teler01] E. Teler and D. Lischinski. Streaming of complex 3D scenes for remote walkthroughs. In *Proceedings of EUROGRAPHICS 2001*, pages 17–25, 2001.

[Wonka99] P. Wonka and D. Schmalstieg. Occluder shadows for fast walkthroughs of urban environments. In *Proceedings of EUROGRAPHICS 99*, pages 51–60, 1999.

[Wonka00] P. Wonka, M. Wimmer, and D. Schmalstieg. Visibility preprocessing with occluder fusion for urban walkthroughs. In *Rendering Techniques 2000 (Proceedings of the Eurographics Workshop 2000)*, pages 71–82, 2000.

[Xia96] J. C. Xia and A. Varshney. Dynamic view-dependent simplification for polygonal models. In *IEEE Visualization '96*, pages 335–344, 1996.

[Zach02] C. Zach and K. Karner. Prefetching policies for remote walkthroughs. In *Proceedings of the WSCG*, pages S153–S159, 2002.