Efficient Rendering of Radiosity using Textures and Bicubic Reconstruction

RUI BASTOS
MICHAEL GOSLIN
HANSONG ZHANG
{bastos|goslin|zhangh}@cs.unc.edu

Department of Computer Science
University of North Carolina at Chapel Hill
CB# 3175, Sitterson Hall
Chapel Hill, NC, 27599-3175
USA

ABSTRACT We present a method to speed up walkthroughs of static scenes. It involves the creation of a continuous C^1 radiosity reconstruction for adaptively sampled regions. This representation is a unified solution to handle unrestricted quadtrees and T-vertices, and allows for the generation of multiple different levels-of-detail of the radiosity function, which is represented as texture maps. The method also involves the use of hardware bicubic filtering for the radiosity display. Both techniques allow improvements in performance and memory usage while preserving visual appearance.

1 INTRODUCTION

An important goal in walkthrough applications is to interact in real time with a model that has rich visual complexity. One way to provide such complexity is by using global illumination techniques to approximate the illumination function of the scene.

Illumination functions can be separated into diffuse (view-independent) and specular (view-dependent) components. This work concentrates on the use of the diffuse component by exploiting its off-line computation (prior to the walkthrough application), and view-independence. We choose to compute the diffuse interchanges between elements in a scene using the radiosity method.

In order to obtain a very rich scene, the traditional radiosity adaptive subdivision greatly increases the number of polygons of the model. This actually does not change the geometrical shape of the model and incurs large cost in terms of the time to compute a dense solution, the space to store the immense model that is produced, and the time to render the greatly increased number of geometrical primitives on contemporary graphics hardware.

Our method decouples shading from geometry and uses C^1 interpolation and filtering schemes to achieve speedups in walkthroughs by minimizing the problems mentioned above without sacrificing visual appearance. We first reconstruct a bicubic representation of the radiosity solution from which we can generate more efficient versions of the model. We then employ the concept of representing radiosity as texture maps in order to decouple geometry from shading. Finally, we use bicubic interpolation in hardware to further improve the performance and appearance of the solution model.

1.1 BACKGROUND AND PREVIOUS WORK

In developing our approach to efficient rendering of radiosity, we drew upon previous work from two fairly disparate areas of research: higher order interpolation and representing radiosity as texture maps.

1.1.1 Higher Order Interpolation

The radiosity process computes samples of the illumination function to an arbitrary degree of accuracy for selected points in the scene. In order to render a processed model, the radiosity function is reconstructed for all visible points of the scene based on the computed samples.

Usually, the radiosity function is reconstructed using bilinear interpolation of neighboring samples. One can see this is a good approximation inside the region being represented, but it is not at the edges [3] [4] [17]. Our visual system is sensitive to derivative discontinuities [13] and the C^0 bilinear interpolation between any neighboring regions leads to noticeable visual Mach band artifacts [1] [15].

According to Bastos [1], a bicubic interpolation scheme can ensure up to C^1 continuity (first derivative) of the reconstructed function. This type of interpolation can prevent Mach band artifacts often resulting in lower order interpolation.

1.1.2 Radiosity As Texture - *RAT*

The fairly regular sampling produced from an adaptive radiosity solution (figure 2) can be effectively represented as a texture map instead of a mesh of gouraud-shaded polygonal elements. This can significantly increase rendering performance on current graphics hardware that support advanced texture mapping capabilities.

Heckbert [6] proposed the use of texture mapping as an alternative to the mesh based radiosity approach in 1990. In his approach the energy arriving at a surface is computed using Monte Carlo ray tracing and stored at texels of radiosity textures (*rexes*). Uniform adaptive sampling is supported organizing rexes into quadtrees. During rendering, radiosity at any pixel in the final image is approximated using bilinear interpolation. Due to the Monte Carlo ray tracing and the bilinear reconstruction step, Heckbert's approach can generate noisy and discontinuous images.

Myszowski and Kunii [11] describe a method to replace the most complex radiosity mesh areas with texture maps based on available texture memory and the number of mesh elements that would be eliminated. Surfaces that are selected are rendered as a gouraud-shaded polygonal mesh and a texture map is retrieved directly from the surrounding rectangular region in the frame buffer. In situations where there are mesh-based artifacts in the lighting solution, they recalculate the solution directly to the texture map itself by applying radiosity sampling at locations corresponding to texels. Radiosity texture maps are rendered using bilinear texture interpolation on a Silicon Graphics RealityEngine.

Möller [9] describes a method to replace radiosity solutions for NURBS (geometry) models with a single texture map. This allows for multiple different levels-of-detail of the model with the same illumination texture map. He generates his textures from a radiosity mesh. Texel values that have no corresponding mesh values are filled in using bilinear interpolation. The textured models are rendered on a Silicon Graphics RealityEngine using bilinear texture interpolation mode.

1.2 OVERVIEW

The remainder of this paper is organized as follows. In the next section we briefly describe our system and the various stages of processing. In section 3 we present our contributions in detail by focusing on several important reconstruction and rendering issues. In section 4, we present some results both in terms of performance and appearance of our processed models. Finally, in section 5, we close with conclusions and future work.

2 SYSTEM

The overall structure of our system is presented in Figure 1. We start with a polygonal model that is processed using radiosity

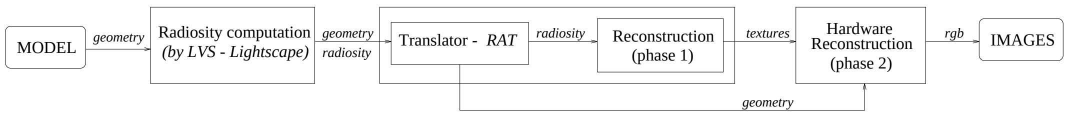


Figure 1: *Overall structure of our system.*

software from *Lightscape© Technologies*. The radiosity function is then reconstructed in two separated phases. In the first phase, three (R, G, B) bicubic reconstructed radiosity patches (*RRP*s) are created for every element in the solution model using the computed radiosity samples [1]. Each *RRP* can then be sampled at arbitrary resolution to generate corresponding radiosity texture maps. These radiosity texture maps are applied to the original polygons of the model so that we can eliminate radiosity mesh geometry artificially created by the adaptive subdivision. Finally, a second reconstruction phase occurs at run time when the *RAT* model is rendered using bicubic texture map filtering on a Silicon Graphics RealityEngine^2.

3 CONTRIBUTIONS

In this section we will attempt to clearly define our contribution to this area of research and distinguish our system from previous work by focusing on several important reconstruction and rendering issues. We begin by defining our default rendering environment and describing metrics we used to evaluate both the performance and appearance of processed models. We then describe some of the salient features of our method and what we consider to be their advantages over previous work.

3.1 Default Environment and Common Metrics for Evaluation

Unless stated otherwise, trials were run on a Silicon Graphics RealityEngine^2 with four RM5 boards. Models were rendered as antialiased, z-buffered, and textured polygons. Hardware lighting was disabled because models with radiosity solutions have lighting that is intrinsic to the polygonal surfaces.

Performance numbers were gathered using statistics available in Iris Performer applications. Some additional data was collected using software developed using the Iris GL graphics library. Most measures are in terms of frames per second or time to render a single frame.

We make quantitative comparisons between images generated by our reconstruction methods and reference images generated by sampling the radiosity function to an extremely fine resolution (nine levels of adaptive subdivision - up to 513×513 samples). Our comparisons are perceptually meaningful, i.e., the quantitative difference between two images indicates their disparity as seen by the human visual system. Perceptual image metrics are discussed in detail in [14].

A good image metric must take into account characteristics of human perception, such as sensitivity to relative luminances, non-linear response to brightness, and different sensitivity to different spatial frequencies. We use the first model described in [14]. In this image metric, the image is first normalized by the mean luminance. Then, the value of each pixel is replaced by its cube root. Next, the resulting image is transformed to the frequency domain and a contrast sensitivity function (modeling frequency sensitivity) is applied to filter the frequencies. Two images are compared by finding the mean square error of their filtered frequencies. The result of comparing two images is a number representing the “perceptual distance” between them. This number is zero when the two images are equal and increases in magnitude with increasing perceptual difference between the two images.

3.2 Radiosity Samples Direct from Quadtree

We generate radiosity reconstruction patches (*RRP*s) directly from the quadtree data structures produced by an adaptive subdivision radiosity calculation. This approach has several advantages over capturing radiosity textures directly from the frame buffer [11]. First, there is no distortion of color values as a result of the rendering process, and second, we can handle radiosity representations that are not right rectangular regions.

3.3 Continuous Representation of Radiosity

We generate a continuous approximation to the radiosity function at each surface in the model using bicubic interpolation in order to ensure up to C^1 continuity. As in [1], the bicubic Hermite form was chosen due to its use of derivative information and the relative ease of approximating such quantities for the radiosity samples. For a detailed description of the radiosity bicubic Hermite reconstruction see [1].

Given a full quadtree data structure representing the radiosity values at the vertices of a regular subdivision of a polygon, we create a bicubic patch per color components (R, G, B) for each leaf in the tree. The bicubic patches are computed such that we can ensure C^1 continuity between any two neighboring patches of the same color component in the quadtree.

The result for an unsubdivided polygon is a single bilinear patch, which is the best interpolation we can do given only four points. For one level of subdivision we get four patches, and so on. Even for shallow quadtrees, we ensure the radiosity reconstruction is C^1 continuous inside the quadtree region.

3.4 Scaling and Levels of Detail from Resampling

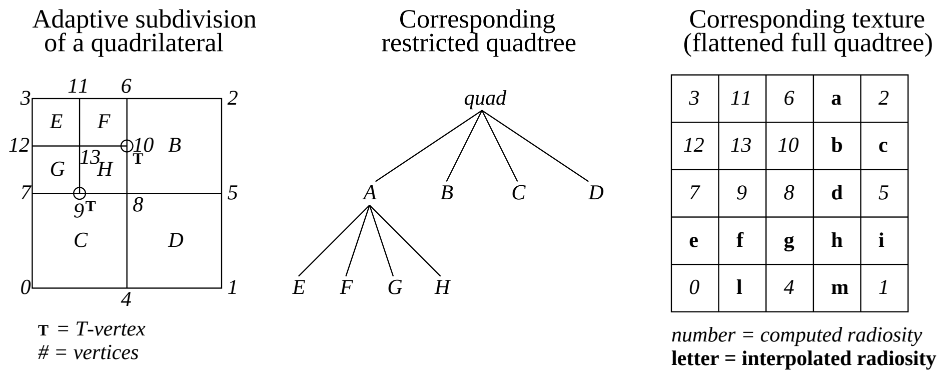
The actual subdivision of a sparse quadtree generates fewer polygons than its corresponding full quadtree. In general, adaptive subdivision generates sparse unbalanced quadtrees. However, when converting a nonfull quadtree into a texture, the nonsampled points in the quadtree have to be evaluated from the surrounding given points leading to a full quadtree. A simple solution is bilinear interpolation [9] which produces almost the same results as the Gouraud shading of the mesh associated with the sparse quadtree and does not improve smoothness if resampled at higher rates. In order to reduce Mach banding artifacts, when using linear interpolation, it is necessary to work with densely sampled radiosity solutions.

We can take advantage of our continuous bicubic representation for the radiosity function by resampling it to arbitrary resolutions in order to generate radiosity texture maps. Notice that the problem of unbalanced quadtrees and corresponding T-vertices is solved by the bicubic reconstruction. In addition, because scaling is accomplished by resampling a continuous function rather than by simple pixel replication or linear interpolation, we can actually improve the smoothness of the image by scaling upward. The ability to scale texture sizes in this fashion has important implications for controlling the use of limited texture memory and providing the ability to have multiple levels-of-detail.

On the RealityEngine^2 platform, textures are stored in memory as blocks that are exact powers of 2 in dimension, so smaller textures still require the same amount of storage as the next power of 2 size up. Radiosity computations that use adaptive subdivision will inherently produce arrays of sample points that are not powers of 2 on a side. A single subdivision produces a 3 by 3 array of samples, two subdivisions produce a 5 by 5 array of samples, and so on. This requires us to scale every radiosity texture in some way in order to take full advantage of available texture memory on the RealityEngine^2. We achieve such desired resolutions resampling the bicubic radiosity reconstruction at the desired rates.

3.5 Handling T-vertices and Unrestricted Quadtrees

Radiosity adaptive subdivision can generate unbalanced quadtrees with nonconforming elements in regions with different mesh densities [3] [17]. Figure 2 presents an example where neighboring elements have different levels of subdivision creating T-vertices in the mesh (vertices 9 and 10). T-vertices are undesirable because the interpolated values at those points will be different from the radiosity value actually computed at that same point. This generates a visual discontinuity along the edges supporting that vertex [13].


Adaptive subdivision of a quadrilateral
3
11
6
2
E
F
12
10
B
G
13
H
T
7
5
9
T
8
C
D
0
4
1
T = T-vertex
# = vertices
Corresponding restricted quadtree
quad
A
B
C
D
E
F
G
H
Corresponding texture (flattened full quadtree)
number = computed radiosity
letter = interpolated radiosity

3	11	6	**a**	2
12	13	10	**b**	**c**
7	9	8	**d**	5
e	**f**	**g**	**h**	**i**
0	**l**	4	**m**	1

Figure 2: *Example of adaptive subdivision, restricted quadtree representation, and corresponding texture map.*

Several solutions to the T-vertex problem have been proposed in the finite element literature. We describe three of them applied specifically to the radiosity domain.

The first approach substitutes the radiosity value computed at the T-vertex by the interpolated shading value at that point so that there is no longer a magnitude (D^0) discontinuity along the edges sharing that vertex [3]. An obvious disadvantage of this approach is that it introduces error by replacing an accurate radiosity sample with an interpolated value.

The second approach triangulates the large element that neighbors the T-vertex so that all the new triangles share the radiosity value at the point of the T-vertex [2]. This approach does not introduce error in the computed solution, but it can produce triangles with poor aspect ratios. Restricted quadtrees are generally used to minimize this problem. In a restricted, or balanced, quadtree, two elements or leaves are allowed to differ by at most one level of subdivision. This implies that the adaptive subdivision will be forced to sample some regions more finely than would otherwise be necessary. This is unfortunate given the goal of adaptive sampling is to avoid oversampling regions where the radiosity function is almost flat and could be approximated well with interpolation.

The final approach uses Delaunay triangulation to avoid the limitation of unrestricted quadtrees. Because this is a mesh-based approach, it can potentially produce a proliferation in the number of final polygons and it may lose some of the hierarchical information of the quadtree.

Our solution is based on a recursive bicubic reconstruction. This technique does not subdivide the original polygon, keeps all the original radiosity samples, and creates a continuous radiosity presentation all over the original polygon.

We begin by creating a bicubic patch per color component for every leaf in the quadtree. This does not ensure C^1 continuity along edges containing T-vertices. Then, bicubic patches at leaf nodes that are not at the deepest level of the quadtree are midpoint subdivided. Already existent radiosity values are shared by the new patches. Radiosity values for new points are approximated interpolating the bicubic patch in the level above in the quadtree. This process is recursively applied until we have a full quadtree.

At this point, we have a completely C^1 continuous representation of the radiosity function inside the original domain. We can flatten this full quadtree into a texture (Figure 2 - right) resampling the bicubic patches (*RRP*s) at the desired rate.

One can easily see how this technique generalizes directly for unrestricted quadtrees handling T-vertices in the same simple way.

3.6 Hardware Bicubic Filtering

We choose to view the rendering of radiosity texture maps that have been generated from *RRP*s to be a second phase of reconstruction. At this final rendering phase we again start with sampled data in the form of a texture map, resulting from the first reconstruction phase, and reconstruct intermediate pixel values on screen as we render the textured polygon.

The usual reconstruction technique for rendering textures is bilinear interpolation. This type of interpolation is readily available and fast on current hardware such as RealityEngine^2, with published fill rates of 320 million pixels per second [16]. We have observed comparable performance in the more restricted case of *RAT* models consisting mainly of large unlit 3-component textures with 12 bits per component (figure 3).

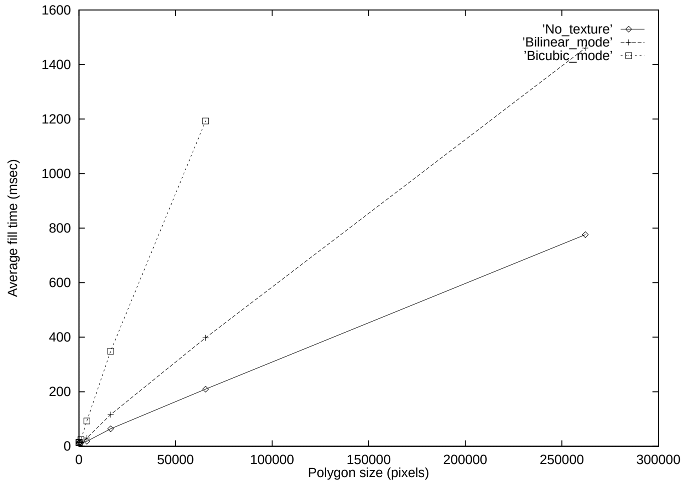


Figure 3: *Performance for bilinear and bicubic rendering modes compared with rendering the same polygon with no textures. In general, we are interested in polygons smaller than 65536 pixels on the screen (*256×256*) where we observe that the bicubic mode is three times slower than the bilinear mode.*

There are several disadvantages to using bilinear interpolation for *RAT* models. Although this type of interpolation can ensure continuity of intensity, it can not represent first derivative continuity, which causes Mach band effects in the reconstructed images (Figure 6.(c)). Notice that Image (c) shows Mach band effects even though the texture was generated from a smooth bicubic *RRP* representation. We would clearly prefer not to introduce severe artifacts during this second reconstruction. A second disadvantage of bilinear interpolation are the same Mach band artifacts that appear when even very dense textures become magnified excessively, as described by Myszowski [11].

Since we assume the shading function is smoothly continuous inside each polygon, we want to have that same smoothness when rendering the radiosity texture. This suggests the use of bicubic filtering [16] at the final rendering phase. Such a mode is available on current RealityEngine^2 systems. In order to render our textures smoothly, we replaced the default filter with a bicubic (1/3, 1/3) filter as described by Mitchell and Netravali [8].

Radiosity textures rendered with bicubic filtering do not show the Mach band artifacts that appear when the same textures are rendered using bilinear interpolation (Figures 6.(d) and (c)). These textures appear smooth even when magnified greatly on screen. However, the bicubic filtering is subject to ringing and blurring effects and it is often necessary to trade off one type of distortion for another [8]. Figure 6.(d) shows the blurring effect when compared to figures 5.(a) and 5.(d). Because the bicubic filter effectively blurs the image, we lose some of the contrast of the original in exchange for elimination of the Mach artifacts.

Bicubic interpolation is slower than bilinear, too. Silicon Graphics claims fill rates of 54 million pixels per second [16], and we have observed similar performance in empirical tests with *RAT* models (figure 3). We can partially compensate for the slower performance by taking advantage of the smoother rendering of textures that comes from bicubic filtering. We can use bicubic filtered textures that are smaller than the bilinearly interpolated ones and that still have less artifacts.

Using bicubic filtering, therefore, allows us to use smaller textures that require less space and render quickly, while still maintaining the smoothness of the radiosity reconstruction. It is possible to perform by hardware Hermite bicubic interpolation identical to the one used in the first reconstruction phase [12]. However, we avoided to use that kind of filtering due to its higher cost in texture memory and computational time and the reduced improvement in image quality.

3.7 Not Just Textures - Rendering *RAT* Models on PixelFlow

Because we store data in continuous form as *RRP*s, we are not restricted exclusively to texture maps as our output format. We have obtained some preliminary results with outputing parametric descriptions of *RRP*s and using these to render directly.

One such implementation makes use of deferred shading available on the PixelFlow architecture under development at UNC [10]. *RRP*s are rendered by rasterizing the bicubic coefficients as well as the parametric coordinates. After all primitives have been rasterized, pixel fragments are shaded in parallel on the SIMD array using the local coefficients and coordinates.

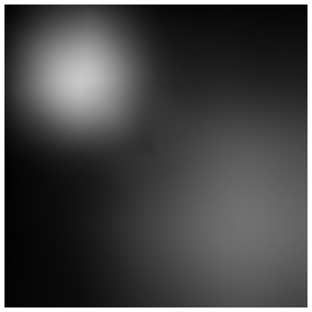


Figure 4: *Image generated using PixelFlow simulator.*

4 RESULTS

4.1 Performance of *RAT* Models

We observed that replacing polygonal radiosity meshes with texture maps dramatically improves rendering performance (table 4.1), even across a variety of real-world models that contain mixes of meshes and textures. Table 4.1 lists the number of triangles and number of vertices for the original (orig) model before radiosity solution, for the adaptively subdivided model (poly) after radiosity solution, and for the *RAT* model. It also presents the frame rates when visualizing the radiosity solution with the adaptively subdivided model and the *RAT* model. Due to saturation effects in the performance meter of the SGI Iris Performer, we were not able to quantify frame rates higher than 30 frames per second. The table also compares the size of the models and the use in texture memory for *RAT* models.

4.2 Appearance of Reconstructed Textures

Figure 5 presents a set of images demonstrating how our first phase bicubic reconstruction can reduce the effects of artifacts resulting from undersampled radiosity solutions. Image (a) is a reference image generated from a deep quadtree (depth 9: up to $(2^9 + 1) \times (2^9 + 1)$ samples). Image (b) is the image corresponding to the quadtree (depth 2: up to 5x5 samples) used to reconstruct textures (c) and (d). Images (c) and (d) are resamplings (*RATs*) of the initial information (image (b)) using bilinear and bicubic interpolation, respectively. Image (c) shows a prominent Mach band (star-shaped) artifact in the center of the bright spot in the upper left corner. This artifact is produced by linear interpolation between samples in the shallow mesh (depth 2) of figure 6.(b). Scaling

Name of the model	Triangles number	Vertices number	Frame rate	Size (MBytes)	Texture memory	Texture number
grotto (orig)	804					
grotto (poly)	158,670	208,052	3.8	2.23		
grotto (*RAT*)	804	1,608	>30.0	0.23	4.64 MB	402
lavapit (orig)	1,336					
lavapit (poly)	55,158	71,648	7.5	1.02		
lavapit (*RAT*)	7,188	9,195	>30.0	1.09	1.41 MB	172
mausoleum (orig)	1,022					
mausoleum (poly)	41,342	51,916	20.0	0.82		
mausoleum (*RAT*)	8,208	10,604	>30.0	1.40	1.90 MB	41

Table 1: *Frame rates for RAT versus polygonal radiosity models.*

the image up to 256 $\times$ 256 pixels with bilinear interpolation, as we see in (c), obviously does not improve the smoothness or reduce the artifacts. Image (d) shows how the bicubic reconstruction is used to eliminate the undersampling artifacts and get smooth results with no Mach band artifacts. Notice that the overall shape of the bright spot is rounder and there is no apparent star-shaped artifact in the center of the spot, as expected from the reference image (a).

A quantitative comparison of the two reconstructed images when compared to the reference image confirms that the bicubicly reconstructed image is closer in appearance to the optimal solution. Applying the perceptual image metric [14] to the reconstructed images and the reference image we get

	reference	bicubic	bilinear
reference	0.0	65.03	69.67

where numbers increase with perceptual differences in the images. The metric appears to agree with the differences that are visibly apparent. Because the bicubic reconstruction image is very close to the reference image, it would appear to be an efficient way to represent the ideal solution by using a much smaller number of radiosity samples.

4.3 Appearance of Rendered Images

Figure 6 presents a set of images demonstrating the use of hardware bicubic filtering of texture maps to produce a rendered image that is smoother and shows fewer shading artifacts. Image (a) shows the initial information used for the first reconstruction phase (same as in figure 5.(b)). Image (b) shows the resampling of the bicubic representation to 8 $\times$ 8 pixels (SGI allocates texture memory assuming powers of 2 pixels on the side of the textures - we scale up to use all the memory allocated). Image (c) shows that even a texture map that has been generated from a bicubic patch can show artifacts if the texel values are interpolated using bilinear mode in hardware. Notice that the same image appears to be much smoother (image (d)) when it is rendered using bicubic filtering mode.

The result of applying the image metric in the hardware shaded images is

	reference	hardware bicubic	hardware bilinear
reference	0.0	167.3	218.3

4.4 Results on a Real Model

Figure 7 presents a series of images of the "grotto" model, for which performance values have already been shown in table 4.1. This model consists of 402 textured quadrilaterals rendered using radiosity textures and bicubic filtering. On the left we see the original wireframe model with 402 quadrilaterals. On the right we see the polygonal radiosity version (more than 150,000 triangles) result of the adaptive subdivision. And in the middle we see the *RAT* model with 402 quadrilaterals and corresponding 402 textures replacing the additional 150,000 triangles.

5 CONCLUSIONS AND FUTURE WORK

Our current system produces models that can be rendered efficiently without sacrificing appearance. We use bicubic Hermite interpolation to reconstruct the radiosity function. This allows us to start with a radiosity solution that has few radiosity samples and

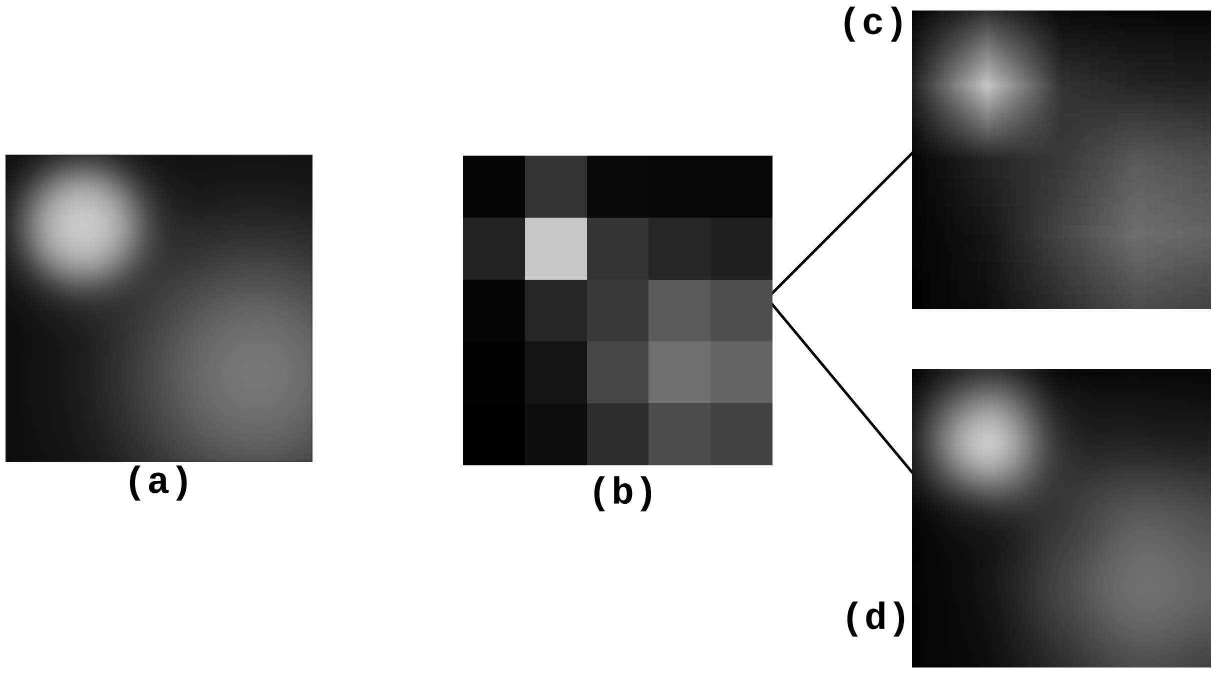


Figure 5: *Results of the first C^1 reconstruction phase for a single plane with two spots on it. (a) Reference image generated from a depth 9 quadtree (up to* 513×513 *samples). (b) Initial image (depth 2 quadtree) used for the first reconstruction phase. (c) RAT: bilinear reconstruction of image (b) (*256×256 *resampling). (d) RAT: bicubic reconstruction of image (b) (*256×256 *resampling).*

that still appears to be smooth (without Mach band artifacts). A simple radiosity solution also takes less time to compute and can be stored more compactly.

The continuous representation of the radiosity function allows us to generate radiosity textures at arbitrary scales by resampling. This enables us to make more effective use of limited texture memory and generate multiple levels-of-detail.

We can further reduce our texture sizes without introducing Mach band artifacts during rendering by using hardware bicubic filtering to render radiosity textures.

PixelFlow : Although texture maps are a more efficient means of storing radiosity samples than polygonal meshes of vertices both in terms of storage and rendering performance on current graphics hardware, we can still do better. We would prefer to both store and render the continuous radiosity reconstruction itself. This representation is inherently more compact than a texture map of samples, and does not require scaling. Additionally, we can avoid a second reconstruction of the radiosity function when we render by computing on-screen pixel values directly from the function itself.

This implies hardware support for a polygonal primitive with an associated function that describes its color value at every point. Such support is available using the deferred shading capability of the PixelFlow architecture [10]. We propose to investigate further implementations of rendering directly from the initial reconstruction of the radiosity function using this platform when it becomes available.

A Priori Discontinuity Meshing : We assume that the radiosity function is fairly continuous within each polygon, but this is often not the case in realistic scenes. In order to guarantee this property in all cases, we can accurately represent discontinuities using discontinuity meshing algorithms [7]. Unfortunately, these algorithms often produce a tremendous increase in the number of polygons in the processed model and this will increase the number of textures for *RAT* models. However, as the adaptive subdivision will not have to capture discontinuities, the quadtrees will be shallower requiring less texture memory.

Multiple Textures : Our current implementation does not allow a polygon to have a surface texture as well as a radiosity texture. We would like to be able to combine textures at different scales so that we could have a brick wall with shadows, for example. This could be accomplished on the RealityEngine^2 by rendering the polygon in multiple passes or composing both color and radiosity in the same texture.

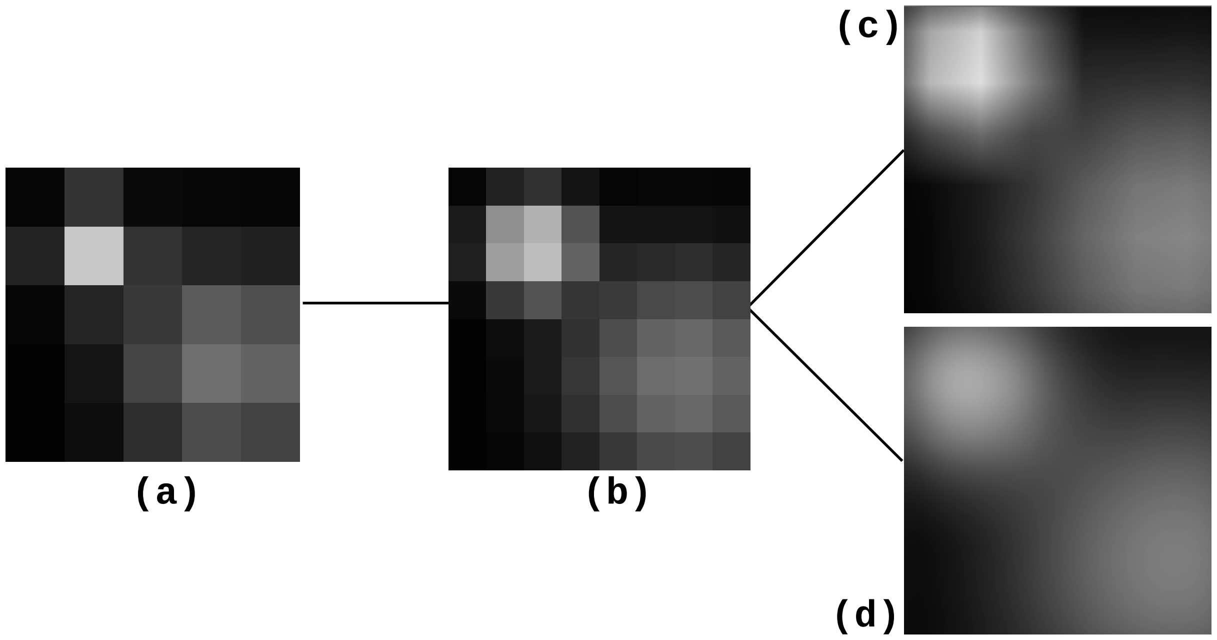


Figure 6: *Results of the hardware shading for a single plane with two spots on it. (a) Initial information (depth 2 quadtree) used for the first reconstruction phase. (b) RAT: First phase bicubic reconstruction of image (a) with* 8×8 *resampling. (c) On screen: Bilinear hardware shading of image (b) (*256×256 *resampling). (d) On screen: Bicubic hardware shading of image (b) (*256×256 *resampling).*

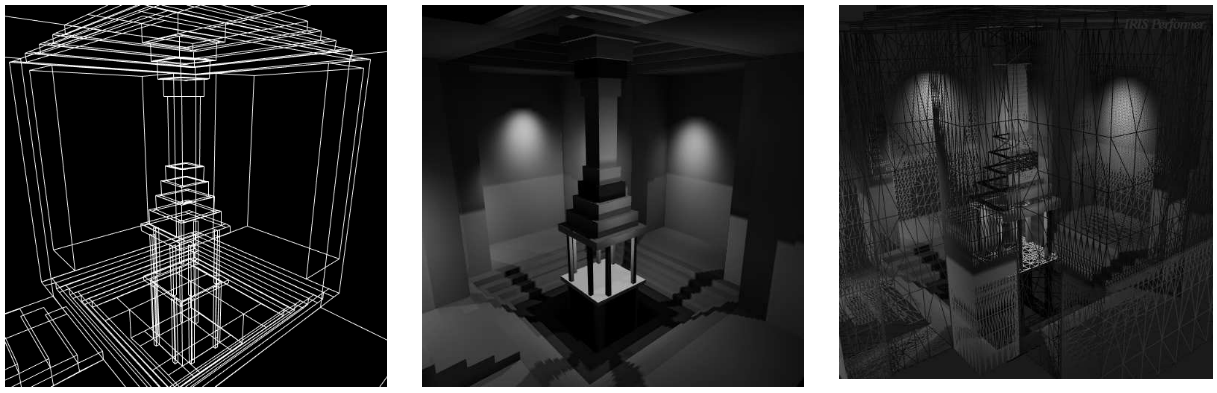


Figure 7: *Grotto model: original model before radiosity solution, RAT, and adaptive subdivision.*

Specular Illumination : To create a scene of even greater realism, we could render the specular component of the illumination and combine this with the diffuse component represented by the radiosity textures. This requires some view dependent processing at run time.

6 ACKNOWLEDGMENTS

This work was partially supported by **ARPA Contract no. DABT63-93-C-0048** and **CNPq - Brasília/BRAZIL**. The authors would also like to thank Eddie Saxe, Lightscape Technologies, and members of the Walkthrough Project at UNC.

References

[1] BASTOS, Rui; AUGUSTO DE SOUSA; and NUNES FERREIRA. **Reconstruction of Illumination Functions using Bicubic Hermite Interpolation**. In 4^{th} *Eurographics Workshop on Rendering* (*Proceedings*), Paris, France, July 1993, pp. 317-326. (Available at http://www.cs.unc.edu/~ bastos/wonr93.ps.gz).

[2] BAUM, Daniel; MANN, Stephen; SMITH, Kevin; and WINGET, James. **Making Radiosity Usable: Automatic Preprocessing and Meshing Techniques for the Generation of Accurate Radiosity Solutions**. *Computer Graphics*, v. 25, no. 4, July 1991, pp. 51-60.

[3] COHEN, Michael and WALLACE, John. **Radiosity and Realistic Image Synthesis**. *Academic Press,* 1993.

[4] GLASSNER, Andrew. **Principles of Digital Image Synthesis**. *Morgan Kaufmann*, 1995.

[5] GOSLIN, Michael. **Illumination As Texture Maps For Faster Rendering**, *Technical Report TR95-042*, Department of Computer Science, University of North Carolina at Chapel Hill, 1995.

[6] HECKBERT, Paul. **Adaptive Radiosity Textures for Bidirectional Ray Tracing**. In *Computer Graphics*, SIGGRAPH *Proceedings*, pp. 145-154, August 1990.

[7] LISCHINSKI, D.; TAMPIERI, F.; GREENBERG, D. **Discontinuity Meshing for Accurate Radiosity**. *IEEE Computer Graphics and Applications*, 12:6, November 1992, 25-39.

[8] MITCHELL, Don; and NETRAVALI, Arun. **Reconstruction Filters in Computer Graphics**. In *Computer Graphics*, SIGGRAPH *Proceedings*, pp. 221-228, August 1988.

[9] MÖLLER, Tomas. **Radiosity Techniques for Virtual Reality - Faster Reconstruction and Support for Levels of Detail**, In *WSCG'96*, 12-16th February 1996, Plzen, Czech Republic.

[10] MOLNAR, Steven; EYLES, John; POULTON, John. **PixelFlow: High-Speed Rendering Using Image Composition**, In *Computer Graphics*, SIGGRAPH *Proceedings*, vol. 26, no. 2, July 1992, pp. 231-240.

[11] MYSZKOWSKI, Karol; and KUNII, Tosiyasu L. **Texture Mapping as an Alternative for Meshing During Walkthrough Animation**. In 5^{th} *Eurographics Workshop on Rendering* (Proceedings), Darmstadt, Germany, June 13-15, 1994.

[12] PARK, Stephen K., and SCHOWENGERDT, Robert A. **Image Reconstruction by Parametric Cubic Convolution**. *Computer Vision Graphics, and Image Processing*, vol. 23, no. 3 , September 1983, pp. 258-272.

[13] RATLIFF, F. **Contours and Contrast**. Scientific American, 226(6):91-101, June 1972.

[14] RUSHMEIER, H.; WARD, G.; PIATKO, C.; SANDERS, P.; and RUST, B. **Comparing Real and Synthetic Images: Some Ideas About Metrics**. In 6^{th} *Eurographics Workshop on Rendering* (*Proceedings*), pp. 213-222, June 1995.

[15] SALESIN, David; LISCHINSKI, Dani and DeROSE, Tony. **Reconstructing Illumination Functions with Selected Discontinuities**. In 3^{rd} *Eurographics Workshop on Rendering* (Proceedings), Bristol, 1992.

[16] SGI White Paper.

[17] SILLION, François X. and PUECH, Claude. **Radiosity and Global Illumination**, *Morgan Kaufmann*, 1994.