Imaging all Visible Surfaces

Wolfgang Stuerzlinger

Department of Computer Science
York University, Toronto, Canada

Abstract

Today many systems exist to generate geometric models of existing scenes and objects. However, very few systems store accurate data about surface appearance such as colors and textures. One way to capture surface texture data is to record a series of images that, collectively, captures all visible surfaces of the object. Finding good viewpoints for this task is not easy.

This paper presents a new heuristic method to find a good set of viewpoints for a given geometric model. Taking images from the computed viewpoints will show every visible part of every surface at least once. The approach uses a hierarchical visibility algorithm to preprocess the scene. A good set of viewing regions is identified with simulated annealing and then good viewpoints are derived. Results and visualizations of the computed solutions are presented.

Key words: Texture Acquisition, Image-Based Rendering and Modeling, Visibility.

1 Introduction

For many man-made objects some form of CAD (computer aided design) data is available today. Prominent examples are buildings, mechanical structures like power plants, and interior rooms like kitchens and bathrooms. Typically the data set describes the geometry of the object exactly, but does not contain accurate or content specific data about surface appearance (such as color or texture). Therefore, a visualization of the object will only show the geometric structure but will not match the visual appearance. For such objects (known geometry but unknown surface appearance) many methods to derive textures from pre-recorded images have been introduced in computer vision. Some approaches allow the user to specify approximate geometry for the scene and map the images onto this approximation (see e.g. [6]). The main application for the presented method is the automated acquisition of textures for environments with known geometry. Subsequent rendering of the textured scene model will provide a more realistic visualization. This can be used in virtual training systems, in the visualization of modifications to real scenes, and all augmented reality applications.

Common to all mentioned approaches is that in general there are some parts of the scene that are not visible in the pre-recorded images, therefore heuristic hole-filling algorithms are used to generate a complete surface description.

It is hard to judge for a human if a set of pre-recorded images is complete in the sense that every part of every surface and object is visible in at least one of the images. Furthermore, it is time-consuming to set up camera and lighting so that high quality images (e.g. without highlights and/or reflections) can be recorded.

This work addresses the visibility portion of the problem with a method to compute a good set of viewpoints from which to capture images. The method assumes that the geometry of the scene is known. Approximate geometry can be used as well, but the quality of the results depends on the quality of the approximation. Taking photographs from the computed viewpoints will ensure that every part of every surface is visible in at least one image. This can also be done with an automated imaging device (e.g. a robot). The method currently assumes that *potentially* a spherical image can be obtained from every viewpoint. Compositing multiple images to a spherical image [17] is one way to generate such images.

The problem of determining an optimal set of viewpoints is NP hard and is related to the art gallery problem known from computational geometry [16] and to the aspect graph [9]. This article presents a heuristic approach that tries to find a near-optimal solution in a reasonable time frame. Furthermore, this work addresses only the visibility part of the problem. Therefore, many other issues such as field of view, image resolution, surface sampling density, depth of field, and the avoidance of highlights or reflections are not addressed in this article as they are of a different nature.

2 Imaging all Visible Surfaces

The method operates on a scene with known geometry. The volume of all possible viewpoints for subsequent viewing needs to be defined as input. To make the problem tractable this volume is subdivided into a number of smaller regions. For each of these subregions, visibility of all surface parts of the scene is precomputed. Based on this discretization of viewing space a set of sub-regions that contain a set of good viewpoints is computed.

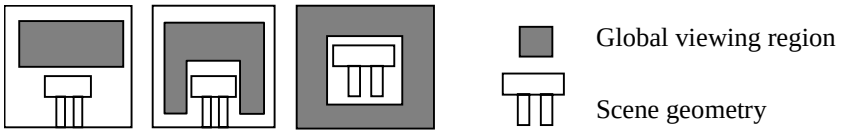

Figure 1: Examples for global viewing regions.

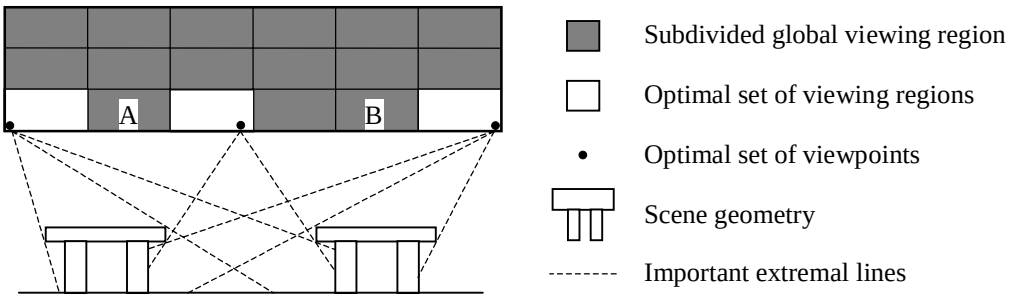

Figure 2: Scene with two tables, a global viewing region, and an optimal set of viewing regions and viewpoints (see text).

2.1 Viewing Regions

A viewing region is defined as a volume of empty space with the interpretation that it represents the union of all viewpoints inside this volume. The volume of all possible viewpoints is called the *global viewing region.* Figure 1 depicts possible viewing regions for some simple scenes.

Clearly, the camera cannot be placed inside objects. Therefore, the global viewing region cannot intersect geometry. In addition, for real (non-zero size) cameras a minimum distance from each object has to be observed. The global viewing region may be only a polygon in space. This appears when the viewpoint is constrained to remain at average eye height level, a common restriction in many walkthrough systems. Note that the global viewing region can also consist of several disjoint viewing regions.

Because the search space for optimal viewpoints in the global viewing region is very large, we subdivide it into a number of smaller viewing regions. A sufficient set of (smaller) viewing regions is then a set that 'sees' every part of every surface at least once. An optimal set avoids redundancies as far as possible and has a minimal number of viewing regions. The same definition applies to sets of viewpoints. Note that the optimal set of viewing regions and viewpoints is not necessarily unique.

Figure 2 shows a simple scene with two objects and a global viewing region. An optimal set of viewing regions and an optimal set of viewpoints is shown. Taking images from the three viewpoints shown will picture every part of the scene that is visible *from within the global viewing region.* To illustrate the optimality the most important extremal lines of sight are shown.

Note that the exact position of the middle viewpoint (and the middle viewing region) is not crucial. In fact, the viewpoint can be anywhere between points A and B in the diagram. This freedom to place a viewpoint is an important property, especially if other objects are added to the scene. Note also that there are scene parts that are invisible from all viewpoints inside the global viewing region. An example is the left leg of the left table.

Computing an optimal set of viewpoints or viewing regions is NP hard [16]. Therefore, we apply the following heuristic approach: First we compute a near-optimal set of viewing regions and then compute a near-optimal viewpoint in each of these regions. With decreasing size of viewing regions, this will converge to a correct solution.

In the approach presented here the global viewing region is subdivided hierarchically into a pre-defined number of viewing regions.

2.2 Hierarchical Visibility

The hierarchical visibility algorithm used in this work is based on the idea of linking all surfaces that are at least partially mutually visible (they can 'interact'). These interactions are subdivided based on the potential error in visibility. A similar idea was introduced previously in the context of hierarchical radiosit [13]. Most recently, Drettakis and Sillion [7] exploited a similar method (termed line-space hierarchy) to store visibilit information in a scene. In the hierarchical radiosity literature, the (smaller) parts of a subdivided surface are called elements, and this convention is used here, too.

Assume that the scene is a set of surface polygons organized in a hierarchy. The interior nodes of this hierarchy called clusters whereas the leaves correspond to surfaces. If needed, these surfaces will be subdivided

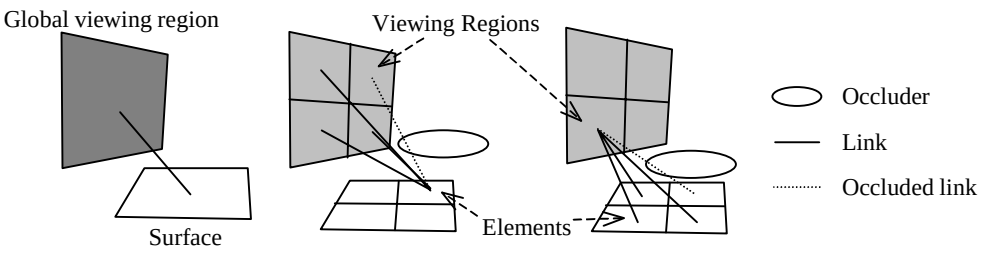

Figure 3: Initial link (top) and two examples for refined links.

further into elements. The hierarchy can be constructed during modeling or automatically by clustering nearby surfaces together (e.g. [2]). A hierarchical visibility method subdivides this scene hierarchy depending on the relative visibility of objects.

The hierarchical visibility algorithm used in this method starts with two nodes. One is the top level bounding box enclosing the whole scene. The other is the global viewing region. The visibility between these two objects is represented as a link. Each link and with it, the objects are subdivided recursively until the p otential error in visibility falls below a predefined threshold. For simplicity, a regular subdivision patter is used. Figure 3 shows a visualization of the initial situation and two examples for subdivided links.

If there is no occluder between the two nodes, a link is stored. Otherwise the larger node is subdivided and the process is applied recursively. Shaft culling [12] is used to speed the refinement process. Note that shaft culling can be optimized in the recursive refinement by temporarily storing the current list of potential occluders during recursive subdivision and using this list during further refinement of the link [1], [7]. For each link created the list of potential occluders is stored for later use. If there is one occluder that occludes the whole shaft, the nodes are mutually invisible and subdivision stops. This test is a simple addition to the shaft-culling test [1].

The result of the hierarchical visibility algorithm is a set of links that associates each viewing region to the el ements, which are at least partially visible from the r egion. Note that the visibility information stored in the links is in some sense conservative: as long as there is one point in the viewing region that can potentially see a part of a surface, a link is created. Unfortunately, it is conservative in the wrong way: links will be created for every element that is at least partially visible. However, there may be no a single viewpoint inside one viewing region that will see all the elements (partially) visible from it. Fortunately, elements are usually visible from more than one viewing region. This means that generally several alternative solutions exist and a solution method is free to choose among them!

2.3 Finding a Set of Good Viewing Regions

The result of the above hierarchical visibility algorith is a set of links that express the visibility relation between the set of viewing regions and the surfaces of the scene. This information is used to determine a good set of viewing regions. The number of combinations to consider increases exponentially with the size of the viewing region set. This prohibits an exhaustive search for non-trivial subdivisions of the global viewing region. This means also that only approximate solutions can be computed in practice. Furthermore, it is very important for any optimization method that the benefit of a proposed combination of viewing regions can be evaluated quickly. The link structure of the hierarchical visibility algorithm is not well suited for this purpose. Therefore, the information is transformed into a different data structure.

The method starts by enumerating all viewing regions and all elements by traversing the corresponding hierarchies. Then a two-dimensional visibility array is created and filled. It is indexed by viewing region and element number. To identify all elements that are visible from one particular viewing region the links of the viewing region and all its parents have to be considered. Each array entry is set if a visibility link exists between the viewing region and the element (or one of its parents).

The problem of finding the best combination of a set of objects is now a combinatorial optimization problem. One simple way to define the benefit or ‘goodness’ of a particular set of viewing regions is the total number of visible elements. As the hierarchical subdivision of the surfaces does not necessarily produce elements with equal areas, it is better to calculate the benefit by summing the surface areas of all visible elements and to weigh them by the element’s visibility. This puts more emphasis on finding large (fully) visible elements first during the global optimization.

To find the best possible solution the presented method uses simulated annealing. It works by changing the vector of solutions randomly. The magnitude of the potential changes is non-monotonically decreased over time. Simulated annealing has proven to be a very gen-

eral and efficient global search method in a variety of applications and for combinatorial optimization pro b-lems. While it cannot be guaranteed that simulated an-nealing will find the global optimum it consistentl finds at least values very close to the optimum in re a-sonable time (see e.g. [14]). The interested reader is referred to the literature for a more detailed description.
Because we cannot determine beforehand how many viewing regions are needed to view every surface part at least once, we decided to adopt the following incre-mental heuristic approach. The method calls the opti i-zation procedure repeatedly with increasing numbers of viewing regions. As soon as the maximum set of visible surface parts is reached, (i.e. at least a local optimum has been found) the loop terminates.

2.4 Computing a Good Set of Viewpoints

The set of viewing regions computed by the above method provides a basis for computing a good set of viewpoints. A good viewpoint inside a viewing region 'sees' (almost) everything that is visible from the whole viewing region. It may be necessary to compute multi-ple viewpoints to cover everything visible from a par-ticular viewing region. However, this does not happen often if the viewing regions are sufficiently small a other viewpoints in other viewing regions will also 'cover' other parts of the scene.
For each viewing region, the method searches for the best possible viewpoint position, i.e. the position that maximizes the total visible surface area. This search is again done with simulated annealing, but the search is now for the best position (instead of the best combina-tion). Only links with partial visibility need to be re-evaluated, and the stored occluder lists for each link can be used to speed processing.

3 Implementation

The hierarchical visibility pre-processing mentioned in section 2.2 was implemented based on a public domain radiosity package [1]. As explained above the visibilit refinement process uses shaft culling to speed the visi-bility tests. If there are polygons within a shaft and no single polygon occludes the shaft completely the cur-rent implementation uses ray casting with 16 random rays to resolve visibility. If all rays are blocked *at the lowest subdivision level* the algorithm falsely concludes that the link is occluded.
The impact of this approximation is not necessaril fatal in this context. First, visibility errors occur *only* at the lowest levels of subdivision, because the hierarchi-cal visibility algorithm automatically refines partial visibility links up to the maximum subdivision limit. This limits potential errors to small regions. Further-more, there will be other viewing regions in general, which can see the corresponding element at least pa r-tially. As long as there is more than one link to an ele-ment, this element will be included in the solution set. Consequently, the implementation *potentially* fails onl for elements that are partially visible from a single viewing region (e.g. something visible through a small hole or crack). Another consequence is that the solution may be sub-optimal, because some small details may be missed. Nevertheless, elements linked to a viewing re-gion are weighted with the visibility estimate in the optimization process and therefore partially visible links are chosen with less preference.
As discussed in section 2.4 the optimization for the best viewpoint positions needs to re-evaluate visibility for partially visible links. The implementation uses again multiple random rays to approximate visibility. Again the same arguments as above holds: if the implementa-tion falsely concludes that an element is invisible, the program may not find the best possible solution a may fail in the presence of small holes or cracks.
The table encoding the mutual visibility of the viewing regions and elements is realized as an array of bit-vectors. All needed operations are then expressed as efficient Boolean operations on elements of this array. Computing the set of visible surface parts from a set of viewing regions is then equivalent to computing the union of the corresponding bit-vectors. Furthermore, the union of all viewing region visibility vectors gives all elements visible from the whole global viewing re-gion. This is used to detect if the optimization has reached the best possible solution.
Our current implementation can also handle only one convex two-dimensional viewing region due to an early implementation choice. Nevertheless, we can argue that the set of everything visible from an empty cubical vol-ume in space is equivalent to the combination of ev e-rything visible from all six faces of this cube. The final result may not be optimal as there are scenes for which a point inside the volume is the best possible answer (consider e.g. a star-shaped scene). As such scenes are rare in practice the restriction to two-dimensional viewing regions was not considered a limit. Further-more, the optimization method presented above does not require one connected global viewing region. This means that the optimization approach is even applicable in the presence of several disjoint viewing regions.

4 Results and Discussion

Experimental results obtained with three real data sets are presented here. All statistics were measured on a SGI Max Impact and timings are given in seconds.

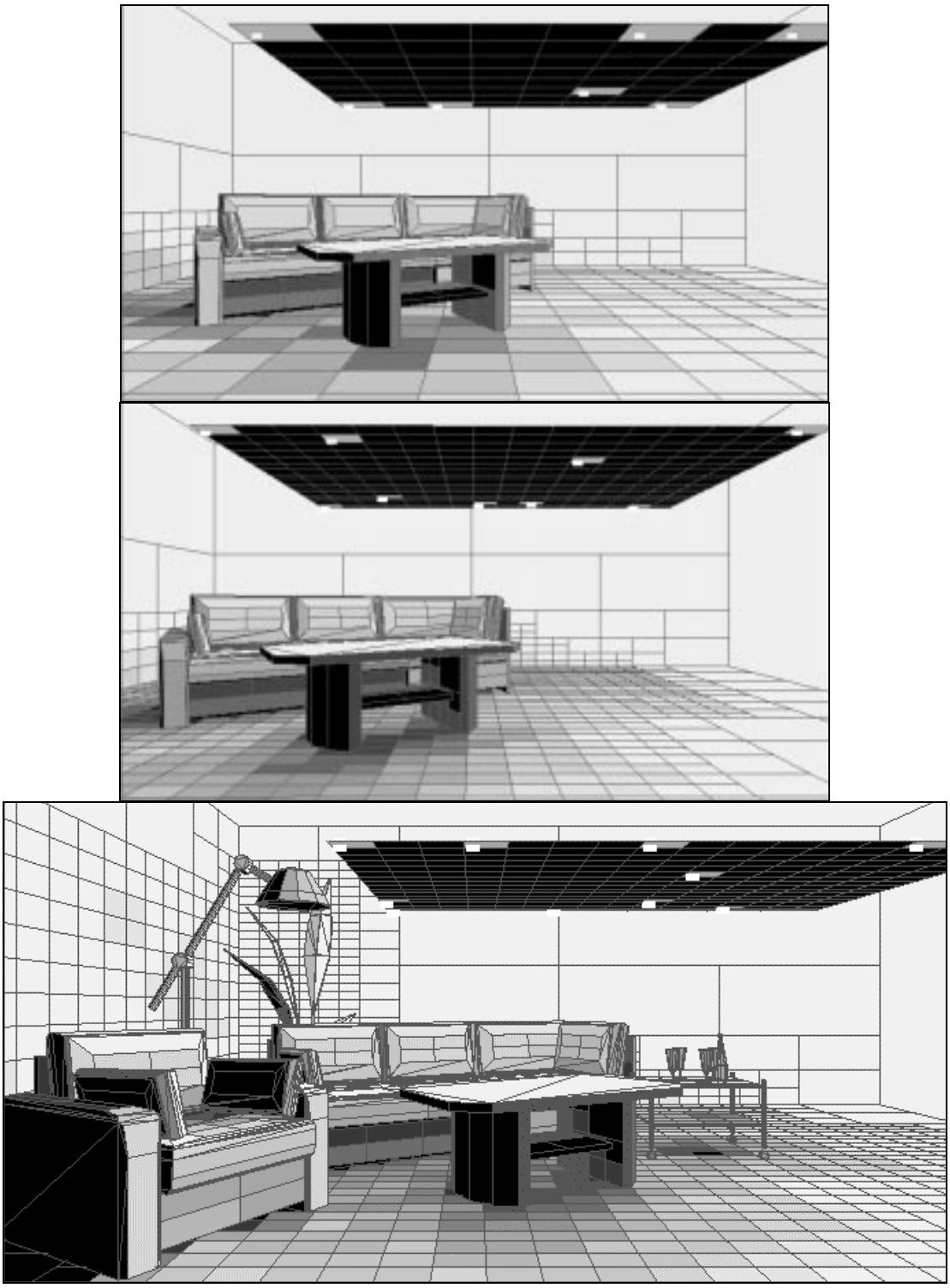

Figure 4: Visualization of solutions for simple environment (top and middle) and for complex environment (bottom). The global viewing region is the large dark polygon near the ceiling, viewing regions are shown in medium gray and small white rectangles depict viewpoints. The color of each surface element encodes how many viewing regions can see it.

First, experiments were performed with a simple living room scene with 316 polygons. The big black polygon at eye height level (visible in the upper part of the image) is the global viewing region.

Two experiments were performed, one subdividing the global viewing region into 64 respective 256 parts. The surfaces were subdivided in the hierarchical visibility-preprocessing phase into 1514 respective 2822 el e-ments.

Figure 4 shows a visualization of the obtained sol u-tions. The global viewing region is depicted as a black polygon, the set of viewing regions is shown in medium gray and the set of viewpoints is depicted as white rectangles. The intensity shown for each surface element encodes how many times each element is visible from the computed set of viewing *regions*. Black signifies that the element is invisible from all viewpoints inside the global viewing region. As the current viewpoint for these images is outside the global viewing region these regions are visible.

The top and middle images of Figure 4 show a set of 7 respective 9 viewing regions. The reason for the difference in the number of viewing regions is that the implementation does not evaluate visibility for partial links at the deepest subdivision level exactly. Consequently, a finer subdivision generates a more correct solution. However, a quick experiment with further subdivision did not yield more viewing regions or viewpoints! Therefore we believe the solution with 9 viewpoint regions to be at least near the optimum.

A more complex living room scene with 1228 polygons was processed with 256 viewing regions. The surfaces were subdivided into 5959 elements. The bottom image in Figure 4 shows a visualization of the solution. B e-cause the scene is more complex 10 viewing regions were necessary to cover all visible surfaces.

The computation of the optimal viewpoints for the mi d-dle image in Figure 4 solution took 166 seconds and for the bottom image 535 seconds to find the shown set of viewpoints.

To empirically verify and visualize the quality of the computed results we generated images that encode the

Figure 5: Visualization of the coverage. Color encodes how many viewpoints see each visible surface point.

visibility of each visible surface point with respect to the computed set of viewpoints.

In Figure 5 each visible surface point was colored according to the number of viewpoints that can see the surface point. Black means that none of the viewpoints could see the corresponding part of the scene. In the images, only surface parts facing away or completel hidden from the global viewing region are black. The black region beneath the table(s) is indeed invisible from the global viewing region. We could detect only one artifact caused by the approximate visibility computation method: The triangular black region under the serving table on the left side in the bottom image.

Table 1 summarizes the runtime and memory statistics for the examples shown. The time spent in visibilit pre-processing grows with the complexity of the environment and the subdivision level. The last entry in Table 1 emphasizes that the optimization times grow more than linearly with the number of viewing regions due to the exponential growth in possible combinations. This is also known as the "curse of dimensionality" in the optimization literature (e.g. [14]), which causes the solution time to grow significantly. The statistics of the optimization process for Figure 5 are summarized in Table 2: Optimization times increase more than linearly with the number viewing regions as more dimensions are added to the problem. The statistics show also that the optimization process quickly finds a good solution for a small number of viewing regions, but converges only slowly to the best possible result. The two-dimensional table for the optimization consumes a small amount of memory. For the more complex example the table was less than 200kB.

To test the scalability of the approach we performed a test on a scene with 62000 polygons on a machine with 128 MB of memory. Unfortunately the program started swapping during the hierarchical visibility calculation and was aborted after 24 hours.

Table 3 shows how long the iterations of the viewpoint optimization for the larger environment took and how good the current solution was. Again, the system quickly reaches a reasonable solution. However, for a good solution the full number of viewpoints is needed. Optimization times for each viewpoint are approximately constant as the stored occluder lists simplify the visibility computation considerably.

5 Conclusions and Future Work

This work introduced a method to compute a good set of viewpoints for a given scene. First, the scene is preprocessed with a hierarchical visibility algorithm. Then a good set of viewing regions is computed. Finally, good viewpoints are identified inside each of the viewing regions. Taking images from the computed set of viewpoints will show everything visible from the global viewing region at least once.

For scenes with known geometry, the presented method can be used to automatically acquire textures for all surfaces. Texture maps for all surfaces can then created by registering the images with the geometry. For the final applications, such as virtual training systems or augmented reality systems, the textured scene geometry will provide images that are much more realistic.

Due to the limitations of our implementation, we cannot guarantee that small regions with complex visibility are correctly included in the solution. Nevertheless, as explained in section 3, as long as there are other viewpoints that see the affected surface parts the algorith will still compute a correct solution. The examples demonstrate that this hold frequently in practice. An implementation of the presented approach that computes visibility exactly (e.g. with the method introduced in [8]) can be guaranteed to compute an exact result.

An interesting fact is that a relatively moderate number of viewing regions suffices for picturing all surfaces in the example scenes. This means that this approach can be used in practice and that it is feasible for automatically gathering texture data in similar environments.

The sampling question was not addressed in this work because of the following issues: Spherical images with finite resolution do not necessarily yield good textures for all surfaces. The following situation demonstrates this: A photograph of a textured wall from across the room will not provide a texture with enough resolution for close-up views. A better sampling strategy would try to minimize the perspective distortion of the images with respect to the textured surfaces. Moreover, there are at least two strategies for acquiring textures that are more detailed - moving the camera closer or zooming. The first alternative may not be feasible if the acquisition device cannot move close enough to a surface because of obstacles. The second requires that a computer controlled, fully calibrated zoom is available on the acquisition device. Both alternatives feature different tradeoffs.

Future work includes:

- Better tuning of the optimization process. Few possibilities for tuning the optimization package were exploited in the current implementation. It might well be that a significant speedup is possible if the parameters of the global optimization algorithm are set appropriately. Another option is to experiment with alternative search algorithms for the combinatorial optimization problem (e.g. Tabu-search [10], [11]).
- The method currently assumes that at each viewpoint a spherical image can be generated. It is easy to show that it might not be necessary to take a full spherical image everywhere. For example in a non-

Figure	Elements	Visibility pre-processing time (sec)	Links created	Opt. viewing regions	Optimization time (sec)	Memory (kb)
4 top	1514	74	18350	7	31	1538
4 middle	2822	311	96936	9	143	5624
4 botto	5959	817	251381	10	956	16252

Table 1: Statistics for viewing region determination.

Number of viewing regions	2	3	4	5	6	7	8	9	10
Percent of final result	97.11	98.37	99.43	99.64	99.85	99.82	99.97	99.97	100
Visibility evaluations.	360	86	249	404	4805	4296	3452	3429	4402
Time (sec)	10	9	19	25	60	98	191	224	317

Table 2: Optimization statistics for optimization process for bottom image of Figure 4.

Percent of final result	91.26	95.72	96.62	97.33	97.48	98.5	99.15	99.52	99.74	100
Optimization time (sec)	52	49	55	87	47	51	47	56	47	44

Table 3: Optimization statistics for viewpoint determination.

textured environment the ceiling may need to be photographed only from one viewpoint. Depending on the sampling strategy this is also applicable in textured environments.

- Another important problem area is that in an automated system side effects such as camera distortion, depth of field, highlights, and reflections need to be handled correctly.

6 Acknowledgements

The author thanks Anselmo Lastra, Gary Bishop, Mary Whitton, and John Amanatides for reading early drafts and the reviewers for their helpful comments.

7 References

[1] Bekaert P., de Laet F. S., Dutre, P., "RenderPark: A Photorealistic Rendering Tool", *http://www.cs.kuleuven.ac.be/cwis/research/graphics/RENDERPARK*, 1997.

[2] Cazals, F., G. Drettakis, C. Puech, "Filtering, Clustering and Hierarchy Construction: a New Solution for Ray-Tracing Complex Scenes", In *Proceedings of EUROGRAPHICS '95* , pp. 371-382, 1995.

[3] Chen, S. E., L. Williams, "View Interpolation for Image Synthesis", In *Proceedings of SIGGRAPH '93*, pp. 270-288, 1993.

[4] Chen, S. E., "QuickTime VR- An Image-Based Approach to Virtual Environment Navigation", In *Proceedings of SIGGRAPH '95*, pp. 29-38, 1995

[5] Darsa, L., B. Silva, A. Varshney, "Navigating Static Environments Using Image-Space Simplification and Morphing", In *Proceedings of 1997 Symposium on Interactive 3D Graphics*, pp. 25-34, 1997.

[6] Debevec, P. E., C. J. Taylor, J. Malik, "Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach", In *Proceedings of SIGGRAPH '96*, pp. 11-21, 1996.

[7] Drettakis, G., F. X. Sillion, "Interactive Update Of Global Illumination Using A Line-Space Hierarchy", In *Proceedings of SIGGRAPH '97*, pp. 57-64, 1997.

[8] Durand, F., G. Drettakis, C. Puech, "The Visibilit Skeleton: A Powerful And Efficient Multi-Purpose Global Visibility Tool", In *Proceedings of SI G-GRAPH '97*, pp. 89-100, 1997.

[9] Gigus, Z., J. Malik, "Computing the aspect graph for line drawings of polyhedral objects. In IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(2), pp. 113-122, 1990.

[10] Glover, F., "Tabu Search - part I", *ORSA Journal on Computing* 1(3), pp. 190-206, 1989.

[11] Glover, F., "Tabu Search - part II", *ORSA Journal on Computing* 2(1), pp. 4-32, 1990.

[12] Haines, E. A., "Shaft Culling for Efficient Ray-Traced Radiosity", In Brunet and Jansen, editors, *Photorealistic Rendering in Computer Graphics*, Springer Verlag, pp. 122-138, 1993.

[13] Hanrahan, P., D. Saltzman, L. Aupperle, "A Rapid Hierarchical Radiosity Algorithm", In *Proceedings of SIGGRAPH '91*, pp. 197-206, 1991.

[14] Ingber, L., "Adaptive Simulated Annealing (ASA)", *http://www.ingber.com/#ASA-CODE* and *http://www.ingber.com/asa_papers*, 1989.

[15] McMillan, L., G. Bishop, "Plenoptic Modeling: An Image-Based Rendering System", In *Proceedings of SIGGRAPH '95*, pp. 39-46, 1995.

[16] O'Rourke J., *Art Gallery Theorems and Algorithms*, Oxford University Press, New York, 1987.

[17] Szeliski R., H. -Y. Shum, "Creating Full Panoramic Image Mosaics and Environment Maps", In *Proceedings of SIGGRAPH '97*, pp. 251-258, 1997.