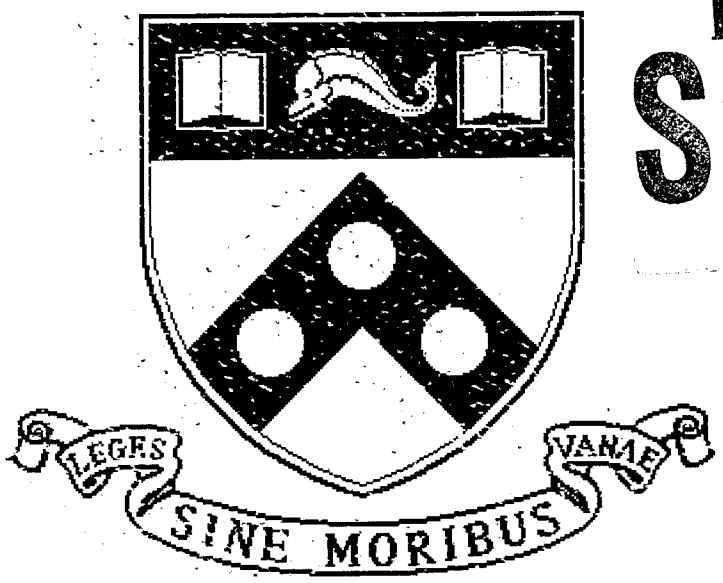
Texture Resampling While Raytracing: Approximating the Convolution Region Using Caching

MS-CIS-94-03
HUMAN MODELING & SIMULATION LAB 60

Jeffry S. Nimeroff
Norman I. Badler
Dimitri Metaxas



University of Pennsylvania
School of Engineering and Applied Science
Computer and Information Science Department
Philadelphia, PA 19104-6389

February 1994

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
		technical report

4. TITLE AND SUBTITLE
Texture Resampling While Raytracing: Approximating the Concolution Region Using Caching

5. FUNDING NUMBERS
DAAL03-89-C-0031

6. AUTHOR(S)
Jeffry S. Nimeroff, Norman I. Badler, Dimitri Metaxas

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Department of Computer and Information Sciences
University of Pennsylvania
200 S. 33rd Street
Philadelphia, PA 19104-6389

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U. S. Army Research Office
P. O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
ARO 26779.34-MA-AI

11. SUPPLEMENTARY NOTES
The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

We present a cache-based approach to handling the difficult problem of performing visually acceptable texture resampling/filtering while ray-tracing. While many good methods have been proposed to handle the error introduced by the ray-tracing algorithm when sampling in screen space, handling this error in texture space has been less adequately addressed. Our solution is to introduce the Convolution Mask Approximation Module (CMAM). The CMAM locally approximates the convolution region in the texture space as a set of overlapping texture triangles by using a texture sample caching system and ray tagging. Since the caching is hidden within the CMAM, the ray-tracing algorithm itself is unchanged while achieving an adequate level of texture filtering (area sampling as opposed to point sampling/interpolation in texture space). The CMAM is easily adapted to incorporate prefiltering methods such as MIP mapping and summed-area tables as well as direct convolution methods such as elliptical weighted average filtering.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Ray-tracing; Texture Resampling; Antialiasing; Filtering; Convolution; Algorithms	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Texture Resampling While Ray-Tracing: Approximating the Convolution Region Using Caching*

Jeffry S. Nimeroff, Norman I. Badler, and Dimitri Metaxas
Center for Human Modeling and Simulation
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, Pa 19104-6389

February 9, 1994

Abstract

We present a cache-based approach to handling the difficult problem of performing visually acceptable texture resampling/filtering while ray-tracing. While many good methods have been proposed to handle the error introduced by the ray-tracing algorithm when sampling in screen space, handling this error in texture space has been less adequately addressed. Our solution is to introduce the Convolution Mask Approximation Module (CMAM). The CMAM locally approximates the convolution region in texture space as a set of overlapping texture triangles by using a texture sample caching system and ray tagging. Since the caching mechanism is hidden within the CMAM, the ray-tracing algorithm itself is unchanged while achieving an adequate level of texture filtering (area sampling as opposed to point sampling/interpolation in texture space). The CMAM is easily adapted to incorporate prefiltering methods such as MIP mapping and summed-area tables as well as direct convolution methods such as elliptical weighted average filtering.

Keywords: Ray-Tracing; Texture Resampling; Antialiasing; Filtering; Convolution; Algorithms

*This research is partially supported by ARO Grant DAAL03-89-C-0031 including participation by the U.S. Army Research Laboratory (Aberdeen), Natick Laboratory, and NASA Ames Research Center; U.S. Air Force DEPTH contract through Hughes Missile Systems F33615-91-C-0001; DMSO through the University of Iowa; NSF Grant IRI91-17110, CISE Grant CDA88-22719, and Instrumentation and Laboratory Improvement Program Grant #USE-9152503.

1 Introduction

Texture resampling is a well researched area of computer graphics. Adequate methods exist for handling the introduction of aliasing errors while "shrinkwrapping" a digital image onto the surface of a computer-generated object[6, 4, 13, 14, 18, 33, 11, 20, 17, 15]. Also, there exists a codification of the steps needed to perform the process in an "ideal" (alias-free) manner[22, 24]. These techniques rely on area information from the rendering algorithm in order to perform their function and the assumption is made that this information is readily available. The renderer is expected to provide the pixel boundary in screen space and the compound mapping (τ) from texture to screen space (surface parameterization combined with the view and screen projection). With this information the filtering module can calculate the pixel's extent in texture space (via inverse projection) and perform filtering within this extent. Exactly how each filtering method uses this information is case dependent, but all the methods referenced require some notion of the pixel's inverse projection into texture space.

Ray-tracing research has given us the ability to accomodate many optical phenomena easily within a computer-modelled environment[32, 8, 7, 19, 28]. A problem exists, however, in that the ray-tracing renderer neither explicitly computes the pixel boundary in screen space, nor explicitly constructs the compound mapping (the screen projection is replaced by the geometric ray intersection process). This appears to preclude using the well-established texture filtering algorithms without changing them severely or compromising the simplicity of the ray-tracer.

We introduce and develop the Convolution Mask Approximation Module (CMAM) and show how this simple caching module and ray tagging system can be used to create an approximation to the texture space filter extent (convolution region) that allows the ray-tracer to perform texture filtering without affecting its inherent simplicity. It will also be shown that this process adds only $O(1)$ volume (time $\times$ space) complexity to the cost associated with any of the adapted texture filtering methods. We conclude with examples of how to use the CMAM in conjunction with MIP maps[33], summed-area tables[11], and the EWA filtering technique[20].

2 Applying Textures while Ray-Tracing

Applying textures while ray-tracing can be thought of as a multi-criteria sampling process. Since only a finite number of rays can be cast for any image, aliasing in screen space is always a concern - for example, undersampling the screen space image function can allow objects to "fall between the cracks." In addition, the presence of texture mapping means that the sample locations will

be used to acquire texture space information. Due to the projective native of the ray tracing "camera" geometry and the nonlinearity of many explicit surface parameterizations, samples that are well placed in screen space are not necessarily well placed in object space or in texture space (Figure 1), therefore, essential texture information maybe missed. Neither area sampling nor increasing the sampling rate solves the problem.

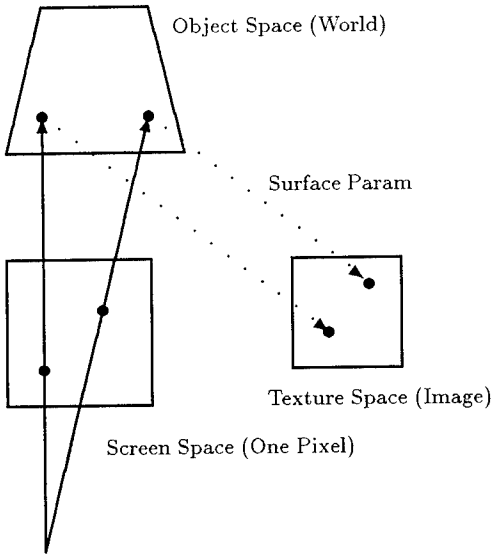


Figure 1: Dense Screen Space Distribution: Sparse in Object and Texture Space

Area sampling requires performing exact integration over the spatial extent of the projected pixel. No information is lost as with point sampling, but performing the integration is expensive, if not intractable. Two early attempts were made at performing this type of area sampling in a ray-style renderer: cone-tracing[1] and beam-tracing[23]. Cone-tracing treats each ray as a cone emanating from the chosen point and having a divergence angle. Beam-tracing projects a bundle of rays as a polygonal beam into the scene along the direction that the infinitesimally thin ray would travel. Both methods require many limiting assumptions to be made about the environment in order to remain tractable. These limitations drastically effect the usefulness of the technique.

Modifying the sampling rate (number of rays processed), using statistically significant samples in an attempt to adequately sample in screen or texture space, can minimize the affects of aliasing energy but does not remove the energy[12, 25, 7, 28]. Since a good portion of the ray-tracer's running time can be

attributed to intersection calculations[32] adding extra samples can significantly affect a ray-tracer's performance. It turns out that many computer graphics textures require an infinite sampling rate to be sampled adequately.

Rather than modifying the sampling rate or allowing the limiting assumptions of an area sampling ray-style renderer, our solution follows from texture filtering research by using a modified point/area sampling method based on a local set of known texture locations. This requires keeping a window of information on the texture sampling pattern for each textured object.

3 Constructing the Convolution Region in Texture Space

The pixel's texture space extent (convolution region) is constructed by projecting the pixel's boundary points into texture space (Figure 2). A ray-tracing algorithm could do this by firing rays through the corners of the pixel and then mapping the intersection points via the surface parameterization. This

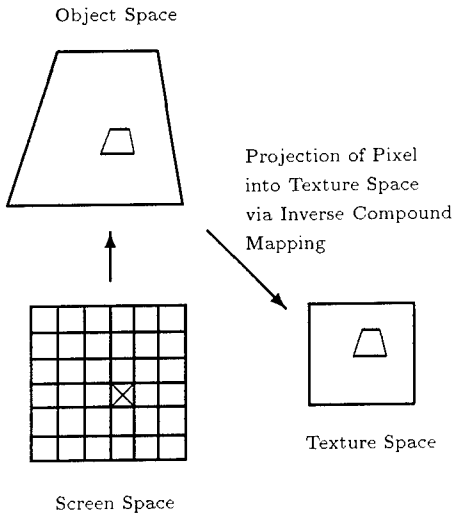


Figure 2: Convolution Region: Pixel Projection

solution suffers from limitations[1, 23] due to the coupling of the rays and also precludes using any of the simple, stochastic approaches to screen-space antialiasing[12, 25, 7, 28]. The rays can still be treated independently if one

is willing to redefine the manner in which the convolution region is defined (constructed).

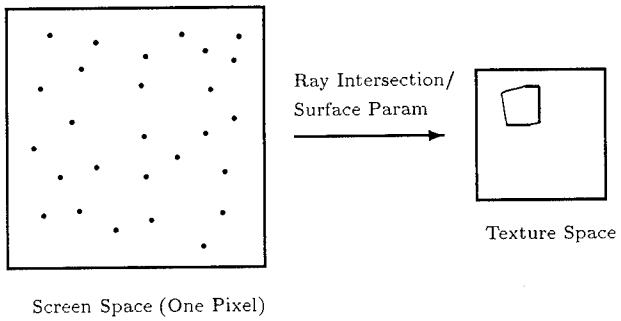


Figure 3: Convolution Region: Convex Hull of Texture Point Samples

If we modify the definition of the convolution region to include that area of texture space inside the convex hull of a set of texture space point samples (Figure 3), an incremental approach to texture filtering while ray-tracing evolves.

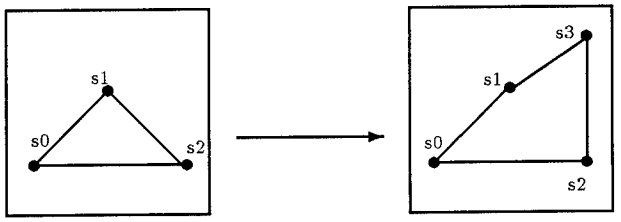


Figure 4: Incremental Construction of the Convolution Region

Our incremental convex hull filtering method approximates the filtering region in texture space as a set of (possibly) overlapping texture triangles (Figure 4). The current sample location along with the two previous sample locations (provided the rays emanate from the same pixel) are used to give a local ap-

proximation to the texture area that needs to be filtered for this sample. This overestimates the convolution region by allowing for the inclusion of a texture sample more than once, but guarantees that only those samples inside the convex hull are included in the filtering operation. A non-incremental approach is not as useful because every point sample of texture space cannot easily/accurately be associated with a filtered texture intensity (filter values are only associated with areas incrementally bound by the point samples).

3.1 Caching and the Convolution Mask Approximation Module

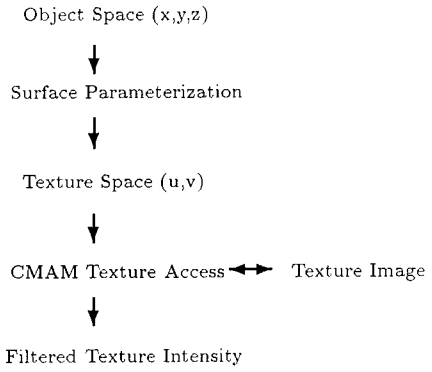


Figure 5: Placement of the Convolution Mask Approximation Module

The convolution mask approximation module (CMAM) is a data structure and a set of routines that resides between the texture image (or data structure) and the surface parameterization (Figure 5) and implements the methodology described above.

```
typedef float Color[COLOR_SPACE];

typedef struct cmam {
    /* Flag for turning CMAM filtering on */
    int filter;

    /* Ray IDs of cached samples */
    int last_id, sec_last_id;

    /* Recursion levels of cached samples */
    int last_level, sec_last_level;

    /* Sample (u,v) of cached samples */
    float last_u, sec_last_u;
```

```
    float last_v, sec_last_v;

    /* Texture data structure */
    Color **map;

    /* Bounds of texture array */
    int rows, cols;
} Cmam, *Cmamptr;
```

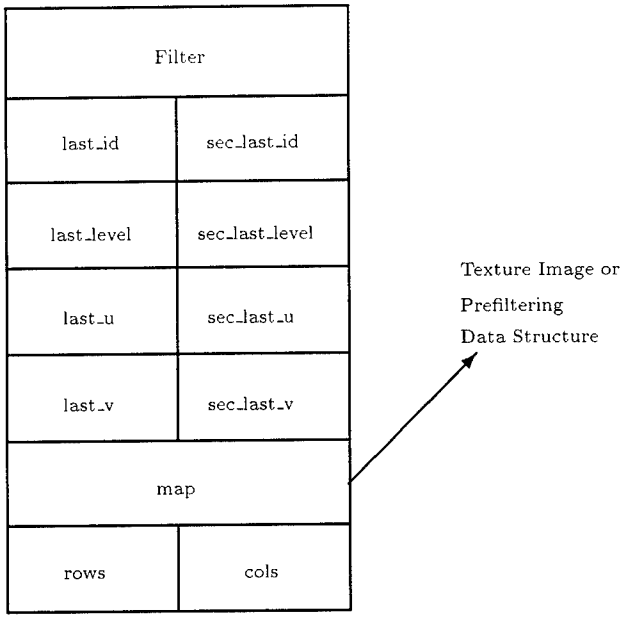


Figure 6: CMAM and Associated Texture Data Structure

Instead of having the ray-tracer accessing the texture image directly and performing filtering itself, the CMAM takes the texture location, and returns the filtered texture value to the ray-tracer.

The convolution region is approximated as above with ray tagging being used to facilitate the process of finding related rays. Rays which are fired through pixels on the same scanline in screen space, and might ultimately be used to bound a region in texture space, are given the same ID and a starting recursion level of zero. IDs, levels, and sample locations are passed to the CMAM which compares them with the most recent CMAM accesses. ID matching facilitates the incremental building of the convolution area along the scanline. Level matching allows the accumulation of texture area information treating proxi-

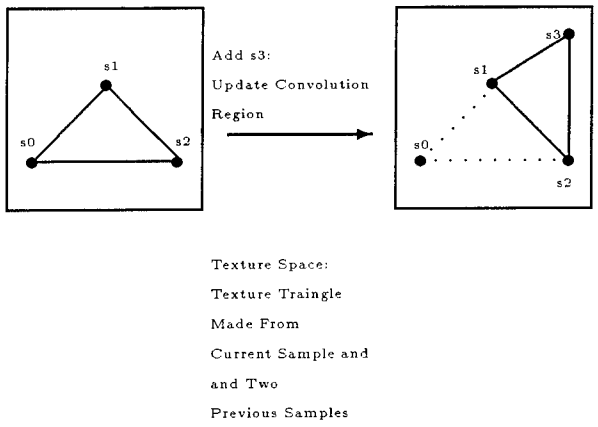


Figure 7: CMAM Approximation of Overlapping Texture Triangles

mate groups of reflected/refracted rays as an approximation to the travelling wavefront[1, 23]. If the IDs and levels match then the region bounded by the samples in texture space is part of an approximation to the true convolution region.

The CMAM then can use known filtering techniques such as MIP mapping, summed-area tables or an EWA scheme to filter the area and return a texture intensity (without any interdependence of the rays). No changes to the basic ray-tracing implementation are necessary. The ray-tracer acts as if it is point sampling in texture space.

3.2 Using the CMAM with Existing Techniques

How the CMAM uses its local approximation of the convolution region is specific to the type of texture filtering that is going to be performed. MIP mapping, summed-area tables, and EWA filtering require their convolution regions to be described in different ways. When the texture sample is sent to the CMAM the ID is checked against the cached values. This current ID can match the IDs of both the cached samples, the ID of the most recently sample only, or any of the IDs in the cache. The problem then reduces to generating an area based on the number of cache hits counted and the texture space sampling pattern.

3.2.1 MIP Map Approximation

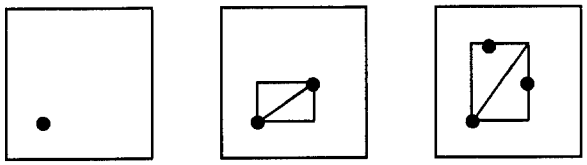


Figure 8: MIP Map Approximation Using the CMAM

Williams' MIP mapping[33] performs texture filtering by accessing a pre-filtered texture pyramid and performing trilinear interpolation. The pyramid is accessed with a d parameter which chooses the two levels which best approximate the filtered region. Intra-level access is via the texture coordinate (u, v) and uses bilinear interpolation to reconstruct the texture value within the levels. The only hard part seems to be constructing d using the CMAM.

We us the following MIP map level approximation algorithm.

1. If the sample does not match the most recent sample, use the highest level of the pyramid (the average intensity of the texture image) to trade blurring for aliasing (texture image with high frequencies), or use the lowest level of the pyramid (point sample) to trade aliasing for blurring (texture image with high frequencies).

   $$d = 0 \quad \text{or} \quad d = MAXLEVEL$$

   Point sampling trades aliasing for blurring, while using the fully averaged texture image trades blurring for aliasing. Since the human visual sustem is more tolerent of blurring than aliasing, we chose to use the averaged texture image.

2. If the sample matches the most recent sample use the length of the line between the two samples in the following calculation:

   $$d = \lg(\text{length of line between samples})$$

3. If the sample matches the both cached samples, fit an axis-aligned bounding box around the three samples. Use the length of the diagonal of this bounding box in the same calculation as above.

   $$d = \lg(\text{length of diagonal of bounding box})$$

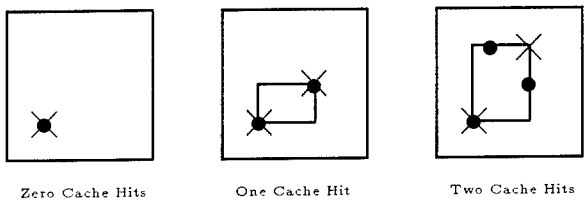


Figure 9: Summed-Area Table Approximation Using the CMAM

3.2.2 Summed-Area Table Approximation

The summed-area table[11] is not restricted to filtering square regions in texture space. It is accessed using the corners of the axis-aligned rectangular region that is to be convolved with a box or Gaussian filter. The summed-area table access is simple for the CMAM.

1. If the sample does not match the most recent sample, use the texture space coordinate to either point sample or average the region from the origin of the summed-area table to the texture coordinate (same criteria as mentioned above).

2. If the sample matches the most recent sample use the two texture space coordinates to create a rectangular region to be filtered using the summed-area table.

3. If the sample matches the both the cached samples, fit an axis-aligned bounding box around the three samples. Use the upper-right and lower-left coordinates to access the summed-area table.

3.2.3 EWA Approximation

The EWA algorithm[20] is a direct convolution algorithm (not prefiltering) and requires the semi-major and semi-minor axes of a texture space elliptical filtering region as well as a warped filter function. Since the ray-tracer lacks the inverse compound mapping τ^{-1}, computing the warped filter function can only be approximated. The axes for the texture space ellipse and the filter kernel access can be done as follows:

1. No matches uses point sampling as with MIP maps or summed-area tables.

2. If the sample matches the most recent sample use the two texture space coordinates to create a line which represents the radius of a texture space

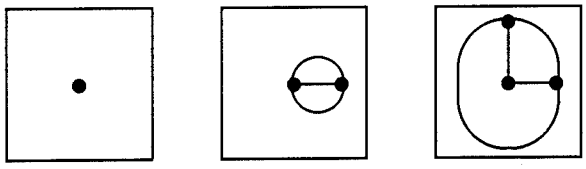


Figure 10: EWA Approximation Using the CMAM

circle (degenerate ellipse). For each texture sample contained within the circle, use its distance from the center to access a circular symmetric filter kernal and weight the sample accordingly.

3. If the sample matches both the cached samples, use the vector from the current sample to one of the cached samples that is the longest as the semi-major axis of he texture space ellipse and the vector between the current point and the other cached sample as the semi-minor axis of the ellipse. The circularly symmetric filter kernel can then be accessed by the distance from the center of the ellipse to the sample, normalized by the distance from the center of the ellipse to the boundary of the ellipse that runs through the sample point and its value used to weight the samples within the texture space ellipse.

4 Convex Hull Weighting

Since the CMAM caches only the last two texture samples it is possible that the area bounding the three most recent samples overlaps a region of the texture image that already has been included in the final filtered intensity for the current pixel. We have begun to investgate a method for incorporating the convex hull of the union of all the approximated convolutions into the CMAM. When a region is to be filtered, the region is differenced with the convex hull to return that part of the convolution region which has not yet been incorporated into the filtered texture intensity (the DC value of the texture image is returned if the current region is completely enclosed within the convex hull of the approximate convolution region). This type of weighting will include each element of the texture image (within the true convolution region) at most once in the final filtered intensity, thereby yielding a better approximation to the true texture intensity.

5 Conclusion

The convolution mask approximation module provides a caching system which is useful for approximating the texture space extent of the screen space filter kernel. It uses ray coherence to locally approximate the filter extent as a set of triangular regions that tile areas of the true convolution region. Filtering then becomes possible using these areas to access either prefiltered data structures or a direct convolution filtering algorithm[33, 11, 20]. With the use of the CMAM we are able to perform texture filtering without changing the simplicity of the basic ray-tracing implementation or limiting any of its photo-realistic features.

References

[1] John Amanatides. Ray Tracing with Cones. *Computer Graphics*, 18(3):129–135, July 1984.

[2] James F. Blinn. Return of the Jaggy. *IEEE Computer Graphics and Applications*, 9(2):82–89, March 1989.

[3] James F. Blinn. What We Need Around Here Is More Aliasing. *IEEE Computer Graphics and Applications*, 9(1):75–79, January 1989.

[4] James F. Blinn and Martin E. Newell. Texture and Reflection in Computer-Generated Images. *Communications of the ACM*, 19(10):542–547, October 1976.

[5] Ronald Bracewell. *The Fourier Transform and its Applications.* McGraw-Hill, 1986.

[6] Edwin A. Catmull. Computer Display of Curved Surfaces. *Proc. Conf. on Computer Graphics, Pattern Recognition, Data Structure*, pages 11–17, May 1975.

[7] Robert L. Cook. Stochastic Sampling in Computer Graphics. *ACM Transactions on Graphics*, 5(1):51–72, Januray 1986.

[8] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed Ray Tracing. *Computer Graphics*, 18(3):139–147, July 1984.

[9] Franklin C. Crow. The Aliasing Problem in Computer-Generated Shaded Images. *Communications of the ACM*, 20(11):799–805, November 1977.

[10] Franklin C. Crow. A Comparison of Antialiasing Techniques. *IEEE Computer Graphics and Applications*, 1(1):40–47, January 1981.

[11] Franklin C. Crow. Summed-Area Tables. *Computer Graphics*, 18(3):207–212, July 1984.

[12] Mark A. Z. Dippé and Erling Henry Wold. Antialiasing Through Stochastic Sampling. *Computer Graphics*, 19(3):69–78, July 1985.

[13] William Dungen Jr., Anthony Stenger, and George Sutty. Texture Tile Considerations for Raster Graphics. *Computer Graphics*, 12(3):130–134, August 1978.

[14] Eliot A. Feibush, Marc Levoy, and Robert L. Cook. Synthetic Texturing Using Digital Filters. *Computer Graphics*, 14(3):294–301, July 1980.

[15] Eugene Fiume and Robert Lansdale. Fast Space-Variant Texture Filtering Techniques. *SPIE 1991 Proceedings*, 1991. Preliminary version of the paper submitted for publication.

[16] James D. Foley, Andries van Dam, Steven Feiner, and John Hughes. *Computer Graphics - Principles and Practice.* Addison-Wesley, 1990.

[17] Alain Fournier and Eugene Fiume. Constant-Time Filtering with Space-Variant Kernals. *Computer Graphics*, 22(4):229–237, August 1988.

[18] Michel Ganget, Didier Perny, and Phillippe Coueignoux. Perspective Mapping of Planar Textures. *Computer Graphics*, 16(1), May 1982.

[19] Andrew Glassner. *An Introduction to Ray Tracing.* Academic Press, New York, 1989.

[20] Ned Greene and Paul Heckbert. Creating Raster Omnimax Images from Multiple Perspective Views Using the Elliptical Weighted Average Filter. *IEEE Computer Graphics and Applications*, pages 21–27, June 1986.

[21] Paul Heckbert. Survey of Texture Mapping. *IEEE Computer Graphics and Applications*, 6(11):56–67, November 1986.

[22] Paul S. Heckbert. Fundamentals of Texture Mapping and Image Warping. Master's thesis, Division of Computer Science, University of California at Berkeley, June 1989.

[23] Paul S. Heckbert and Pat Hanrahan. Beam Tracing Polygonal Objects. *Computer Graphics*, 18(3):119–128, July 1984.

[24] Robert C. Lansdale. Texture Mapping and Resampling for Computer Graphics. Master's thesis, Department of Electrical Engineering, University of Toronto, January 1991.

[25] Mark E. Lee, Richard A. Redner, and Samuel P. Uselton. Statistically Optimized Sampling for Distributed Ray Tracing. *Computer Graphics*, 19(3):61–67, July 1985.

[26] Dimitri Metaxas and Evangelos Milios. Reconstruction of a Color Image from Nonuniformly Distributed Sparse and Noisy Data. *CVGIP: Graphics Models and Image Processing*, pages 103–111, March 1992.

[27] Don P. Mitchell. Generating Antialiased Images at Low Sampling Densities. *Computer Graphics*, 21(4):65–69, July 1987.

[28] Don P. Mitchell. The Antialiasing Problem in Ray Tracing. *Advanced Topics in Ray Tracing, SIGGRAPH 1990 Course Notes*, 1990.

[29] Don P. Mitchell and Arun N. Netravali. Reconstruction Filters in Computer Graphics. *Computer Graphics*, 22(4):221–228, August 1988.

[30] Donald E. Pearson. *Transmission and Display of Pictorial Information.* Halsted Press, John Wiley & Sons, New York, 1975.

[31] Ken Shoemake. Personal communication, January 1992.

[32] Turner Whitted. An Improved Illumination Model for Shaded Displays. *Communications of the ACM*, 23(6), June 1980.

[33] Lance Williams. Pyramidal Parametrics. *Computer Graphics*, 17(3):1–11, July 1983.

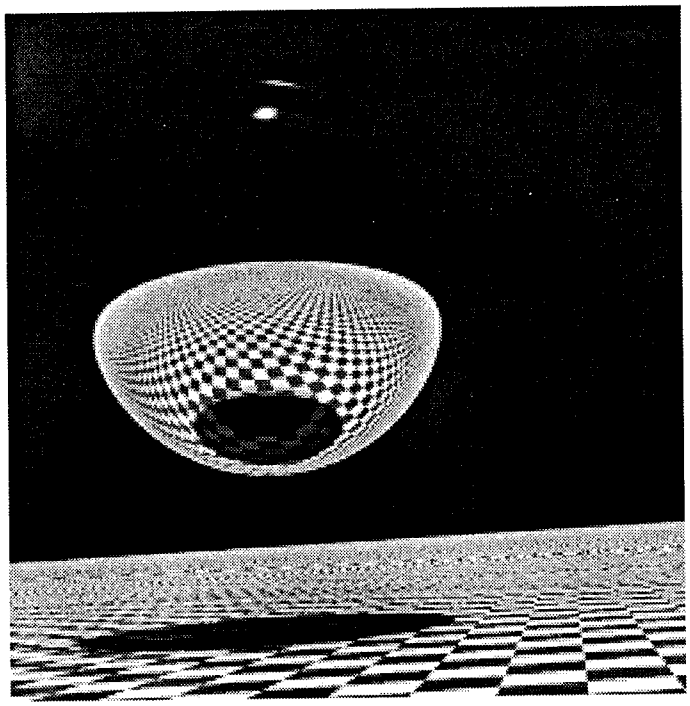


Figure 11: Test Scene 1 with CMAM on

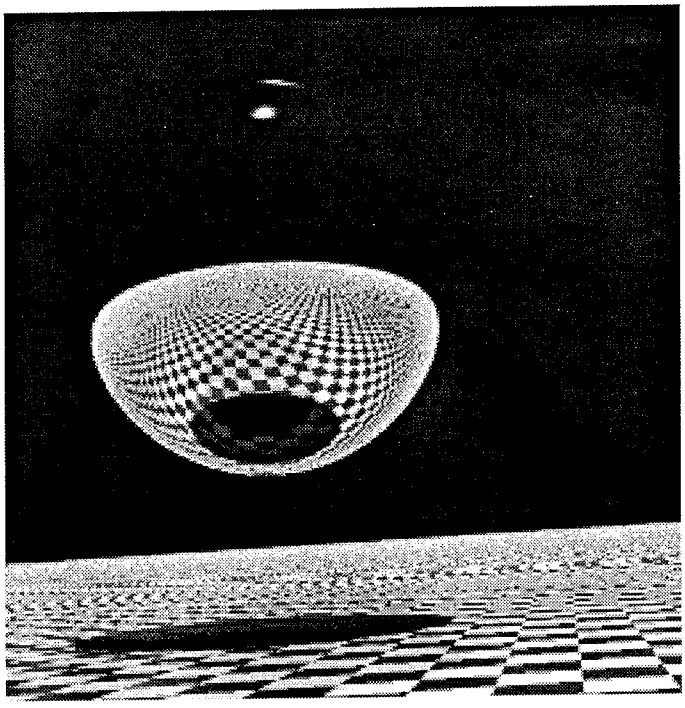


Figure 12: Test Scene 1 with CMAM off

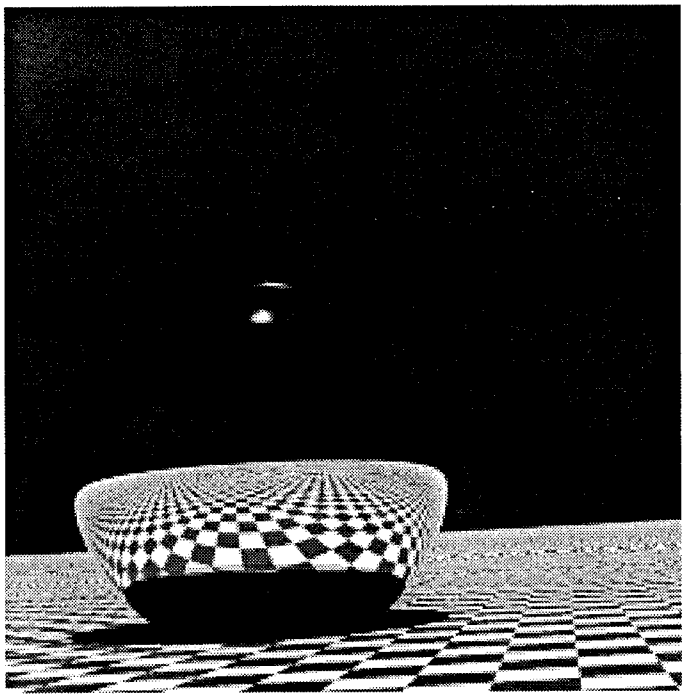


Figure 13: Test Scene 2 with CMAM on

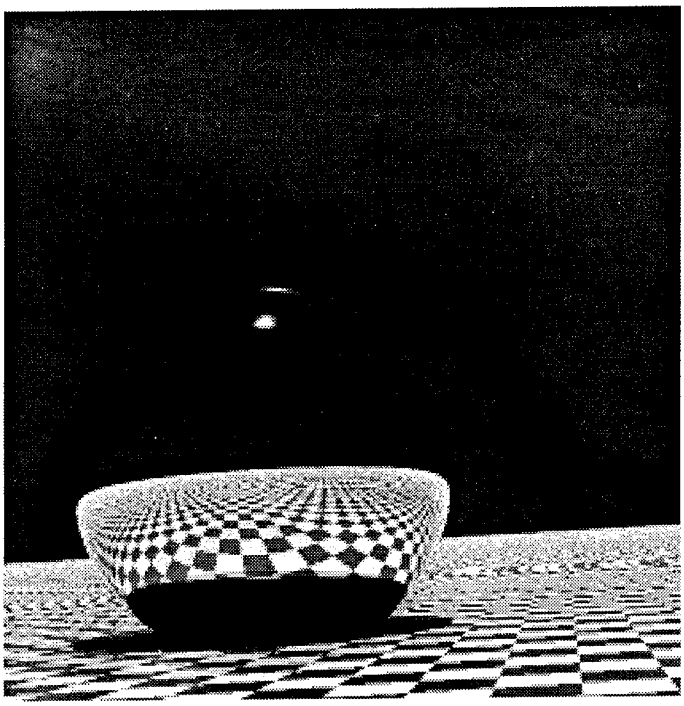


Figure 14: Test Scene 2 with CMAM off

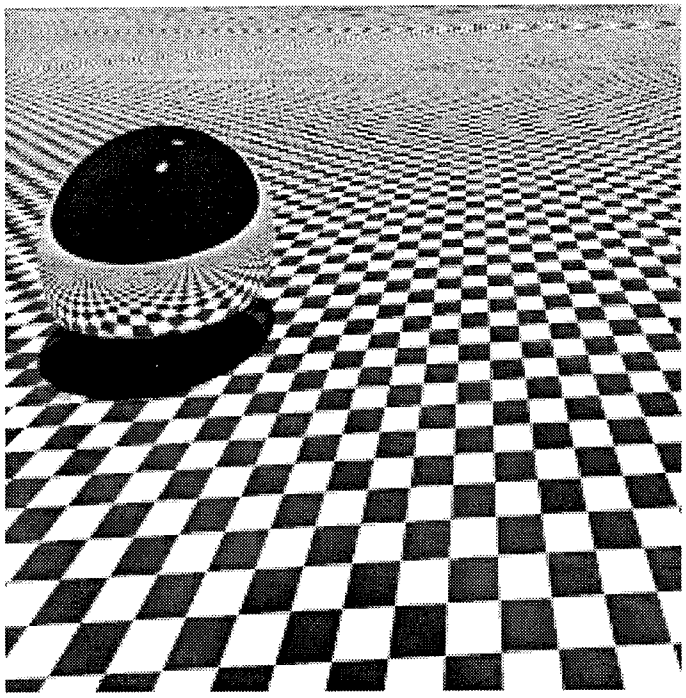


Figure 15: Test Scene 3 with CMAM on

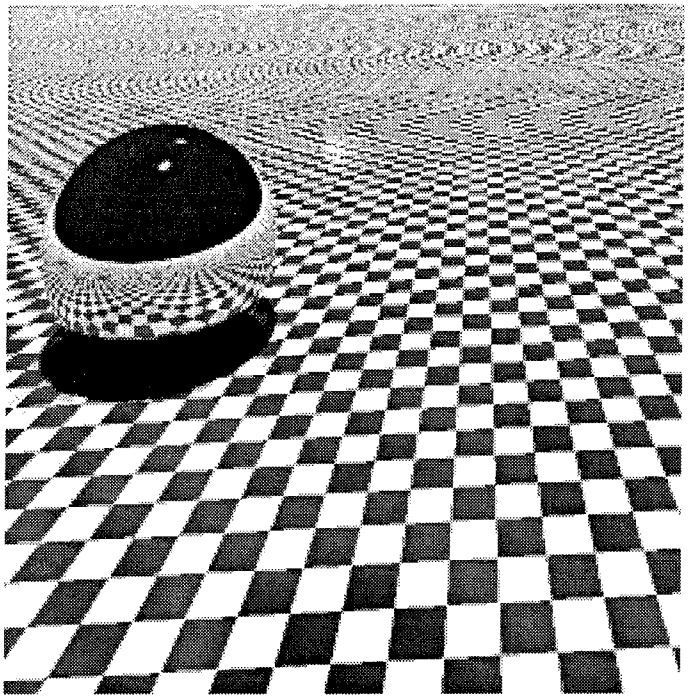


Figure 16: Test Scene 3 with CMAM off

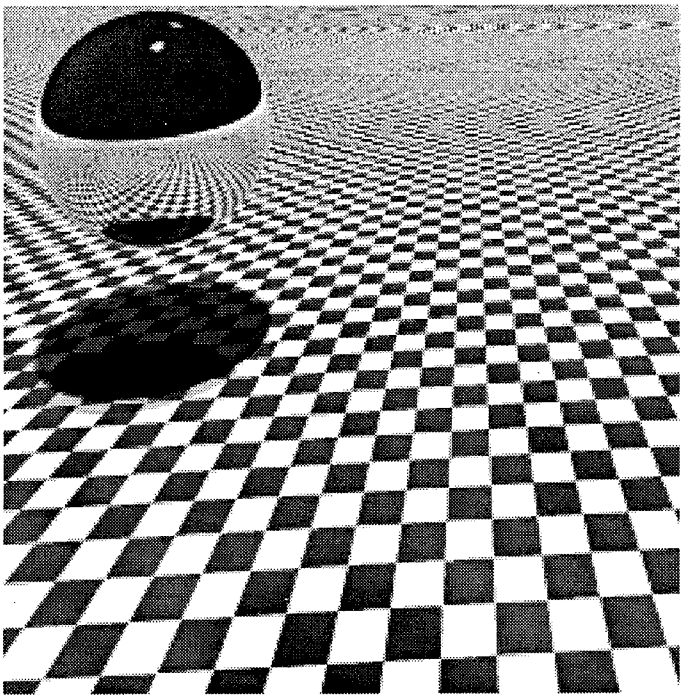


Figure 17: Test Scene 4 with CMAM on

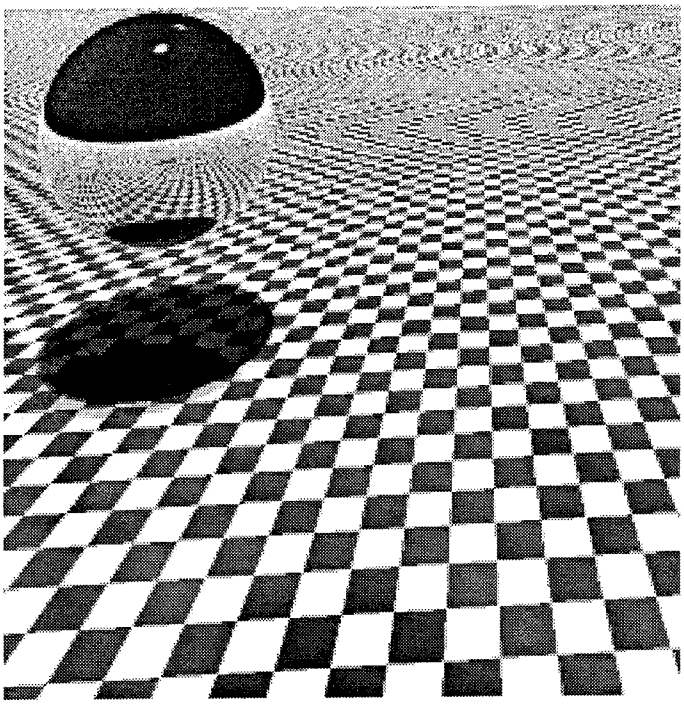


Figure 18: Test Scene 4 with CMAM off