Interner Bericht

Evaluation of a BRDF using BSP Trees

Dr. rer. nat. Philip Jacob
272/95

Fachbereich Informatik

Universität Kaiserslautern · Postfach 3049 · D-67653 Kaiserslautern

Evaluation of a BRDF using BSP Trees

Dr. rer. nat. Philip Jacob
272/95
AG Computergraphics
Postfach 3049
67653 Kaiserslautern
University of Kaiserslautern

67653 Kaiserslautern (Germany)
e-mail:jacob@informatik.uni-kl.de

Abstract

In this paper an analytic hidden surface removal algorithm is presented which uses a combination of 2D and 3D BSP trees without involving point sampling or scan conversion. Errors like aliasing which result from sampling do not occur while using this technique. An application of this algorithm is outlined which computes the energy locally reflected from a surface having an arbitrary BRDF. A simplification for diffuse reflectors is described, which has been implemented to compute analytic form factors from diffuse light sources to differential receivers as they are needed for shading and radiosity algorithms.

1 Introduction

One of the main problems in computer graphics is the visibility in complex scenes. Related problems like the evaluation of illumination models and the computation of form factors can often be reduced to this problem. The most popular visibility algorithms used today are ray tracing and z-buffering techniques. They generate a 2D image from a set of discrete samples taken from the environment. BSP trees (*Binary Space Partitioning Trees*) can be used for both as a space subdivision technique to speed up rendering. Like quad- and octrees, BSP trees can also directly store the geometry of an object. These two applications of BSP trees are combined in an analytic visibility algorithm presented in paragraph 3.1.

2 The BRDF

A BRDFBRDF describes surface attributes like reflectivity which depends on the angles of incident and reflected energy. This function is also influenced by the wavelength and the location for which it is computed. As most surfaces are neither completely diffuse reflectors nor prefect mirrors, this function depends at least on four angular parameters (Figure 1).

2.1 Definition

The BRDF (*Bidirectional Reflectance Distribution Function*) is defined as the radiance $dL_r(\Theta_r, \varphi_r)$ reflected from a surface which is illuminated by a certain irradiance $dE_i(\Theta_i, \varphi_i)$.

$$\delta(\Theta_i, \varphi_i, \Theta_r, \varphi_r) = \frac{dL_r(\Theta_r, \varphi_r)}{dE_i(\Theta_i, \varphi_i)} = \frac{dL_r(\Theta_r, \varphi_r)}{dL_i(\Theta_i, \varphi_i)\cos\Theta_i d\Omega_i} \quad \left[\frac{1}{sr}\right] \qquad \text{Eqn. 1}$$

The BRDF of a diffuse reflector is a constant function $\delta(\Theta_i, \varphi_i, \Theta_r, \varphi_r) = \rho/\pi$. For a perfect mirror it is a dirac function which is non zero only for the angles defined by a perfect reflection. Considering a single direction (Θ_r, φ_r) of reflected radiance, the BRDF defines a weight for each direction of incident energy. Figure 1 shows a BRDF for a fixed viewing angle having diffuse and

specular components.

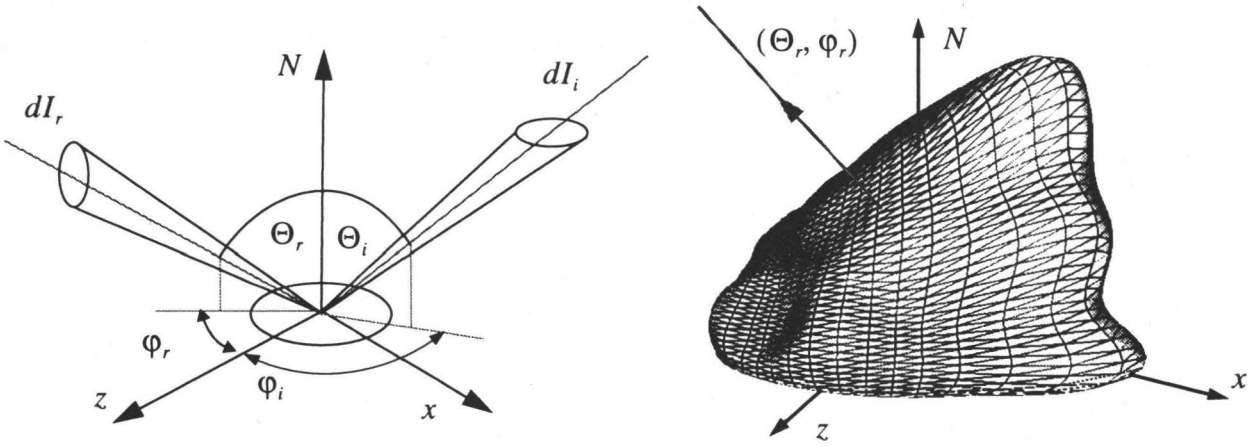

Figure 1: BRDF of a Surface

2.2 Representation and Storage of a BRDF

During the last 20 years numerous empirical and theoretical models for local reflection of light have been developed. Most illumination models store the BRDF implicitly for certain surface models like diffuse, specular or e.g. metal surfaces. The alternative to this storage model is the explicit representation by basis functions over the hemisphere like spherical harmonics. Implicit models can be evaluated very fast in comparison to an explicit representation, but most of them are not able to handle a complex reflection behavior like an anisotropic reflection.

2.3 Illumination with Point and Area Light Sources

For an object which is illuminated by point light sources the BRDF has to be evaluated for all directions in which the light sources appear from the viewpoint of the shaded surface. Several algorithms have been developed which compute the BRDF using a cascaded model of spherical harmonics [WEST92].

To compute the radiance reflected by a surface which is illuminated by an area light source or a light reflecting patch an integration over the surface of the radiating medium has to be performed. If this step is done by sampling, two problems arise. First, the value of the BRDF has to be computed, then a ray has to be sent in the direction of the light source to determine whether it is visible or not. The next paragraph describes a visibility algorithm and a method of computing illumination or form factors in complex scenes.

3 The BSP-Algorithms

BSP trees have been used in computer graphics since 1980 to solve visibility. These algorithms first transform a scene into its BSP-representation which is a binary structure. Using special traversal techniques, visibility is solved for any point of view. BSP trees have also been used to store shadows from a point light source by dividing a scene into shadowed objects and objects which are illuminated [CROW77].

3.1 An Analytic Hidden Surface Algorithm

The visibility algorithm presented in this paper works in two stages. In the first step all objects of the scene are stored in a BSP tree similar to the algorithm developed by Fuchs [FUCH80]. The clipping of the objects against the partitioning planes leads to a fragmentation of the objects which is shown in figure 2.

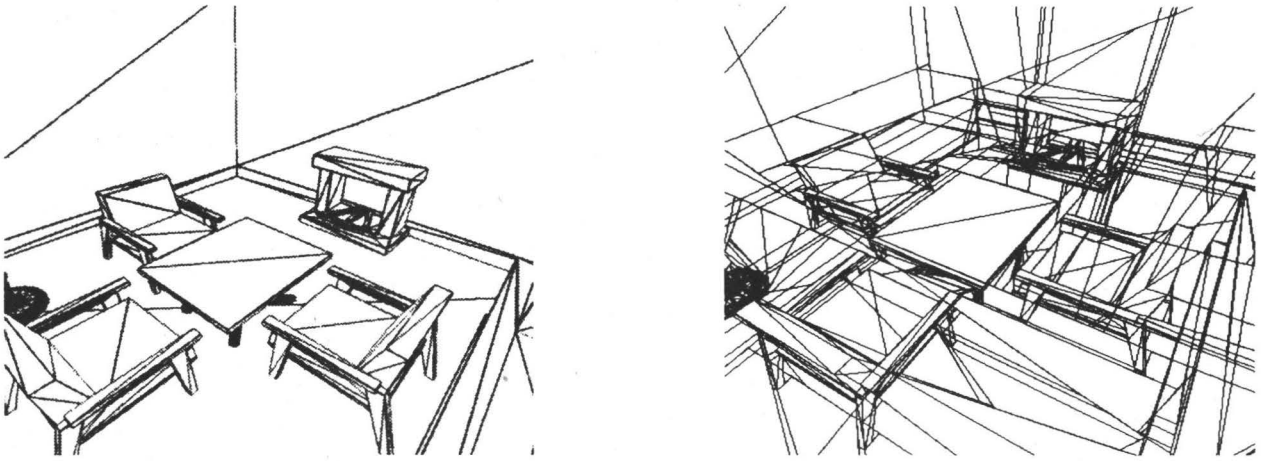

Figure 2: Original Scene and its BSP-Representation

This global BSP tree is used in the second, the rendering pass, to generate a 2D contour image from a specific point of view. Traditional BSP methods use a *back-to-front* traversal where more distant objects can be obscured by those which lie closer to the viewer. The algorithm presented here uses a *front-to-back* traversal and stores the contours of objects which already have been traversed in a 2D BSP tree lying in the image plane. This contour image stores all visible parts of the scene with object precision. During the traversal each object is first projected on the image plane an then clipped against the existing 2D BSP tree. Figure 3 shows a triangle, its projection on the image plane and the corresponding representation as a BSP tree.

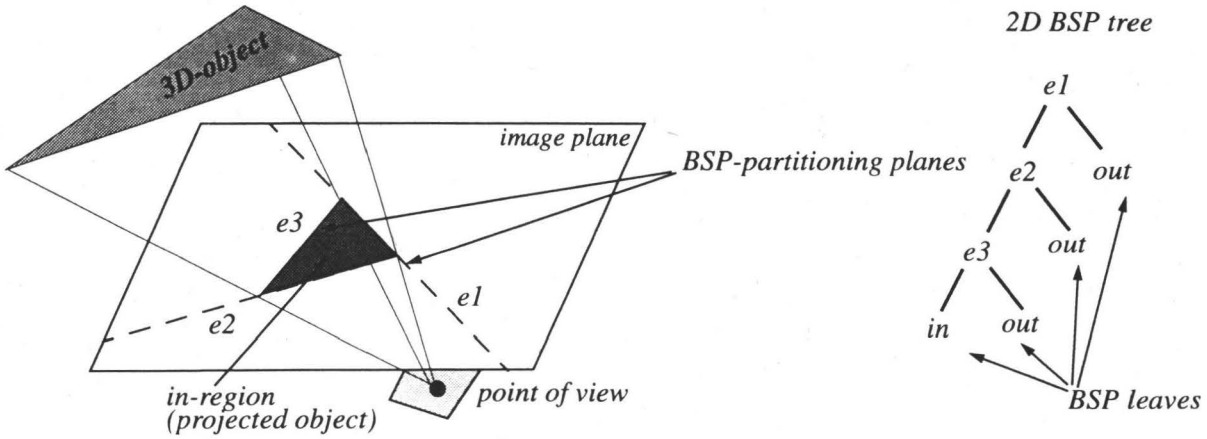

Figure 3: 2D BSP Image of a Single Triangle

Thus the *in-leaves* of the BSP tree represent a region on the image plane consisting of all visible objects which have been traversed. Objects or parts of objects which lie outside this *in-region*

(Figure 4) can't be obscured by other objects, so they are used to extend the 2D BSP tree.

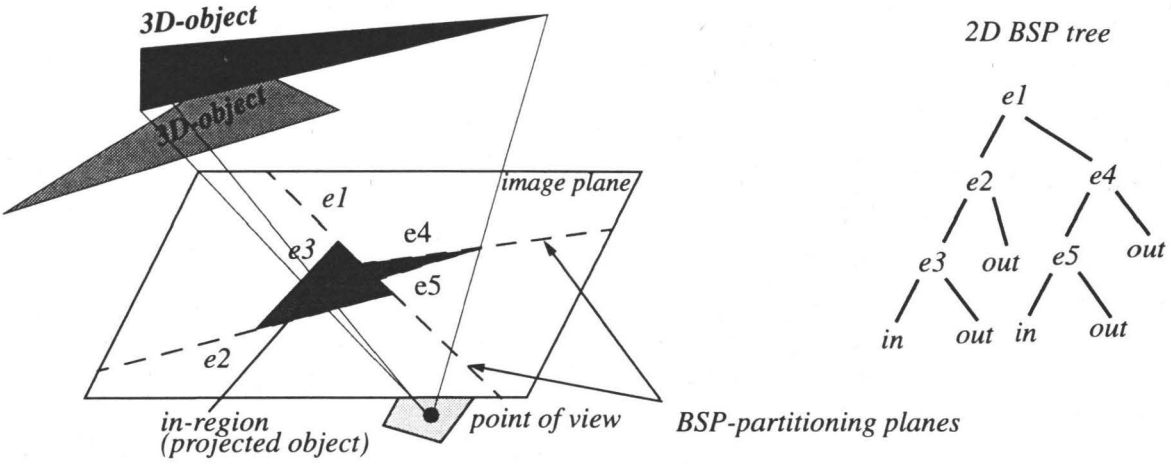

Figure 4: Extension of the 2D BSP Tree

The difference between traditional BSP-techniques and the algorithm presented here is that no part of the scene is overwritten. The main advantage of this method is that no oversampling or resampling techniques have to be applied to reduce aliasing. Although clipping is done in the image plane instead on a 3D BSP volume, most of the computation time is consumed by clipping a projected object against the existing 2D BSP tree. If the BSP-partitioning planes near the root node are chosen independent of the scene, so that they build a quadtree-like subdivision, the clipping time can be reduced immensely (Figure 5). For extreme viewing angles (170° – 180°) the subdivision can be refined towards the center of the image plane, so that approximately the same number of objects fall into each BSP pixel.

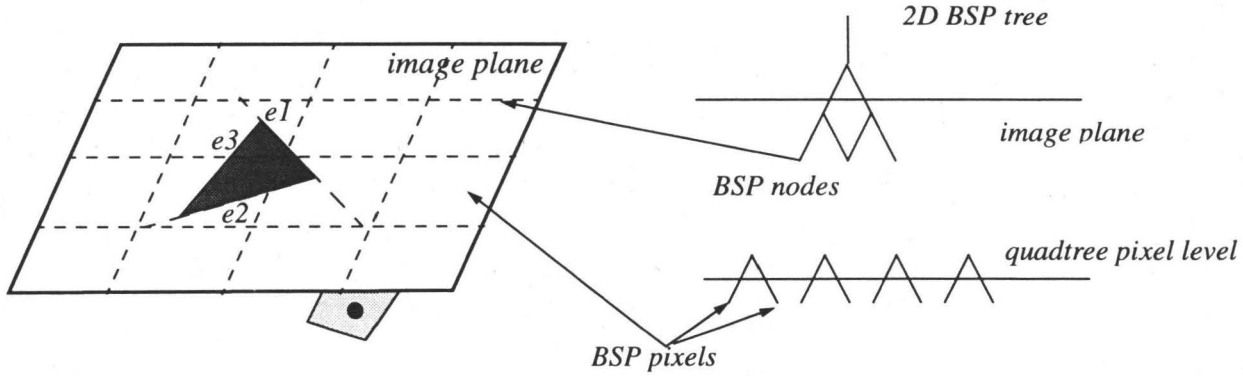

Figure 5: Quadtree-subdivision of the Image Plane

Besides this application in rendering the algorithm can be used to evaluate a BRDF or compute precise form factors to finite patches.

3.2 Evaluation of a BRDF using BSP Trees

To compute the energy locally reflected from a differential surface element an integration over

all light emitting patches of the scene has to be done. This integral can be reduced to an integral over all visible patches by solving the visibility problem in a preprocessing step. During a second step, only the visible parts are considered for the evaluation of the BRDF similar to the hemicube algorithm [COHE85]. The BRDF is then computed using sampling techniques or analytical expressions.

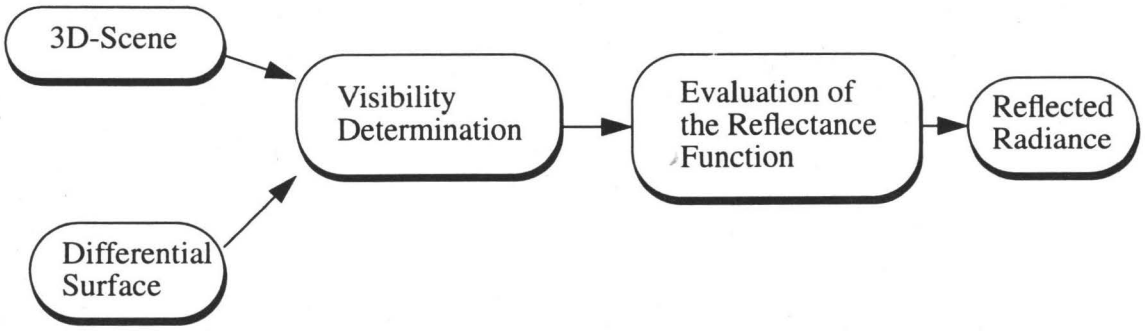

Figure 6: Radiance Computation

3.2.1 Surfaces with General BRDFs

To compute the energy reflected from a surface having a non-trivial BRDF, an integral over all visible objects needs to be evaluated, with the incident energy $(dE_i(\Theta_i, \varphi_i))$ weighted by the BRDF.

$$L_r(\Theta_r, \varphi_r) = \int_{hemisphere} \delta(\Theta_i, \varphi_i, \Theta_r, \varphi_r)\, dE_i(\Theta_i, \varphi_i) \qquad \text{Eqn. 2}$$

The algorithm described in paragraph 3.1 is used to solve the visibility problem. Therefore the scene is rendered from the view of the surface element. The hidden surface algorithm generates a contour image with a viewing angle of nearly 180 degrees. This image defines the borders of integration for the BRDF. To get linear integration borders and to speed up computation, the BRDF can be projected on the image plane for which the contour image has been generated. Figure 7 shows a contour image mapped on the projected BRDF.

The projection of the BRDF implies a multiplication with the ratio of the modified solid angles.

A constant (diffuse) BRDF is thus transformed in a $\cos^4\Theta$ function (Figure 7).

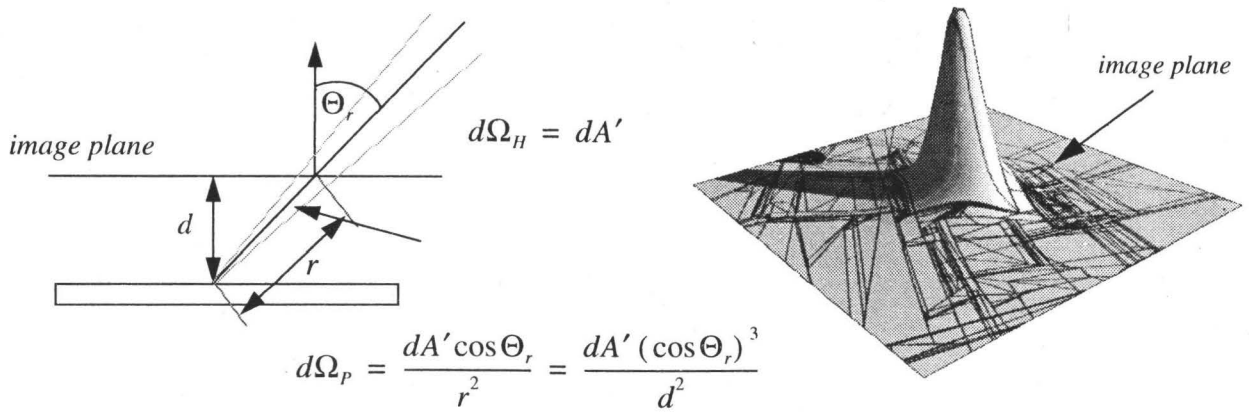

Figure 7: Contour Image mapped on the BRDF

The following images have been generated via ray tracing and the algorithm described above. When a ray hits a surface, a 2D BSP tree of the environment is created, which is used to compute the reflected radiance for the appropriate BRDF. Specular surfaces and anisotrophic reflections can be simulated without affecting computation time.

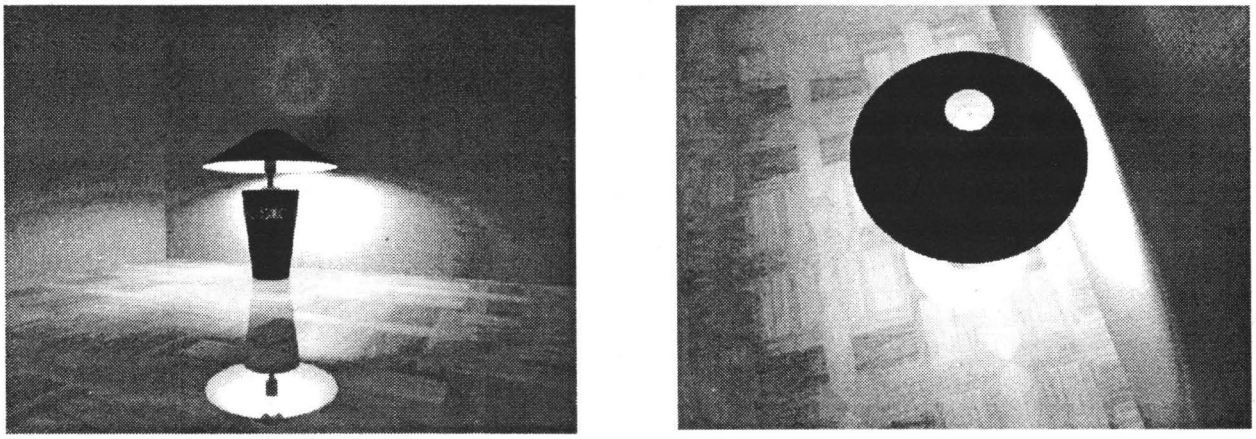

Figure 8: Raytracing with a General BRDF

Figure 8 shows two snapshots from a sequence* for which diffuse radiosities have been precalculated using the progressive refinement technique [COHE88]. The reflected radiance has been computed with the method described above.

3.2.2 Diffuse Surfaces

For a diffuse surface the reflected radiance can be computed directly from the contour image using the form factor from a patch to a differential receiver [BAUM89]. While the hemicube method and ray tracing are approximations, the algorithm suggested here is an analytic solution for complex scenes and does not depend on parameters like the resolution or sampling rate. The

*. Mpeg-video clips and color images are available via www at http://davinci/projects/jacob_1.html

difference between Baum's analytical form factors and the approach described in this paper is that the visibility is also solved analytically and not by using a z-buffer. Even the number of analytical form factors computed is reduced heavily because the computation is not done at pixel level but on objects which have been clipped against the BSP tree.

To compare form factors computed by this algorithm with those computed by the hemicube, an illumination model called hemicube ray tracing [MEYE90] was used, which needs form factors to simulate shadows and penumbra from area light sources.

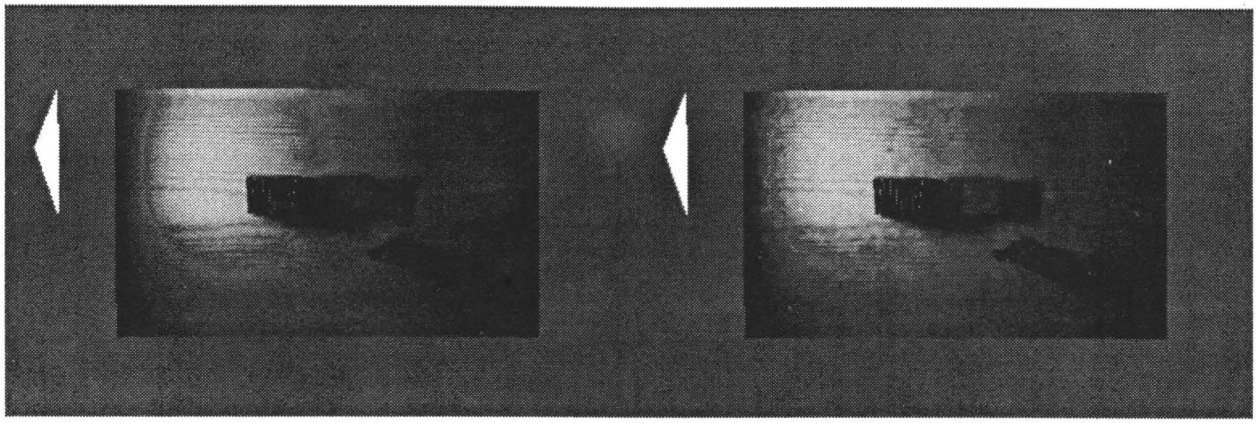

Figure 9: Leitz-Scene

The image on the right side of figure 9 was rendered by hemicube ray tracing with a 64x64 hemicube. The radiance field (Figure 10) shows aliasing errors in the hemicube image, especially in regions where the projection of the light source on the hemicube is very small. The radiances computed with the algorithm described in this paragraph are error free, which is proved by the smooth distribution shown on the left side of figure 10 and the corresponding image in figure 9.

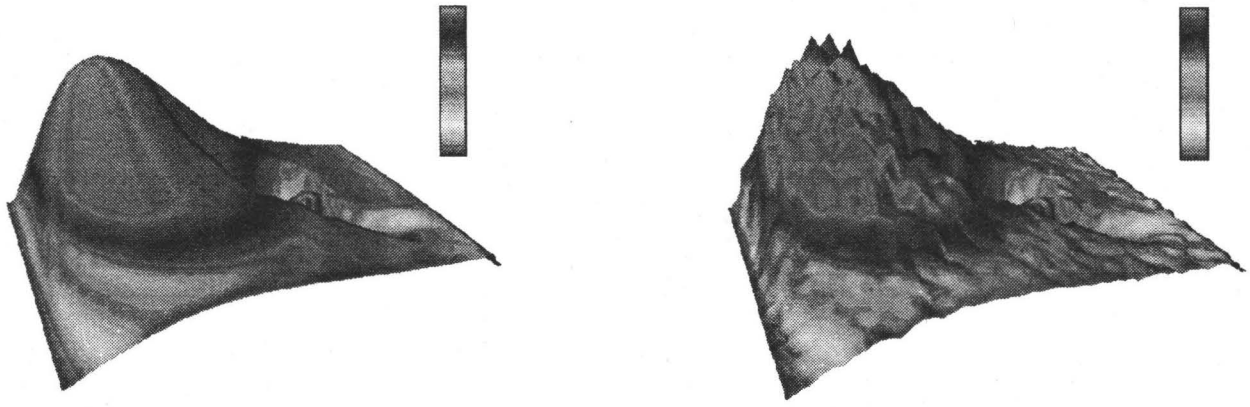

Figure 10: The Radiance Fields of Figure 9

The implementation of this algorithm has been tested with several polyhedral scenes with a size ranging from 1.000 to 50k objects. With a resolution of less than 128x128 pixels the hemicube needs less computation time, using a resolution of more than 512x512 pixels the analytic method gets faster than the hemicube algorithm which still has a maximal relative error of 1% in the form factors. For this comparison the original hemicube method was used. Improvements like the one suggested by Baum [BAUM89] can help to reduce these errors but they will fail when objects are

completely lost during scanconversion.

4 Conclusion

The hidden surface algorithm suggested in this paper is an alternative to ray tracing or z-buffer techniques. It produces a 2D image for a scene composed of planar objects, but unlike ray tracing and z-buffer algorithms neither oversampling nor resampling techniques have to be applied to reduce aliasing. The algorithm can be used as a tool for computing visibility in illumination models. Especially, if more samples, caused by an increasing variance of the direction of reflected rays, are needed to keep aliasing at a minimum, this new method can save a great amount of computation time. Furthermore, it is independent of the surface attributes and can be used for specular and diffuse surfaces without modification.

5 References

[BAUM89]

R. Baum, H. Rushmeier, J. Winget: *Improving Radiosity Solutions Through the Use of Analytically Determined Form-Factors*, ACM Computer Graphics Vol. 23. Number 3, pp. 325-334

[COHE85]

M. Cohen, D. Greenberg: *The Hemi-Cube, A Radiosity Solution For Complex Environments,* ACM Computer Graphics Vol. 19, Number 3, pp. 31-40, 1985

[COHE88]

M. Cohen, S.Chen, J.Wallace, D.Greenberg: *A Progressive Refinement Approach To Fast Radiosity Image Generation*, ACM Computer Graphics Vol. 33 Number 3 pp. 75-84

[CROW77]

F. Crow: *Shadow Algorithms for Computer Graphics*, ACM Computer Graphics Vol. 11 Number 3, pp. 242-248

[FUCH80]

Fuchs, Kedem, Naylor: *On Visible Surface Generation by A Priori Tree Structures*, ACM Computer Graphics Vol. 14, Number 3, pp. 124-133,1980

[MEYE90]

Urs Meyer, Hemi-Cube Ray Tracing: *A Method for Generating Soft Shadows*, Proc. Eurographics 1990, pp. 365-376

[WEST92]

S. Westin, J. Arvo, K. Torrance: *Predicting BRDFs from Complex Surfaces*, ACM Computer Graphics (SIGGRAPH'92) Vol. 26 pp. 255-264