Workshop on Local Illumination Proceedings Graphics Interface '92

Vancouver, British Columbia, Canada

These proceedings gather the texts accompanying the presentations given at the *Workshop on Local Illumination* held within Graphics Interface '92 in Vancouver, May 11th, 1992.

We thank the presenters and the participants, and we hope that these proceedings will stimulate further thoughts and research on that important field of computer graphics.

Alain Fournier and Pierre Poulin
Department of Computer Science
University of British Columbia
Vancouver, BC
V6T 1Z2
Canada
Email: {fournier|poulin}@cs.ubc.ca

Hierarchical 3D-Texure

Mikio Shinya

NTT Human Interface Laboratories
3-9-11 Midori-cho
Musashino-shi, Tokyo, 180 Japan
email: shinya@nttarm.ntt.jp

Abstract

A hierarchical 3D-texture model is proposed that ensures smooth transition from geometry to **mapping and allows efficient anti-aliasing of complex scenes. Several** problems with 3D-texture **are pointed out, and possible solutions are described. Early experiments** indicate the potential of **the model.**

1 Introduction

Local illumination models are computational models that endeavour to replicate the reflectance distribution function of surfaces and their micro-structures. Since the term "micro" is relative to scaling and distance, it is very important to establish a smooth transition from global structures, i.e., geometry, to local illumination models.

Kajiya introduced the idea of a *hierarchy of detail* [KAJ85], where three basic levels are introduced. These are the lighting (reflection) level, the mapping level (texture, bump, displacement, etc.), and the model (geometry) level. The mapping level approximates the geometry level in the form of images. The reflection level represents stochastic properties of the mapping necessary for shading calculation.

Obviously, there are excessive gaps among the levels to allow a smooth transition, and many studies have attempted to improve the models by introducing three-dimensional features, like occlusion and self-shadowing [MAX,CABD,POULIN]. Among these, 3D-texture [KAJ89,PERLIN] explicitly represents discretized three-dimensional shapes, and is expected to bridge the gap between geometry and mapping.

Another important aspect of local illumination models is their need for a *pre-filtering* feature to ensure accurate anti-aliasing. For example, thousands of tiny mirror-like facets are "filtered" in existing reflection models to form a distribution function of facet normals; this prevents aliasing. 3D-texture could provide a new type of anti-aliasing method for complex objects: the pre-filtering of 3D objects.

This paper proposes a hierarchical 3D-texture model that achieves a smooth transition from geometry to mapping, as well as effective anti-aliasing through its pre-filtering feature. The hierarchical 3D-texture has a similar structure to the hierarchical two-dimensional textures such as pyramidal textures, and an appropriate level is chosen according to the resolution required by rendering.

For this purpose, we need a 3D-texture model which:

- can be generated from geometric objects.
- can provide the information needed for rendering.
- can be summed up, or filtered, to build lower resolution 3D-textures.

The following sections emphasize several unsolved problems with 3D-texture, and possible approaches are described together with an early experimental result.

2 3D-texture

3D-texture was developed by Kajiya et al. [KAJ89] and by Perlin et al. [PERLIN]. Perlin's model (hyper-texture) emphasizes the modeling tasks using set operations and modulation functions, while Kajiya's model (texel) is more rendering-oriented. These two papers were the starting point for this paper.

In Kajiya's model, each voxel contains three kinds of information: a scalar density ρ for light attenuation, a coordinate system $B = (\vec{n}, \vec{t}, \vec{b})$ for the shading model, and shading model Ψ. 3D-textures are rendered by ray tracing, sampling points on rays, and summing up the light intensity. This model works well for fur objects, as demonstrated in the Teddy Bear images. However, as pointed out in his paper, there remain several open problems for further extension:

- how to construct 3D-texture from geometric models,
- how to sum up voxels.

In addition, the following problems should be also solved to realize hierarchical 3D-textures:

- how to reduce the memory requirement,
- what kinds of information must be kept and how to use them in rendering.

3 Approaches

3.1 Reducing memory requirement

To achieve a smooth transition from geometry, we need a series of 3D-textures ranging from fine to coarse resolutions. Unfortunately, storing a three-dimensional array of voxels at high resolution, say 512×512×512, requires several gigabytes of memory, and is too expensive for current computers. Thus, solving the storage problem is the most important issue from the practical viewpoint.

The key idea is that *most voxels are not visible to the eye*, and that *we don't need to process invisible voxels*. There are two kinds of invisible voxels:

- voxels containing no objects,
- voxels completely occluded from all directions.

Sparse visible voxels can be efficiently stored using a list structure (Figure 1).

The number of visible voxels is, at most, the sum of the projected areas of all facets, and is expected to be $O(n^2)$ for n-pixel resolution in most cases. The worst case would require $O(n^3)$ storage with this strategy, which would occur, for example, when tiny particles, like fog or smoke, are uniformly distributed in space. In this case, however, we can apply other data compression techniques, such as run-length coding, that can take advantage of the uniformity.

3.2 Rendering

In general, the rendering process involves two tasks: visibility (occlusion) check and light intensity calculation (shading and shadowing). Thus, each voxel should store information on occlusion and shading. With this information, rendering can be achieved through the ray tracing scheme.

Information to store for rendering To allow occlusion and shading calculation, voxels should contain the following information:

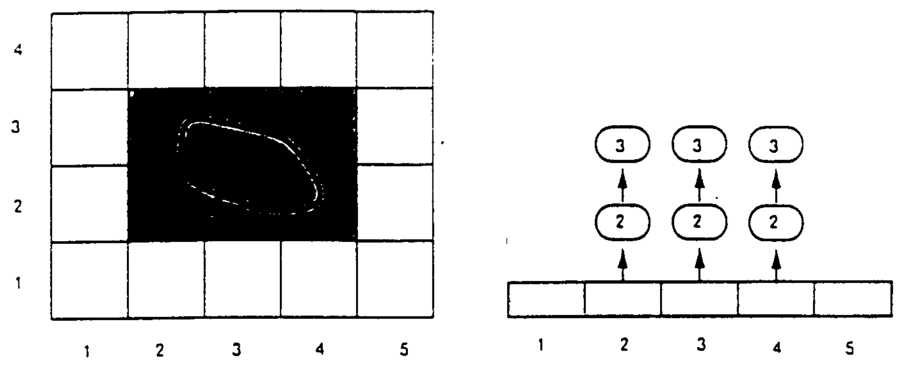


Figure 1: List-structure.

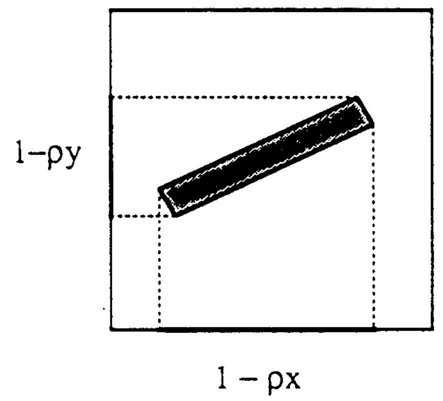


Figure 2: Density of a voxel.

Density To describe occlusion by a single voxel, the density, or the rate of occlusion, should be stored in each voxel. Because occlusion by surfaces is significantly anisotropic, the density strongly depends on direction (Figure 2). At least, ρ should be a vector representation indicating the occlusion ratios in the x-, y-, and z-directions.

Correlation To describe occlusion by multiple voxels, correlation among voxels should be stored. In Kajiya's model, density values are simply multiplied along the ray path, analogous with light absorption in volumetric media. This is a good approximation when correlation among the voxels is low; for example, when tiny surfaces are randomly placed in the voxels (Figure 3-a). However, voxels may have strong correlation, as shown in Figure 3-a and -b.

Shader To describe shading information, shading functions (Ψ) and their frames (B) should be stored. Since multiple surfaces may lie within a single voxel, shading information should be stored as a list.

Ray tracing 3D-texture Figure 4-a shows conventional ray tracing, where infinitely thin rays are traced. In this case, occlusion (or absorption) and light intensity should be evaluated on each line

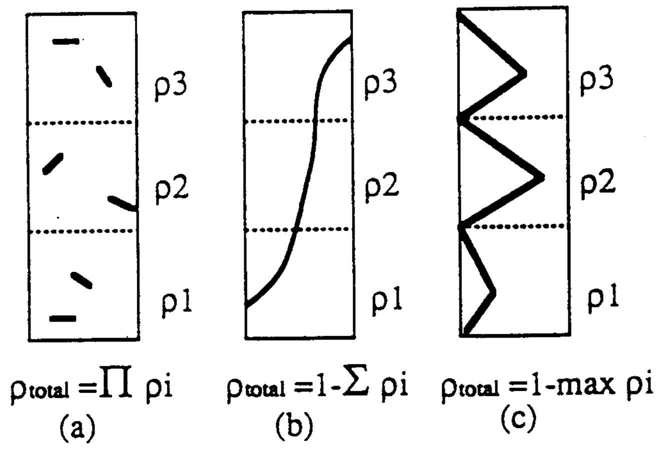

p3
p2
p1
$\rho_{total} = \Pi\ \rho i$
(a)
$\rho_{total} = 1 - \Sigma\ \rho i$
(b)
$\rho_{total} = 1 - \max\ \rho i$
(c)

Figure 3: Correlation.

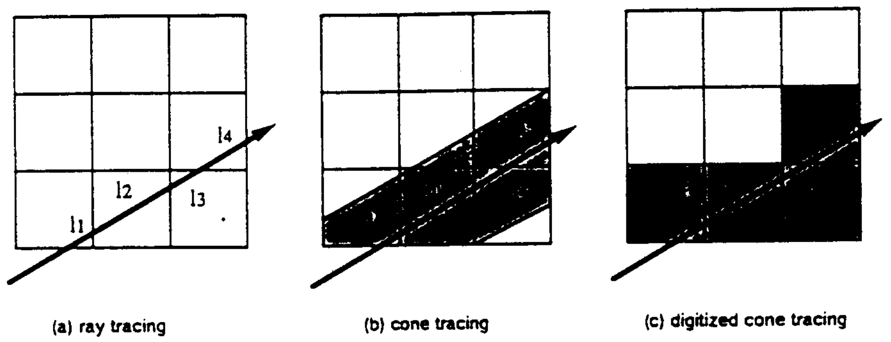


Figure 4: Rendering 3D-texture.

segment (l_i), and then the total intensity is summed up along the whole path. For this evaluation, occlusion should be calculated in the ray direction. Since the ray direction could be any direction, the voxels should store occlusion information for all directions, which would be too expensive.

The ideal ray tracing could be achieved by cone tracing [AMA] (Figure 4-b). In this case, occlusion and light intensity are evaluated on each *volume* segment. However, this algorithm involves expensive volume integral calculations.

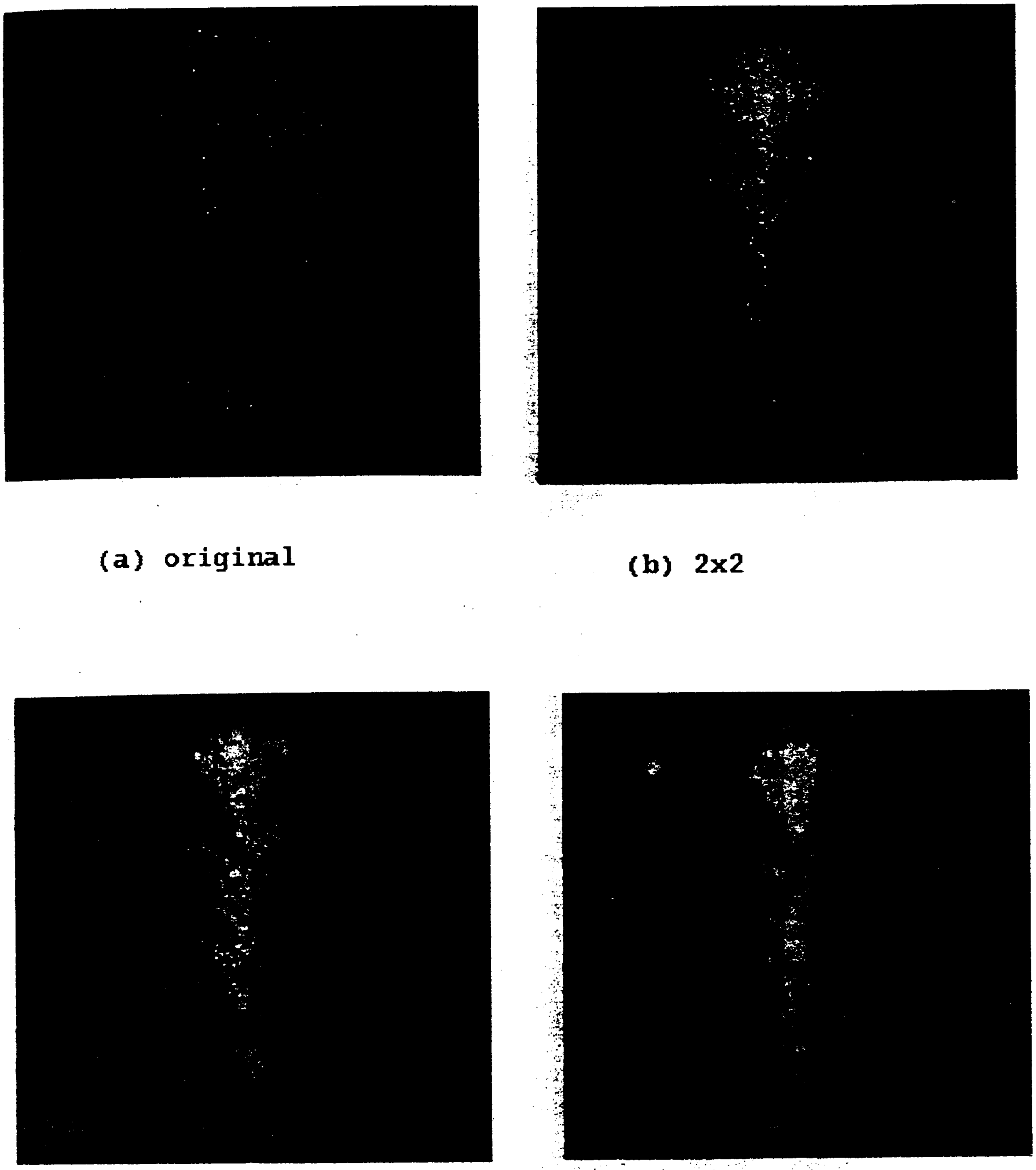
When the width of a cone is comparable with the voxel size, the cone can be approximated by its digitized line in the voxel space. In the example shown in (Figure 4-c), the cone is approximated by the successive voxels, v_1-v_2-v_3-v_4. This considerably simplifies the situation as follows:

- the direction of each voxel traversal becomes ±x, ±y, or ±z.
- volume segments of cones become identical to voxels.

Thus, voxels should store occlusion information only for the six directions. Furthermore, pre-computation of the volume integral in each voxel can avoid expensive volume integrals in the rendering phase.

3.3 Summing up voxels

This process involves filtering the information stored in voxels, i.e., density, correlation, and shaders. The calculation of density and correlation is straight forward, but filtering shaders requires representing



(a) original

(b) 2x2

(c) 4x4

(d) 8x8

Figure 5: Hierarchical 3D-texture of bamboo

Table 1: Number of non-empty voxels

level	resolution	number of non-empty voxels
original	204x825x173 (29.1M)	82K
2x2	102x413x87 (3.6M)	41K
4x4	51x207x44 (460K)	19K
8x8	26x104x22 (59K)	6.4K

the distribution functions of normal vectors in some way.

Recently, Fournier has developed an elegant method to filter bump maps [[FOU92]]. In his method, distributions of normal vectors are approximated by an adequate number of Phong-like functions using the non-linear least square technique. This method can be applied to voxel filtering.

3.4 Building a hierarchical 3D-texture

Once the techniques described in the above sections are established, construction of a hierarchical 3D-texture from geometry is straight forward. First, geometric objects are 3D-scan-coverted at a high resolution, and all voxels containing surfaces are stored as a 3D-texture. The 3D-texture is then successively summed to make a series of lower resolution 3D-textures. When the 3D-texture becomes one-layer, it has reached the mapping level, i.e, shader-mapping plus displacement mapping.

4 Preliminary Experiment

A simple experiment was made to examine the memory requirement. A bamboo stalk consisting of 41K polygons was first converted into a 3D-texture at the resolution of 102×413×173, and then the original texture was filtered to the levels of 2x2x2, 4×4×4, and 8×8×8 (Figure 5).

In the original texture, the number of non-empty voxels was found to be about 82K from 29.1M voxels (102×413×173). (See Table 1).

5 Conclusion

This paper proposed a hierarchical 3D-texture model that ensures smooth transition from geometry to mapping and allows efficient anti-aliasing of complex scenes. Several problems with 3D-texture were pointed out, and possible solutions were described. Early experiments indicated the potential of the model, and extremely complex scenes, like mountains covered with trees, or a stadium filled with thousands of people, are expected to be accurately rendered.

Acknowledgments

The author would like to thank Alain Fournier for useful discussions on bump map filtering. He also thanks Takahiko Kamae and Rikuo Takano for their administrative support, Marie-Claire Forgue for helpful suggestions, and Atsushi Kajiyama and Hiroki Kobayashi for their technical support.

References

[AMA] J. Amanetides, 'Ray Tracing with Cones', Computer Graphic **18**, No.3, pp.129-135, 1984.

[CABD] B. Cabral, N. Max, R. Springmeyer, 'Bidirectional Functions from Surface Bump Maps', Computer Graphics **21**, No4, pp.273-281, 1987.

[KAJ85] J.T. Kajiya, 'Anisotropic Reflection Model', Computer Graphics **19**, pp.15-21, 1985.

[KAJ89] J.T. Kajiya, T. L. Kay, 'Rendering Fur with Three Dimensional Textures', Computer Graphics **23**, pp.271-280, 1989.

[MAX] N. Max, "Shadows for Bump Mapped Surfaces", Advanced Computer Graphics, T. Kunii Ed. Springer Verlag, Tokyo, pp.145-156, (1986).

[PERLIN] K. Perlin, 'Hypertexture', Computer Graphics **23**, pp.253-262, 1989.

[POULIN] P. Poulin, A. Fournier, 'A Model for Anisotropic Reflection', Computer Graphics **24**, No.4, pp. 273-282, 1990.

[FOU92] A. Fournier, 'Filtering Normal Maps and Creating Multiple Surfaces', submitted for publication.