Analysis of Visibility Masks and Resultant Image Quality

Leo Talbot and Carol O'Sullivan

Image Synthesis Group
Department of Computer Science,
Trinity College, Dublin 2, Ireland
{Leo.Talbot,Carol.OSullivan}@cs.tcd.ie

Abstract. Virtual Interfaces and Visibility Masks are an extremely efficient method for parallelising the Radiosity Method. They exploit data locality, and keep communication between nodes as low as possible. Previous papers on this technique have been mainly concerned with raw performance, and have failed to analyse how these techniques can impact on the quality of the final image. This paper explores the efficiency of generating visibility masks, and analyses the tradeoff between image quality and speed.

1 Introduction

As computing power has increased in the last few years, it has become possible to render larger and more complicated scenes within a reasonable time-frame. As these scenes grow larger, problems are encountered when using the radiosity method to render them. Since the radiosity method exhibits a lack of data locality, large scenes cannot effectively exploit the memory hierarchy of the computer.

When using the ray-tracing algorithm, the computation of two adjacent pixels will most probably result in a similar set of rays being cast, and a similar set of occlusions being tested for. The resultant memory access patterns can make effective use of the computer's memory hierarchy, as the data which is used for calculating the second pixel will have been read into the cache as a result of the previous calculation. However, when we look at the radiosity method, and its one-to-many shooting patterns, it is immediately obvious that these patterns will not gain as much benefit from caching.

Since the arrival of radiosity onto the rendering arena, most work aimed at improving computational efficiency has concentrated on reducing the number of necessary computations, or on using tightly coupled multiprocessor machines. Few solutions have concentrated on the issue of data locality. Those that have are mostly based on divide and conquer strategies: By dividing the scene into local environments, we get improved data locality in the subscenes.

2 Related Work

Some of the earliest divide and conquer approaches to parallel radiosity were proposed by Xu et al. [6] and Neumann et al. [5]. Both approaches utilised the same idea: By subdividing the scene into local environments using virtual walls, the efficiency of the algorithm can be improved. The virtual wall is a temporary storage area for energy as it travels from one local environment to the other. The wall consists of patches which receive energy from the local environment. These patches are then transmitted

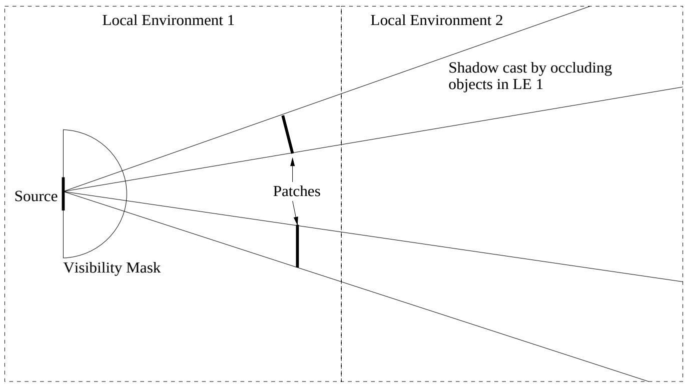


Fig. 1. Visibility Mask

to the neighbouring environment, where their energy is distributed. This technique was refined by Arnaldi et al. [2], who introduced a directional element to the patches on the virtual wall. This lead to more accurate results, but also resulted in extra computation and storage requirements.

The local environment concept was further refined by Arnaldi et al. [1] with the Visibility Masks method. Here, the environment is once again subdivided into local environments, but there are no virtual walls dividing them. Instead, when a patch shoots into the local environment, it generates a visibility mask. The mask acts as an occlusion buffer, and stores the visibility of the patch from outside the local environment (Fig 1).

3 Theory

The progressive refinement [3] technique, with Monte Carlo integration for form factor calculation, is used to evaluate the radiosity solution for the local environment. In this technique, the patch with the greatest unshot energy is chosen, and this energy is then distributed around the scene. The patch is then propagated to neighbouring environments, where it is placed on a queue. When this patch has the greatest amount of unshot energy in the new environment, its energy is distributed, and so on until the patch has fully distributed its energy. During the distribution in an environment, the visibility mask is filled in. The mask is represented by a discretised hemisphere. Each pixel of the hemisphere corresponds to a boolean value in the visibility mask. A value is set to false if a ray passing through that pixel encounters an object in the environment.

One of the more important decisions that must be taken when initialising the visibility mask is its resolution. Masks with lower resolutions take less time to fill in, and use less network bandwidth, but obviously suffer from a deterioration in definition. This can have a adverse effect on image quality. Shadow edges become less defined, and discontinuities in shadow edges can occur at the boundaries between environments.

4 Algorithm

The Hemisphere of the visibility mask is represented as a 2D array of bits: A 0 represents false (an occlusion), and a 1 represents true (no occlusion). The indices for the array are θ and ϕ of the outgoing ray which originates at the centre of the shooting patch. We decided to utilise a uniform pixel structure, as this would be extremely efficient in the transmission of patches between local environments. Since the 2D array is basically a flat array of integers in memory, it is trivial to pack it into the outgoing packet, and just as importantly, trivial to unpack at the destination. The choice of θ and ϕ as indices also gives the mask better resolution in the normal direction, where it is needed most.

The hemisphere's resolution is specified by an integer variable r. The resolution along ϕ is 4 times as great as that along θ.

There are two choices for filling in the hemisphere:

- Method A: Fill in the visibility mask while shooting rays when calculating the form factor. No extra rays need to be shot, and hence the only extra computation necessary is the actual filling in of the mask. However, pixels may be marked off multiple times and there is no efficient way of preventing this wastage. The resultant wasted computation can prove expensive if the number of rays being shot is very large. This method does not guarantee 100% accuracy. If the flux of the rays in a particular direction is not high enough, not all relevant pixels will be marked. This can lead to "light leaks" in the mask, and hence shadows may not be as dark as they should be.
- Method B: Fill in the visibility mask after form factor computation. This involves shooting a ray through each pixel in the hemisphere, and marking the pixel if the ray hits an object. This method incurs extra computation, as extra ray casts are needed. As the resolution of the mask increases, so does the number of ray casts required. This method does guarantee the correctness of the mask however.

5 Results

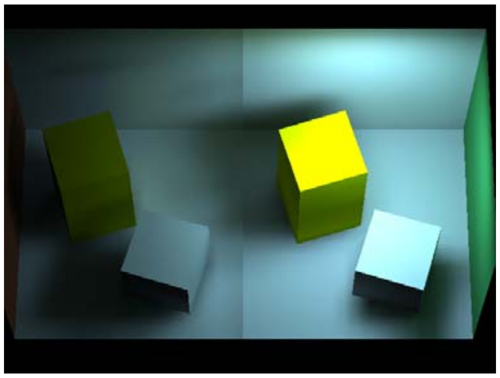
A modified Cornell Box was used as the test scene for analysing image quality. This particular scene was chosen due to the shadow cast by the tall yellow box on the left hand side, which was particularly good at picking up errors.

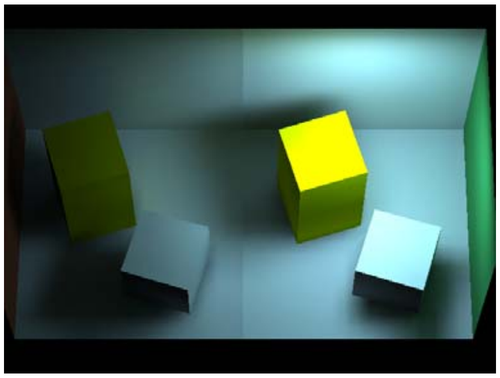
Figure 2 shows this scene rendered with varying resolutions[1], and a reference image. The reference image was rendered using the same scene data, but with a resolution of 250. When the resolution is 20, there is a clear deterioration in the image quality at the edge of the above-mentioned shadow. Due to the poor resolution of the mask, the shadow has poorly defined shape, with jagged edges. There is also a noticeable discontinuity in the shadow edge as it crosses from one environment to the other. This discontinuity is also present in the other images, but it is much less noticeable. If the reference image is compared with the other images, there is very little difference between it and when the visibility mask resolution is 100. When the resolution is reduced to 50, there is some loss in definition, but it is barely noticeable.

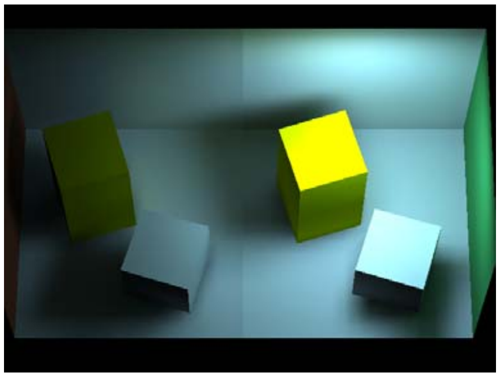
Figure 3 contains an image that has been rendered using method A with a very high mask resolution. The visibility masks have not been filled in properly, and light has leaked through into the environment on the right. Hence the shadow cast by the tall yellow box is completely wrong.

Table 1 contains details of execution time for the scene using both methods at various resolutions. Computation was done on a uniprocessor sytem. This was to in order to prevent network latency from interfering with the results. It is immediately obvious that method B is far quicker at lower resolutions. The two methods are nearly equal

[1]In this section, resolution refers to the resolution of the visibility mask


(a) Resolution = 20


(b) Resolution = 50


(c) Resolution = 100

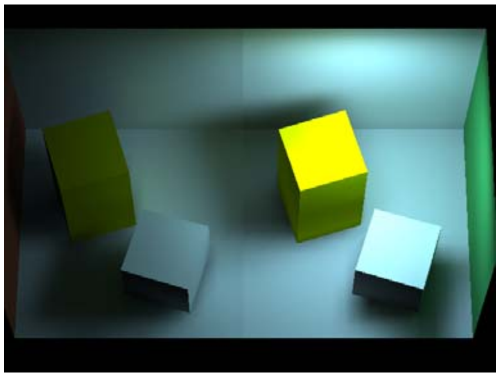

(d) Reference Solution

Fig. 2. Quality vs. Resolution

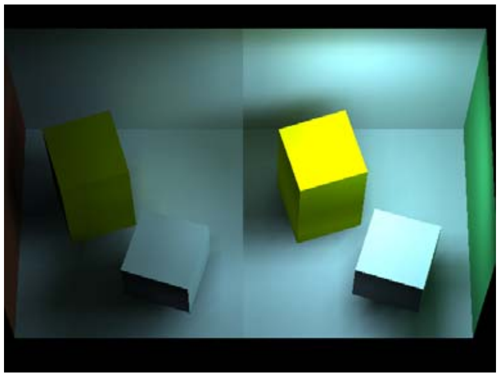


Fig. 3. Light leaking through visibility mask

Resolution	Method A	Method B
20	196m 38s	171m 10s
50	205m 11s	179m 30s
100	209m 24s	202m 37s

Table 1. Execution Time

when the resolution approaches 100. However, method B would still be the preferred choice, as it is at such resolutions that light leaks start to occur in the mask. The probability of light leaks occuring can be reduced by making the patches in the scene smaller or by increasing the number of Monte Carlo samples when calculating the form factor, thus increasing the chances that a pixel on the hemisphere will be hit by an outgoing ray. Both these solutions will add greatly to computation time however, rendering them effectively useless.

6 Conclusion

The resolution of visibilty masks can have a profound impact on image quality. Increased resolutions clearly give better image quality, but this quality comes at a price. Higher resolutions for method B give longer computation time, while with method A, we must face the probability of light leaks.

Method B could have problems dealing with extremely complex scenes however. Since this method uses extra ray casts, the number of surfaces in the scene will have an impact on performance. The use of a well-tuned acceleration system (e.g. a SEADS grid [4]) should reduce this to a minimum.

7 Acknowledgments

We would like to thank William Leeson and Gareth Bradshaw for their help. This project has been supported by an Enterprise Ireland strategic research grant and Hitachi Dublin Laboratory.

References

1. Bruno Arnaldi, Thierry Priol, Luc Renambot, and Xavier Pueyo. Visibility Masks for Solving Complex Radiosity Computations on Multiprocessors. In *Proc. First Eurographics Workshop on Parallel Graphics and Visualisation*, pages 219–232, Bristol, UK, September 1996.
2. Bruno Arnaldi, Xavier Pueyo, and Josep Vilaplana. On the Division of Environments by Virtual Walls for Radiosity Computation. In P. Brunet and F. W. Jansen, editors, *Photorealistic Rendering in Computer Graphics (Proceedings of the Second Eurographics Workshop on Rendering)*, pages 198–205, New York, NY, 1994. Springer-Verlag.
3. Michael Cohen, Shenchang Eric Chen, John R. Wallace, and Donald P. Greenberg. A Progressive Refinement Approach to Fast Radiosity Image Generation. In *Computer Graphics (ACM SIGGRAPH '88 Proceedings)*, volume 22, pages 75–84, August 1988.
4. Akira Fujimoto, Christopher G. Perrot, and Kansei Iwata. Environment for fast elaboration of constructive solid geometry. *Advanced Computer Graphics (Proceedings of Computer Graphics Tokyo '86)*, pages 20–33, 1986.
5. Laszlo Neumann and Attila Neumann. Photosimulation: Interreflection with Arbitrary Reflectance Models and Illumination. *Computer Graphics Forum*, 8:21–34, 1989.

6. Hau Xu, Qun-Sheng Peng, and You-Dong Liang. Accelerated Radiosity Method for Complex Environments. In *Eurographics '89*, pages 51–61. Elsevier Science Publishers, Amsterdam, North-Holland, September 1989.