From Geometry to Texture:
Experiments Towards Realism in Computer Graphics

Thesis by
Timothy L. Kay

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California
1992
(Defended May 21, 1992)

Acknowledgements

My first thanks go to my parents, Eugene Kay and Virginia Pringle, for always supporting and encouraging my academic pursuits, and for ingraining in me the importance of a good education. I know they will be delighted when I *finally* put my education to use.

Arthur Karshmer led me towards a career in computer science when I was in the seventh grade, by loaning me some dusty old computer programming books. Hidden between FORTRAN and Cobol manuals, I discovered *Computer Programming in BASIC*[25], from which I learned to program before I ever saw a computer. His brother Bernard Karshmer expanded my intellectual horizons and always encouraged me to pursue my peculiar interests (and shared with me some of his own).

Thomas Banchoff and Andries van Dam introduced me to computer graphics at Brown University and provided opportunities and encouragement while I was at Brown and Caltech. Eva Kalin taught me pure mathematics and an appreciation for rigor.

Alan Norton and International Business Machines Corporation funded our lab and my research. Due to good fortune, Alan's interests at T.J. Watson Research Center coincided with my work as a graduate student, providing me the opportunity to pursue both research and consulting at the same time.

Thanks to Atsushi Takagi of Toyota Motor Corporation for teaching a useful seminar on color theory and for providing the geometric model of a car seat used in

Chapter 5.

Thanks to Leonid Rudin and Stanley Osher of Cognition Technologies Incorporated, for allowing me the use of their well-equipped laboratory to compute results and prepare this document.

I enjoyed sharing a laboratory with John Snyder, who was always available for enlightening discussions, and who designed our parametric Teddy bear model. My thanks also go to Ronen Barzel, Clark Brooks, Kurt Fleisher, Devendra Kalra, Dave Kirk, David Laidlaw, Jon Leech, Mike Newton, John Platt and Brian Von Herzen for many interesting conversations, all of which contributed to this document. To David Laidlaw, extra thanks for providing me an IGES manual when I needed it most.

From Alan Barr I learned many things both within and beyond the scope of computer science. Al taught me to present my work in a bold fashion; good work will go unnoticed unless accompanied by a strong presentation.

I thank James Blinn for teaching the fundamentals of computer graphics (and for having invented many of them). But more importantly, I thank Jim for teaching me the true meaning of intellectual honesty.

My heartfelt thanks to James Kajiya, my advisor, for his tremendous guidance, for his amazing scope of knowledge, and for providing me a great opportunity at Caltech. He never expected anything in return for his tutelage and support, but he always made opportunities available for me to lend a hand. I have learned a great deal from Jim during my many years assisting and collaborating.

Finally, I thank Rosemary Macedo, my wife, for incredible patience, inspiration, encouragement, and just the right amount of prodding, without which this thesis might never have been finished.

Abstract

This thesis presents a new computer graphics texture element called a *texel* as well as an associated rendering algorithm, which together produce an appearance never before achieved in computer graphics. Unlike previous modeling primitives, which are limited to solid, crisp appearances (e.g., metal, plastics, and glass), texels have a soft, fuzzy appearance, and thus can be used to create models and images of soft objects.

This thesis presents a solution to the problem of creating fur. As an example, a Teddy bear is modeled and rendered. As part of the process, a new BDRF is developed for texels which can produce back lighting effects. A model deformation technique using trilinear solids is developed.

This thesis then addresses a more complex example, that of creating a microscopic swatch of cloth by computationally "weaving" threads. The process of converting the resulting geometric model into texels is presented. The swatch of cloth is then replicated to cover the infinite plane seamlessly.

A new phenomenon, the *texture threshold effect* is presented. It is the point at which geometry turns into texture. When viewed from beyond a certain distance threshold, the appearance of a microscopic model will converge to a macroscopic model. The position of the texture threshold is calculated. The infinite cloth model is then rendered from beyond the texture threshold, and its cloth BDRF is extracted computationally. This BDRF is then used to render a cloth-covered car seat.

The BDRF extraction process involves sampling an image which contains spectral energy above the Nyquist limit. Hence, the use of point sampling in computer graphics is analyzed to verify that aliasing energy is controlled. The process of jittered subsampling is analyzed, correcting and completing previous attempts. The results confirm that it is possible to render complex computer graphics imagery avoiding artifacts from aliased energy.

Contents

List of Figures

Chapter 1

Introduction

Many of the original problems in computer graphics have been solved (e.g., shadows, reflections, refractions, motion blur, and depth of field), yet the goal of visual realism still lays beyond the state of the art. Using ever more complex geometric models, better physical simulation, and better rendering, researchers have succeeded in generating realistic scenes for only a few special cases. Traditional computer graphics remains limited to creating scenes which contain exclusively hard objects, e.g., glass, plastic, and metallic surfaces.

Commonly, computer graphics models are represented in one of two ways, as boundary representations, or as constructive solid geometry. Reflecting the mathematical origin of computer graphics, the appearance of these traditional primitives is strong on geometry but weak in texture. This thesis introduces a new modeling primitive, the *texel*, a truly three-dimensional primitive which must be rendered in a new way, and which provides an entirely new category of visual effects.

The texel primitive is a natural extension to the computer graphics construct called a *volume density*, which has been used primarily for the purposes of scientific visualization rather than as a tool in the quest for visual realism. The design of the texel combines techniques from the two usually disparate fields of scientific visualization and realistic image synthesis.

This thesis demonstrates that texels can realistically represent a range of soft objects. The texel primitive is explored in a series of three experiments, each moving further along the path from geometry to texture. The first experiment joins traditional computer graphics modeling with the texel primitive to create a Teddy bear. In the process, a fur texel is developed in conjunction with an appropriate bidirectional reflectance function (BDRF).

The second experiment leaves the realm of traditional primitive-based modeling: a complete computer graphics model of a swatch of cloth is created entirely within the confines of a single texel. A relaxation technique for weaving thread into a cloth model is presented.

The third experiment completes the trip from geometry to texture. A newly discovered phenomenon called the *texture threshold effect* is presented, which in essence, defines the point at which geometry turns into texture. Taking advantage of this effect, the BDRF of the cloth model is extracted. The resulting BDRF is used to shade a polygonal model of a car seat which then appears upholstered.

In the end we realize that the road from geometry to texture is circular; in traveling from geometry to texture, we end up again working with geometry. Via texels, we start with microscopic geometry and end with macroscopic geometry.

1.1 Organization of the Thesis

This thesis consists of seven chapters. This chapter, "Introduction," outlines the problems to be addressed and discusses previous work in the subject.

Chapter 2, "Texels," lays the ground work for the remainder of the thesis. The new modeling primitive, called a *texel,* is defined. Texels are first considered in the most general sense, as a collection of certain properties defined over an arbitrary region in space. The chapter then shifts its focus to the practical matter of actually

rendering texels. Several simplifying assumptions are made, and the texel rendering algorithm is presented.

Chapter 3, "Texels as Modeling Primitives: A Teddy Bear," explores the first of two uses of texels. Texels supplement more traditional computer graphics techniques in the creation of a Teddy bear model. Chapters 2 and 3 significantly expand upon the material published by Kajiya and Kay [19].

Chapter 4, "Texels as Models: A Cloth Texel," extends the use of texels beyond their role as modeling primitives to become models in their own right. A method for turning a geometric model into a texel is described.

Chapter 5, "Computationally Derived BDRFs," presents a method for creating bidirectional reflection models using computer graphics techniques. A model of the microscopic properties of a particular material is created. Using ray tracing, the microscopic model is rendered in such a way that the macroscopic BDRF is extracted. As an example, the BDRF is extracted from the cloth texel of the previous chapter.

Chapter 6, "Point Sampling and Computer Graphics," addresses an important issue that surfaces in the previous chapter. Point sampling has long been used in computer graphics although its effect has never been fully analyzed. This chapter analyzes point sampling and demonstrates that it is possible to perform the computations outlined in Chapter 5 with confidence that the measured signal is free from aliasing artifacts.

Chapter 7, "Conclusion," reviews the results achieved and poses ideas for future work.

1.2 Original Contributions

This thesis contributes the following original work:

- Texels. Texels naturally extend the concept of volume densities. Whereas volume densities encode a scalar density for each point in space, texels encode a scalar density as well as a coordinate frame and a bidirectional reflection function (BDRF) at each point in space.

- Fur model. A texel which effectively represents the visual appearance of fur is presented.

- Radially-symmetric hair BDRF. A radially symmetric hair lighting model is presented which, in conjunction with an appropriate texel density and coordinate frame, produces natural back lighting effects.

- Relaxation process to weave cloth. A method is presented which can be used to weave threads together to produce cloth-like patterns.

- Texel creation via scan conversion. A method for creating texels by scan conversion of geometric models is presented.

- Texture threshold effect. The texture threshold is the point at which geometry turns into texture. The existence of the texture threshold effect is established, and the position of the texture threshold is computed.

- Computational measurement of anisotropic BDRFs. A method for extracting anisotropic BDRFs from computer graphics models is presented. The BDRF extraction process is a significant enhancement of a previously published attempt at BDRF extraction.

- Cloth BDRF. A cloth BDRF is extracted from a microscopic cloth model and is used to render a cloth car seat.

- Class I and Class II image functions. Image functions are categorized by the nature of their spectra. The distinction between Class I and Class II spectral energy is presented.

- Computer graphics point sampling analysis. Point sampling via jittered sub-sampling in computer graphics is analyzed. Existing rendering methods are shown to be effective at attenuating aliasing energy. For a given selection of rendering parameters, a bound on the the amount of aliasing energy is derived.

1.3 Previous Work

Complex Scenes

Complexity is an essential ingredient in the quest for reality in computer graphics. Many attempts to generate scenes of high complexity have been made. Csuri *et al.* [12] generated images of smoke and fur using thousands of polygons. Weil [36] modeled cloth using thousands of Lambertian cylinders. In both cases the results appeared brittle rather than soft.

The fundamental problem with such geometric approaches to complexity is that it overwhelms computer graphics renderers. The complexity can never be increased to a degree necessary to produce the desired effect.

Volume Rendering

The volume density rendering model used in this thesis was introduced to computer graphics by Blinn [5], who was interested in rendering the rings of Saturn. He analytically integrated the appearance of a large collection of microscopic spherical particles uniformly distributed in a plane.

Kajiya and Von Herzen [18] introduced the idea of ray tracing volume densities

to include shadows. Their volume densities were used to represent clouds and were modeled procedurally with $\frac{1}{f}$ noise.

A great deal of work has been done in the area of scientific visualization, which typically involves the rendering of medical volume data, such as MRI scans. See, for example, Max [21], Nishita, Miyawaka, and Nakamae [23], Rushmeyer and Torrance [30], Sabella [31], Upson and Keller [35], Drebin, Carpenter, and Hanrahan [14].

Perlin and Hoffert [26] took a novel procedural approach to designing volume densities, which they call hypertexture. Their technique produces a new range of shapes and is able to make some convincing hair textures. Due to the global nature of their specification technique, they are unable to locally control the hypertexture adequately to produce elements which can be used in conjunction with traditional models.

Trilinear Solids

A trilinear solid is a special case of the hyperpatch, presented in Casale and Stanton [8]. While the hyperpatch is much more expressive geometrically, the trilinear solid affords precisely the degree of freedom needed to map the fur onto the bear in Chapter 3. Using this special case simplifies both the modeling and ray-trilinear patch intersection processes.

BDRF Extraction

Cabral, Max, and Springmeyer [7] proposed that a bidirectional reflectance function could be derived by computationally measuring a multi-faceted surface.

Analysis of Jitter Sampling

Jitter has always been an unavoidable nuisance in engineering sampling problems. Many engineers, e.g., Shapiro and Silverman [32], Balakrishnan [1], Brown [6], Beutler [3], and Masry [20], analyzed the effects of jitter in the hopes of determining the effect that such jitter had on the validity of their sampling. As much of this work predated the emergence of sampling algorithms in computer graphics, it is not surprising that they have little direct relevance in the field of computer graphics.

Cook [11] introduced the concept of jitter sampling to the field of computer graphics. Dippé [13] and Cook [10] both analyzed jitter sampling as it applies to computer graphics. Their analyses left large gaps which are addressed in this thesis.

Chapter 2

Texels

2.1 Definition of a Texel

For modeling and rendering soft objects, the traditional computer graphics view of the world as two dimensional surfaces embedded in $\mathbb{R}^3$ is inadequate. To render soft objects, we introduce a new, fully three-dimensional computer graphics primitive. The set of properties assigned to each point within a three-dimensional texel are a superset of the properties assigned to each point on a two-dimensional manifold in traditional computer graphics. Just as a two-dimensional surface model specifies properties such a normal and tangent vector and a BDRF to each point on the surface of the object, our new three-dimensional object will specify normal and tangent vectors as well as a BDRF to each point within the model.

Definition. A *texel* is a set $\{\alpha, \rho, \mathbf{B}, \Psi\}$ defined over a region of $\mathbb{R}^3$, consisting of an attenuation coefficient $\alpha(\mathbf{x})$, a scalar density $\rho(\mathbf{x})$, a coordinate frame $\mathbf{B}(\mathbf{x})$, and a BDRF $\Psi(\mathbf{x}, \boldsymbol{\theta}_l, \boldsymbol{\theta}_v)$.

The attenuation coefficient $\alpha(\mathbf{x})$ specifies how effectively texel material attenuates light intensity as the light propagates through the texel material. The most general definition of a texel allows $\alpha(\mathbf{x})$ to vary as a function of the position $\mathbf{x}$. For the purpose of this paper, however, the control provided by an attenuation coefficient

which is constant over the texel is adequate. Thus, the attenuation coefficient will be referred to without arguments, as simply α.

The density $\rho(\mathbf{x})$ encodes the geometry of the texel. Where the density is zero, no material exists, and where the density is large, the material is solid.

The coordinate frame $\mathbf{B}(\mathbf{x})$ describes a coordinate system at each point within the texel, consisting of a normal vector, a tangent vector, and a binormal vector. Any two of the three vectors will completely describe the coordinate frame; the third can be computed as the cross product of the other two.

The BDRF $\Psi(\mathbf{x}, \boldsymbol{\theta}_l, \boldsymbol{\theta}_v)$ describes how each point $\mathbf{x} = (x, y, z)$ in the texel interacts with a light source. The orientation of the light is given by Euler angles $\boldsymbol{\theta}_l = (\theta_l, \phi_l, \psi_l)$, and the position of the viewer is given by Euler angles $\boldsymbol{\theta}_v = (\theta_v, \phi_v, \psi_v)$. The coordinate frame $\mathbf{B}$ contributes to the lighting. (Its participation is implicit in that the angles $\boldsymbol{\theta}_l$ and $\boldsymbol{\theta}_v$ are measured using the local coordinate frame as coordinate axes.) The particular choice of BDRF is determined by the type of material that is being modeled.

2.2 Rendering Texels

To complete the description of a texel, we must specify how a texel interacts with light. In nature, light rays emanate from light sources and interact with objects that lie in their path. After traveling potentially complicated paths, a small fraction of the light rays reach the viewer. Figure 2.1(a) shows a light source, two objects and a viewer. Light rays start at the light source and interact with the objects, thereby being scattered in all directions. A few possible paths for the light rays are shown, one of which is shown hitting the viewer.

Rendering a geometric model in a manner that accurately accounts for such interreflections of light between objects encompasses solving Kajiya's rendering equation

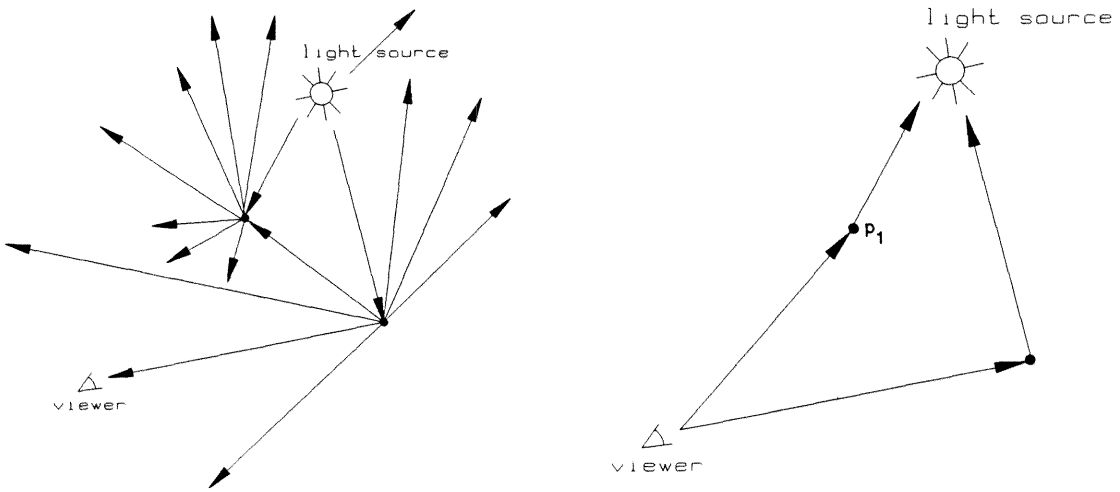


Figure 2.1: Nature versus ray tracing. (a) In nature, light emanates from the light sources; very few light rays reach the viewer. (b) In ray tracing, rays emanate from the viewer, thereby guaranteeing that no computation is wasted.

[17]. Due to its generality, the rendering equation is very difficult to solve. Instead, we will rely on an approximation of the rendering equation that has been in use in computer graphics for many years. Whereas nature casts rays from light sources, *ray tracing* [29] [11] casts rays from the viewer, as is shown in Figure 2.1(b). When a ray encounters an object p_1, a single additional ray is cast towards each light source. If the secondary ray, called a *light ray,* hits another object before it hits the light, then object p_1 is known to be in shadow with respect to that particular light.

When a geometric model contains traditional computer graphics primitives, ray-object interactions always occur at the surface of those objects. When texels are involved, however, rays interact not only at the surface, but at all the points along the ray for which the ray and the object intersect.

In designing a model for the interaction of light with texels, we treat a texel as a translucent material such that any point within the material can reflect light, and light that propagates through the material encounters exponential attenuation. Solid

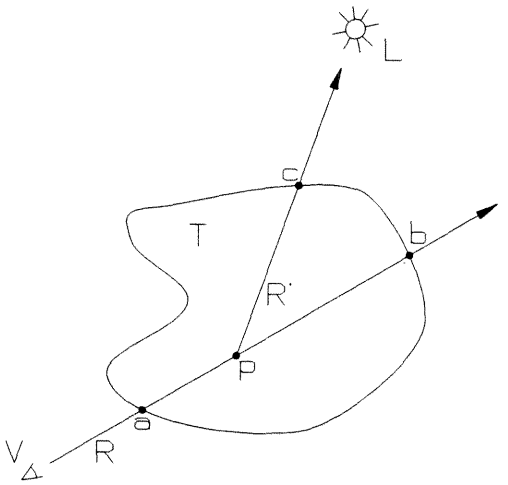


Figure 2.2: Ray R enters texel T at a and leaves at b.

surfaces are treated as very dense translucent materials, in which the parameters have been adjusted so that light attenuates completely before descending far into the material. At the other extreme, gases are modeled with parameters which cause the attenuation term to have a relatively small effect.

The texel illumination model presented here is essentially identical to the volume density illumination created by Blinn [4]. Our model is extended to incorporate the additional information available at each point within the texel.

As light propagates through a material, its intensity attenuates by the formula

$$I' = Ie^{-\alpha \int_a^b \rho(\mathbf{x}(s))ds}. \tag{2.1}$$

In words, the original intensity I is attenuated exponentially, resulting in a reduced intensity I'. The amount of attenuation is computed by integrating the density of the material along the path of the light ray.

Figure 2.2 shows a ray R emanating from the viewer and passing through texel T. The ray enters the texel at point a and leaves at point b. Along the entire segment between a and b, light interacts with the object and may be scattered towards the viewer. To determine the amount of light reflected from the light source L by object

T along ray R, first consider a test point p at some point along the ray, as diagramed in Figure 2.2. A ray that starts at R, interacts at p, and arrives at L must propagate through the texel material between points a and p, interact with p, and then propagate through the texel material between points p and c.

Applying Equation 2.1 to Figure 2.2, two attenuation terms arise. The *primary attenuation* term reduces the intensity of the light as it propagates from point p to point a, while the *secondary attenuation* term reduces the intensity of the light as it propagates from point c to point p.

The intensity of the light at point p, accounting for the attenuation along the ray R', is,

$$I_{R'}(p) = I_L(p)e^{-\alpha \int_p^c \rho(\mathbf{x}(t))dt},$$

where $I_L(p)$ is the intensity of light L that would have fallen on point p if the texel were not in its path. $I_L(p)$ is calculated in the typical computer graphics manner: if the light is infinitely far away, the intensity is a constant I_L; alternatively, if the light is at a finite distance, the intensity of the light falls off as the inverse square of the distance,

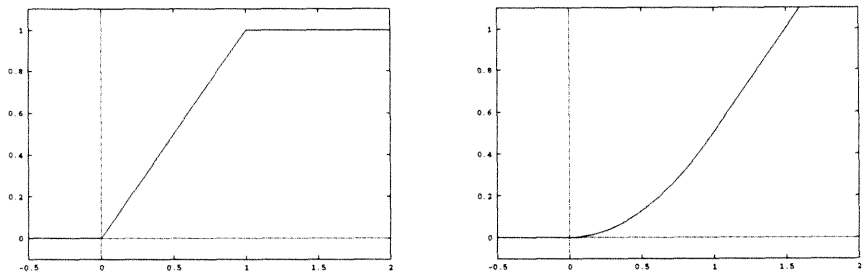
$$I_L(p) = \frac{I_L}{||L - p||^2}.$$

The light reflected from p back along R, $I_{R'}(p)$, is modulated by three factors. First, the material at p reflects light energy from L to V according to the BDRF $\Psi(p, R, R')$. Second, the reflected light travels between a and p and is then attenuated according to Equation 2.1,

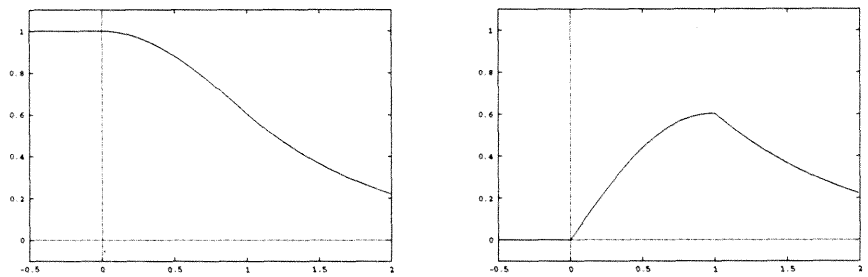
$$I_R(p) = \Psi(p, R, R')I_{R'}(p)e^{-\alpha \int_a^p \rho(\mathbf{x}(s))ds}. \tag{2.2}$$

Finally, the total intensity reflected back along R to the viewer is the integral of the pointwise intensity $I_R(p)$ along every point of R,

$$I_R = \int_a^b \rho(p)I_R(p)dp = \int_a^b \rho(p)\Psi(p, R, R')I_{R'}e^{-\alpha \int_a^p \rho(\mathbf{x}(s))ds}dp. \tag{2.3}$$



(a) $\rho(\mathbf{x})$ (b) $\int_{-\infty}^{x} \rho(u)du$

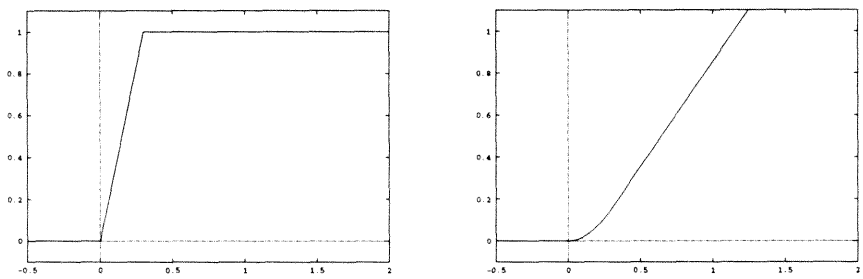


(c) $e^{-\alpha \int_{-\infty}^{x} \rho(u)du}$ (d) $\rho(x)e^{-\alpha \int_{-\infty}^{x} \rho(u)du}$

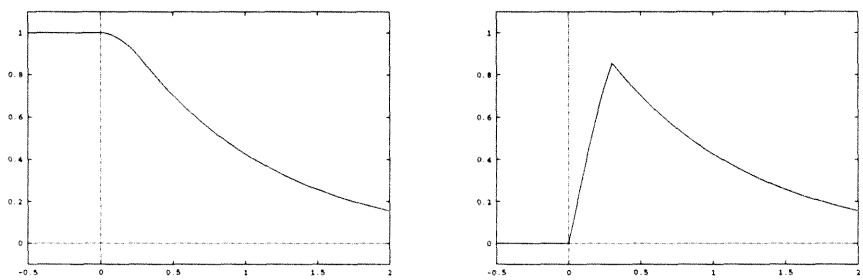
Figure 2.3: Texel behavior is most interesting in regions where the density changes. In (a) the density $\rho(x)$ ramps from 0 to 1 in one unit of distance. (b) shows the exponent $\int_{-\infty}^{x} \rho(u)du$ in the attenuation. (c) shows the attenuation factor $A(x) = e^{-\alpha \int_{-\infty}^{x} \rho(u)du}$ with $\alpha = 1$. In (d) the product $\rho(x)A(x)$ illustrates the short-term behavior of the texel integrand. ($\rho = 1, \alpha = 1$.)

Note the appearance of the $\rho(p)$ term in the integrand. The density at p is proportional to the probability that there was material at p to reflect light back to the viewer. Such an interpretation of ρ is consistent with the idea that ρ specifies the actual geometry of the texel. If the density is high, then a large amount of light is reflected to the viewer, and if the density is low, no light is reflected.

Figures 2.3 through 2.7 show various terms in the integrand of Equation 2.3 as applied to a simple one-dimensional texel. The figures demonstrate the effects of the primary attenuation by assuming that the BDRF term Ψ and light attenuation term $I_{R'}$ are a constant value 1.



(a) $\rho(\mathbf{x})$ (b) $\int_{-\infty}^{x} \rho(u)du$



(c) $e^{-\alpha \int_{-\infty}^{x} \rho(u)du}$ (d) $\rho(x)e^{-\alpha \int_{-\infty}^{x} \rho(u)du}$

Figure 2.4: Sharp density transition. If the density transitions more quickly (a), the resulting integrand contains a sharper peak. ($\rho = 1, \alpha = 1, \text{boundary} = .3$.)

Figure 2.3(a) shows the density ρ of the sample texel, which ramps from 0 to 1 in one unit of distance. Figure 2.3(b) shows the integral of the density $\int_0^t \rho(t)dt$. Figure 2.3(c) shows the complete attenuation term, $e^{-\alpha \int_0^t \rho(t)}$, computed with an attenuation coefficient $\alpha = 1$. Figure 2.3(d) shows the product of the density in(a) and attenuation term in (c).

Figure 2.3(d) demonstrates an important feature of Equation 2.3. Clearly, because the density ρ is one of its factors, the integrand is zero whenever the density is zero. Notice, moreover, that the exponential attenuation factor causes the integrand to tend to zero whenever the density ρ has been positive for a while. In other words, the integrand is nonzero only in the narrow region where the density first climbs away from zero.

Figure 2.5 illustrates this behavior more dramatically. The density profile ρ is the same in this case as it was in the previous figure, except that it and the attenuation coefficient α have been scaled by a factor of 50. Rather than rising from 0 to 1 in one unit, the density ρ rises from 0 to 50 in one unit. The exponential attenuation (c) is much more severe, and the non-zero region of the integrand is much narrower.

Solids

If the densities of Figures 2.3 and 2.5 were to be rendered, they would appear as solids. To understand this, consider what happens at various test points, specifically at the following positions. If the position $x = M$ represents the point along the positional axis at which the integrand assumes its maximal value, then for

$x < 0$ The value of the integrand is zero, so the contribution is zero. Such sample points are considered to be in empty space.

$0 < x \ll M$ The value of the integrand will be small. Such points fall within the boundary layer of the material being rendered. Their contribution will usually be overwhelmed by contributions from additional points further inside the material.

$x \approx M$ Sample points in the region near M will contribute substantially to the result of the integration.

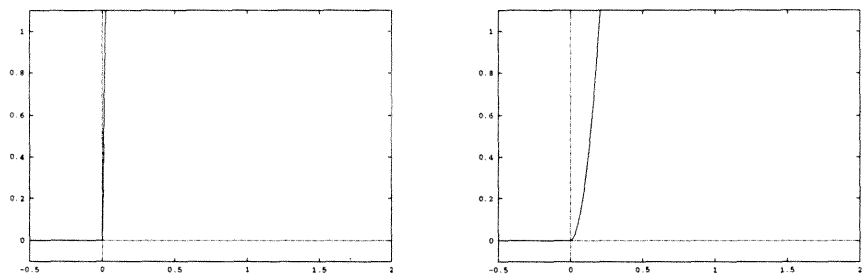
$x \gg M$ Such samples are located either well within the solid material or beyond the region of high density. Due to the exponential attenuation factor, such samples contribute very little or nothing to the resulting integral.

In the case that the integrand of Equation 2.3 contains a sharp peak, the important samples occur near the boundary layer of the solid. Outside the surface, no energy is reflected towards the viewer; inside the surface, the attenuation term eliminates any reflected energy before it reaches the viewer.

Gases

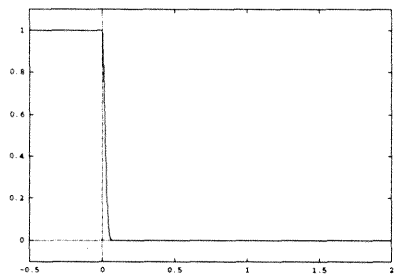
Figure 2.6 demonstrates what happens when the attenuation coefficient is small. The exponential attenuation term becomes much less significant, and the resulting integrand fails to demonstrate the same short-term behavior as in the previous examples. Instead, the value of the integrand climbs to a maximum and then slowly decreases over a long period. Such behavior is typical of gases.

Figures 2.4 and 2.7 demonstrate the effect of shortening the distance in which the density term climbs from zero to its maximum value. In Figure 2.7, the density transitions instantaneously from zero to its maximum, thereby causing a discontinuity in the integrand. It will be shown later that such extreme behavior produces aliasing artifacts, and that simple steps can be taken to avoid such discontinuities.

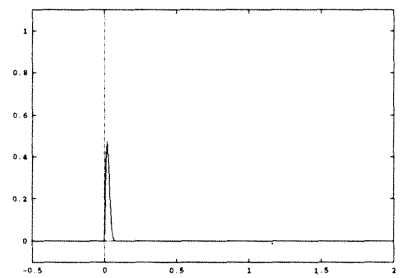


(a) $\rho(\mathbf{x})$

(b) $\int_{-\infty}^{x} \rho(u)du$

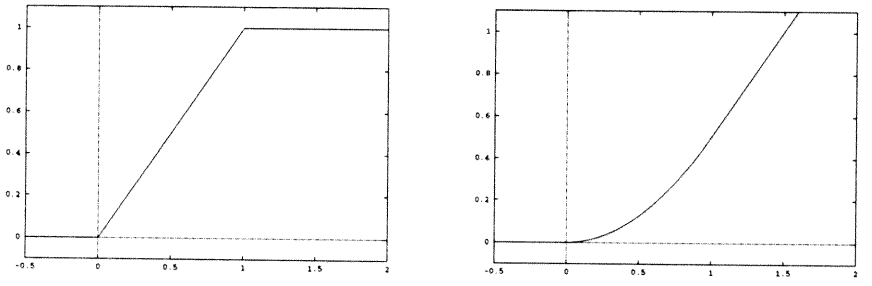


(c) $e^{-\alpha \int_{-\infty}^{x} \rho(u)du}$



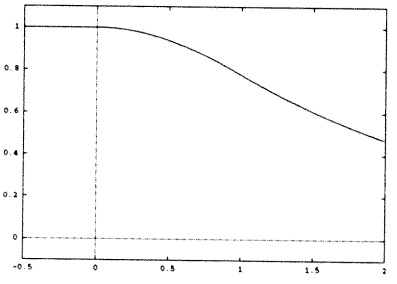
(d) $\rho(x)e^{-\alpha \int_{-\infty}^{x} \rho(u)du}$

Figure 2.5: Texel containing solid material. Both the density and attenuation factor have been scaled by 50, resulting in a much sharper peak. ($\rho = 50, \alpha = 50$.)

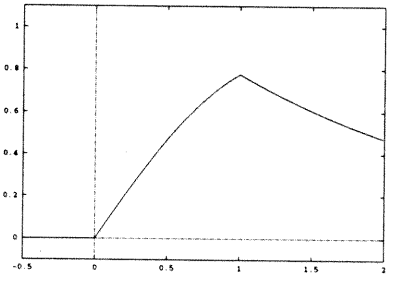


(a) $\rho(\mathbf{x})$

(b) $\int_{-\infty}^{x} \rho(u)du$

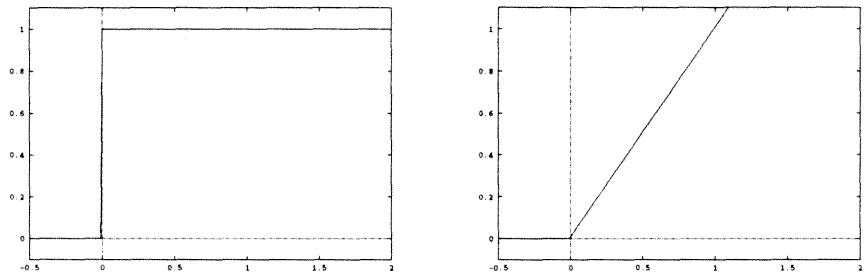


(c) $e^{-\alpha \int_{-\infty}^{x} \rho(u)du}$



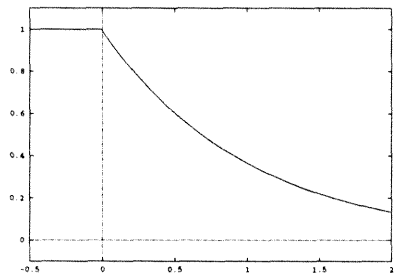
(d) $\rho(x)e^{-\alpha \int_{-\infty}^{x} \rho(u)du}$

Figure 2.6: Texel containing gaseous material. The attenuation factor has been reduced by a factor of two, resulting in a slightly less severe peak. ($\rho = 1, \alpha = 0.5$.)

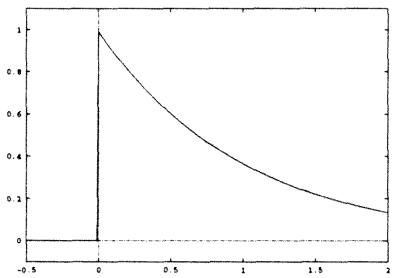


(a) $\rho(\mathbf{x})$

(b) $\int_{-\infty}^{x} \rho(u)du$



(c) $e^{-\alpha \int_{-\infty}^{x} \rho(u)du}$



(d) $\rho(x)e^{-\alpha \int_{-\infty}^{x} \rho(u)du}$

Figure 2.7: Discontinuous integrand. If the density increases in a discontinuous fashion as in (a), the resulting integrand (d) contains a discontinuity. ($\rho = 1, \alpha = 1, \text{boundary} = 0.$)

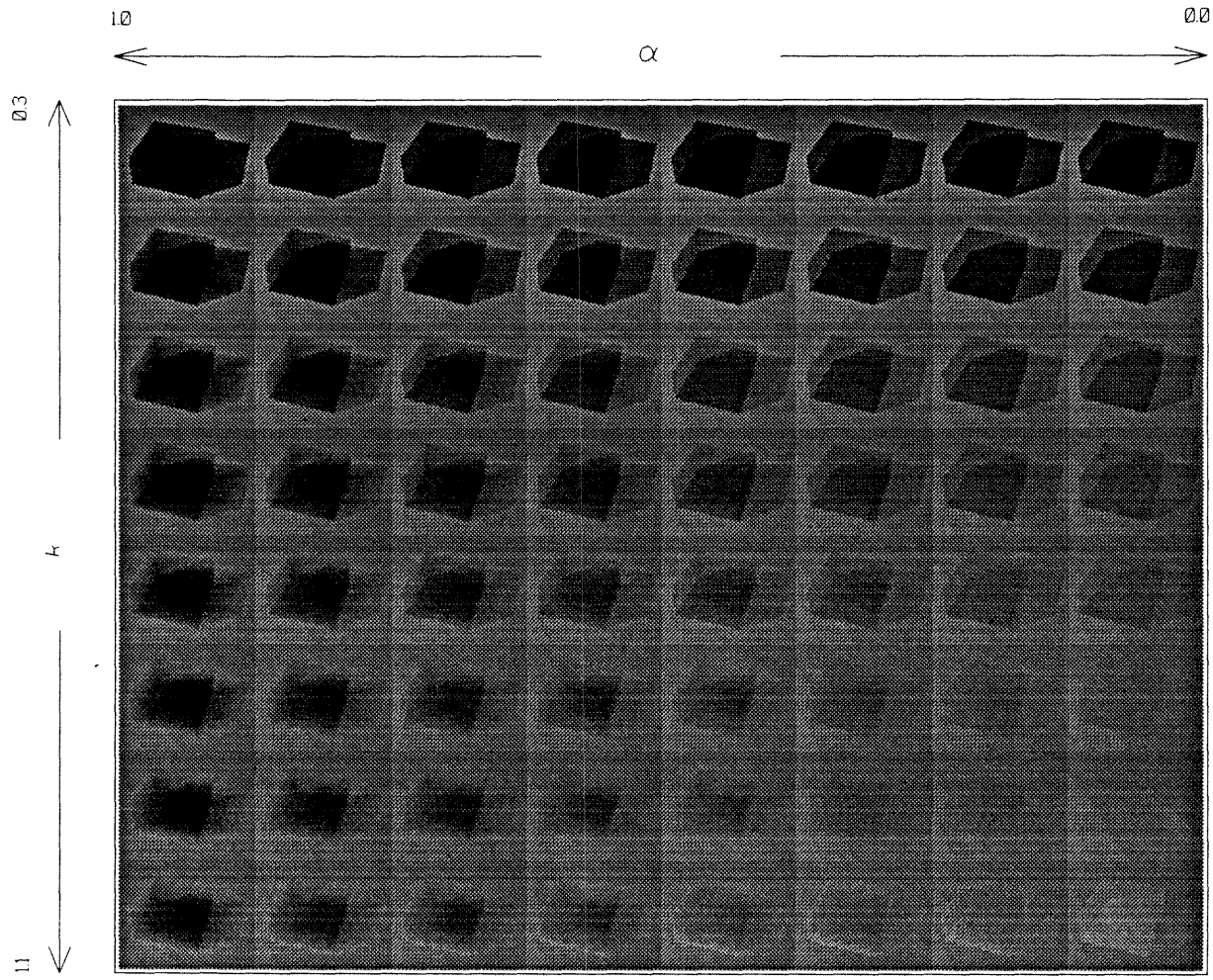


Figure 2.8: Texel attenuation. Constant density texels demonstrate the attenuation term $ke^{-\alpha \int \rho(x)dx}$ of the texel rendering equation. The attenuation coefficient α changes along horizontal axis. The scale factor k changes along the vertical axis ($\alpha = 1.0$ at the left edge; $\alpha = 0.0$ at the right edge; $k = 0.3$ across the top; $k = 1.1$ across the bottom. The bottom row glows where the attenuation term exceeds 1.0.)

Figure 2.8 visually illustrates the effect of various parameter settings on the attenuation factor $ke^{-\alpha \int \rho(x)dx}$. The cube in the picture is a zero albedo object. It can be seen only because it causes attenuation; no light reflects from it. Along the horizontal axis, α changes from 1.0 at the left edge, where the exponential has maximal effect, to 0.0 at the right edge, where the exponential has no effect. Along the vertical axis, k changes from 0.3 at the top, where it causes the objects to be very dark, to 1.1 at the bottom, where it causes the objects to intensify the light reflected from the ground plane, thereby causing halos. Such light amplification is not physically realistic.

This section has presented the continuous version of the texel rendering equation and presented examples to help build intuition. The next section discretizes the theory so that it can be used for calculations in computer graphics.

2.3 Computing the Solution Numerically

So far, texels have been discussed as purely mathematical constructs. Using the computer, we wish to render models which contain texels. To do so, numerical approximations to the previously discussed constructs and integrals must be developed.

Array Representation of a Texel

By definition, a texel is an arbitrary region in space defining an arbitrary density, coordinate frame, etc. at each point within the region. In practice, it is impractical to manipulate arbitrary regions within a computer. Instead, two simplifying assumptions are made. First, the shape of a texel is restricted to be a rectangular solid. This assumption can be made without loss of generality. When a shape other than rectangular solid is desired, that shape can be embedded within a bounding rectangular solid, and the density of the texel outside the desired shape can be set to zero.

The second simplification does make the implementation of texels less general than

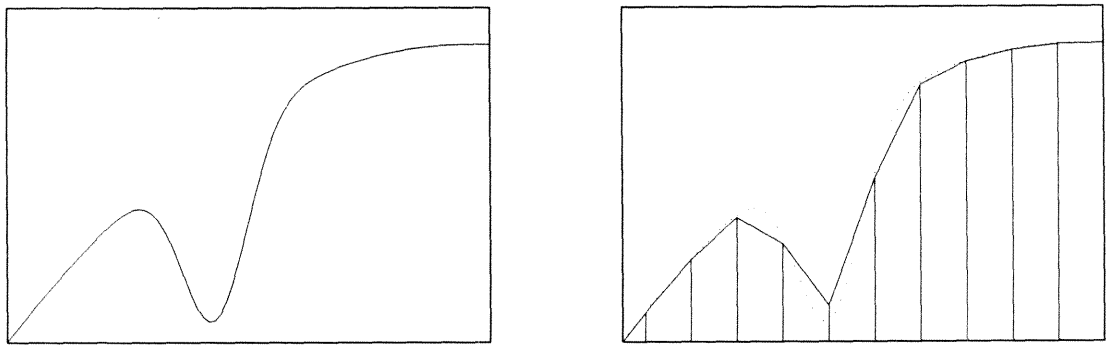
possible within the definition. For the purposes of this paper, texels are approximated as three-dimensional arrays of floating point numbers. A texel's scalar components, such as the density ρ, are represented by individual three-dimensional arrays of numbers, while the vector components, such as the tangent $\mathbf{T}$, are represented by triples of three-dimensional arrays.

Having decided on a discrete representation for texel data, we must next decide how to interpolate values that fall between the discrete data points. There are many possible interpolation techniques that trade off computational expense for various degrees of accuracy, smoothness, and continuity. For the sake of simplicity, simple trilinear interpolation is used. This choice further restricts the family of texels that can be represented. At the same time that it guarantees continuity, linear interpolation precludes sharp transitions. Therefore, the actual position of solid surfaces cannot be represented precisely.

Figure 2.9(a) shows the density function for a solid that we wish to represent as a texel. The approximation as an array with interpolation only loosely represents the original solid. Figure 2.9(b) shows the approximation of a one-dimensional object by a one dimensional texel. The process of converting an object to a texel array and then reconstructing it using linear interpolation results in a new object which closely represents the original. When the representation is too coarse, the designer can always increase the number of cells in the texel.

Texel Array Boundary Constraints

As illustrated in Figure 2.10, a texel represented by an array of numbers can be interpreted in two different ways, depending on whether the numbers are intended to assign values to the centers or the corners of the cells of the array. To determine which interpretation is better, consider a special case for texel arrays. It is often



(a) (b)

Figure 2.9: Approximating an object with a texel array. Plot (a) shows a solid object that is to be represented by a texel array. In (b) the texel array is drawn as a sequence of vertical bars. An approximation to the original object is reconstructed from the texel values using linear interpolation.

useful to design a texel that can abut with itself or another texel. The texels must abut in such a fashion that no seams occur at the boundary. Using the "centers" interpretation, it is very difficult to determine texel values because it is necessary to interpolate across the the texel boundary using values from both texel. The "corners" interpretation offers the advantage that the value of any point within any texel can be determined by interpolating values from just one texel. This property simplifies the implementation of a variety of algorithms, and is therefore the interpretation that we choose.

The corners approach does mandate a constraint on the adjacent boundaries of two abutting texels. Because no information is communicated between texels during the interpolation process, the abutting faces must be identical, or the rendered seam will appear discontinuous.

Care must also be taken when the face of a texel does not abut another texel. Using the "corners" interpretation of texel data, a non-abutting texel ends abruptly at its faces. Figure 2.7 shows the integrand of the texel rendering equation in the

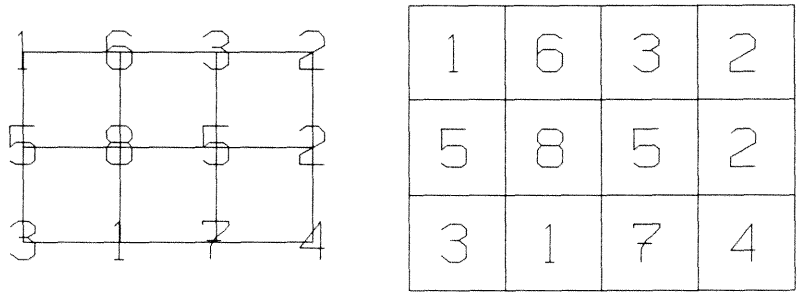


Figure 2.10: Texel boundary conditions. The boundary of a texel can be interpreted in two different ways. On the left, the texel values are specified at the corners of the grid cells. On the right, the values are specified at the center of the grid cells.

case when the boundary cells of a texel are non-zero. As will be discussed in the next section, the discontinuity in the integrand can create situations that are difficult to render. In some cases, it is useful to design texels that end in such an abrupt fashion, but more often, such behavior is undesirable. Therefore, it is usually desirable to insure that non-abutting texel faces assume the value zero.

In short, the two boundary constraints required by texels are that abutting faces should be identical and that non-abutting faces should be zero.

Rendering Texel Arrays

The intersection of a ray with the boundary of a texel returns an interval $[t_{\text{near}}, t_{\text{far}}]$ over which the ray tracer must integrate the texel. To perform the integration, Equation 2.3 is approximated by the sum

$$I_R = \Delta t \sum_{t_{\text{near}} \le t \le t_{\text{far}}} \rho(R(t)) I_{R'} e^{-\alpha \Delta t \sum_{t_{\text{near}} \le u \le t} \rho(R(u))}. \tag{2.4}$$

The value Δt is the step size and is specified as a global parameter.

Figure 2.11 illustrates the texel array rendering algorithm. The texel is rendered by stepping along the ray R within the interval $[t_1, t_2]$. At each position p_i along the way, shading calculations are performed for a virtual particle. In the figure, the steps

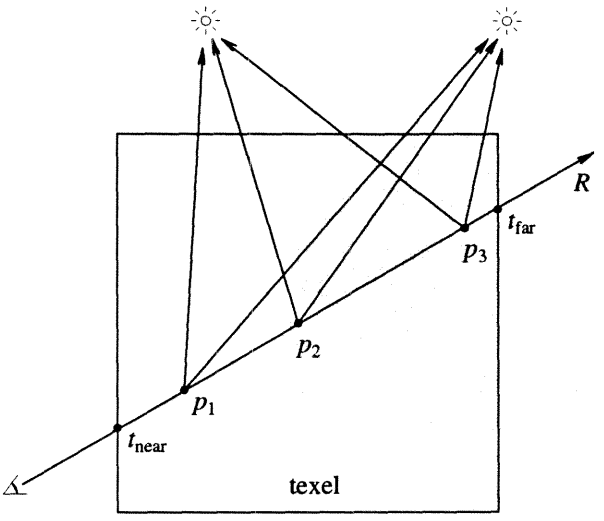


Figure 2.11: Rendering a texel array. The texel integration is replaced by a sum. The integrator steps along the ray, shooting shadow rays from each sample point, and summing texel contributions.

occur at positions p_1, p_2, and p_3. The texel is evaluated at each sample point, yielding values for α, ρ, B, and Ψ.

The inner summation of Equation 2.4 is computed incrementally. As the integrator steps along the ray, the successive values of ρ are accumulated into variable σ. At each point along the ray, an individual sample's contribution to I_R is

$$I_R(p_i) = \Delta t \rho(R(t)) I_{R'} e^{-\alpha \Delta t \sigma}.$$

At each p_i, shadow rays are cast towards each light source. The shadow ray calculation determines whether the object is shadowed by another object. It also computes the attenuation term, which affects the intensity $I'_R(p_i)$ using the same integration function, via a recursive function call. Because our rendering model uses Blinn's approximation, it does not incorporate multiple scattering, the shadow ray integration need not perform shading calculations nor spawn additional shadow rays.

Two optimizations are made based on the values of ρ and σ. Frequently, the integrator traverses empty space within the texel. In that case $\rho = 0$, and the

```
texel_integrate(I_R, a, τ, R, t_near, t_far, Boolean shade)
    I_R = σ = 0
    t = t_near
    while t ≤ t_far
        p = R(t)
        (ρ, B, Ψ) =linear interpolate τ at p
        σ+= ρΔt
        if shade=Yes and ρ > 0 and σ < Σ then
            for each light at L_i, of intensity l_i
                R'(t) = (L_i − p)t + p
                texel_integrate(·, a', τ, R', ε, t'_far, No)
                I_R+= ρΨ(R, R') a' l_i / ||L_i − p||² e^{−ασ}
        a = e^{−ασ}
        t+= 2Δt rand()
```

Figure 2.12: Outline of the texel integration function. The integral of texel τ produces an intensity I_R and an attenuation factor a.

contribution of $I_R(p)$ to I_R is zero. At other points, the integrator traverses space within the depths of a solid. In that case $\sigma \gg 1$, and the attenuation term approaches zero, as does the contribution of $I_R(p_i)$ to I_R. In either case, all further calculations at the particular point p_i may be skipped.

Figure 2.12 outlines the integration function. The texel τ is integrated between points $R(t_{\text{near}})$ and $R(t_{\text{far}})$. The function returns intensity I_R and attenuation factor a. Initially `texel_integrate` is called with `shade=Yes`, which indicates that shadow ray calculations and shading calculations are to be performed. Each recursive call to `texel_integrate` is made with `shade=No`.

Step Size and Monte Carlo Sampling

The approximation of the integral Equation 2.3 with the sum Equation 2.4 requires careful consideration regarding the choice of step size Δt as well as the placement of the samples. The computational requirements increase inversely with the size of Δt.

A large Δt decreases running time at the expense of quality of the integration.

As shown in Figure 2.3 through Figure 2.5, the integrand is non-zero over very narrow regions only. If the integrator is to detect all such regions, then the value of Δt must be smaller than the width of the smallest such region. It is not possible to specify such information *a priori* for an arbitrary texel. In the specific case of the array representation of texels, one can estimate a bound. The character of the integrand cannot change on a scale smaller than the size of an array cell. By setting Δt to be smaller still, we assure that the integrator encounters all features of a texel.

The value Δt is used to select successive sample points along the ray. If the sample points are chosen naively, as $p_i = p_{i-1} + \Delta t$, then problems occur: the position of the sample points between adjacent rays correlate, causing phantom surfaces to appear within the texels. To avoid such problems, a technique from Monte Carlo theory is employed to decorrelate the positions of the sample points. As indicated by the function in Figure 2.12, the positions of the sample points p_i are chosen using *additive random sampling* [32]: $p_i = p_{i-1} + 2\Delta t\, X$, where X is a uniform random variable over the interval $[0, 1]$. The expected value $EX = 0.5$, so each step is Δt units on the average.

The Self Shadowing Problem

Density gradients in a texel cause features to appear in the integrand of Equation 2.3. Texels are integrated using an average step size Δt, which is is adjusted to be small enough to assure that the integrator will encounter any such feature. Subject to this constraint, Δt is adjusted to be as large as possible to reduce the computational requirements of the integrator.

Consider the case that only a single sample falls, either at point p or at point q, within the integrand feature indicated in Figure 2.13. Each integrand feature

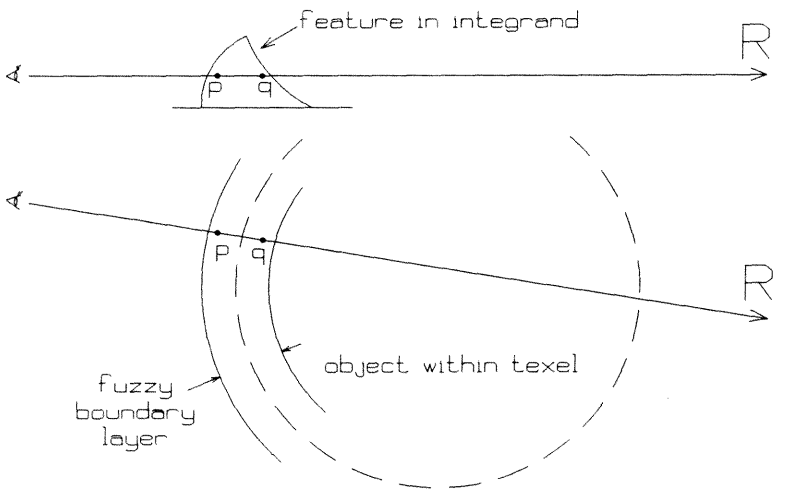


Figure 2.13: Texel object fuzzy boundary layer due to a feature in the integrand. Integration samples can fall near the outside of the fuzzy boundary layer or deep within the boundary layer.

represents a boundary layer at the surface of an object. Ideally, an integration sample (either at p or at q) should be shaded equally because the appearance of the object should be independent of the actual placement of the sample.

It might appear that a sample at point q, which falls well within the boundary layer, will suffer more attenuation than a sample at point p, which falls near the outside of the boundary layer. Conveniently, the integrator's knowledge of the integrand is derived exclusively from the sampling process: a sample that lands at position q *has no knowledge of the material that occurs between itself and the viewer.* At q, just as at p, the value of σ will be zero, and the sample will be rendered without primary attenuation.

The situation is different for shadow rays. A shadow ray shot from point p typically will traverse much less material than a shadow ray shot from q. Although the point q is not attenuated due to material between itself and the viewer, energy from the light to point q is attenuated by a larger factor than the same energy to p. The effect of this attenuation is that a sample that falls well within the boundary layer is

illuminated poorly. In this regard, Equation 2.4 fails to approximate Equation 2.3.

The problem just posed is actually a variant of the ray tracing problem known as the *self shadowing problem.* In traditional ray tracing, at the point p where a ray hits an object x, a shadow ray is shot towards the light source. The problem is that that shadow ray always hits object x, and the ray tracer determines that object x is (always!) in shadow. The solution to the self shadowing problem is to specify a tolerance ϵ. Any shadow-ray-object intersections closer to p than ϵ is ignored. Obviously, if ϵ is specified as a very large number, then nearby objects fail to cast shadows on each other. Typically with traditional ray tracing ϵ can be a very small number (e.g., 10^{-10}), but not always (e.g., Barr's superquadric intersection code [2]). In texel ray tracing, the value of ϵ must be adjusted appropriately to alleviate the self shadowing problem.

Specifying a minimum distance ϵ does not completely alleviate the texel version of the self shadowing problem. Fortunately, the consequence of self shadowing is not as catastrophic as in the case of traditional ray tracing. Surfaces appear darker than they would if a perfect integration were performed, and the magnitude of the effect is, to a small degree, proportional to the choice of Δt, a result which is marginally disturbing. Ideally, the choice of Δt should have no systematic effect on the appearance of the image.

The images in this thesis were all rendered as described, and we find them to be of acceptable quality. It is always possible to consider fancier (i.e., adaptive) integrators, but the complexity of the algorithm increases significantly.

Results

Figure 2.14 shows a sphere texel rendered using various values of the density and Δt. The density decreases towards the top of the figure. Δt decreases towards the right

side of the figure. The plate labels indicate the maximum density value and number of samples per unit length.

The sphere texel density was created by scan conversion, which will be described in Chapter 4. The normal vector component of the sphere texel is given by the 2×2×2 array

(-1,1,-1)	(1,1,-1)	(-1,1,1)	(1,1,1)
(-1,-1,-1)	(1,-1,-1)	(-1,-1,1)	(1,-1,1)

.

The normal vectors produced by the array approximate but do not exactly match the real sphere normal vectors. A traditional Lambertian lighting model was used in conjunction with the sphere texels.

Figure 2.15 shows a texel created using low-pass filtered white noise. The value of the noise determined the density at the base of the texel as well as the height of the column. The bear fur lighting model, presented in Chapter 3 is used. Tangent vectors are constant $(0, 0, 1)$. The two tori serve as a demonstration of the importance of shadows in visualizing texels. The texels are affixed to the surface of a torus as described in Chapter 3.

2.4 Chapter Summary

A texel was defined as an attenuation coefficient, a density, a coordinate frame, and a BDRF over a region in space. An integration scheme (similar to that of [4]) was presented for integrating arbitrary texels defined over arbitrary regions in space.

With an eye towards implementation, a simplified texel representation was introduced, in which texels are approximated by three-dimensional arrays with trilinear interpolation. To simplify texel interpolation, the "corners" interpretation was chosen rather than the "centers" interpretation, and two texel boundary conditions logically followed.

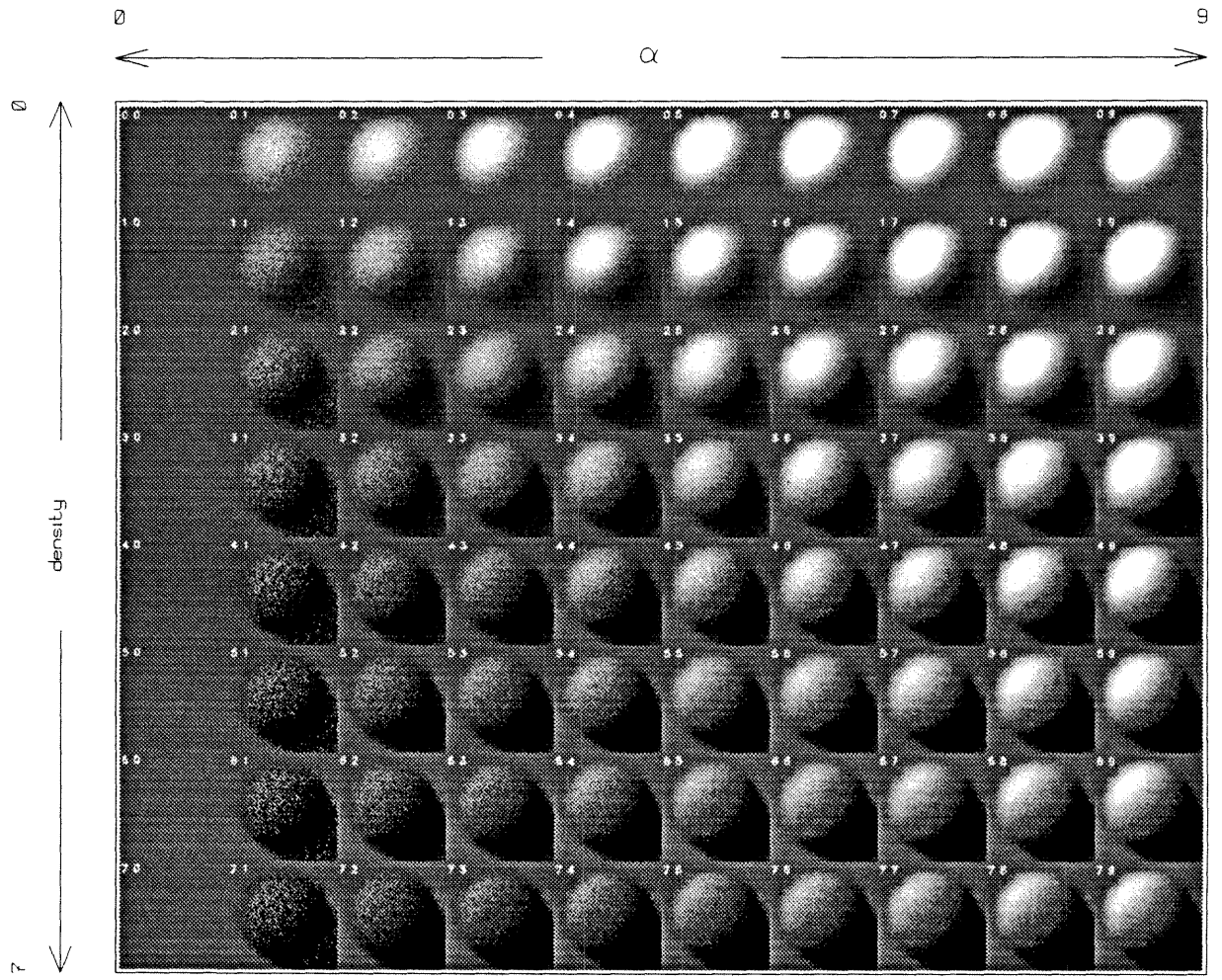


Figure 2.14: A texel representation of a sphere. The texel is rendered for various values of the density along vertical axis. Both the attenuation coefficient and samples per unit length vary along the horizontal axis. As the number of samples increases, the attenuation coefficient is decreased linearly.

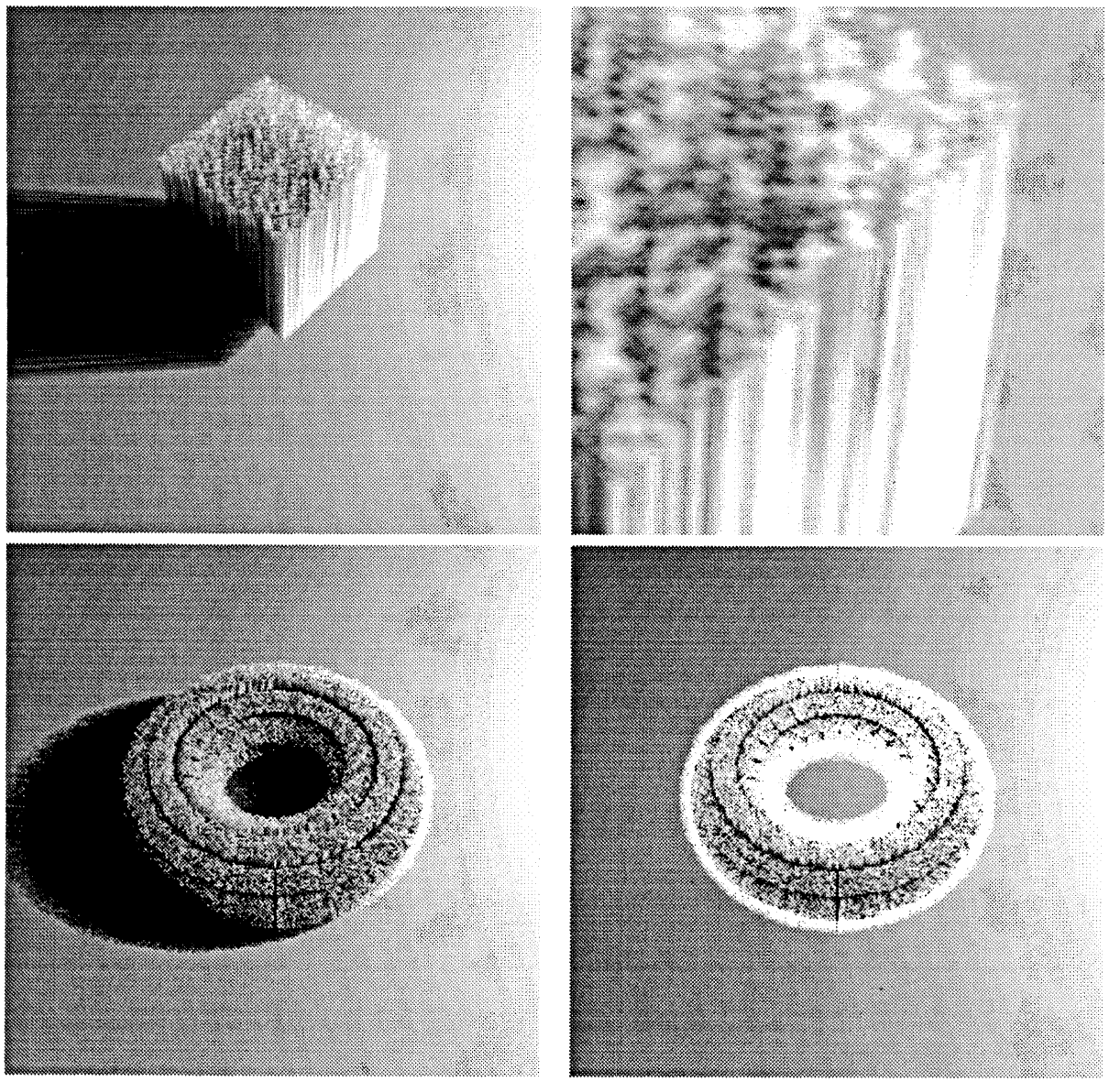


Figure 2.15: A texel generated with white noise. The texel is "glued" onto the tori. One torus is rendered with shadows on, the other with shadows off.

Texel integration was approximated with a summation, and the step size of summation operation was bounded to be smaller than the size of a cell in the texel. Uniform random sampling was employed to decorrelate samples between adjacent rays.

To illustrate the use of texels, several images of rendered texels were presented.

Chapter 3

Texels as Modeling Primitives: A Teddy Bear

A texel can be used in computer graphics in two fundamentally different ways, either as a modeling primitive, or as the model itself. This chapter and the next give an example of each approach. This chapter considers texels as a modeling primitive. A texel with desired characteristics is designed using new techniques and is then applied as a geometric primitive in the traditional computer graphics sense. Chapter 4 takes a more radical approach to computer graphics modeling, wherein an entire computer graphics model is built entirely within the confines of a texel.

This chapter describes the creation of a computer graphics model of a Teddy bear. The creation process involves three steps:

- A fur texel is created. The texel contains a fur density function and corresponding BDRF, which are both designed in an *ad hoc* fashion to mimic natural properties of fur.
- A parametric model of the Teddy bear body is created.
- The fur texels are mapped onto the surface of the parametric model.

The creation of the Teddy bear model introduces concepts which are new to computer

graphics. Designing texels is an entirely new subject; this section will describe a few examples of texel design processes. The texel design involves the creation of both geometric texel components (ρ, B, α) and an appropriate BDRF.

The design of the parametric model uses a technique called *generative modeling*, which is not new to this thesis and has been detailed in [33]. For the sake of completeness, a brief description of the generative modeling process is included here.

The mapping of the texels onto the bear body uses a restricted version of *hyperpatch mapping* [8], called *trilinear mapping*. This mapping process uses a structure called a trilinear cube which is composed of six bilinear patches. Bilinear patches are related to other patches but offer fewer degrees of freedom, and are used here because they afford precisely the degree of freedom required by the texel-to-parametric-surface mapping problem.

3.1 Fur Texel

The process of creating a texel which models a real-world object can involve different degrees of realism. For some applications, it might be necessary to produce a representation of an object that reproduces that object faithfully in many different senses. For example, some applications might attempt to extract physically significant measurements or perform dynamic simulations of the modeled objects. As far as this chapter is concerned, the only measure of realism to be applied is visual appearance. The objective is to produce a texel representation of a fur patch for which the visual appearance is convincing.

In designing the fur texel, several decisions are made for the sake of simplicity. First, only a single texel is created to represent a small patch of fur, and it is replicated to tile the entire surface of the bear. Second, the fur texel is designed as a cube, with all hairs sticking straight upwards. The trilinear warping process, discussed in a later

section, will map the multiple instances of each texel into their final resting position on the body of the bear. As will be explained in that section, it suffices to design a texel that properly tiles the infinite plane. The space in which we design the texels will be called *object space.* The space in which the bear model exists will be called *world space.*

Fur Texel Density

The fur texel will be modeled within a three-dimensional array whose shape is a rectangular solid. The faces to which the hairs will be anchored will be called the *scalp.* The first task in designing the fur texel is to choose the placement of the hairs on the scalp. By inspecting fur samples and human hair, it was decided to place the hairs in a *Poisson disk* pattern. A Poisson disk is generated by randomly placing points within an area, subject to the constraint that no two points fall closer than d units of distance. When enough points have been placed such that the addition of more points is impossible, the resulting collection of points forms a Poisson disk.

Generating a Poisson Disk

An approximate Poisson disk can be generated with algorithm `Poisson` of Figure 3.1. `Poisson` repeatedly chooses random points in the unit interval. If a new point is more than d units away from all previously-selected points, the point is added to the set of selected points. If N successive points fail to meet the minimum-distance criterion, `Poisson` terminates, and the points selected so far form an approximate Poisson disk. Figure 3.2 shows one such approximate Poisson disk, generated with $d = 0.2$.

The parameter d determines the density of the Poisson disk. A smaller d results in more points and, therefore, more hairs in the fur texel.

`Poisson` does not necessarily generate a true Poisson disk. To be a true Poisson

```
n = 0
loop
    failed = −1
    do
       failed = failed + 1
       if failed > N then terminate
       p = random (x, y) ∈ [0, 1]×[0, 1]
    while ||p_i − p|| < d for some i ∈ {1, ..., n}
    n = n + 1
    p_n = p
repeat
```

Figure 3.1: Algorithm `Poisson` creates an approximate Poisson disk.

disk, it must be the case that any new point added to the disk would violate the distance criterion. To determine if a particular configuration chosen by `Poisson` is a true Poisson disk, imagine drawing filled circles of radius d centered at each point. If the circles completely cover the unit square, then the disk is a true Poisson disk.

It is possible to create a configuration of points for which the circles cover all but a very small area within the square. In that case, an additional point could be added without violating the distance criterion. However, if the uncovered area is small enough, it becomes unlikely that `Poisson` will find the area and complete the Poisson disk. For this reason, disks created by `Poisson` are only approximate Poisson disks. For the purposes of creating a fur texel, approximate Poisson disks suffice. Hence, the distinction between true and approximate Poisson disks will now be dropped.

To facilitate the covering of the Teddy bear, the fur texel must be able to tile the infinite plane. To do so, the left edge of the texel must be able to abut the right edge without artifacts, as must the top and bottom edges. To make such a Poisson disk, it is necessary to modify the distance metric $||\mathbf{p}_i - \mathbf{p}||$ in algorithm `Poisson`. The distance between the test point and the selected points must be measured as if the

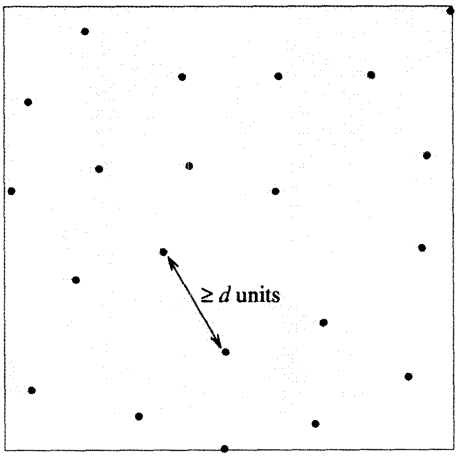


Figure 3.2: Poisson disk. A *Poisson disk* is the unit square packed with random points such that no two points fall closer than d units.

Poisson disk were being created on the surface of a torus. In other words, the distance must be measured from test point $\mathbf{p}$ to each of the points within the Poisson disk, as well as to the points in the eight surrounding tiles, as illustrated in Figure 3.3.

Bear fur is formed of two different lengths of hair called the *undercoat* and *overcoat*. The undercoat is a thick covering of relatively short hairs, while the overcoat is longer and sparser. The two coats were placed on the scalp using an interactive version of algorithm `Poisson`. First, a Poisson disk with $d = 0.1$ was created, and the points tagged as overcoat hairs. `Poisson` was then run a second time without clearing the previous values for n and $\mathbf{p}_i$, but with the change $d = 0.2$. The effect of the second pass was to add additional hairs, which were marked as undercoat.

Finally, the chosen scalp positions must be turned into a texel density ρ. Based on the number of hairs generated by the Poisson disk process, it was decided that an array with a 40×40 element base would be used to store the fur texel density. The 40×40 base, which abuts to the scalp, identifies with the Poisson disk, so each cell of the array represents 0.025 units of width.

Each point within the Poisson disk was "snapped" to the nearest texel array cell.

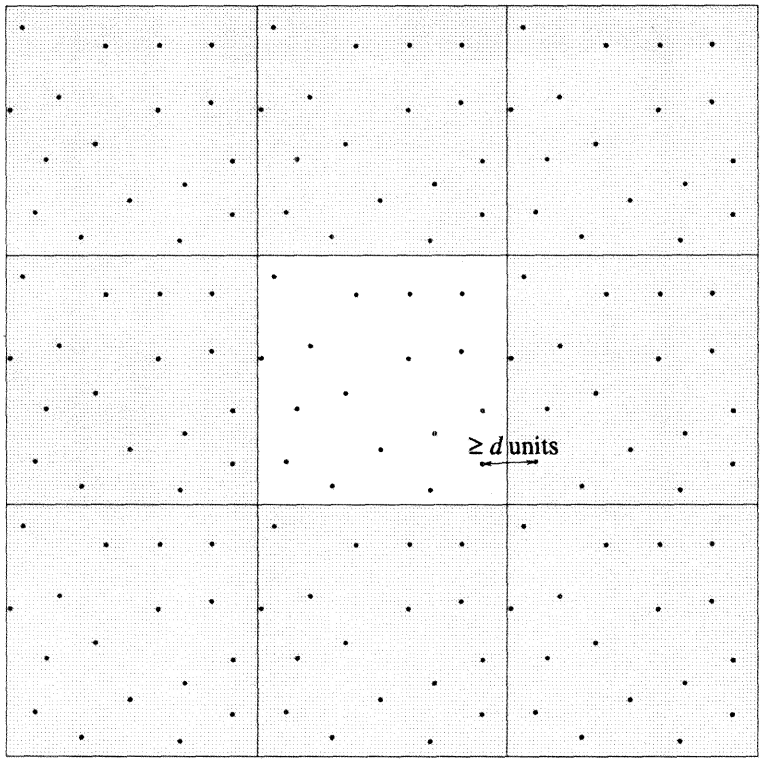


Figure 3.3: Poisson disk on a torus. A *Poisson disk on a torus* is a Poisson disk with the additional constraint that the minimum-distance criterion must be satisfied across the boundaries of the tiled unit square.

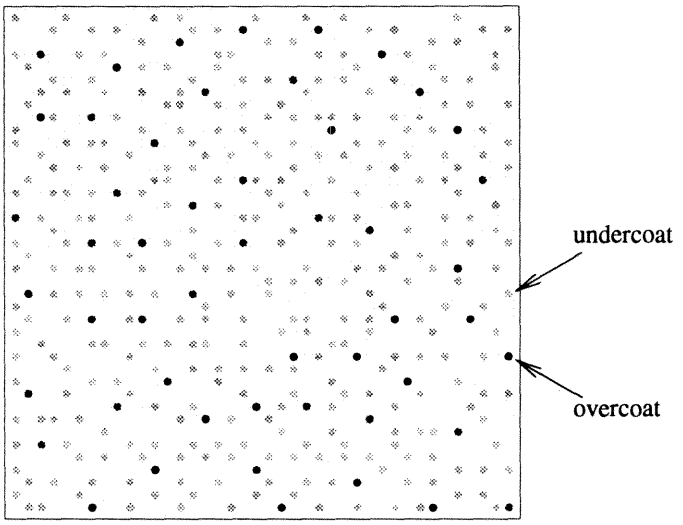


Figure 3.4: The base of the fur texel. The overcoat and undercoat are positioned in a Poisson disk pattern. The darkness of the dot indicates the length of the hair.

Any cell containing a hair was assigned the value 1. Other cells were assigned the value 0. Figure 3.4 illustrates the base of the fur texel used in the Teddy bear model. Darker dots represent overcoat hairs, while lighter dots represent undercoat hairs.

The fur texel array was created with 10 vertical cells, which were used to encode the length (or height) of each hair. Each texel array cell therefore represents 0.1 units of height. Each hair placed according to the Poisson disk distribution was assigned a random height. Undercoat hairs were assigned heights of either 0.5, 0.6, or 0.7 units, and overcoat hairs were assigned heights of 0.8 or 0.9. Wherever a hair was placed by the placement process, the vertical dimension of the texel array was filled with the value 1. The rest of the array was filled with 0.

Fur Texel Coordinate Frame

The coordinate frame for the fur texel is very simple. As presented in the next section, the fur BDRF is isotropic. The BDRF is a function of the vector tangent to the hair direction, but independent of the other two vectors in the coordinate frame. So, it is

necessary to specify only the tangent vector for the fur texel.

Because the design of the fur texel has all hairs pointing in a single direction, the tangent vector to the hairs will always point in a constant direction. If we define the plane containing the scalp to be the x-y plane, then the fur texel tangent vector is the constant vector (0,0,1). To formulate the constant vector (0,0,1) as a texel, we need two $1 \times 1 \times 1$ arrays named `zero` and `one`, whose single cells contain the values 0 and 1 respectively. Then the texels tangent can be specified as `(zero,zero,one)`.

Attenuation Coefficient

The value of α, which is a constant scalar value over the entire texel, is left unspecified. It is adjusted at rendering time (in a similar manner as the color of the objects is adjusted at rendering time), to obtain the specific appearance desired.

3.2 Fur Lighting Model

At a point $\mathbf{p}$ within a texel, the various texel values are determined by linearly interpolating the values of the array `density`. The interpolation, in combination with the texel rendering algorithm, has the effect of making each hair a finite thickness that is proportional to the size of a cell in the `density` array. Although the hairs have a finite thickness, the BDRF of the fur texel is designed (in an *ad hoc* fashion) to behave as we would expect the BDRF of an infinitely thin hair to behave.

Specifically, we make one important assumption about the reflection of light off thin hair: the light reflects uniformly in a radially symmetric fashion. Suppose an individual hair lies along the (vertical) z axis, and we are interested in the light reflecting from a point at the origin. Place the viewer at an angle of elevation ϕ, and place the light at an angle of elevation θ. Our desire to have radially symmetric light reflection from the hair means simply that the amount of reflected light depends only

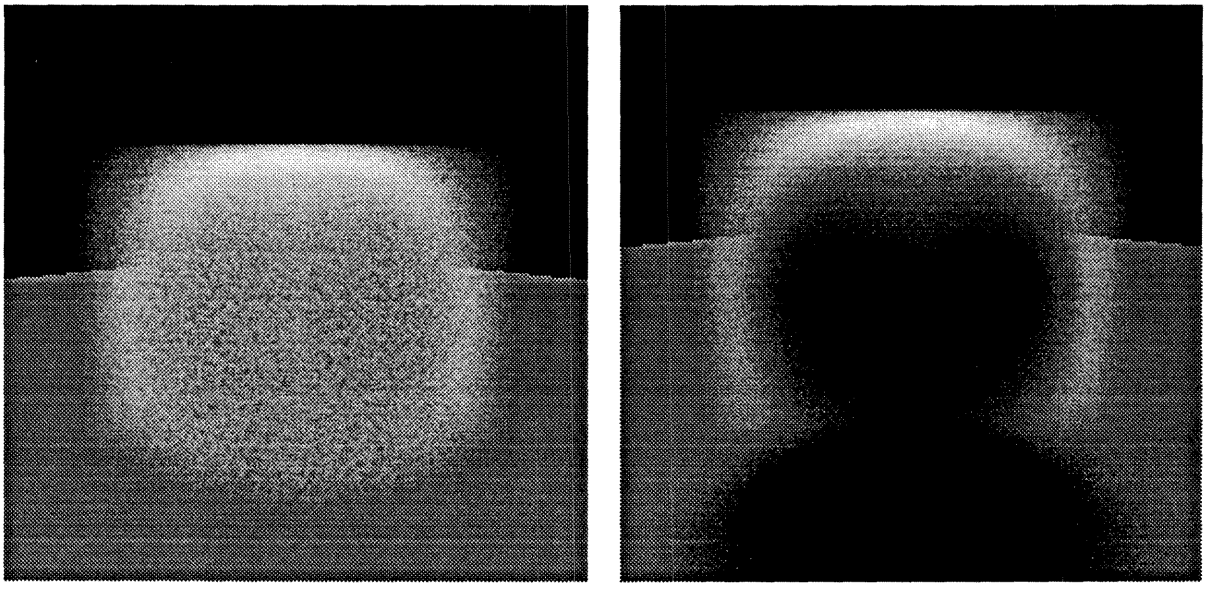
on ϕ and θ, and is independent of the azimuthal angles of the viewer and light.

The assumption of radial symmetry has two important effects on the fur BDRF. First, the mathematics of the model is simplified to a small degree, as we need not consider the azimuthal angles. Second, and much more important, the assumption of radial symmetry means that our hairs can be back lit. Whereas a traditional computer graphics object, illuminated from the rear, reflects no light to the viewer, our fur model will allow such light to be transmitted through the fur towards the eye. (Such transmitted light will, of course, be subject to the attenuation factors described in the previous chapter.)

Figure 3.5 shows a simple texel viewed from the illuminated side and from opposite the illuminated side. The ability of the fur BDRF to incorporate back lighting contributes a great deal to the realism of the images of the Teddy bear, which are presented at the close of this chapter. In fact, it is the back lighting effect that makes objects appear soft and fuzzy.

Aside from the property of radial symmetry, the BDRF of the hair is modeled in a fashion similar to traditional computer graphics BDRFs, as two independent components called the *diffuse* component and the *specular* component. The diffuse component of a BDRF specifies the overall light reflection characteristics of a material if the shininess of the material is discounted. The specular (mirror-like) component specifies the light reflected in the direction of the viewer, thereby contributing the appearance of shininess to the material.

Figure 3.6 illustrates the derivation of the diffuse component of the fur BDRF. At some test point $\mathbf{p}$ within the texel, we need to compute the amount of energy reflected from the light $\mathbf{l}$, off the hair, towards the viewer $\mathbf{e}$. The diffuse component of the BDRF ignores the position of the viewer. Furthermore, the previously stated simplification–that the BDRF is to be radially symmetric about each hair–implies



(a) (b)

Figure 3.5: Texel back lighting. (a) The illuminated side of the texel whose density ρ is the 3×3×3 array

0 0 0	0 0 0	0 0 0
0 0 0	0 k 0	0 0 0
0 0 0	0 0 0	0 0 0

. (b) The side opposite the illumination.

that, of the three vectors that make up the coordinate frame, only use of the tangent vector is appropriate.

In the traditional computer graphics diffuse reflection model, the amount of reflected energy is proportional to $\mathbf{n} \cdot \mathbf{l}$. The same will be true for our fur BDRF. If $\mathbf{l}'$ is the projection of $\mathbf{l}$ onto the plane normal to $\mathbf{t}$, then the diffuse component is simply $\Phi_d = \hat{l} \cdot \hat{l}'$. In terms of $\mathbf{t}$, which is perpendicular to $\mathbf{l}'$, the diffuse component is $\Phi_d = \sqrt{1 - (\hat{l} \cdot \hat{t})^2}$.

As a consistency check, notice that the formula for the diffuse component should be insensitive to the sign of $\mathbf{l}$. A negative $\mathbf{l}$ is equivalent to an $\mathbf{l}$ reflected through the normal plane. The formula for the diffuse component contains a square term, which enforces the desired behavior.

The formula for the specular component is a bit more complicated. In the Phong model, the specular component can be thought of as a measure of how close the view vector is to the normal reflection of the light vector. To compute the specular component, first reflect the light vector $\mathbf{l}$ about the normal $\mathbf{n}$, yielding $\mathbf{l}'$. The specular component is then $(\hat{e} \cdot \hat{l}')^s$. The power s controls the tightness of the specular highlight. Higher values of s will cause the highlight to be narrower.

Similarly, in the case of the specular component of the hair BDRF, we wish to measure the amount of coincidence of the normal reflection of the light vector $\mathbf{l}$ with the view vector $\mathbf{e}$, while maintaining radial symmetry about the hair. Referring to Figure 3.7, the cone R contains the reflection of light vector $\mathbf{l}$ about the hair normal vector. As we desire radial symmetry, we ignore the azimuth of the reflection vector, causing the reflection vector we are interested to lay somewhere on the cone R.

To measure the coincidence of the vector $\mathbf{e}$ with the reflection of the light, vector $\mathbf{e}'$ is the projection of view vector $\mathbf{e}$ onto the cone R shown in the figure. The specular component of the fur BDRF is then simply $\hat{e} \cdot \hat{e}'$.

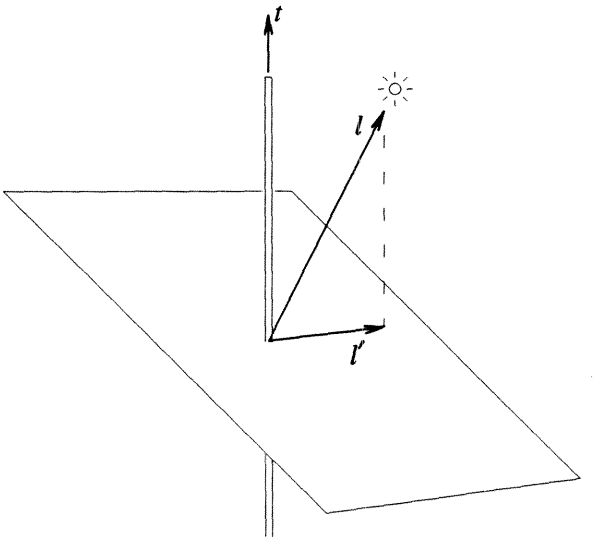


Figure 3.6: Diffuse component. The diffuse component of the hair BDRF is $\hat{l} \cdot \hat{l'}$. (The hat ^ indicates a unit vector.) Expressed in terms of t, the diffuse component is $\sqrt{1 - (\hat{l} \cdot \hat{t})^2}$.

To derive an expression for $\hat{e} \cdot \hat{e'}$ in terms of only $\mathbf{e}, \mathbf{t}$, and $\mathbf{l}$, we employ some trigonometric identities:

$$\Phi_s = \hat{e} \cdot \hat{e'} = \cos(\phi - \theta) = \cos\phi\cos\theta + \sin\phi\sin\theta$$

$$= (-\hat{e} \cdot \hat{t})(\hat{l} \cdot \hat{t}) + \sqrt{1 - (\hat{e} \cdot \hat{t})^2}\sqrt{1 - (\hat{l} \cdot \hat{t})^2}.$$

In the specular case, unlike the diffuse case, the direction of the light vector $\mathbf{l}$ is significant. Rather than canceling, a negative sign in $\mathbf{l}$ causes the value Φ_s to change sign.

The combination of the radially-symmetric BDRF and the density derived from the Poisson disk completes the specification of the fur texel. Figure 3.8 shows the fur texel and how it reacts to back lighting conditions.

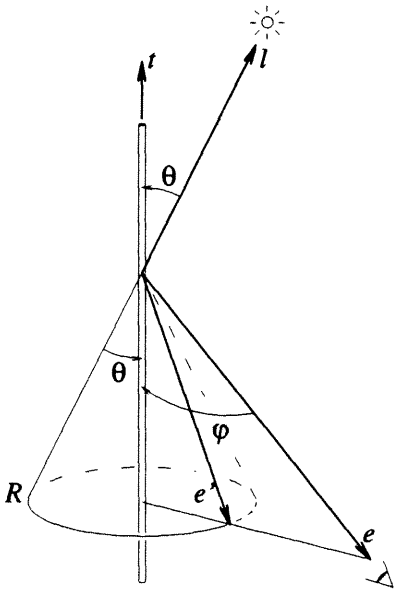
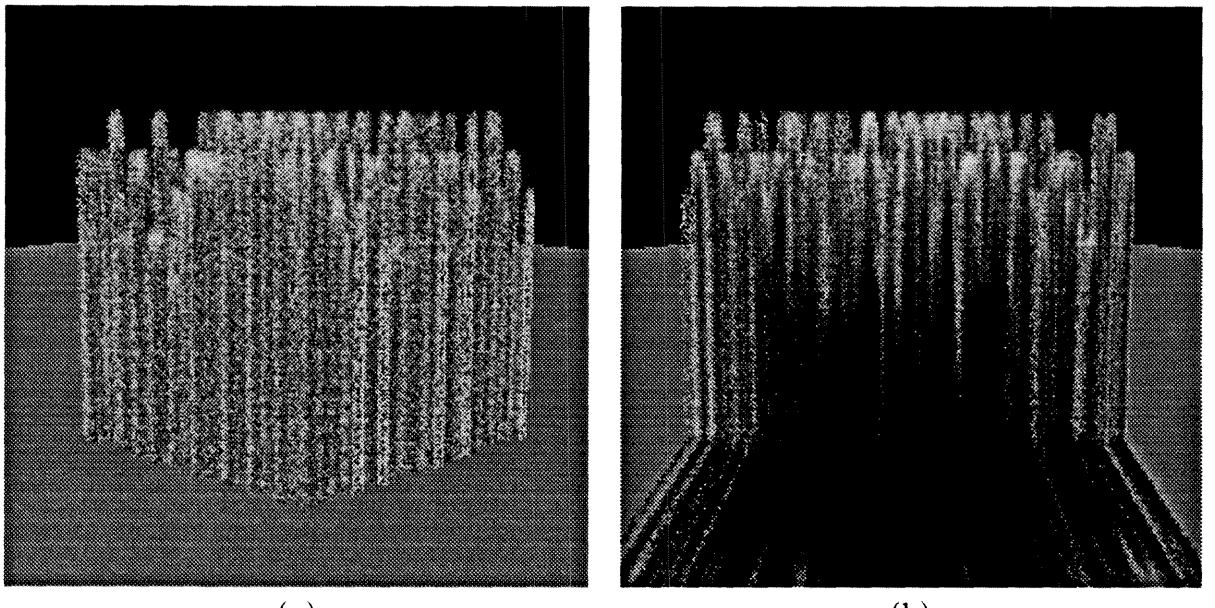


Figure 3.7: Specular component. The specular component of the hair BDRF is $\hat{e} \cdot \hat{e'}$.



(a) (b)

Figure 3.8: Fur texel. The fur texel incorporating the Poisson disk overcoat and undercoat of Figure 3.4, and illuminated using the radially-symmetric fur BDRF. (a) The illuminated side of the texel. (b) The side opposite the illumination.

3.3 Trilinear Mapping

The previous section defines the fur texel, which, on its own, is only a modeling primitive. To create a coat of fur, the texel must be mapped onto the surface of a parametric model. The fur texels will be placed contiguously so as to completely tile the surface of the model. The fur texel has been designed, via a Poisson disk on a torus, so that no seams will show at the boundary between texels.

A parametric model is the mapping of a square parameter space, parameterized by u and v, onto a surface in $\mathbb{R}^3$, and is specified as a mapping function $\mathbf{f}(u,v) : \mathbb{R}^2 \mapsto \mathbb{R}^3$. We chop the parameter space along lines of constant u and v values into a *parametric grid* of *parameter squares.* The function $\mathbf{f}(u,v)$ transforms the parameter grid onto the surface of the three-dimensional parametric model. It is over this grid that we wish to tile the fur texels.

The fur texels are mapped onto the surface of the parametric model using $\mathbf{f}(u,v)$ in the same way that the parameter squares are mapped onto the surface. The base (scalp) of the fur texel is identified with a parameter square with corners at the coordinates $(u_i, v_i), (u_{i+1}, v_i), (u_{i+1}, v_{i+1})$, and (u_i, v_{i+1}). These four points are mapped into $\mathbb{R}^3$, taking the fur texel along with them. The mapping of a fur texel onto a parameter square on the surface of the parametric model is illustrated in Figure 3.9. The gray square in Figure 3.9(a) indicates a parameter square with which we have identified the base of the fur texel. Figure 3.9(b) shows the parametric model with a texel "glued" onto one of the parametric squares on its surface.

The process of choosing a parametric grid determines the final positions of the base corners of every texel in the fur coat. However, the position of the rest of the points within the texel have not been specified. We must construct a 1-to-1 identification between points on the base of our texel and points within the parameter square after

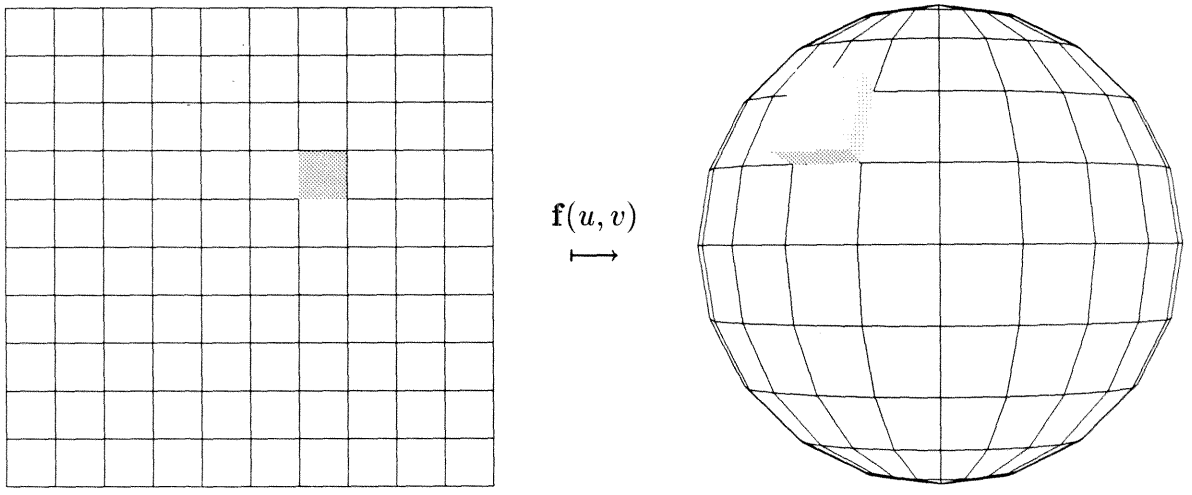


Figure 3.9: Parameter squares. The parameter space (a) is chopped into parameter squares. The fur texel is identified with a particular parameter square and is mapped onto the surface of the parametric model (b).

it has been mapped into $\mathbb{R}^3$.

In mapping of the texel onto the surface of the parametric model, the planar rectangular texel base has been mapped into a more general shape which is specified by four arbitrary points in $\mathbb{R}^3$. Any four non-colinear points p_1, p_2, p_3, p_4 in $\mathbb{R}^3$ uniquely define a surface called a *bilinear patch*, as defined by the parametric surface

$$S(u,v) = (1-v)\left[(1-u)p_1 + up_2\right] + v\left[(1-u)p_4 + up_3\right].$$

We identify each point (u, v) on the base of the texel with the point $\mathbf{S}(u, v)$ in $\mathbb{R}^3$. Now the positions of every point on the base of each fur texel has been uniquely specified.

Next we need to determine the 3-space positions of the rest of the points in the transformed texel. This task is not as easy as the previous one; the parametric model, which is a two-dimensional surface embedded in $\mathbb{R}^3$, contains only enough information to specify positions for the two-dimensional base of the texels. To determine a unique position for the rest of the texel, additional constraints beyond the parametric

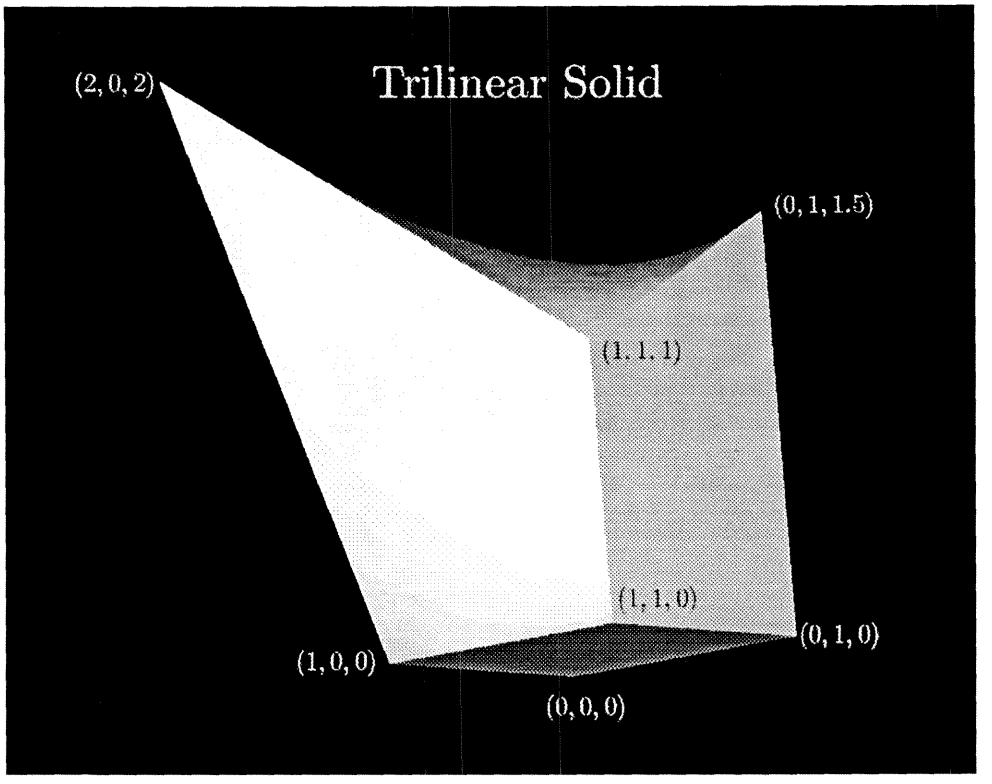


Figure 3.10: A trilinear solid. Eight ordered points in 3-space, uniquely determine a trilinear solid.

mapping function need to be specified.

We start by pointing the hairs of the texel in a direction normal to the scalp. To do this, unit vectors which are normal to the parametric surface are placed at the corners of the base of the texel which have already been mapped into $\mathbb{R}^3$. The tips of the normal vectors specify new positions in $\mathbb{R}^3$ to which the remaining corners of the texel are mapped.

An obvious feature of the normal-based mapping is that texels which start out sharing faces in parameter space continue to share common corners in 3-space.

As with the base of the texel, we use bilinear interpolation to specify a unique mapping between points on the faces of the warped texel and points on the faces of the original unwarped texel. Figure 3.10 shows the shape of one such warped texel.

To complete the mapping between texel points in the two spaces, interpolate between corresponding points on opposite faces of the texel. The resulting trilinear mapping provides a homomorphism between the texel in the two spaces.

The trilinear mapping herein described suffers from a problem commonly encountered in computer aided design when computing an offset surface. In regions where a parametric surface is concave, a large offset causes the offset surface to buckle back on itself. The problem is circumvented in this paper by avoiding parametric surfaces with sharply concave regions.

Combing the Fur

The final step in defining the mapping of the texel from parameter space to world space involves a relaxation of the constraints that positioned the top four corners of the texel. The process called *combing the fur* entails adding perturbations to the positions of each of the upper corners of the warped texels. Figure 3.11 shows the effect of applying the fur texel to the surface of a torus. The positions of the upper corners of the texels were perturbed by adding a tangential component.

A more complicated perturbation process is used in combing the fur on the Teddy bear. A Fourier map is created to generate a perturbation vector $\mathbf{r}_{uv}$ at each point (u, v) in parameter space:

$$\mathbf{r}_{uv} = (\sum_i \sin \zeta_i^u u + \phi_i^u, \sum_i \sin \zeta_i^v v + \phi_i^v).$$

The designer specifies a sequence of (frequency,phase) pairs $\{(\zeta_1, \phi_1), (\zeta_2, \phi_2), \ldots\}$ for both the u and v coordinates. At each grid point $\mathbf{r}_{uv}$ is calculated and mapped into $\mathbb{R}^3$, where it becomes a tangent vector to the parametric surface. The perturbation is then added to the normal vector, and the result specifies a new position for the corresponding texel corner. Upon viewing the results, the designer can then change the Fourier coefficients, and a new combing of the fur will result.

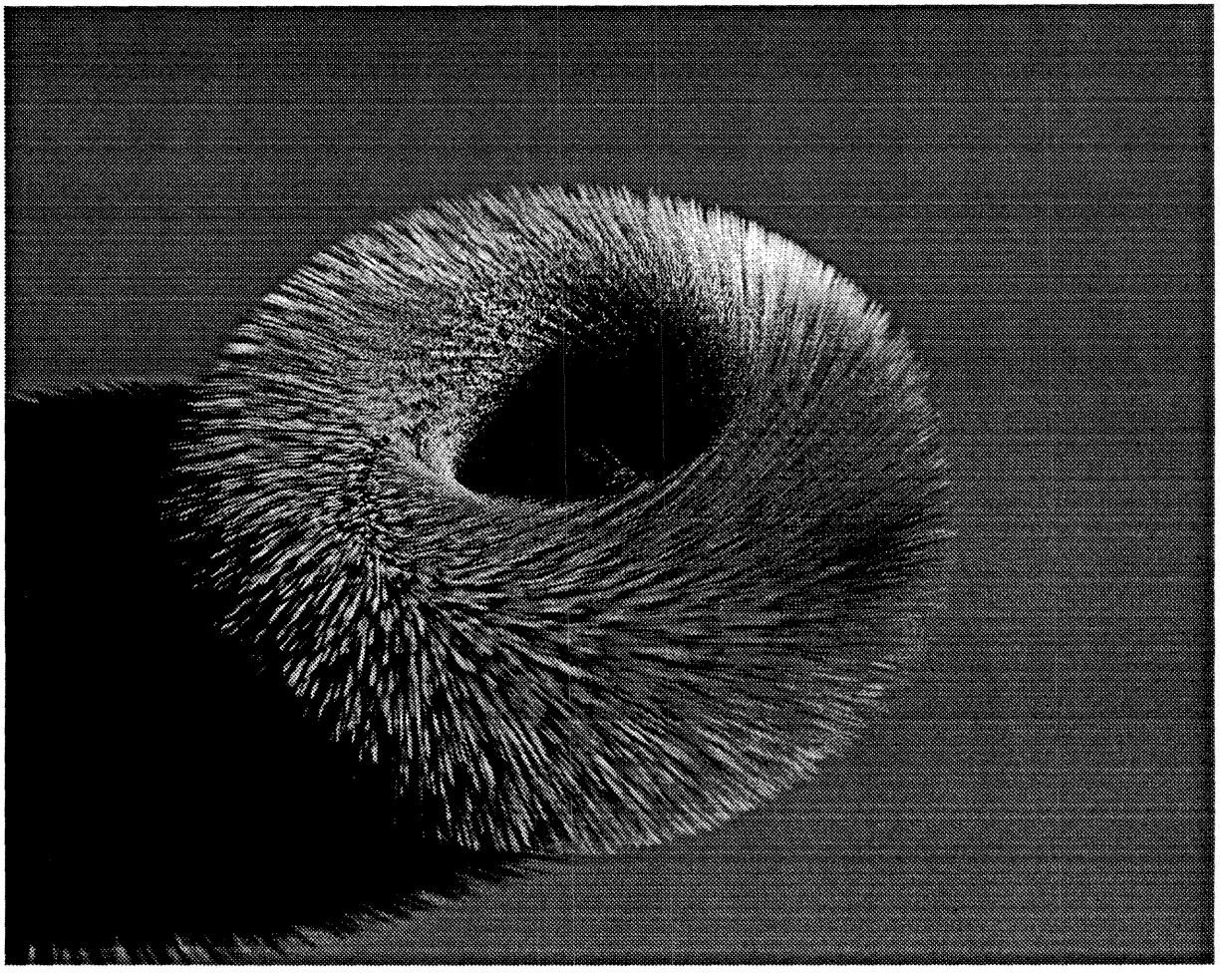


Figure 3.11: A furry torus. The bottom of the image is to the right of the page. Notice the dark spot on the right side of the image, where the hairs point directly at the light source.

In practice, it is found that the Fourier map perturbation process is very difficult to control. An alternative, more appealing method for combing the fur is now presented. (This method has not been implemented.)

The new fur combing is similar to the Fourier map method in that a perturbation field is specified in parameter space. Unlike the Fourier map method, the perturbation is specified in a completely different manner. Using the mouse, the designer draws a collection of rays $\{\mathbf{a}_1 t + \mathbf{b}_1, \mathbf{a}_2 t + \mathbf{b}_2, \ldots\}$ on a square that represents the parameter space. The direction $\hat{b}$ and length $||\mathbf{b}||$ of the rays are both significant, and roughly represent strokes of the comb over the fur in the neighborhood of the origin of each ray $\mathbf{a}$. The direction of a ray represents the direction of that stroke, and the length specifies the strength of that stroke.

The stroke information can be transformed into perturbation information by computing a weighted sum of the stroke vectors. The perturbation at a given position (u, v) is given by the vector sum

$$\mathbf{r}_{uv} = \sum_i \frac{\mathbf{b}_i}{||(u,v) - \mathbf{a}||^2}.$$

Ray–Bilinear Patch Intersection

A bilinear patch is a surface of degree 2. Therefore, there will be at most two intersections of a ray with a patch. We derive an analytic solution to the intersection, taking care to ensure that the solution avoids degenerate cases.

A ray is defined by the equation $\mathbf{R}(t) = \mathbf{a}t + \mathbf{b}$ with $0 \leq t$. The vector quantities $\mathbf{a}$ and $\mathbf{b}$ specify the origin and direction cosines of the ray. A bilinear patch is an equation of the form $\mathbf{P}(u, v) = \mathbf{A}uv + \mathbf{B}u + \mathbf{C}v + \mathbf{D}$ with $0 \leq u \leq 1$ and $0 \leq v \leq 1$ where $\mathbf{A}, \mathbf{B}, \mathbf{C}$, and $\mathbf{D}$ are also vector quantities.

To compute the values of t, u, and v, for which $\mathbf{R}(t) = \mathbf{P}(u, v)$, expand the vector

equation into scalar equations into three equations of the form

$$A_1uv + B_1u + C_1v + D_1t + E_1 = 0,$$

$$A_2uv + B_2u + C_2v + D_2t + E_2 = 0,$$

and

$$A_3uv + B_3u + C_3v + D_3t + E_3 = 0.$$

These equations should be reordered so that $D_1 \geq D_2 \geq D_3$. This will assure that, in the case of a patch aligned with an axis, the denominators in the equations that follow will be reasonable (thereby avoiding floating point overflows).

Solving the first equation for t, yields

$$t = -\frac{A_xuv + B_xu + C_xv + E_x}{D_x}.$$

The value for t can be substituted into the remaining two equations thereby removing all references to t. This leaves two equations of the form

$$F_2uv + G_2u + H_2v + I_2 = 0 \tag{3.1}$$

and

$$F_3uv + G_3u + H_3v + I_3 = 0. \tag{3.2}$$

These two equations can be multiplied by F_3 and F_2 respectively, and the uv terms can be eliminated, giving a linear equation relating u and v. Solving for u and back substituting into Equation 3.1 or Equation 3.2 results in a quadratic equation in v. Once v is determined, u and t quickly follow.

When solving the quadratic equation (of the form $ax^2 + bx + c = 0$), there is a possibility that the coefficient a on the square term may be very small. This could occur, for example, when the four points of the bilinear patch are coplanar. Since we

are looking only for values of $0 \leq u \leq 1$, we can determine if a is too small using the equation

$$b + sgn(b)\sqrt{b^2 - 4ac} < 2a.$$

If the inequality fails to hold, then any root would be outside the range $-1 \leq u \leq 1$ and need not be computed. This and similar tests will help avoid floating point overflows.

The preceding solution yields up to two intersections, (t_1, u_1, v_1) and (t_2, u_2, v_2), of ray with the bilinear patch.

Ray–Trilinear Cube Intersection

A trilinear cube is composed of six bilinear patches. The intersection of a ray $\mathbf{R}(t)$ with a trilinear cube is calculated by intersecting $\mathbf{R}$ with each of the faces of the cube. The intersections are then sorted according to their distance along $\mathbf{R}$.

If the ray just happens to nick the corner of the trilinear cube, then only one intersection might be found. When this happens, the intersection is ignored. When the ray intersects with the trilinear cube at exactly two points, the texel integration is performed between them.

Given that the faces of the cube are quadratics, it is possible to get more than two intersections. When more than two intersections are computed, all intervening intersections are ignored. The texel integration is performed between the nearest and the furthest of the intersection points.

For example, suppose one side of the trilinear cube sags inward. The ray may enter the cube via a perpendicular face at $t = t_1$, intersect with the sagging face twice at $t = t_2$ and $t = t_3$, and leave the cube through the opposite perpendicular face at $t = t_4$, producing four intersections. Although the ray entered the cube at $t = t_1$, exited at $t = t_2$, entered again at $t = t_3$ and exited for good at $t = t_4$, for the sake

of simplicity, we ignore the inner two intersections $t = t_2$ and $t = t_3$ and assume that the ray stayed within the cube over the entire interval $[t_1, t_4]$.

To simplify implementation, the texel integration occurs along a straight line between the entry and exit points *in texel space.* In other words, once the entry and exit points have been determined and mapped back to texel space, the integration proceeds ignoring the fact that the texel had been trilinearly warped. In order to traverse a linear path in texel space, the ray traverses a curved path within the warped texel in world space. The approximation, while potentially grossly incorrect, produces reasonable results.

3.4 Parametric Teddy Bear Model

Having defined a fur texel modeling primitive and a procedure to map that primitive onto the surface of a parametric model, all that is left to create our model of a Teddy bear is a parametric model of the scalp of the Teddy bear. The Teddy bear parametric model was created by Snyder and Kajiya using a technique called *generative modeling*, which has been described by Snyder in [33]. Here we need only a brief description of the technique as it applies to the creation of the Teddy bear model.

Generative modeling provides a programming language in which the user expresses the geometric interaction of various generative modeling primitives. The main data type in the generative model is a manifold, declared with `MAN`. Figure 3.12 shows one such program `bear_leg` used to create the leg of the Teddy bear. The program `bear_leg` operates on three two-dimensional curves, `legtop`, `legbot`, and `legcross`, which are created separately using a curve editing program (and stored in disk files named `*.crv`).

At the heart of `bear_leg` are the two lines which begin with `MAN mat = ...` and `MAN leg = ....` The `mat` manifold is a linear array (created with the `@(...)`

```
MAN bear_leg()
{
   MAN u = m_x(0);
   MAN v = m_x(1);

   MAN p1 = m_crv("legtop.crv",u);
   MAN p2 = m_crv("legbot.crv",u);

   MAN prof1 = m_resample_x(p1);
   MAN prof2 = m_resample_x(p2);

   MAN cross = m_crv("legcross.crv",v);
   MAN xmin,xmax,ymin,ymax;
   curve_extent(cross,&xmin,&xmax,&ymin.&ymax);

   MAN mat = @(m_inter_scale(xmin,xmax,prof1[1],prof2[1]),
               0,0,m_inter_trans(xmin,xmax,prof1[1],prof2[1]),

               0,m_inter_scale(ymin,ymax,prof2[1],prof1[1]),0,0,
               0,0,1,prof1[0],
               0,0,0,1);

   MAN leg = m_transform3d(cross,m_matrix(mat,4,4));
   return leg;
}
```

Figure 3.12: Generative modeling program example. The program for the generative model for the Teddy bear leg combines three two-dimensional curves to produce a three-dimensional parametric model. The three curves are included as Figure 3.13.

operator) containing 16 entries, which form a 4×4 transformation matrix. The effect of the matrix is to scale and translate another manifold so that one of its axes always touches the curves `legtop` and `legbot`.

The `leg` manifold is created by actually applying the `mat` matrix to the `legcross` curve. Refer to curves in Figure 3.13. The curves `legtop` and `legtop` form a profile which controls the scaling of the cross section curve `legcross`.

In a similar fashion, the Teddy bear's head, ears, nose, body, and arms were created. The resulting Teddy bear parametric model is presented in Figure 3.18.

`legtop` and `legbot` `legcross`

Figure 3.13: Curves used in modeling the bear's legs.

`earcross`

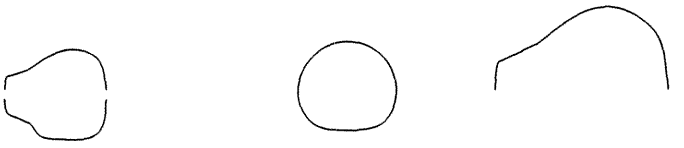
Figure 3.14: Curve used in modeling the bear's ear.

`bodyprof` `bodycross`

Figure 3.15: Curves used in modeling the bear's body.

`armtop` and `armbot` `armcross` `armside`

Figure 3.16: Curves used in modeling the bear's arm.



`headscl1` and `headscl2` `headcross` `headscl3`

Figure 3.17: Curves used in modeling the bear's head.

3.5 Results

By applying the fur texels to the Teddy bear model in the manner described in the previous section, the Teddy bears in Figure 3.19 and Figure 3.20 were produced.

The only difference between the two Teddy bears is the combing of the hair. The hair on the Teddy bear in Figure 3.19 (Lilac) was created before the rendering program was finished. Once the renderer was completed, the hair in Figure 3.20 (Herbert) was designed to more closely match our desires. The finished renderer provided good feedback for choosing the Fourier map coefficients.

3.6 Chapter Summary

As an example of using texels as computer graphics modeling primitives, the process of creating a computer graphics model of a Teddy bear was described. The process involved three steps: the creation of a fur texel, the creation of a Teddy bear parametric model, and the mapping of the fur texel onto the parametric model.

Creating the fur texel involved two goals: a convincing visual appearance was desired, and fur texel needed to tile the infinite plane without visible seams at the boundaries. To achieve these goals, the fur texel was modeled as a Poisson disk on a torus. Furthermore, the fur texel was designed to have an overcoat and an undercoat,

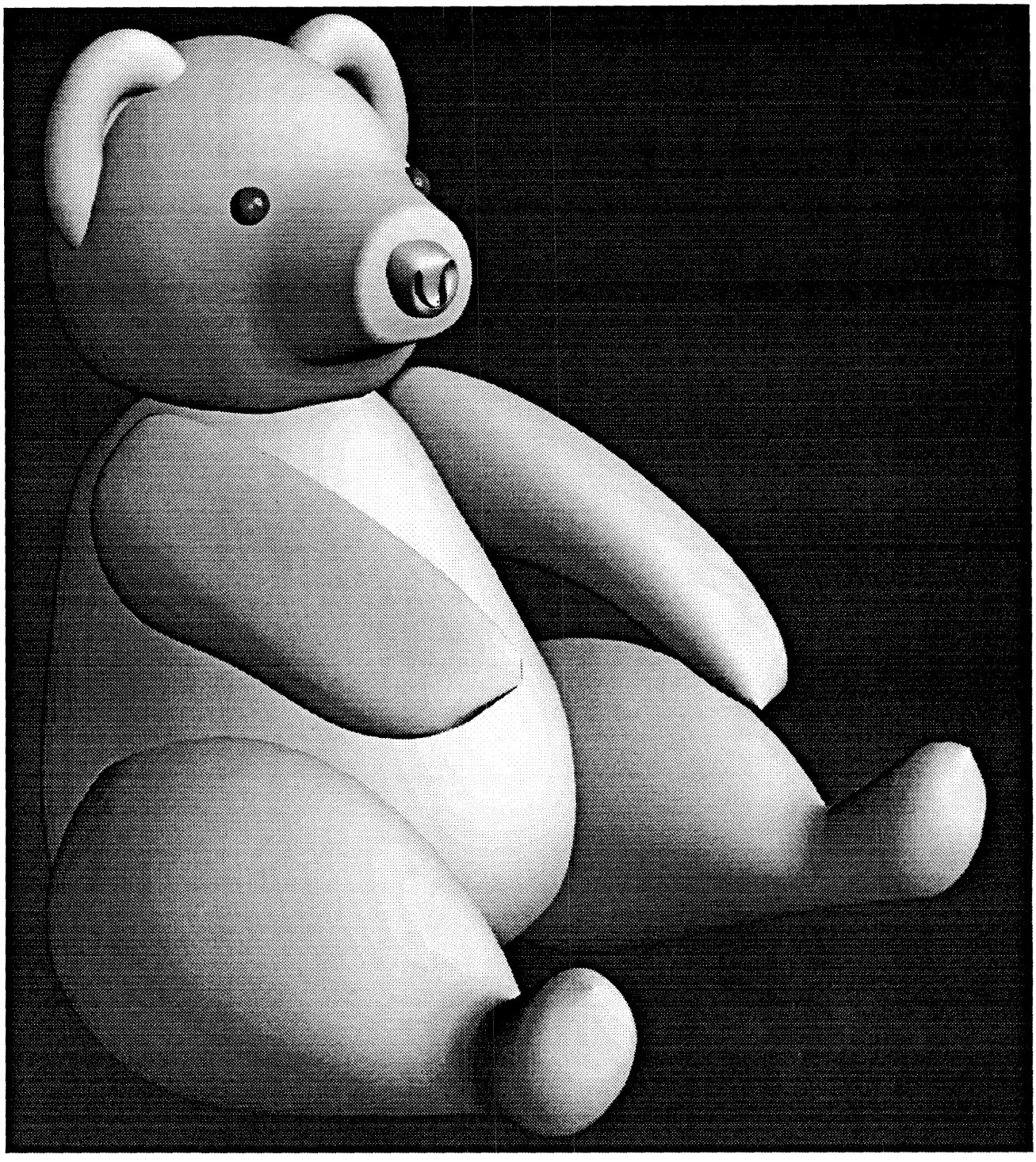


Figure 3.18: Bare Teddy bear. The Teddy bear "scalp" was created using a technique called *generative modeling.*

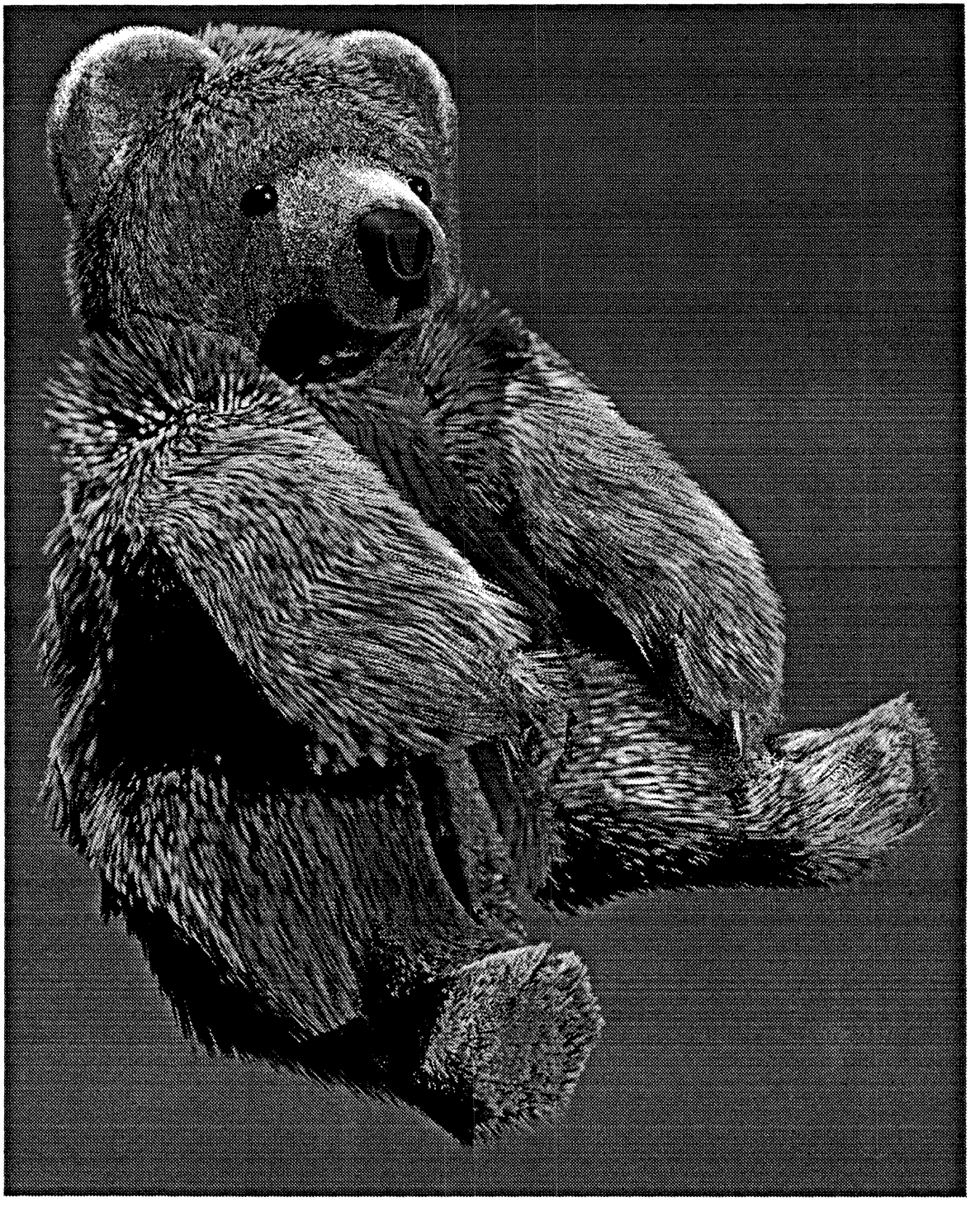


Figure 3.19: Lilac. Lilac the Bear was the first bear picture computed after all the pieces of the problem fell together.

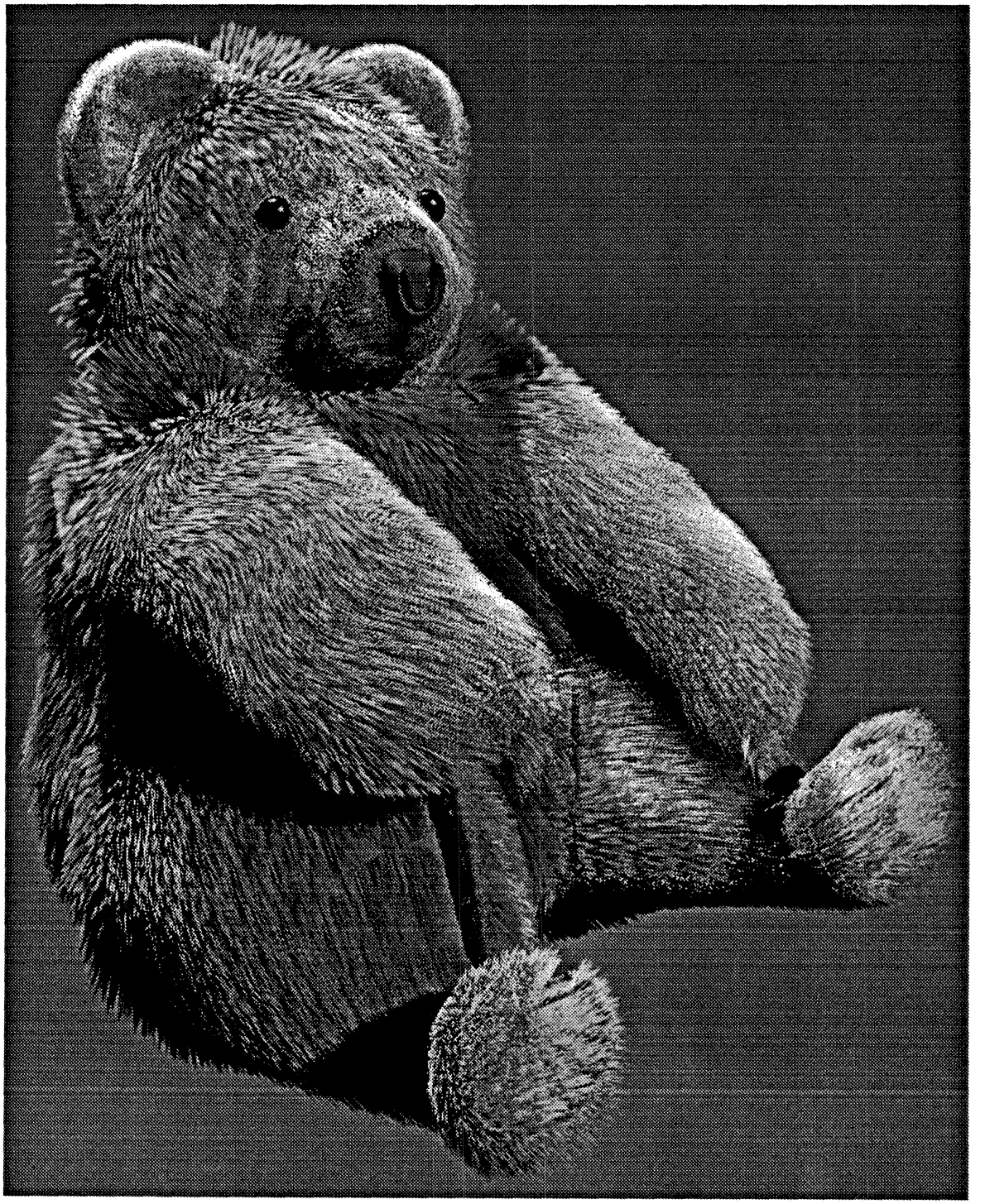


Figure 3.20: Herbert. Herbert the Bear resulted from tuning many parameters. The main difference between Lilac and Herbert is that, in combing Herbert's hair, we had a fully functional renderer and, thereby, had good feedback which was useful in choosing Fourier map coefficients.

just as real fur has.

In conjunction with the fur texel, a fur BDRF was created. The fur BDRF was designed to be radially symmetric so that light would be scattered towards the viewer even when the texel was being illuminated from behind, relative to the viewer. Such back lighting effects are largely responsible for the fuzzy appearance of the fur texel.

The fur texel was mapped from parameter space to world space using a trilinear mapping, which affords just the right degree of freedom to uniquely specify the requirements. To enable a designer to control the position of the fur, two methods of "combing the fur" were described.

An analytic ray-trilinear intersection algorithm was described, as was the method for mapping world-space intersections into texel space.

Finally, a furry torus and two Teddy bear images were presented.

Chapter 4

Texels as Models: A Cloth Texel

In Chapter 3 we created a fur texel which was used as a modeling primitive. The fur texel was combined with other computer graphics primitives into a Teddy bear model. This chapter takes an opposite approach: we start by constructing a geometric model of a patch of cloth and finish by converting that geometric model into a cloth texel. In this chapter, the texel is used not as a modeling primitive, but as the entire computer graphics model.

Construction of the cloth texel serves as an example of the texel creation process as well as an example of a texel more complicated than the fur texel presented in Chapter 3. The same cloth texel will also play an important role in Chapter 5, as an example of a microscopically modeled surface from which the BDRF will be extracted.

The use of texels is not without precedent in computer graphics. The scientific visualization community has worked with a restricted form of texels, called *volume densities*, for many years. The issue here is one of how the texels are created. Previously volume densities have been created in three ways:

- via measurement: CT and MRI scanners measure volume data directly,
- via simulation: physical simulations create volume data using physical models, and

- via recursive functions: various cloud densities have been created using iterative or recursive functions [18] [21] [23] [30].

The cloth texel presented here illustrates a new, fourth technique for creating texels:

- *scan conversion*: a geometric solid model is sampled directly into texel data.

4.1 Weaving a Cloth Model

There are many different types of cloth, such as woven, knit, felt, and Malino, each of which acquires particular visual characteristics from the manner of its construction. The cloth we choose to model has a very simple structure: a plain weave which is composed of two sets of perpendicular threads. The methods used here to model this simple weave are equally applicable to more complex weaves.

The set of threads affixed to the loom is called the *warp*. The other set of threads, which is woven between the warp, is called the *weft*, also know as *woof* or *filling* [27]. For the simple cloth model presented here, we assume that the weft threads remain perfectly straight during the weaving process. The threads of the warp weave alternately above or below each thread of the weft. Warp threads alternate in phase so that, if one warp thread goes above a weft thread, then the next warp thread goes below the same weft thread.

Cloth Design Using Relaxation

As the first step in the design of the cloth texel, we must determine the placement of the weft and warp. We have specified that the weft is perfectly straight. Therefore the warp threads must be modeled so that it suitably weaves around the weft threads.

The warp placement problem is treated as a constrained relaxation problem. A single warp thread is modeled as a sequence of nodes $\mathbf{n}_i$ connected by springs. The

warp thread is stretched between weft threads, and an iterated relaxation is performed.

As long as the boundary conditions are treated with the proper symmetry constraints, the simulation can be restricted to half of the region between two adjacent weft threads. Figure 4.1 illustrates the configuration of this relaxation problem. The warp thread begins in an initial configuration as shown by the straight line.

At each step in the relaxation, each node position $\mathbf{n}_i$ is adjusted by an amount $\mathbf{r}_i$, which is a function of the length of the springs connected to node $\mathbf{n}_i$ as well as the distance between $\mathbf{n_i}$ and the weft thread.

When stretched beyond their rest length l, the springs exert linear force. The weft thread exerts exponential force in a radial direction, but only if a node of the warp contacts the weft thread, which is of thickness t.

Assuming that all springs are stretched beyond their rest state and the weft thread is at the origin, the node perturbation function is

$$\begin{aligned}\mathbf{r}_i &= A\left[(||\mathbf{n}_{i-1}-\mathbf{n}_i||-l)\frac{\mathbf{n}_{i-1}-\mathbf{n}_i}{||\mathbf{n}_{i-1}-\mathbf{n}_i||}+(||\mathbf{n}_{i+1}-\mathbf{n}_i||-l)\frac{\mathbf{n}_{i+1}-\mathbf{n}_i}{||\mathbf{n}_{i+1}-\mathbf{n}_i||}\right] \\ &\quad +B(t>||\mathbf{n}_i||)e^{(t-||\mathbf{n}_i||)}\frac{\mathbf{n}_i}{||\mathbf{n}||}. \qquad (4.1)\end{aligned}$$

The term $t > ||\mathbf{n}_i||$ evaluates to either 1 or 0, depending on the validity of the inequality. The term is included to nullify the attractive force between the warp and weft threads that would occur when the warp and weft do not make contact.

The coefficients A and B adjust the relative strengths of the two forces. Their specific values are unimportant; they should be assigned values small enough to cause the relaxation to run slowly. Otherwise oscillations will occur, causing the time to convergence to increase.

Equation 4.1 must be modified to treat the left boundary symmetrically and the right boundary antisymmetrically. Assuming that the left-most node in the relaxation

is $\mathbf{n}_1$ and the right-most node is $\mathbf{n}_N$, that the components of $\mathbf{n}_{N-1}$ are (x_{N-1}, y_{N-1}), and that the coordinate of the right-most weft thread is $(2, 0)$, then the boundary conditions can be handled by the assignments

$$\mathbf{n}_0 = -\mathbf{n}_2$$

and

$$\mathbf{n}_{N+1} = (x_{N-1}, 1 - y_{N-1}).$$

Figure 4.1 shows eight iterations of the relaxation process, after which the system reaches a steady state. Figure 4.2 shows the resulting solution appropriately reflected to create the final warp thread configuration for our plain weave example. As a last step in the geometry modeling process, the weft thread from the relaxation result is thickened, and the warf threads are thinned, as illustrated in Figure 4.3. (More accurate results would be achieved if the relaxation were performed using the final thread thicknesses. This approximation is considered to yield minimal impact on the results and is used here to simplify the implementation.)

The relaxation technique presented in this section can be expanded easily to two dimensions and can be used to model more complex weaves as well.

4.2 Texel Creation by Scan Conversion

Once the position of the threads is determined, a texel can be created. The cloth texel is more complicated than the fur texel in that it must contain coordinate frame information as well as density information. To render the cloth texel, we use the radially-symmetric lighting model introduced in the Chapter 3. Because that BDRF involves only the tangent vector, only the tangent vector will be computed and stored with the texel.

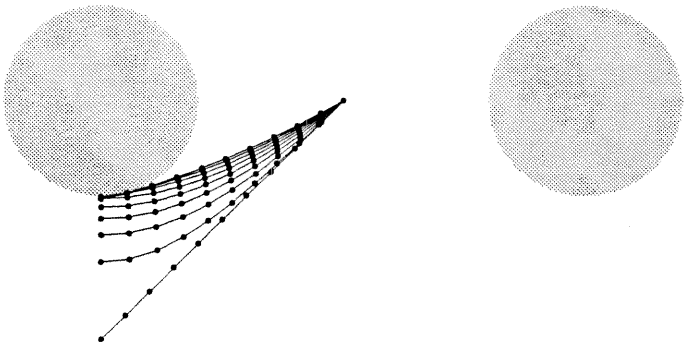


Figure 4.1: Thread placement by relaxation. The gray circles are weft threads. The straight line is the initial position of the warp thread before relaxation. The succession of curves shows the results of eight iterations of the relaxation process.

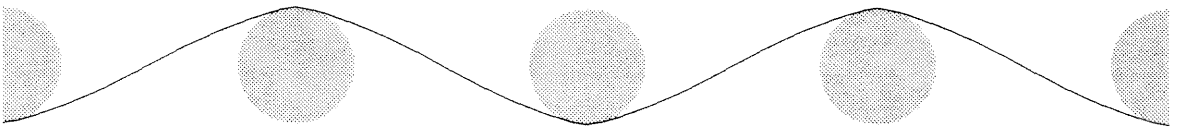


Figure 4.2: Relaxation results replicated. The gray circles are weft threads. The line shows eight quarter cycles of the warp thread.

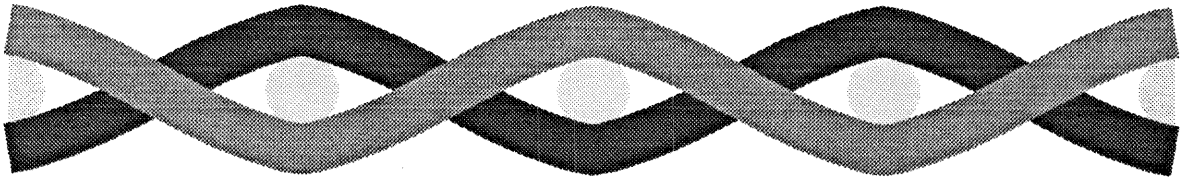


Figure 4.3: Relaxation results showing two adjacent warp threads at their final thickness. The thickness of the weft threads is reduced proportionally.

The cloth texel must be large enough to contain a full cycle of the weave pattern in both the warp and weft directions. Therefore, the texel must contain an integral multiple of at least two warp threads and two weft threads.

The cloth texel must contain enough samples so that the geometry of the model is preserved. Based on the structure of the weave and after experimenting with several sampling frequencies, we decide that the cloth texel can be well represented in an array roughly 80 cells on each side. (It is wise to overestimate the size of the array when scan converting.) The next section will discuss a method for resizing the array after the scan conversion is complete.

The process of converting our geometric thread model into a cloth model involves scanning through every cell in the texel and determining if that cell falls within one of the threads. If so, the density for that cell is set to 1. Also, the thread tangent is computed and stored as the tangent for that cell.

If it is determined that no thread passes through the cell, then the cell occupies empty space, and the density for that cell is set to 0. If the density of the cell is 0, the tangent value need not be specified.

In designing the texel scan conversion algorithm, it is important to keep in mind the texel boundary constraints discussion of Chapter 2, i.e., the value assigned to a texel cell represents the value of the texel at the corners, rather than at the center of the texel cells. With this constraint in mind, the cloth texel can be constructed so that it is able to abut with itself seamlessly.

4.3 Antialiasing and the Texel Editor

The scan conversion process turns a geometry model into a texel. If the resulting texel is not exactly as desired, it is always possible to modify the scan conversion process to produce different results. Such modifications can complicate an otherwise

simple piece of code.

As an analogy, consider a computer graphics renderer. It is the responsibility of the renderer to transform a three-dimensional model into a two-dimensional image. Is it also the responsibility of the renderer to anti-alias? To composite? To resize, scale, or gamma correct? The answer is, sometimes yes and sometimes no. In practice, it is useful to have another program, often called an *image processing program*, which operates exclusively on images.

Likewise, it is very useful to have a program called a *texel processing program* or *texel editor*. The texel editor can perform many useful operations. For example, the texel produced by the scan conversion process is a certain size, 80×80, which was chosen to generously represent the features in the texel. Because the amount of data generated by the sampling process is unwieldy, the texel editor is used to filter the cloth texel down to roughly 20 cells on a side.

Resizing the texel changes the number of cells and modifies the contents of each cell. When several cells are combined into a single cell, the various components of the texel are simply averaged coordinate by coordinate. For example, if two adjacent cells contain tangent vectors that point in different directions, the combined cell will contain a tangent vector that points in a direction which is in some way a compromise between the two original vectors.

A few useful operations that the texel editor can perform are listed here.

- Scaling. It is often useful to scale the density component of a texel by a scalar.
- Rotating and mirroring. Rotating a texel by 90 degrees is particularly useful. Rotations by arbitrary angles involves a great deal of resampling, but will be useful for more-sophisticated texel processing operations.
- Resizing. It is necessary to consider texel boundary constraints when resizing

a texel. In many cases, such as with texel abutment, it is desirable to preserve the actual value of the boundary cells of a texel.

- Concatenation. A new texel can be created by abutting two texels. The process might not make sense unless the two texels share common values along the face where they abut. The resulting texel will contain one fewer cell along the axis of the abutment.

4.4 Results

The cloth texel resulting from the relaxation, scan conversion, and texel editing is shown in Figure 4.4. The texel construction process was careful to align the texel boundaries with complete cycles of the weave pattern, so that the texels can abut seamlessly, as shown in Figure 4.5.

4.5 Chapter Summary

As an example of using texels to represent entire computer graphics models, the process of creating a model of a swatch of cloth was presented. A geometric model of the swatch was created by positioning individual threads using a relaxation technique.

Different techniques for creating volume densities were discussed. The fourth technique, called scan conversion (new to this thesis) converts geometric models directly into texels.

The idea of a texel editor is proposed. The texel editor is a utility program that can manipulate texels in a fashion appropriate to their nature. Rather than create texels, the texel editor allows the designer to combine, resize, or otherwise process texels.

An image of the swatch of cloth was presented. The cloth was modeled using the

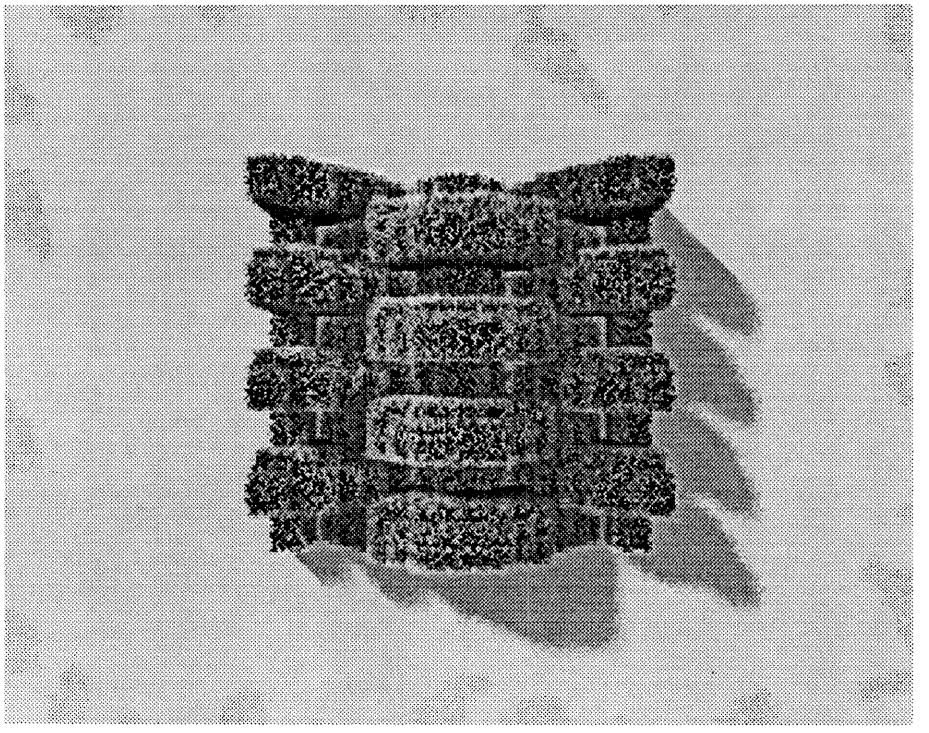


Figure 4.4: The cloth texel. The cloth texel is created by scan converting the output from the thread relaxation program. To better indicate the shape of the texel, the green ground poly is place so as to cut the texel in half.

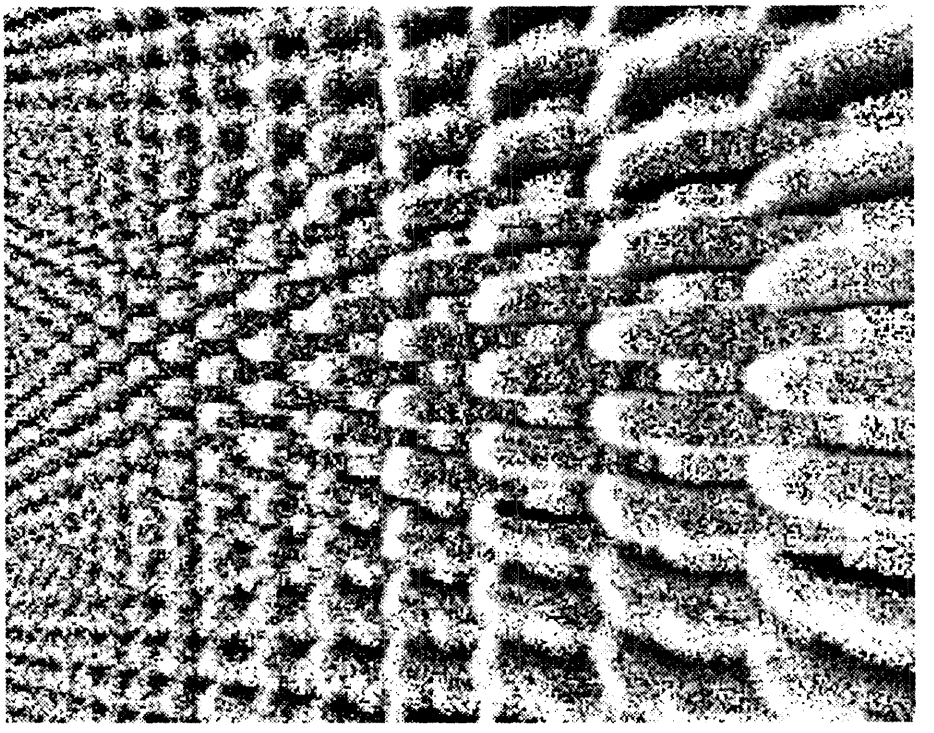


Figure 4.5: Cloth texel replication. The cloth texel replicated infinitely many times.

relaxation technique, converted to a texel using the scan conversion technique, and was rendered using the texel rendering technique and the fur BDRF.

Chapter 5

Computationally Derived BDRFs

This chapter considers the process of deriving bidirectional reflectance functions (BDRFs) computationally using computer graphics rendering techniques. The computational method is contrasted with other methods of obtaining BDRFs. Next, some theoretical work demonstrates the existence of the *texture threshold effect*, thereby establishing that BDRFs can indeed be obtained computationally. The discovery of the existence of a texture threshold effect is perhaps the most significant contribution of this thesis. Finally, as an example, the BDRF is extracted from the cloth texel of Chapter 4 and applied to a car seat.

5.1 Methods of Creating BDRFs

BDRFs are mathematical models of the interaction of light with various materials. Such models are used in computer graphics rendering to specify the appearance of objects constructed of the various materials. In the past, BDRFs have been constructed in four ways:

- *Ad hoc:* a BDRF is invented to satisfy the constraints of the situation. Examples: Lambertian model, Phong model.

- Physically based: the physics of the material are simplified to an appropriate level of abstraction. Examples: Blinn model [4], Cook metal model [9].

- Measured: the BDRF is measured directly from the material in question. Examples: Minato [22], Takagi [34].

- Computationally derived: A geometric model of the microscopic characteristics of the material is constructed. The model is then computationally reduced to a macroscopic BDRF. Example: Cabral, Max, and Springmeyer [7].

The *ad hoc* approach has served the computer graphics community well since the time of the first solid shaded images, providing simple BDRFs approximating a limited range of materials. The physically based and measurement methods are more difficult to apply, but can yield BDRFs for a wider range of materials.

If the first three approaches are providing useful BDRFs, why do we need to consider computational derivation? The traditional methods successfully model a limited set of materials. The physically based approach has been applied successfully only to materials whose microscopic features have been mathematically modeled or which can be described as simple random processes. For example, the Cook metal model was derived from existing physical models. A researcher cannot, however, refer to the *Handbook of Chemistry and Physics* to find a physical model of a hillside covered with grass, or even the most basic weave of cloth. Nor can such textures be easily modeled using a random process. Similarly, the measurement approach must be used in very controlled situations. A paint sample can be measured, but a field of wheat is more difficult.

This chapter studies the fourth approach, computational derivation of BDRFs and, as a new contribution, extends the range of materials which it is possible to model to include anything that can be described geometrically. The ability to characterize

a wider range of materials enables the construction of more-complex scenes and, thereby, the achievement of greater realism. In addition, the computational approach to the creation of BDRFs is inherently interesting for the same reason that the field of computer graphics is interesting: they could just step outside and snap a photograph. once realism is achieved, it will be possible to surpass it. Similarly, the ability to create realistic BDRFs computationally enables one can explore new materials that do not yet exist in the real world.

5.2 Extending Previous Work

The work by Cabral, Max, and Springmeyer [7] introduced the concept of BDRF extraction for isotropic models. The shortcomings of their approach were primarily caused by two factors. First, they used a spherical harmonics representation, which tends to low-pass filter their computed BDRF. Second, rather than computing samples by direct rendering, their procedure involved numerous mathematical approximations. In fact, their approach really is a hybrid approach between the physically based and purely computational methods.

The approach of this chapter diverges from that of Cabral, Max, and Springmeyer in almost every respect. It is the contention of this chapter that the computational approach can and should be applied in its pure form. There are essentially just two steps to the purely computational approach, as opposed to the numerous approximation steps proposed by [7]: a microscopic model of the material of interest is constructed in as fine detail as possible, and that model is sampled using the best rendering method available.

The simplicity of this approach precludes the use of mathematical approximations. Such approximations reduce the computational requirements of the BDRF extraction process at the expense of more important considerations. Any such approximations

first must be formulated, which is an human-intensive process. Also, by approximating, one risks biasing the extraction process and yielding an incorrect BDRF. Foremost, the overriding reason to avoid approximations is that *there simply is no need to approximate.* This chapter demonstrates that BDRFs can be derived without simplifying approximations in a reasonable amount of time.

5.3 The Texture Threshold Effect

We start by considering whether it is possible to extract a correct BDRF from a repetitive model by computational means. If we create a graphics model and start measuring, will the results converge to a unique solution? After all, the shift from microscopic appearance to macroscopic appearance might rely on some physical phenomenon (e.g., diffraction) which our renderer does not incorporate.

The answer to the question is, yes. When viewed from a distance greater than the *texture threshold*, the appearance of a graphics model will remain forever fixed. This behavior is called the *texture threshold effect.*

For example, Figure 5.1 shows images of the infinite cloth model from Chapter 4, illuminated by an infinite-distance light source. Each successive image is viewed via perspective transformation from twice the distance of the previous image. As the texture recedes into the distance, microscopic features disappear, and macroscopic features emerge. After the seventh iteration in the sequence, all images appear identical, except for noise which is added unavoidably during the rendering process.

The individual threads of the cloth texel disappear as the viewer crosses the texture threshold, the point at which geometry turns into texture. The existence of the texture threshold effect means that a complex computer graphics model need not be stored entirely as discrete modeling elements. For repeated elements that are viewed from beyond the texture threshold, a polygonal approximation to the model along

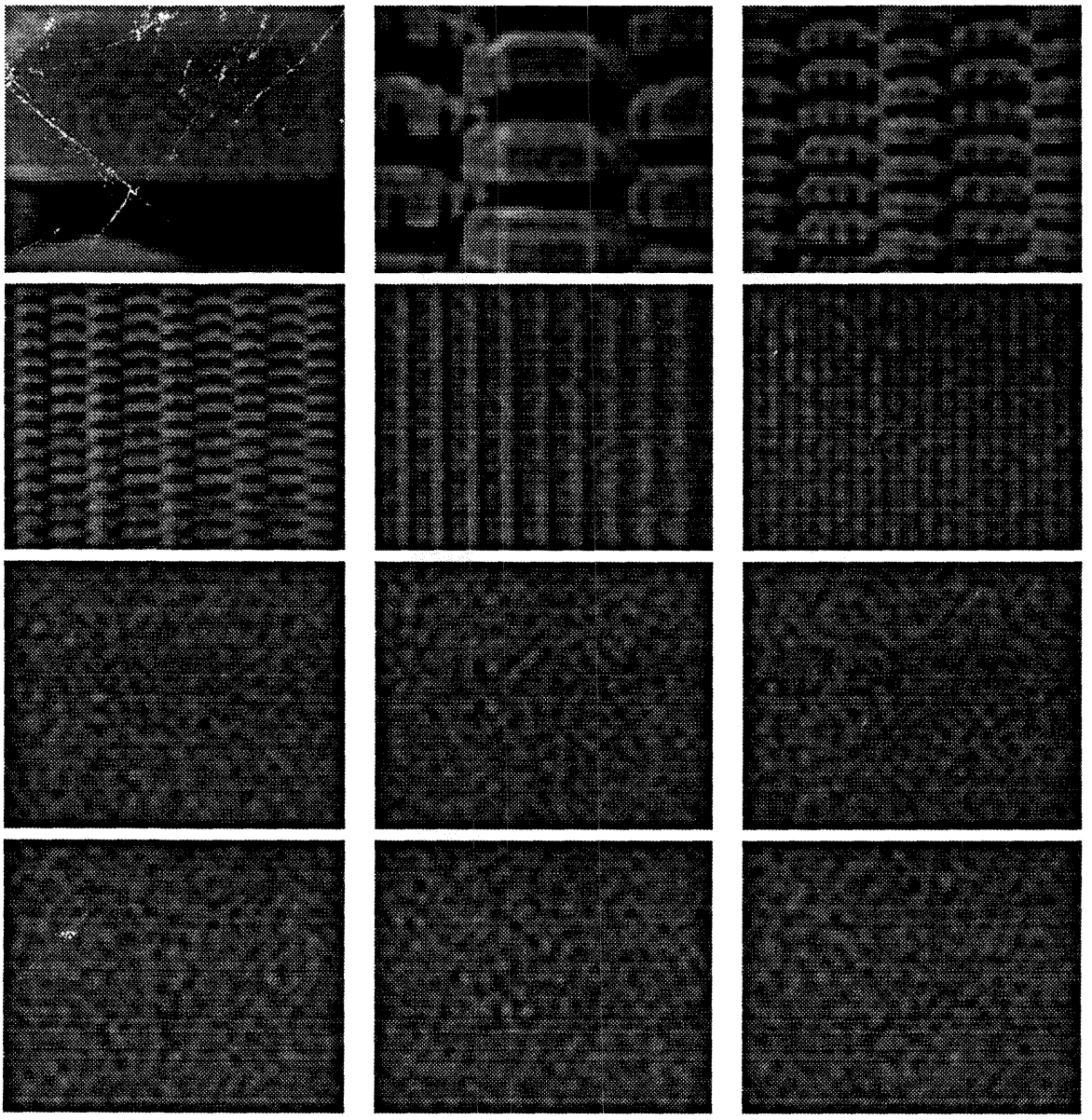


Figure 5.1: The cloth texel seen from an increasing distance. Each frame in the sequence is viewed from twice the distance as the previous frame. After a certain point, the frames appear identical.

with an appropriate BDRF may be substituted. For example, a field of grass needs to be stored as individual elements only if the viewer is close enough to resolve the individual blades. If the viewer is beyond the texture threshold, then a grass BDRF can be substituted for the individual blades without changing the appearance of the field.

Definition. The *fundamental cell* of a repeating model is the smallest repeating unit into which the model can be broken. In the case of the cloth of Chapter 4, for example, the fundamental cell is a unit two threads on a side. Call the frequency at which the fundamental cell repeats ν_{cell}.

Definition. The *pixel size* is the size of a pixel when projected onto the plane of the repeating model.

As can be seen in Figure 5.1, the texture threshold effect occurs exactly when the fundamental cell of the repeating texture changes from being larger than a pixel to being smaller than a pixel. The texture threshold effect occurs there due to the design of computer graphics renderers. In this chapter, we assume that a renderer is capable of producing a two-dimensional image by ideally low-pass filtering a three-dimensional computer graphics model M at Nyquist frequency ν_{pixel}. The next chapter more rigorously studies the capabilities of the renderer and analyzes specifically how well a renderer can low-pass filter a three-dimensional model.

Theorem. When the light is at infinity, and when an orthographic projection is used, the texture threshold effect occurs when the size of the fundamental cell becomes smaller than the size of a pixel, or $\nu_{\text{cell}} > \nu_{\text{pixel}}$.

Proof. The Fourier transform of a repeating pattern is a sequence of equally spaced delta functions. The DC term is the integral of M over one fundamental cell. The rest of the pulses occur at integral multiples of ν_{cell}. When $\nu_{\text{cell}} > \nu_{\text{pixel}}$, all delta functions are eliminated by the renderer except the DC term. The resulting image is,

therefore, a flat field.

The texture threshold has to occur at exactly the point indicated. If there were features in the model smaller than a pixel, and the renderer produced something other than a flat field, then the renderer must be producing aliasing artifacts.

The value of the flat field corresponds to the value of the BDRF for the corresponding positions of the light and viewer.

5.4 The BDRF Extraction Process

The BDRF extraction process involves the following steps:

1. Model. First a microscopic model of the material is created in such a way that it can be tiled to cover the infinite plane. The tiling process must be one that allows adjacent texels to interact. For example, it is important that the individual tiles shadow each other.

2. Light and viewer at infinity. The light source must be positioned at infinity. In computing a given sample, the angle to the light source must remain constant even though the image is rendered over a finite (non-infinitesimal) area. For the same reasons, the viewer must also be positioned at infinity, by choosing an orthographic (e.g., non-perspective) projection.

3. Isotropic or anisotropic? A determination must be made as to whether the model is isotropic (dependent on the rotation of the material around the vertical axis) or anisotropic.

4. The BDRF table size must be selected. An isotropic model generates a three-dimensional BDRF, while an anisotropic model generates a four-dimensional BDRF. Each cell in the table corresponds to an increment in elevation or az-

imuth for either the light or the viewer. In the isotropic case, any one axis may be omitted. The number of cells along each axis depends on the nature of the BDRF to be extracted.

5. The sampling process. For each direction of the light and the viewer, an image is rendered. The viewing parameters must be chosen so that the rendering happens beyond the texture threshold. The center pixel of each image is used as the value of the BDRF sample.

The BDRF extraction process produces a three- or four-dimensional *tabular BDRF* which accurately represents the sampled material whenever that material is being viewed from beyond the texture threshold. To use the tabular BDRF, the macroscopic geometry of the material is modeled as a polygonal mesh. At each ray-polygon intersection, normal and tangent vectors are computed using Phong interpolation to form a local coordinate frame. The direction cosines to the light and viewer can be transformed into the local coordinate system and then used as indices into the tabular BDRF, resulting in an intensity value for that point on the model.

As indicated by the last step, each sample is extracted by rendering an entire image, although each such image need be only a few pixels on a side. There are two reasons for rendering the whole image. First, as discussed in Chapter 6, a low pass filter must be employed to control aliasing. Such filtering requires the computation of successive pixels. Second, in the event that the renderer is behaving incorrectly, aliasing will be immediately apparent.

Two points must be made regarding the sampling of the microscopic geometry at low angles of incidence. First, an incidence angle of exactly zero degrees cannot be measured. At exactly zero, the direction of view is parallel to the plane of the model, and no intersections occur. Fortunately, such measurements are not required

to produce a useful BDRF table. When applying the tabular BDRF to an ordinary polygonal model, the incidence angle between the view ray and model is zero degrees only at the silhouette edge which is of zero thickness. Thus, the grazing angle needs to be measured near the horizon, but not exactly at the horizon.

Second, special consideration must be given to rendering infinitely repeated texel models. Chapter 2 mentioned that a ray-texel integration can be terminated as soon as the density sum σ becomes larger than a given threshold. Such termination is vital when rendering an infinite texel; near the horizon, the distance between the integration limits t_{near} and t_{far} approaches infinity. Without the termination criterion, the computation of the samples near the horizon may never finish.

Having outlined the BDRF extraction process, we now turn to a concrete example.

5.5 Cloth BDRF

The BDRF was extracted from the cloth texel of Chapter 4 and is presented in Figure 5.2. Figure 5.2(a) shows the BDRF as a telescoped spherical projection. Viewed sideways, so that the long dimension is horizontal, the bottom of the table represents the equator of a hemisphere, and the top represents the north pole. Horizontal positions represent various angles of azimuth of the light source, while vertical positions represent various angles of elevation. At each sampled position of the light source, an entire spherical projection of the behavior of the viewer is given. Figure 5.2(b) is identical to Figure 5.2(a) with the roles of the light and viewer reversed.

The cloth texel is oriented so that the warp direction is zero degrees of azimuth. Each azimuth (light and viewer) was sampled at ten positions, and each elevation was sampled at four positions. At the poles, where multiple sampling is redundant, only one actual sample was computed (resulting in a total of $(10\times3+1)^2$ samples). The resulting samples were placed into a table of dimension $11\times4\times11\times4$ (to simplify

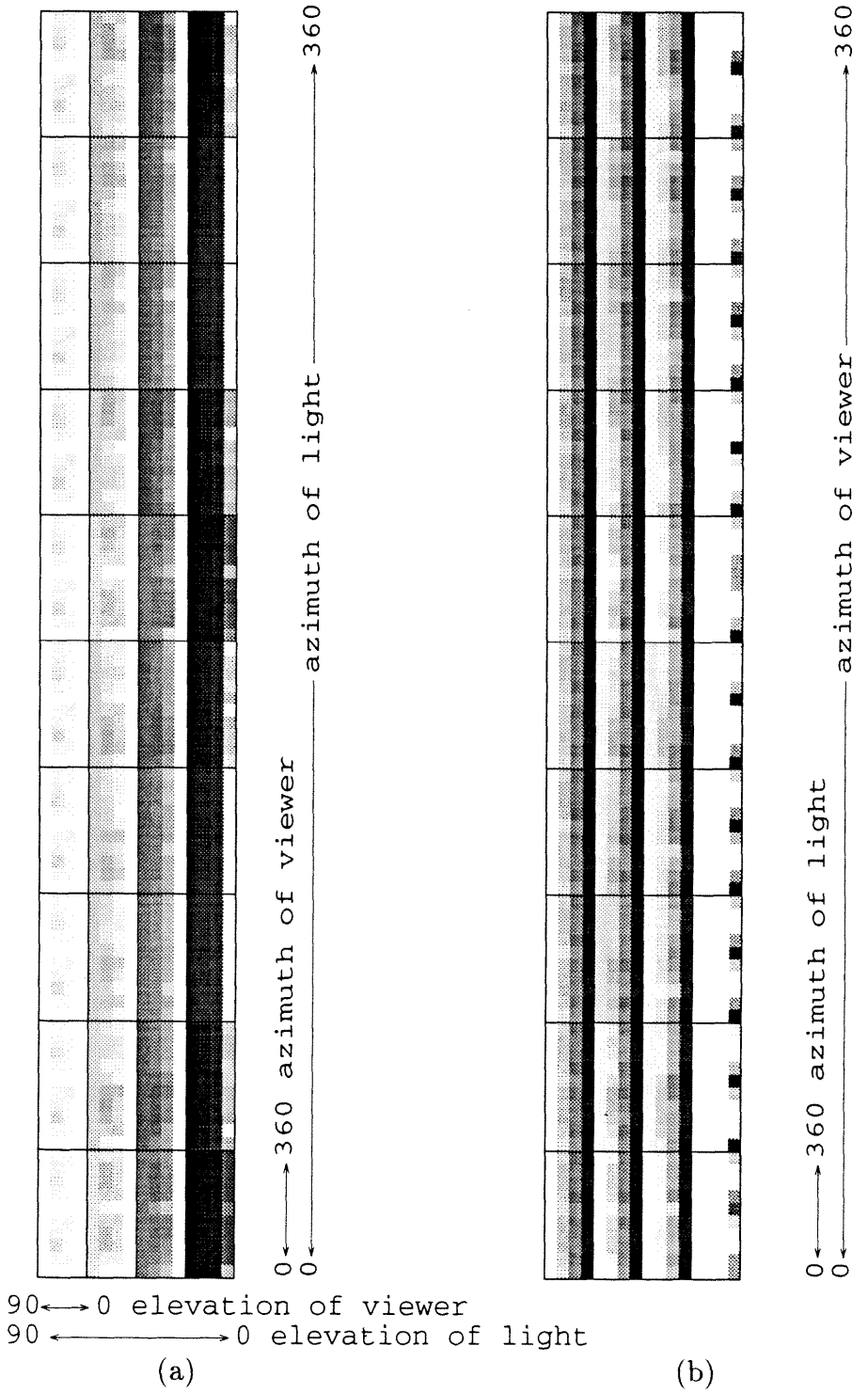


Figure 5.2: The computationally derived cloth BDRF.

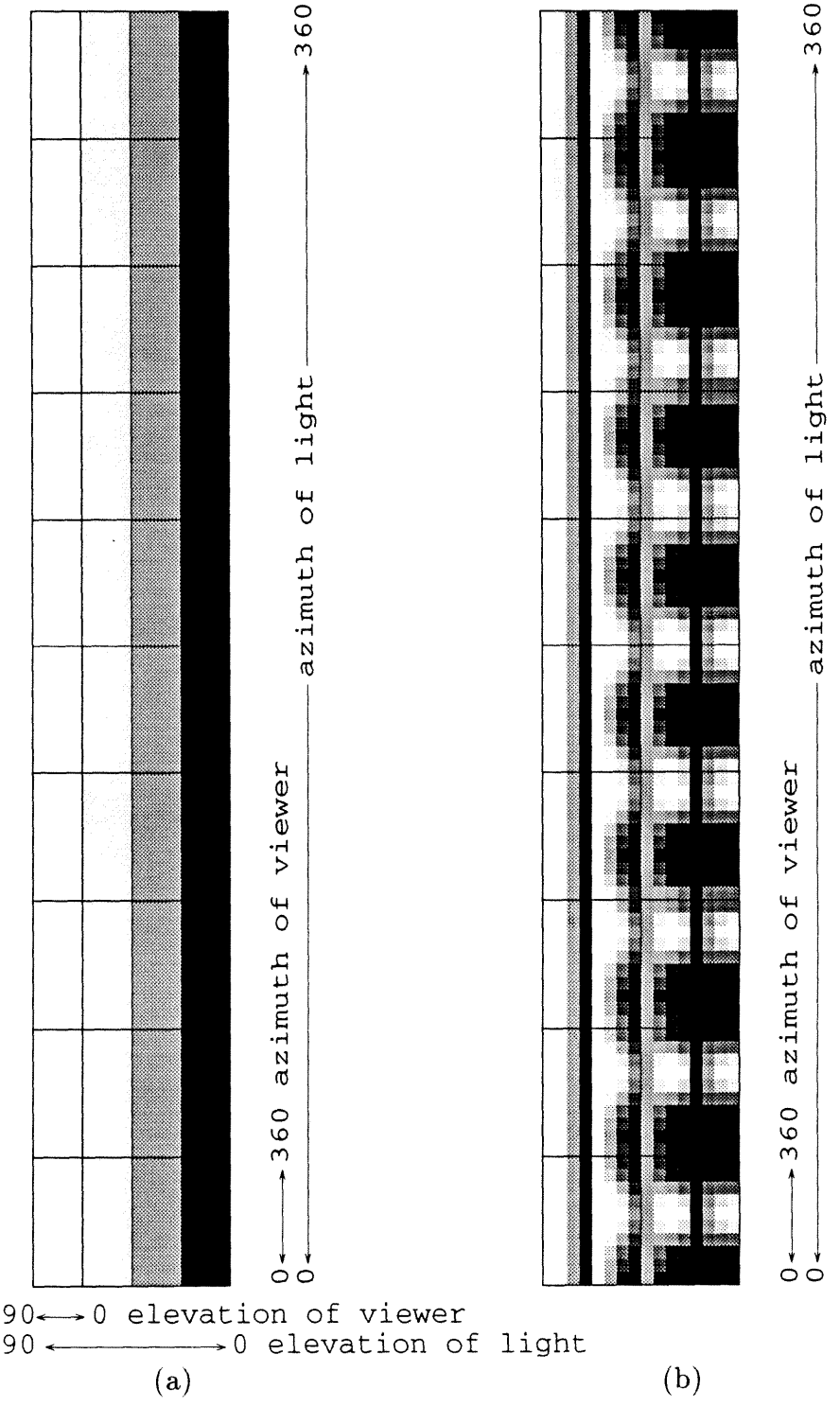


Figure 5.3: Lambert diffuse and Phong specular BDRF.

calculations, the first azimuthal slice is duplicated as the last).

Included for comparison, Figure 5.3 shows (a) the Lambert BDRF, and (b) the Phong BDRF. The Lambert BDRF is very simple. Each telescoped spherical projection is a constant intensity, due to the fact that the Lambert BDRF is independent of the position of the viewer. The Phong BDRF is isotropic, and therefore repeats in the azimuthal direction.

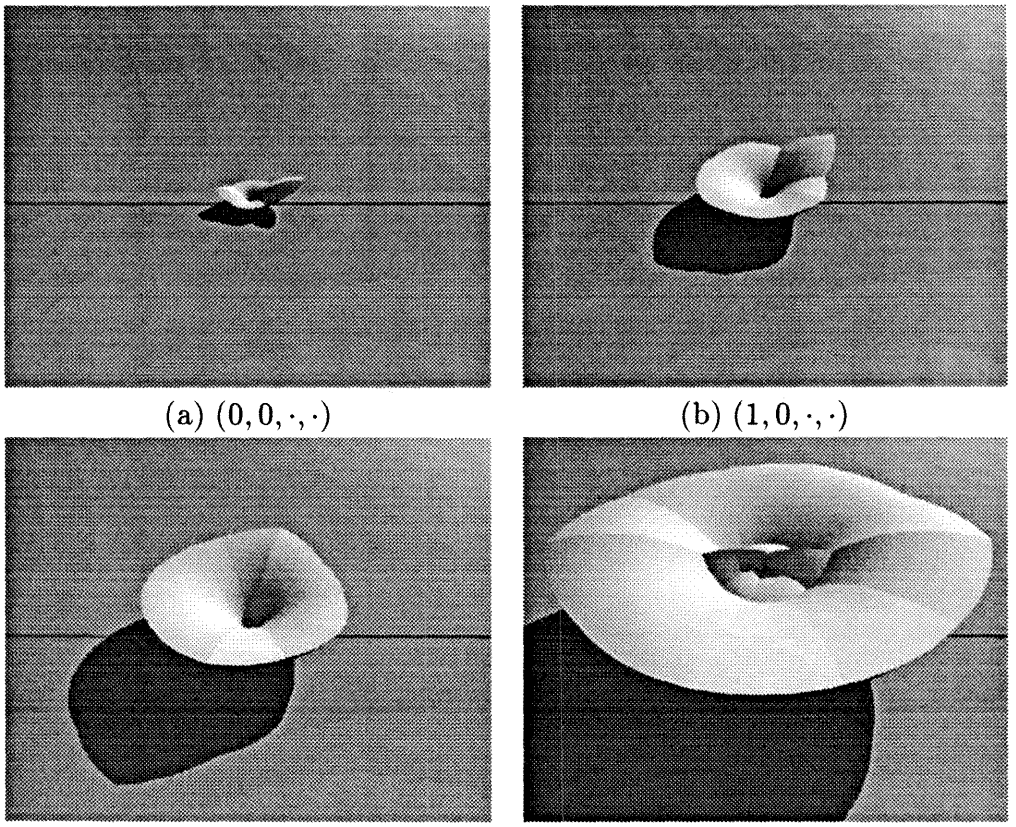
We consider several features of the cloth BDRF of Figure 5.2(a). The diagram will be indexed using a four-tuple (a_l, e_l, a_v, e_v), where a_l and e_l are the azimuth and elevation of the light, and a_v and e_v are the azimuth and elevation of the viewer. An entire telescoped projection is referenced as $(a_l, e_l, \cdot, \cdot)$. The azimuth and elevation angles are specified in degrees.

The projection $(0, 0, \cdot, \cdot)$ represents the view-dependent BDRF, given that the light is on the horizon and at 0 degrees of azimuth. Within the projection, there are two bright-positions, corresponding to indices $(0, 0, 0, 0)$ and $(0, 0, 180, 0)$. In other words, when the light and viewer are both at 0 degrees, about half the light is reflected from the light to the viewer. When the light is at 0 degrees, and the viewer is at 180 degrees, about a third of the light is reflected.

As the light and viewer both move to 90 degrees of azimuth (indicated on the diagram three cells to the right) almost all the light is reflected to the viewer.

Referring to the top of the diagram, where the light is at 90 degrees of elevation, an interesting situation occurs. If the viewer is at the horizon, $(\cdot, 90, \cdot, 0)$, a large fraction of the light is reflected towards the viewer. Similarly, when the viewer is at 90 degrees, $(\cdot, 90, \cdot, 90)$, a large fraction of the light is reflected towards the viewer. However, between 0 and 90 degrees, less light is reflected towards the viewer. This bi-modality is unlike any lighting model commonly used in computer graphics.

Figure 5.4 shows a few projections of the cloth BDRF in a more traditional com-



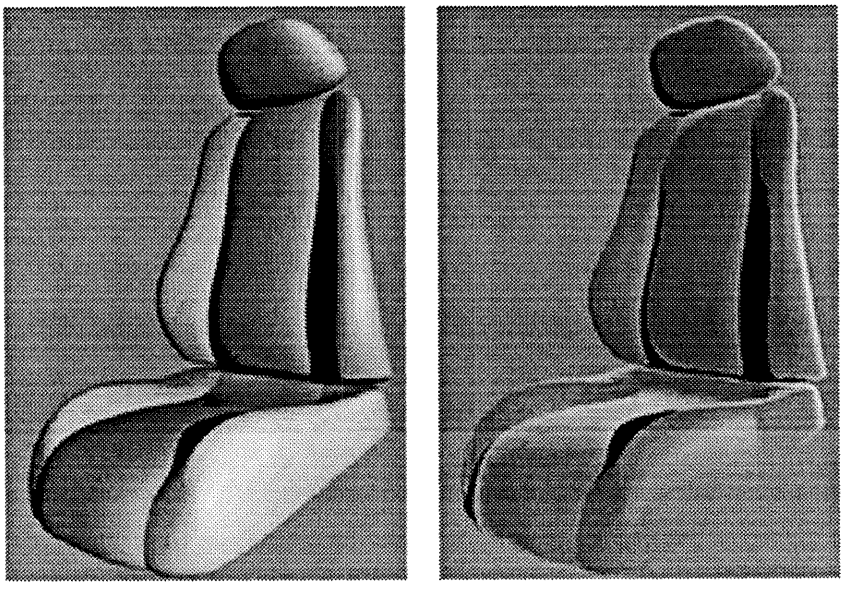
(a) $(0, 0, \cdot, \cdot)$ (b) $(1, 0, \cdot, \cdot)$

(c) $(2, 0, \cdot, \cdot)$ (d) $(\cdot, 3, \cdot, \cdot)$

Figure 5.4: Traditional view of cloth BDRF. At four selected positions of the light, the figures show the cloth BDRF for all possible view positions. The banding occurs at discontinuities in the derivatives, which are introduced during the linear reconstruction of the continuous BDRF from the tabular BDRF.

puter graphics manner. Figure 5.4(a) shows the entire hemisphere at index $(0, 0, \cdot, \cdot)$, drawn with the radius modulated by the magnitude of the BDRF. Similarly, Figure 5.4(b) shows the hemisphere $(72, 0, \cdot, \cdot)$. The latter figure is much larger because the BDRF assumes much larger values for the corresponding light position. Figure 5.4(c) shows the hemisphere $(\cdot, 90, \cdot, \cdot)$.

Figure 5.5(a) shows a car seat model rendered using Lambertian shading. The seat was created as a mesh of bicubic patches. Each patch was adaptively subdivided to satisfy a flatness criterion and then approximated with triangles. Each triangle was



(a) (b)

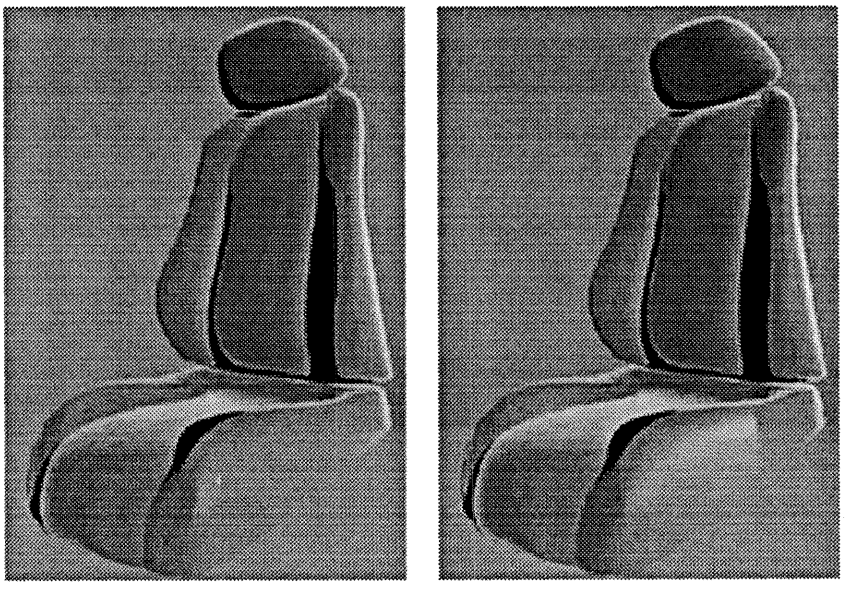
Figure 5.5: Diffuse and cloth car seats. Seat (a) was shaded using the traditional Lambertian diffuse model, while seat (b) was computed using the extracted cloth BDRF. Car seat courtesy of Toyota Motor Corporation.

stored as a set of three vertices containing position, normal, and tangent information.

Tangent vector information should be created along with the rest of the model as part of the design process because the orientation of the cloth material must be specified as well as its shape. Unfortunately, the Toyota car seat model provides no tangent information.

The parameter lines of the bicubic patches are inappropriate as tangent vectors because they fail to lie along the desired grain of the cloth. Instead, we must rely on the fact that most of the surfaces of the car seat are roughly aligned with the coordinate axes, and constant direction tangent vectors make good approximations, as explained below.

The same ray tracer that was used to extract the BDRF is used to render the car seat. Each ray-triangle intersection provides surface normal and tangent vectors, as well as directional information to the viewer and light source. The normal and



(a) (b)

Figure 5.6: Car seats with different tangent vectors.The tangent vectors point (a) upwards and (b) rearwards.

tangent vectors define a local coordinate system. The vectors to the viewer and light source are transformed into (elevation,azimuth) pairs in the local coordinate system and are used to look up a BDRF value from the table.

Figure 5.2(a) was rendered with the tangent vectors pointing upwards, which is a reasonable approximation of the tangent vectors for the back of the car seat. Figure 5.2(b) was rendered with tangent vectors pointing rearwards, which is a reasonable approximation of the tangent vectors for the seat. The seat from Figure 5.2(a) was combined with the back from Figure 5.2(b) to produce Figure 5.5(b).

Two noteworthy features in Figure 5.5(b) give the impression that the car seat is upholstered. The cloth BDRF illuminates the seat much more uniformly than does the Lambertian model, and the silhouette edges of the cloth are very bright. Upon inspection of real cloth, these effects are readily confirmed.[1]

[1]I would like to make a simple observation about psychophysics, which takes us beyond the scope of this work: the illumination of the car seat by the two BDRFs yield results which are drastically different. Nevertheless, the human visual system has no apparent difficulty discerning the true shape

Implementation Details

Two implementation details, which are important to ray tracing in general, are mentioned here because they are particularly important when using the cloth BDRF.

First, when rendering macroscopic models shaded as cloth, it is very important to antialias. The cloth BDRF has a sharp feature that occurs at the silhouette edge. In the absence of antialiasing, such features tend to get lost in the "jaggies."

Second, when viewing macroscopic models shaded as cloth, it is very important to *gamma correct*: although the computer graphics rendering process assumes linearity, computer displays do not respond linearly to the voltages produced by the frame buffer. The non-linearity is called the *gamma* of the monitor. To compensate, an inverse non-linear ramp must be applied either to the image before it is stored in the frame buffer, or to the color look up tables in the frame buffer itself. In the absence of gamma correction, the quality of images containing cloth BDRFs seem to degrade much more severely than similar images containing simpler (e.g., Lambertian, Phong) BDRFs.

5.6 Chapter Summary

A new phenomenon, the texture threshold effect, was presented. The texture threshold is the point at which geometry turns into texture, and occurs when the size of the fundamental cell of a repeated texture becomes smaller than the size of a pixel. A proof of the existence of the effect was presented.

The cloth texel, repeated over the infinite plane, was rendered from beyond the texture threshold for a variety light source and viewer positions. The resulting samples constitute a BDRF for the cloth model which was presented as a telescoping spherical

of the model.

projection.

The cloth BDRF was applied to a polygonal car seat model, yielding a realistic, upholstered appearance.

Chapter 6

Point Sampling and Computer Graphics

Chapter 5 described a technique for computationally measuring the BDRF of arbitrary surfaces, whereby a microscopic surface patch is modeled as a texel and is replicated to tile the infinite plane. By tracing rays, the average intensity of the architected surface can be measured for various positions of the light source and viewer. The collected measurements constitute a BDRF for the modeled surface. The tabulated BDRF can then be applied to a traditional computer graphics model, enabling the macroscopic appearance of the surface to be reproduced with minimal calculation.

The former discussion of the BDRF calculations sidestepped an important issue. The BDRF measurement process involves point sampling of a continuous-domain function, which must be performed with care lest incorrect results be obtained due to aliasing. In particular, the process described in Chapter 5, consisting of patterns repeating to infinity, constitutes a worst-case scenario as far as sampling problems are concerned.

If we are to trust the results of our BDRF calculations, we must be certain that these results are free from aliasing. Unlike rendering a computer graphics image, for which the existence of aliasing artifacts is immediately obvious, the rendering of

BDRFs makes determination of the existence of aliasing artifacts much more difficult. It is, therefore, especially important to address the sampling problems of Chapter 5 from a theoretical point of view.

This chapter discusses the point sampling technique called *jitter sampling* [11] as it applies to the specific application of computer graphics rendering discussed in Chapter 5, as well as to computer graphics rendering in general.

This chapter begins by creating a new classification for computer graphics images, called Class I and Class II. Class I images are those that contain high-frequency spectral energy due only to discontinuities. Class II images contain high frequency spectral energy due to both discontinuities and to high frequency details in the model. Images from both classes contain spectral energy at arbitrarily high spectral energy. The chapter discusses the typical engineering approach to sampling signals with spectral energy at arbitrarily high frequencies and explains why such an approach cannot be used in computer graphics, where Class II images are increasingly common.

The chapter presents the often-used computer graphics method for rendering Class I images, called subsampling. The effect of subsampling is then explained mathematically using simple Fourier analysis. The analysis demonstrates that it is possible to render computer graphics images although the signals being sampled contain spectral energy at arbitrarily high frequencies. A formula is derived for the spectral energy due to aliasing as a function of the amount of subsampling. While these results are not new, they lay important ground work for considering the Class II case that follows.

Finally, the chapter studies jitter sampling as it applies to computer graphics. Jitter sampling was introduced to computer graphics by Cook *et al.* [11], and has been analyzed by Cook [10] and Dippé and Wold [13]. The analysis presented in each paper is incomplete. To derive the spectral attenuation of jitter sampling, [10] cited Balakrishnan [1], who restricted the domain of mathematical derivation to input

signals that are already band width limited to below the Nyquist limit. It is, of course, necessary to consider input signals which are not band width limited. The analysis presented in [13] leaves some large gaps, also. The conclusions in [13] may be correct, but their analysis does not demonstrate their correctness.

Using a more direct approach, this chapter derives the frequency attenuation result similar to those upon which [10] based his results. The derivation presented in this chapter is valid for arbitrary signals, not just signals which have been band width limited before sampling. The attenuation result is then used to explain how jitter sampling allows Class II images to be treated in the same way as Class I images.

6.1 Terminology and Assumptions

As illustrated in Figure 6.1, computer graphics rendering algorithms involve a mapping Φ of a three-dimensional model M into a two-dimensional image function $f(u, v)$. The image function $f(u, v)$ is used for one of two purposes, depending upon whether an image is being rendered or a BDRF is being computed. In the case of rendering, $f(u, v)$ is point sampled to produce an array of pixels s_{ij}, called an *image*. As shown in Figure 6.2, the pixels are stored in a *frame buffer*, which reconstructs an approximation $\tilde{f}(u, v)$ of the original $f(u, v)$.

A frame buffer contains pixels that occur at a period T units of distance per pixel. If it were ideal, the frame buffer could display a maximum frequency $\nu_{\text{pixel}} = \frac{2\pi}{T}$ radians per unit distance. In practice, imperfect electronics limit the response of the frame buffer to $\nu < \nu_{\text{pixel}}$.

The nature of the reconstruction performed by the frame buffer is largely beyond our control. It will be assumed that such reconstruction is *perfect* (that the frame buffer is capable of displaying all frequencies up to a frequency ν) without attenuation. (The frequency ν might be slightly smaller than the Nyquist limit corresponding to

the pixel rate of the frame buffer.)

As discussed in Chapter 5, $f(u,v)$ might instead be measured via point sampling to build a tabular BDRF. The resulting table can then be interpolated to produce a continuous BDRF $\tilde{f}$.

The two key questions to be addressed in this chapter are, what strategy should be used to sample $f(u,v)$, and how well does the resulting $\tilde{f}(u,v)$ represent $f(u,v)$? Because the frame buffer can display frequencies no higher than ν, $\tilde{f}(u,v)$ cannot exactly equal $f(u,v)$. Somehow, $\tilde{f}(u,v)$ must be a band width limited version of $f(u,v)$. We will assume that an ideal representation of $f(u,v)$ would be that which presents a version of $f(u,v)$, the spectrum of which has been perfectly band width limited to frequencies less than ν.

It is often useful to sample $f(u,v)$ at a frequency η which is greater than ν_{pixel}. In that case, the samples eventually will be resampled to one per pixel.

The sampling problems discussed in this chapter are all two-dimensional. But because sampling on a regular grid is *separable*, meaning that they can be treated one dimension at a time, the mathematics in the remainder of the chapter will refer to only one dimension. The image function $f(u,v)$ will be called $f(x)$, and its spectrum will be called $f(\omega)$. Similarly, the sampled signal s_{ij} and the reconstructed signal $\tilde{f}(u,v)$ and its spectrum will be called s_i, $\tilde{f}(x)$, and $\tilde{f}(\omega)$, respectively.

6.2 The Shannon Sampling Theorem

The Shannon sampling theorem states that a band width limited function $f(x)$ with maximum frequency ω must be sampled with frequency 2ω to be represented faithfully. It is appropriate for computer graphics to consider the contrapositive of this statement: if we wish to sample a function f using T samples per unit distance, then f must contain no spectral energy at frequencies above $\frac{\pi}{T}$ radians per second (the

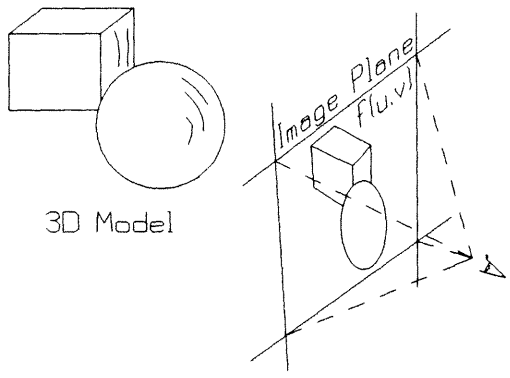


Figure 6.1: Two-dimensional image function. The two-dimensional image function $f(u, v)$ is created by a transformation Φ of a three-dimensional analytic model M onto the two-dimensional *image plane*. The transformation process Φ encompasses, among other things, shading, reflection, shadow, hidden surface elimination, and viewing transformation calculations.

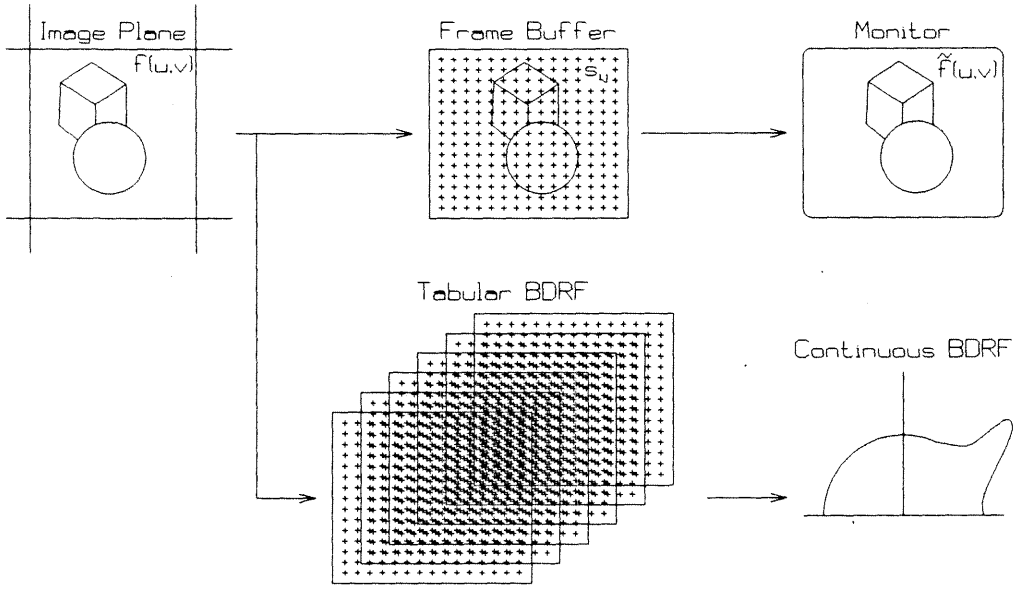


Figure 6.2: Rendering and extracting BDRF. An image function can be used for two purposes. Across the top of the diagram, the image function $f(u, v)$ is sampled to create the *image* s_{ij}, which is stored in the frame buffer. The frame buffer reconstructs the samples into an approximation $\tilde{f}(u, v)$. Across the bottom of the diagram, the image function is sampled for various positions of the light and viewer, creating a four-dimensional tabular BDRF. From the table, $\tilde{f}$, the continuous BDRF, is reconstructed.

Nyquist limit). Otherwise, the high frequency components of f will appear as low frequency energy in the samples.

In real-world applications, the Shannon sampling theorem specifies a constructive precondition which must be established before classical Fourier analysis techniques can be applied. For example, if an audio signal is to be processed digitally, the application itself will dictate a desired band width, say, 20 kHz. The Shannon sampling theorem states that the signal must be sampled with a frequency of at least 40 kHz. The same theorem requires, moreover, that the incoming analog signal must be band width limited to 20 kHz *before* the samples are taken. In the real world, it is easy to limit the incoming band width by using an analog filter. Although the incoming analog signal might contain spectral energy at arbitrarily high frequencies, a simple analog filter can be designed to pass only the appropriate band of frequencies.

In computer graphics, we have analytic descriptions of signals rather than analog signals. For example, the description of a sphere as an origin and a radius analytically describes an analog signal. Via sampling, various processes in computer graphics transform an analytic description of a model into a discrete representation of some transformation of the model. Just as in the real world, the Shannon sampling theorem states that computer graphics sampling cannot be performed accurately unless the band width of the original analytically-defined model be limited *before* the samples are taken. Unlike the real world, the digital nature of the computer precludes the use of an analog front end to the sampling process. So if it is to be band width limited before sampling, the analytic function will have to be band width limited analytically, which is, in most cases, an intractable problem.

Since it is impractical to band width limit the continuous-domain function f before it is sampled, whenever a computer graphics model contains spectral energy of frequency higher than ν, any computational sampling process will fail to meet the

hypotheses of the Shannon sampling theorem. This chapter will show that essentially all interesting computer graphics models contain such problematic spectral energy. In other words, computer generated imagery will always contain aliasing.

Furthermore, this chapter will show that, despite the lack of applicability of the Shannon sampling theorem, it is still possible to solve computer graphics problems using point sampling, via techniques called subsampling and jitter sampling.

6.3 Spectral Energy Classification

To understand point sampling in computer graphics, it is first necessary to characterize the types of image functions $f(x)$ that occur in computer graphics processes. A computer graphics image function is created by projecting a three-dimensional model M onto the image plane. The model M is composed of objects whose surfaces are smooth. This collection of surfaces is projected onto the image plane. Due to the smoothness of the projection, and the nature of the hidden surface elimination, the resulting image function is a piecewise smooth function.

The Fourier transform of a piecewise smooth function will have high-frequency spectral energy from two sources. First, discontinuities exist between the smooth pieces. These discontinuities are sources of arbitrarily high frequency spectral energy. Such spectral energy due to discontinuities will be termed *Class I spectral energy.* Second, high frequency spectral energy comes from high-frequency patterns that make up the piecewise smooth image function. Such patterns contribute high frequency energy called *Class II spectral energy.*

If we sample $f(x)$ with a sampling frequency η, and if $f(x)$ contains spectral energy at a frequency greater than $\frac{\eta}{2}$, *which is not Class I spectral energy,* then the image is said to contain Class II spectral energy.

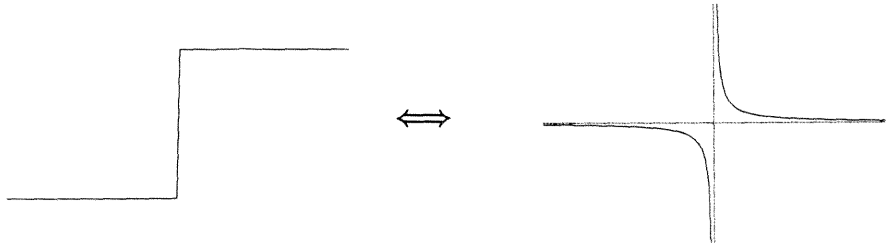
The distinction between the two classes of spectral energy is fundamental; one type

of energy cannot be mapped into the other by changing the viewing transformation Φ. As long as the scene contains a discontinuity, Class I spectral energy will exist. It does not matter how much the Φ map magnifies M; discontinuities will continue to be discontinuities.

Class II spectral energy is more difficult to characterize. A model M cannot, without also considering the mapping Φ, be said to contain Class II spectral energy. To any model M containing arbitrarily high spatial frequencies, we can assign a viewing transformation which will magnify the high-frequency area sufficiently so that, relative to the sampling frequency η, the frequencies of $f(x)$ will be exclusively Class I. Similarly, to any model M, we can assign a viewing transformation that shrinks even low-frequency areas until they represent (relative to η) arbitrarily high frequency spectral energy.

For the purposes of discussing sampling in computer graphics, the canonical model M is that of an infinite checkerboard. The checkerboard contains Class I spectral energy, contributed by every (discontinuous) transition between the black and white checks. Does an image function of the checkerboard contain Class II spectral energy? It depends upon the map Φ. If the checkerboard covers the entire field of view, and the checks are larger than the pixel spacing, then only Class I spectral energy exists. For the checkerboard image function to contain Class II spectral energy, it is necessary for one of two conditions to hold. Either the checks must be small enough that their spatial frequency exceeds η, or the viewing transformation must be such that the horizon is visible, in which case the checks in the distance become vanishingly small.

The two classes of spectral energy defined above require separate treatment. We start by analyzing Class I energy and finish with Class II.



$H(x)$ $\qquad\qquad$ $iH(\omega)$

Figure 6.3: Class I spectral energy: The spectrum of a discontinuity falls off as $\frac{1}{\omega}$. The graphs show the step function $H(x)$ and its Fourier transform $H(\omega)$.

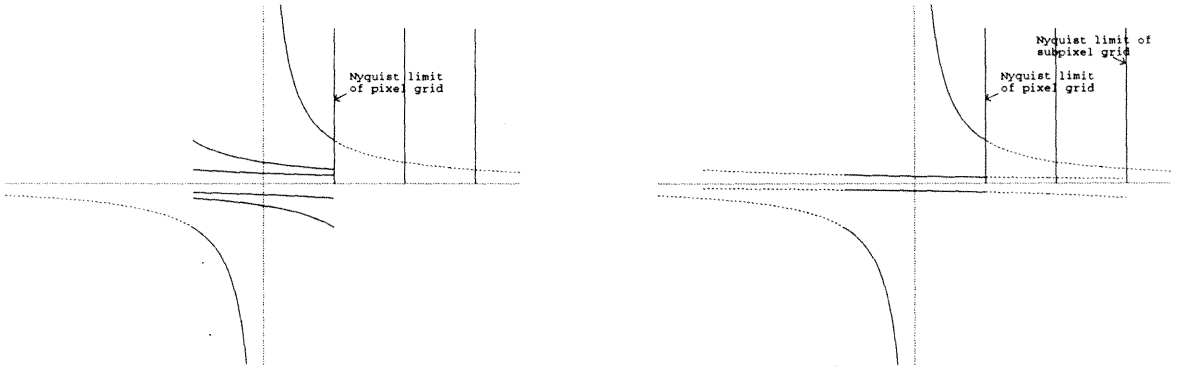
6.4 Class I Spectral Energy: Discontinuities

Class I image functions contain large, smooth features separated by discontinuities in the form of silhouette edges. To characterize the spectral behavior of discontinuities, we consider an image function which contains a single discontinuity. Let $H(t)$ be a discontinuous function, such that $H(t) = 0$ if $t \leq 0$, and $H(t) = 1$ if $t > 1$. The Fourier transform of $H(t)$ is

$$H(\omega) = \int_{-\infty}^{\infty} H(x)e^{-i\omega x}dx = \int_{0}^{\infty} e^{-i\omega x}dx = \int_{0}^{-\infty} e^{u}\frac{du}{-i\omega} = \frac{1}{i\omega}\int_{-\infty}^{0} e^{u}du = \frac{1}{i\omega},$$

by substituting $u = -i\omega x$ and $du = -i\omega dx$. In other words, while the spectrum of a discontinuity contains arbitrarily high frequencies, the spectral energy of the high-frequency components falls off as $\frac{1}{\omega}$.

Computer graphics image functions contain discontinuities, and, therefore, will contain spectral energy at arbitrarily high frequencies. When such an image function is sampled at the rate of one sample per pixel, high frequency energy (the energy above the Nyquist limit ν of the frame buffer) will alias and produce low frequency artifacts. Figure 6.4(a) illustrates, in the frequency domain, how such aliasing occurs. The spectral energy (which falls off as $\frac{1}{\omega}$) folds back on itself at the Nyquist frequency.



(a) (b)

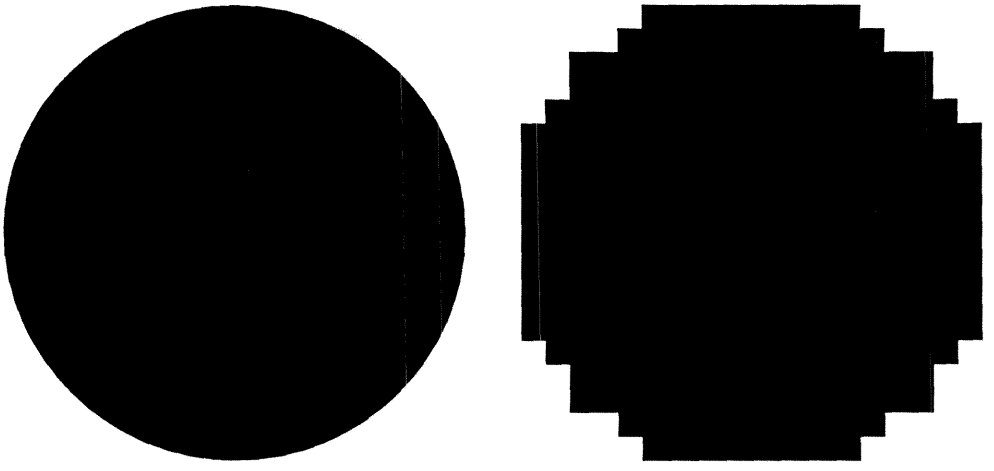
Figure 6.4: Subsampling reduces aliasing energy. When rendered at one sample per pixel, as in (a), the $\frac{1}{\omega}$ spectral energy aliases at the Nyquist limit corresponding to the pixel rate. When subsampling, as in (b), the $\frac{1}{\omega}$ spectral energy aliases at a higher frequency. A low-pass filter eliminates the band of spectral energy that falls between the two Nyquist limits.

Figure 6.5(a) show a two-dimensional image function $f(u, v)$ containing a light-gray circle against a dark-gray background. Figure 6.5(b) shows $\tilde{f}(u, v)$, the effect of rendering the circle using one sample per pixel. The jagged edges of the circle are a result of undersampling $f(u, v)$.

6.5 Subsampling

Because the spectrum of a discontinuity contains spectral energy at arbitrarily high frequencies, computer graphics images will always contain aliasing artifacts. Fortunately, the high-frequency spectral energy of a discontinuity decreases monotonically, so a process called *subsampling* is traditionally used to reduce the aliasing energy to an arbitrarily small magnitude.

In order to take advantage of the $\frac{1}{\omega}$ behavior of the spectral energy, it is necessary to increase the frequency at which the spectrum is folded back onto itself. At one sample per pixel, frequencies above ν will alias. An increase in the sampling frequency



(a) (b)

Figure 6.5: Class I aliasing. A continuous-domain circle (a) exhibits aliasing (or "jaggies") when sampled at one point per pixel (b).

will cause an increase in the frequency at which aliasing occurs. Due to the $\frac{1}{\omega}$ spectral behavior of Class I images, postponing aliasing until a higher frequency results in lower overall aliasing energy.

The process of subsampling involves sampling $f(x)$ with a period $\frac{T}{N}$, which corresponds to a sampling frequency N times higher than the pixel frequency. Figure 6.6(a) shows the sampling grid for an image four pixels on a side. Figures 6.6(b) and 6.6(c) show the sampling grid for the same image when $N = 2$ and $N = 4$ respectively. While rendering using (b) or (c) requires, respectively, four or sixteen times as many samples as (a), the quality of the resulting images improves proportionally.

Figure 6.4(a) illustrates, in the frequency domain, the aliasing that occurs when sampling at the pixel rate. Figure 6.4(b) demonstrates sampling at a higher rate. Notice that the areas under the aliasing curves are significantly smaller than the same areas in Figure 6.4(a).

After sampling at the higher rate, we now have N times as many samples as will fit into the frame buffer. Also, the maximum frequency represented in the samples is

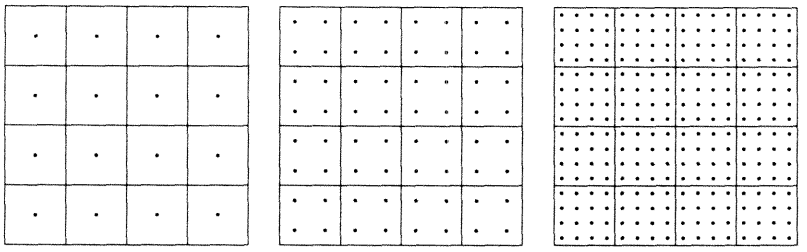


Figure 6.6: Subsampling schematic. An image that is four pixels high by four pixels wide can be created by sampling with one sample per pixel, as in (a). To reduce the spectral energy due to aliasing, the image function is sampled at a higher sampling frequency, as in (b) and (c). The values for the actual pixels are then computed by applying a low-pass filter and resampling the image at the lower sampling frequency.

$N\nu_{\text{pixel}}$, which is above the Nyquist frequency for the frame buffer. It might appear that the situation is the same as before: we still have an input signal with a band width larger than the frame buffer can display. The fact that the input signal is discrete, however, allows further processing to be carried out digitally. To reduce both the number of samples and the maximum frequency, a digital low-pass filter is used. Any low-pass filter which rolls off sharply at frequency ν is adequate and easy to implement in a computer. We use a Lanczos filter [15],

$$L(x) = \begin{cases} \text{sinc}(\nu x)\text{sinc}(\frac{\nu x}{k}) & \text{if } -\pi < \frac{\nu x}{k} < \pi \\ 0 & \text{otherwise} \end{cases},$$

which is a sinc function windowed by the first lobe of a second sinc function. The parameter k determines the width of the kernel, which will be $2k$ lobes (or, if $\nu = \nu_{\text{pixel}}$, $2k$ pixels) wide. Figure 6.7 shows a Lanczos filter.

Figure 6.8 illustrates computer graphics rendering both with and without subsampling. The top schematic diagram shows an image function sampled at one sample per pixel and stored directly in the frame buffer. The bottom schematic diagram shows the image function subsampled with $N = 2$. The subsamples are low-pass filtered, and the results are stored in the frame buffer.

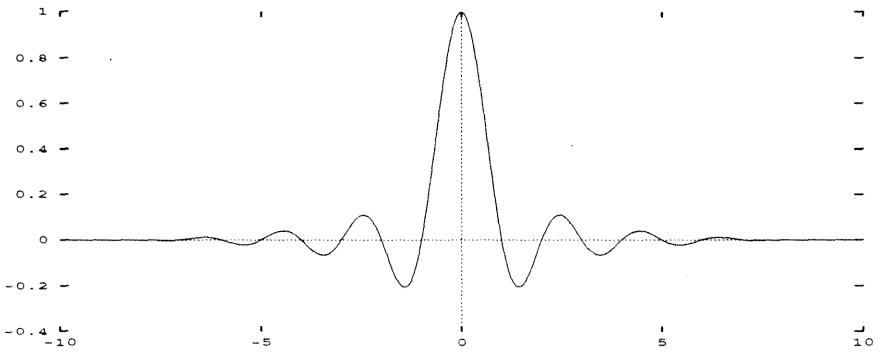


Figure 6.7: Lanczos filter is a windowed sinc function. The zeros of the filter occur at the centers of neighboring pixels.

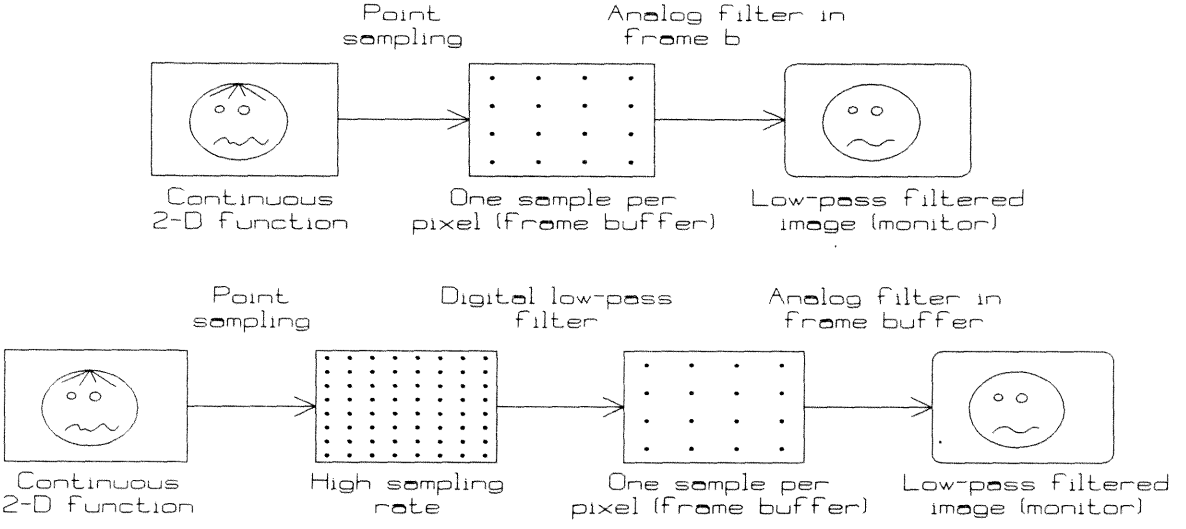


Figure 6.8: Subsampling schematics. The upper figure illustrates the process of rendering a 4×4 image by sampling at one point per pixel. The lower figure illustrates the process of rendering the same image via subsampling. The intermediate step of sampling at a higher sampling rate reduces aliasing artifacts.

Magnitude of Aliasing with Subsampling

So far this section has demonstrated the known result that the magnitude of the aliasing energy decreases using subsampling. Now we will derive the actual magnitude of the aliasing energy as a function of the amount N of subsampling performed. Let $f^N_{\text{alias}}(\omega)$ be the aliased version of $f(\omega)$, subsampled by a factor of N. Figure 6.4 shows $f(\omega)$ with dotted lines and $f^N_{\text{alias}}(\omega)$ with solid lines. To determine the actual amount of aliasing energy, we simply sum all the pieces of the $f(\omega)$ as they fold back on themselves to create $f^N_{\text{alias}}(\omega)$. By folding $f(\omega)$ at frequency $N\nu$, we arrive at

$$f^N_{\text{alias}}(\omega) = \frac{1}{\omega} + \sum_{k=1}^{\infty}\left[\frac{1}{\omega + 2N\nu k} + \frac{1}{\omega - 2N\nu k}\right] = \frac{1}{\omega} + \sum_{k=1}^{\infty}\left[\frac{2\omega}{\omega^2 - (2N\nu)^2 k^2}\right] \tag{6.1}$$

$$= \frac{1}{\omega} + \frac{\omega}{2(N\nu)^2}\sum_{k=1}^{\infty}\left[\frac{1}{(\frac{\omega}{2N\nu})^2 - k^2}\right].$$

The first term, $\frac{1}{\omega}$, is the ideal result. If it were possible to render a discontinuity without aliasing, the first term alone would be the correct answer. The second term is the error term, which arises due to high spectral energy aliasing itself to appear as low spectral energy. If we can show that the value of the sum decreases monotonically as N increases, then we can conclude that the error term falls off as $\frac{1}{N^2}$ as the subsampling factor N is increased.

For the sake of simplicity, we assume that the digital filter employed in the subsampling process is an ideal low-pass filter which cuts off at frequency ν. The low-pass filtered $f_{\text{alias}}(\omega)$ assumes the value zero outside of that interval $\omega \in [-\nu, \nu]$. Within the interval, Figure 6.9 shows the sum

$$\sum_{k=1}^{\infty}\left[\frac{1}{(\frac{\omega}{2N\nu})^2 - k^2}\right]$$

for values of $N = 1, 2, 4, 8$, and 16. The lower curve shows the amount of aliasing as a function of ω, when $N = 1$. For that curve, the term $\frac{\omega}{2N\nu}$ assumes values in $[-\frac{1}{2}, \frac{1}{2}]$.

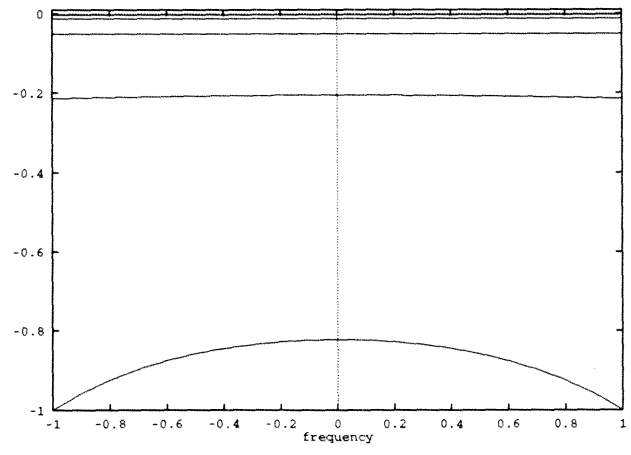


Figure 6.9: Summing the aliasing energy. The sum $\frac{1}{2N^2}\sum_{k=1}^{\infty}\frac{1}{(\frac{\omega}{2N\nu})^2-k^2}$, for values of $N = 1, 2, 4, 8$, and 16, from the lowest curve to the highest curve, respectively. The frequency axis is in units of $\frac{\omega}{\nu}$.

The upper curve, corresponding to $N = 16$, shows that, as $\frac{\omega}{2N\nu}$ vanishes with respect to k^2, the sum approaches

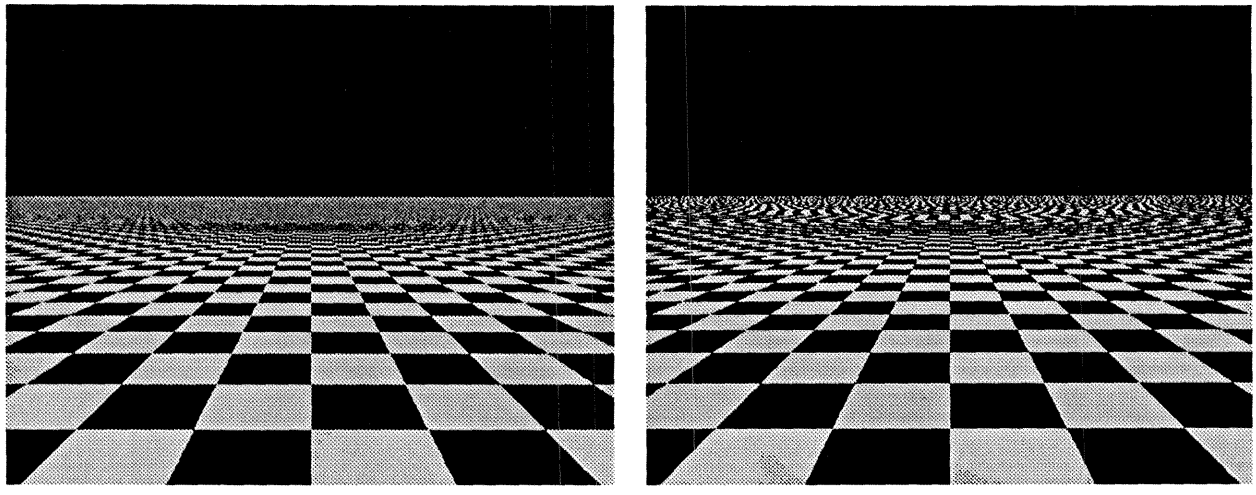
$$\sum_{k=1}^{\infty} -\frac{1}{k^2} \approx -1.64493.$$

Thus, as the subsampling factor N increases, the value of sum decreases monotonically. In the worst case (with no subsampling, $N = 1$), the sum takes maximal absolute value of 2.0 when $\omega = \nu$. Therefore, the aliasing energy is bounded by $\frac{1}{N^2\nu}$.

6.6 Jitter Sampling and Class II Spectral Energy

The previous section showed that it is possible to render a specific subset of computer graphics image functions. As long as the image functions are carefully chosen to be Class I, the spectral energy at higher frequencies is well behaved, and images can be rendered with a controllable amount of aliasing energy.

In the early years of computer graphics, people were careful to avoid Class II image functions because known subsampling techniques produced images which ex-



(a) (b)

Figure 6.10: Class II aliasing. The checkerboard image function (a), when undersampled, results in an aliased version of the checkerboard (b). Class II aliasing artifacts are called "Moiré patterns."

hibited large aliasing artifacts. Cook *et al.* [11] applied a Monte Carlo technique called *stochastic sampling* to computer graphics and showed that Class II image functions could be rendered effectively. This section will consider one type of stochastic sampling called *uniform jitter sampling*, and derives a mathematical explanation as to why it works.

The spectra of Class II image functions do not exhibit the nice behavior as seen in the Class I case. It is quite simple to design a scene in which there exists spectral energy at arbitrarily high frequencies. Consider the image function of an infinite check in Figure 6.10(a).[1] The spectrum of the image contains energy at *every* frequency of magnitude comparable to that at any other frequency.

Jitter sampling is a sampling process very similar to regular sampling, except that the position of the sample point is jittered before an actual sample is taken. The jittered position of each sample is computed by adding a random variable to the

[1]It is, of course, not possible to include an actual picture of an image function. Any presentation here must by definition involve some sampling process. While not strictly correct, the figure is included here to aid in the presentation.

original position of each sample location. (For two-dimensional images, a random number is added to each of the x- and y-components of the original sample position. This paper concerns itself with just one of the dimensions.)

Various types of jitter might be considered, such as Gaussian, Poisson and uniform jitter. We will consider only uniform jitter sampling in this section for reasons which will become clear in the following section.

Engineers need to consider jitter sampling, but for a reason very different from that of computer graphics. Typical real-world sampling processes often involve an amount of uncertainty as to the actual time or position of each sample. To properly analyze a system which contains sampling uncertainties, engineers model the sampling process as that of jitter sampling. They do so because, in the real world, jitter is an unavoidable nuisance. In contrast, in computer graphics jitter sampling is performed because it yields an elegant solution to the problem of sampling Class II image functions.

Figure 6.12 shows schematically the uniform jitter sampling process. The image function can be jitter sampled at frequency ν as in (a). For reasons which will become clear soon, it is advantageous to perform uniform jittered *sub*sampling, as in (b).

To analyze jitter sampling, we define $J_a^b(f(x))$ to be an operator that jitter samples the signal $f(x)$ at a point a, with jitter magnitude b. In other words, a jitter sample $J_a^b(f(x))$ is the value of the function $f(x)$ evaluated in the interval $x \in [a - \frac{b}{2}, a + \frac{b}{2}]$.

Consider a signal composed of a single sine wave $s(x) = e^{i\omega x + \phi}$ with frequency ω, and phase ϕ. Using sampling period T, the sample at position a, obtained by jitter sampling with uniform jitter of magnitude kT, yields the result,

$$\tilde{s}(a) = E[J_a^{kT}(s(x))] = \frac{1}{kT}\int_{a-\frac{kT}{2}}^{a+\frac{kT}{2}} e^{i\omega x+\phi}dx = \frac{e^{\phi}}{i\omega kT}\left[e^{i\omega(a+\frac{kT}{2})} - e^{-i\omega(a+\frac{kT}{2})}\right]$$

$$= e^{i\omega a+\phi}\frac{1}{i\omega kT}\left[e^{i\omega\frac{kT}{2}} - e^{-i\omega\frac{kT}{2}}\right] = e^{i\omega a+\phi}\frac{2}{\omega kT}\frac{\left[e^{i\omega\frac{kT}{2}} - e^{-i\omega\frac{kT}{2}}\right]}{2i} \qquad (6.2)$$

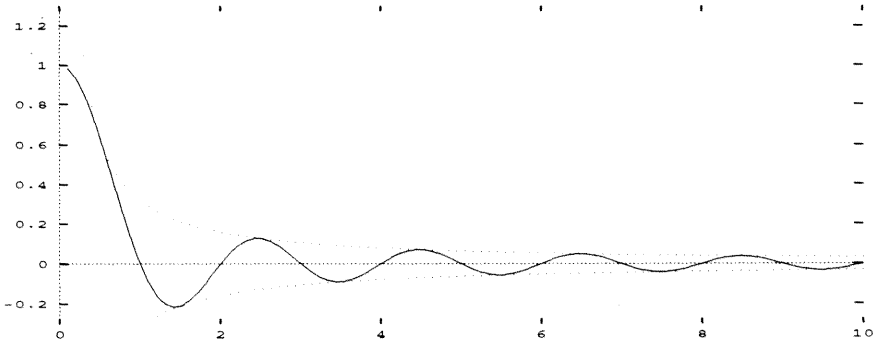


Figure 6.11: Sinc function. The attenuation of a sine wave of frequency ω. The horizontal axis represents the frequency ω, and is in units of $\frac{2\pi}{kT}$. The first zero crossing occurs when $\omega = \frac{2\pi}{kT}$, which occurs when an entire cycle of the sine wave fits exactly within one sampling interval. The dotted line is a graph of the function $\pm\frac{1}{\omega}$, which strictly bounds the attenuation function.

$$= e^{i\omega a+\phi}\frac{2}{\omega kT}\sin\left[\omega\frac{kT}{2}\right] = s(a)\mathrm{sinc}\left[\omega\frac{kT}{2}\right].$$

The signal $\tilde{s}(x)$ is the same as the original signal $s(x)$ with the spectrum attenuated by $A_{kT}(\omega) = \mathrm{sinc}\left[\omega\frac{kT}{2}\right]$. $A_{kT}(w)$ is independent of the phase ϕ of the input signal.

The jitter sampling of individual sine waves behaves as expected. As the period of $s(x)$, which is $\frac{2\pi}{\omega}$, increases with respect to the sampling period T, the product ωT approaches zero, the attenuation term $A_{kT}(\omega)$ approaches unity, and $\tilde{s}(x)$ approaches $s(x)$. Low-frequency sine waves are left essentially undisturbed by the jitter sampling process. At the other end of the spectrum, as the period of $s(x)$ decreases with respect to T, ωT approaches infinity, and $A_{kT}(\omega)$ approaches zero. Sine waves of very high frequencies are completely attenuated by the jitter sampling process.

Figure 6.11 shows the attenuation of the single frequency sine wave as a function of the frequency ω. For any given input frequency ω, the graph in Figure 6.11 can be used to determine the amount of attenuation caused by a signal of that frequency when jitter sampling is applied.

The above proof is generalized when we consider an image $f(x)$ and its Fourier transform $f(\omega)$. The inverse Fourier transform of $f(\omega)$ is

$$f(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(\omega)e^{i\omega x}d\omega. \tag{6.3}$$

We now explore the effect of uniform jitter sampling on the signal $f(x)$ in the same manner that we studied its effect on a single sine wave. To do so, the single sine wave $s(x)$ in the derivation of Equation 6.2 is replaced with $f(x)$ represented as the inverse Fourier transform integral of Equation 6.3.

$$\tilde{f}(a) = E[J_a(f(x))] = \frac{1}{kT}\int_{a-\frac{kT}{2}}^{a+\frac{kT}{2}} f(x)dx = \frac{1}{kT}\int_{a-\frac{kT}{2}}^{a+\frac{kT}{2}} \frac{1}{2\pi}\int_{-\infty}^{\infty} f(\omega)e^{i\omega x}d\omega dx.$$

After changing the order of integration, the resulting ω-integrand is essentially identical to that found in Equation 6.2. Applying a derivation similar to that of Equation 6.2 yields

$$\tilde{f}(a) = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(\omega)\frac{1}{kT}\int_{a-\frac{kT}{2}}^{a+\frac{kT}{2}} e^{i\omega x}dxd\omega$$

$$= \frac{1}{2\pi}\int_{-\infty}^{\infty} f(\omega)\text{sinc}\left[\omega\frac{kT}{2}\right]e^{i\omega a}d\omega = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(\omega)A_{kT}(\omega)e^{i\omega a}d\omega.$$

The spectrum of the signal $\tilde{f}(x)$ is the same as the spectrum of $f(x)$ multiplied by $A_{kT}(\omega)$,

$$\tilde{f}(\omega) = f(\omega)A_{kT}(\omega). \tag{6.4}$$

The same result can be derived more directly using the convolution theorem. By defining the box

$$B(x) = \begin{cases} 1 & \text{if } a - \frac{kT}{2} < x < a + \frac{kT}{2} \\ 0 & \text{otherwise} \end{cases},$$

the integral

$$\int_{a-\frac{kT}{2}}^{a+\frac{kT}{2}} f(x)dx$$

can be rewritten as

$$\int_{-\infty}^{\infty} f(x)B(t-x)dx = f(x) * B(x),$$

the convolution of $f(x)$ and $B(x)$. By the convolution theorem, the spectrum of $\tilde{f}(x)$ is the product of the Fourier transforms of $f(x)$ and $B(x)$. The Fourier transform of $B(x)$ is the sinc function $A_{kT}(\omega)$.

Reduced to a Previous Problem

Consider a Class II image function $f(x)$. We can compute $\tilde{f}(a) = E[J_a^{kT}(f(x))]$ from $f(x)$ via jitter sampling, thereby attenuating the spectrum of $f(x)$, so that the spectrum of $\tilde{f}(x)$ exhibits $\frac{1}{\omega}$ behavior similar to that of Class I image functions. To render $\tilde{f}(x)$ correctly, then, we can refer to Section 6.5, which discussed the technique of subsampling for rendering Class I image functions.

By combining jitter sampling with subsampling, we arrive at an effective method of rendering Class II image functions. If subsampling alone were used, the original image function $f(x)$ would be sampled on a fixed sampling grid. In the combined process, $\tilde{f}(x)$ is substituted for $f(x)$.

A point sample of $\tilde{f}(x)$ at position a is simply $\tilde{f}(x)$ evaluated at a. The computation of $\tilde{f}(a)$ from $f(x)$ requires integrating $f(x)$ in a neighborhood of a. We approximate the integral by averaging M uniform jitter samples of $f(x)$ chosen from the interval $[a - \frac{kT}{2}, a + \frac{kT}{2}]$. The combined jittered subsampling technique uses such integration to compute each subsample and then digitally low-pass filters the subsamples to produce an image at the frame buffer resolution.

We must modify our analysis of the effects of jitter to accommodate subsampling. The jittering that was performed over a fraction k of a pixel must instead be performed over the same fraction of a *sub*pixel. The change is accomplished by simply substituting $\frac{kT}{N}$ for kT in the analysis of jitter sampling. The result restated is that jittered sampling over a subpixel causes spectral attenuation $A_{\frac{kT}{N}}(\omega) = \operatorname{sinc} \omega \frac{\mathrm{kT}}{\mathrm{2N}\omega}$, which is enclosed by a $\frac{2N\omega}{kT}$ envelope.

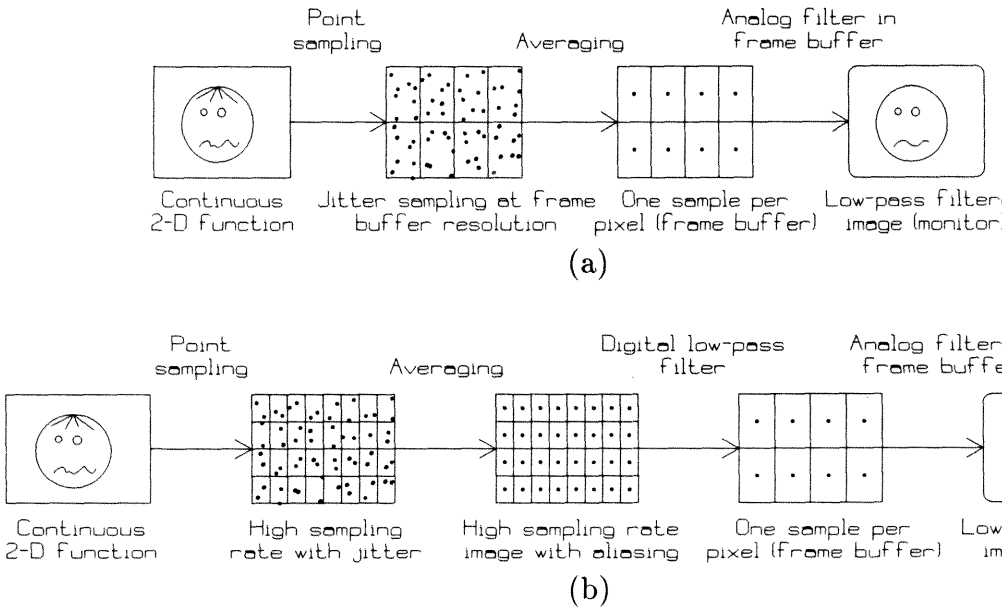


Figure 6.12: Schematic of jittered subsampling. (a) Illustrates the process of jittered sampling to create a 4×2 image. To compute $\tilde{f}(x)$, the integral over each pixel is estimated using eight jittered samples. (b) Illustrates the process of jittered subsampling. An 8×4 image is sampled using jitter sampling and then low-pass filtered to create a 4×2 image. To compute $\tilde{f}(x)$, the integral over each subpixel is estimated using two jittered samples.

Equation 6.1 demonstrated that, for factor of N subsampling, a $\frac{1}{\omega}$ signal exhibits order $\frac{1}{N^2}$ aliasing energy. Thus, combining jittering and subsampling results in order $\frac{1}{N}$ aliasing energy.

The process of jittered subsampling reduces aliasing energy. The jittered sampling process involves an ensemble average to compute $\tilde{f}(\omega)$ from $f(\omega)$, and the subsampling process involves a spatial average to band width limit the subsampled image function. The next section discusses the tradeoff between the two averages.

Variance Reduction versus Subsampling

Figures 6.12(a) and (b) illustrate the process of jittered sampling with and without subsampling. In Figure 6.12(a) no subsampling is performed, and the integral to compute $\tilde{f}(x)$ is approximated using $M = 8$ jittered samples per pixel. In Figure 6.12(b),

subsampling with $N = 2$ is performed, and the integral is approximated using $M = 2$ jittered samples per subpixel.

Both processes illustrated in Figure 6.12 use the same number of samples of $f(x)$ (64) to compute the final 4×2 image. According to the theory presented in previous sections, which process in Figure 6.12 will have less aliasing energy? Figure 6.12(a) computes a better approximation of $\tilde{f}(x)$ at each sample point. However, no subsampling is employed, so aliasing energy will be large. Figure 6.12(b), on the other hand, uses a poorer approximation to $\tilde{f}(x)$ at each sample point, but subsampling is employed, so aliasing should be greatly reduced.

We have already explained that subsampling reduces aliasing energy, and thereby improves the rendering process. In practical terms, subsampling has a cost. Samples dedicated to the subsampling process reduce the samples available to estimate the integral $\tilde{f}(x)$. To determine which sampling scheme is better, we must consider how the quality of the integration effects the outcome of the rendering process. To answer the question of which scheme in Figure 6.13 to use, we need to know how a better estimate of the convolution integral effects aliasing. The answer is that a better estimate of the convolution integral only affects the amount of noise in the final image, but it in no way affects the amount of aliasing energy.

In practice, a limited amount of computing resources is available to render an image. Given that a fixed number of samples are to be used per pixel, how can that budget be best spent? Figure 6.13 demonstrates a range of the possible ways to spend a budget of 64 samples within a single pixel. Figure 6.13(a) illustrates the use of all 64 samples to estimate the value of $\tilde{f}(x)$ at the center of each pixel. In that case, no budget is available to subsample. At the other extreme, Figure 6.13(d) illustrates the process of using only a single sample to estimate each integral, while reserving the entire budget for subsampling. How do we choose among the various alternatives?

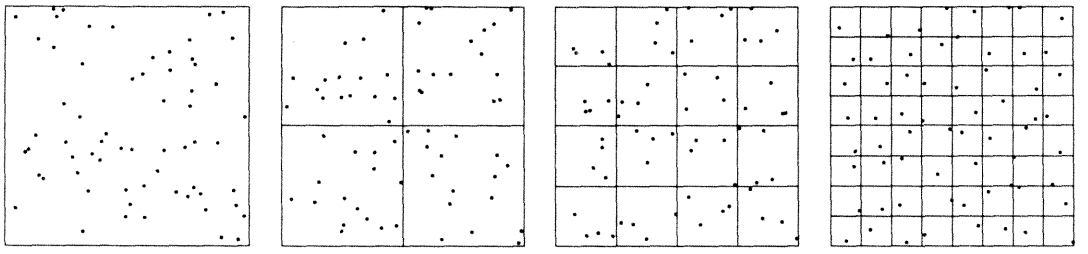


Figure 6.13: Pixel budget. A budget of 64 samples per pixel can be used in at least four different ways. In pixel (a), no subsampling is performed, and the rendering process is that of convolving the spectrum with a sinc function. Pixels (b), (c), and (d) show different levels of stratification, and the convolution kernel approaches that of an ideal box.

When we use a Monte Carlo method, it is important to design a process which is a *consistent estimator* of the underlying problem. A consistent estimator is one whose behavior converges to the correct answer as the number of samples increases.

The process indicated by Figure 6.13(a) will never converge to a correct answer; an infinite number of samples will yield perfect integration results, so the resulting $\tilde{f}(u, v)$ will exhibit $\frac{1}{\omega}$ spectral energy. Despite the high quality integration, the absence of subsampling results in an image which exhibits terrible aliasing artifacts.

The process indicated by Figure 6.13(d) does converge to the correct answer. As the number of samples increases, the amount of aliasing decreases as $\frac{1}{N}$. In the end, however, the subsampled image will exhibit a large amount of noise. Fortunately, sampling is not the last step in the subsampling process. The subsampled image must be low-pass filtered to produce the final image. If we treat the sampling process as an ergodic process, then the spatial average performed by the low-pass filter not only reduces the band width, but also acts as an ensemble average, thereby reducing the noise.

The process indicated by Figure 6.13(d), when considered in concert with the low-pass filter, constitutes a consistent estimator with respect to both aliasing energy and

noise. Therefore, it is always better to maximize the amount of subsampling, thereby erring on the side of reducing aliasing. The noise will be reduced in the second half of the subsampling process.

Better Than $\frac{1}{\omega}$ Attenuation

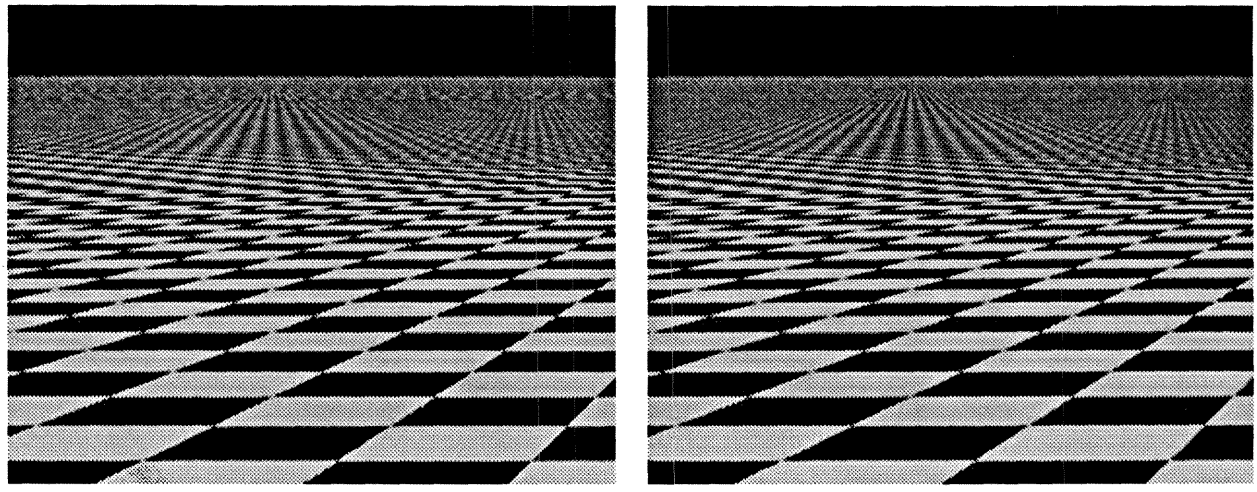
Depending on the particular perspective transformation, image functions of three-dimensional models which contain repetitive features often exhibit behavior known as the *picket fence effect.* As one follows the repetitive feature further and further to the horizon, the local frequency spectrum contains a single dominant wavelength which increases without bound.

At some point, the dominant wavelength of such images will approximately equal the Nyquist frequency $\frac{2\nu}{N}$ corresponding to the subsampling frequency. At that particular position in the image function, the harmonics near the Nyquist limit all alias to roughly the same lower frequencies and beat against each other.

The checkerboard image in Figure 6.14(a) demonstrates the picket fence effect. While aliasing appears at many places in the image, the most-visible aliasing is located in the distance, at the point in which the frequency of the checks nears the Nyquist frequency of the subsampling grid.

Our previous analysis bounds the aliasing energy to order $\frac{1}{N}$, where N is the subsampling factor. Figure 6.14(a) demonstrates that $\frac{1}{N}$ aliasing energy yields objectionable aliasing artifacts even for large values of N. In special cases, by looking more closely at the shape of the attenuation function $A_{kT}(\omega)$, it is possible to reduce the aliasing energy to unobjectionable levels, as in Figure 6.14(b).

Consider a region of the image function in Figure 6.14(a) consisting of the scan lines in the neighborhood of the Moiré pattern. Figure 6.15 schematically shows the spectrum of the region. After $A_{kT}(\omega)$ attenuation, a peak at the dominant wavelength



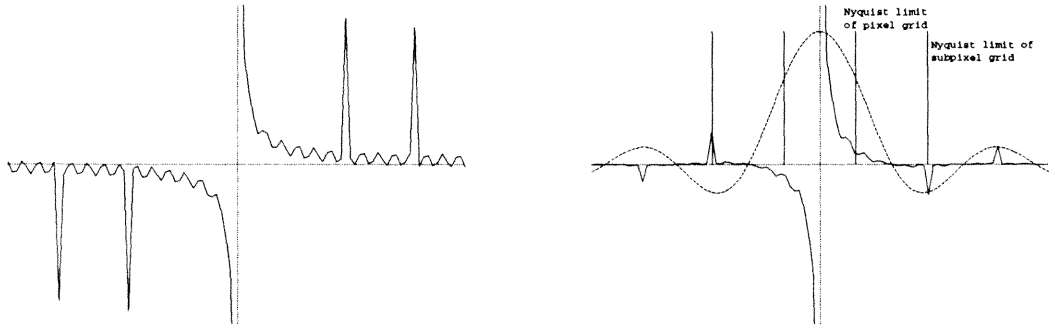
(a) (b)

Figure 6.14: Differing amounts of jitter. (a) Half-subpixel jitter ($k = 0.5$), 10×10 subsampling. (b) Full-subpixel jitter ($k = 1.0$), 5×5 subsampling. Theoretically, (a) aliasing energy in (a) is attenuated by twice as much as (b). The aliasing energy in (b) just happens to fall at a zero crossing of the attenuation function.

survives. When the signal is sampled, the peak will add significant aliasing energy to the otherwise low energy signal, thereby causing the Moiré patterns in the figure.

Figure 6.15(b) shows the same image function rendered the same way as Figure 6.15(a), except that the k parameter, which controls the amount of jitter, was changed from 0.5 to exactly 1.0. At $k = 1.0$, the jitter sampling process is free to choose samples placed anywhere within the subpixel. The corresponding attenuation function $A_{kT}(\omega)$ has zero crossings coinciding exactly with the harmonics of the Nyquist frequencies in Figure 6.15(a), and the spectral energy peak gets attenuated completely. (This point was made in [13], but the conclusions drawn were incorrect.)

To summarize, if the jitter magnitude $k = 1.0$, and the image function being rendered happens to have a pattern that contributes a single dominant frequency, then the zeros of the attenuation function $A_{kT}(\omega)$ will coincide with the dominant frequency peaks right where those peaks would otherwise cause aliasing artifacts.



(a) (b)

Figure 6.15: Spectrum near Nyquist limit.(a) The spectrum of a region of Figure 6.14(a). (b) The spectrum attenuated by $A_{kT}(\omega)$. A dominant wavelength survives at the subpixel Nyquist frequency. After aliasing, the dominant wavelength contributes significant aliasing energy to the otherwise low energy signal.

It is possible to create an image function which has multiple dominant peaks, and which would cause Moiré patterns at a location other than the Nyquist limit. For that image function, a jitter magnitude other than $k = 1.0$ might produce a better rendering.

Therefore, the best setting for the parameter k depends on the nature of the image function being rendered. For an arbitrary image function, it would require a large amount of analysis to determine the best value. Furthermore, the behavior demonstrated by the infinite checkerboard is by far the most common type of repetitive pattern found in typical computer graphics image functions, so we conclude that the setting $k = 1.0$ will provide the best overall performance for a computer graphics renderer.

Historically, the infinite checkerboard has been a popular test case. Many computer graphics researchers have presented papers which attempt to improve rendering performance, and they use the infinite checkerboard as a test case [24] [13] [16]. Notably, few researchers have posed the question, "Independent of speed, what does it

take to render a correct infinite checkerboard without aliasing artifacts?" Figure 6.16 shows an infinite checkerboard rendered with full-subpixel jitter and 10×10 subsampling. Our understanding of the jitter sampling process has allowed us to produce an image of the infinite checkerboard free from aliasing artifacts.

The repeated cloth texel in Chapters 4 and 5 is an infinite checkerboard. The analysis in this chapter proves that the BDRF extraction methods can sample the cloth model from well beyond the texture threshold and arrive at a correct result.

6.7 Chapter Summary

Image functions were classified according to the nature of their spectral content into Class I or Class II image functions. The traditional method for rendering Class I image functions, subsampling, was presented. Although subsampling is well understood, a thorough analysis was presented to lay groundwork for the rest of the chapter.

The process of rendering Class II image functions, jitter sampling, was presented, along with a new and complete analysis of the method. It is shown that the jitter sampling process reduces the problem of rendering a Class II image to the simpler problem of rendering a Class I image function. A tradeoff between variance reduction and subsampling concluded that maximum subsampling should be employed at the expense of variance reduction.

Finally, the process of rendering an infinite checkerboard image function was detailed. Under specific circumstances it is possible to render such repetitive patterns with essentially no aliasing artifacts.

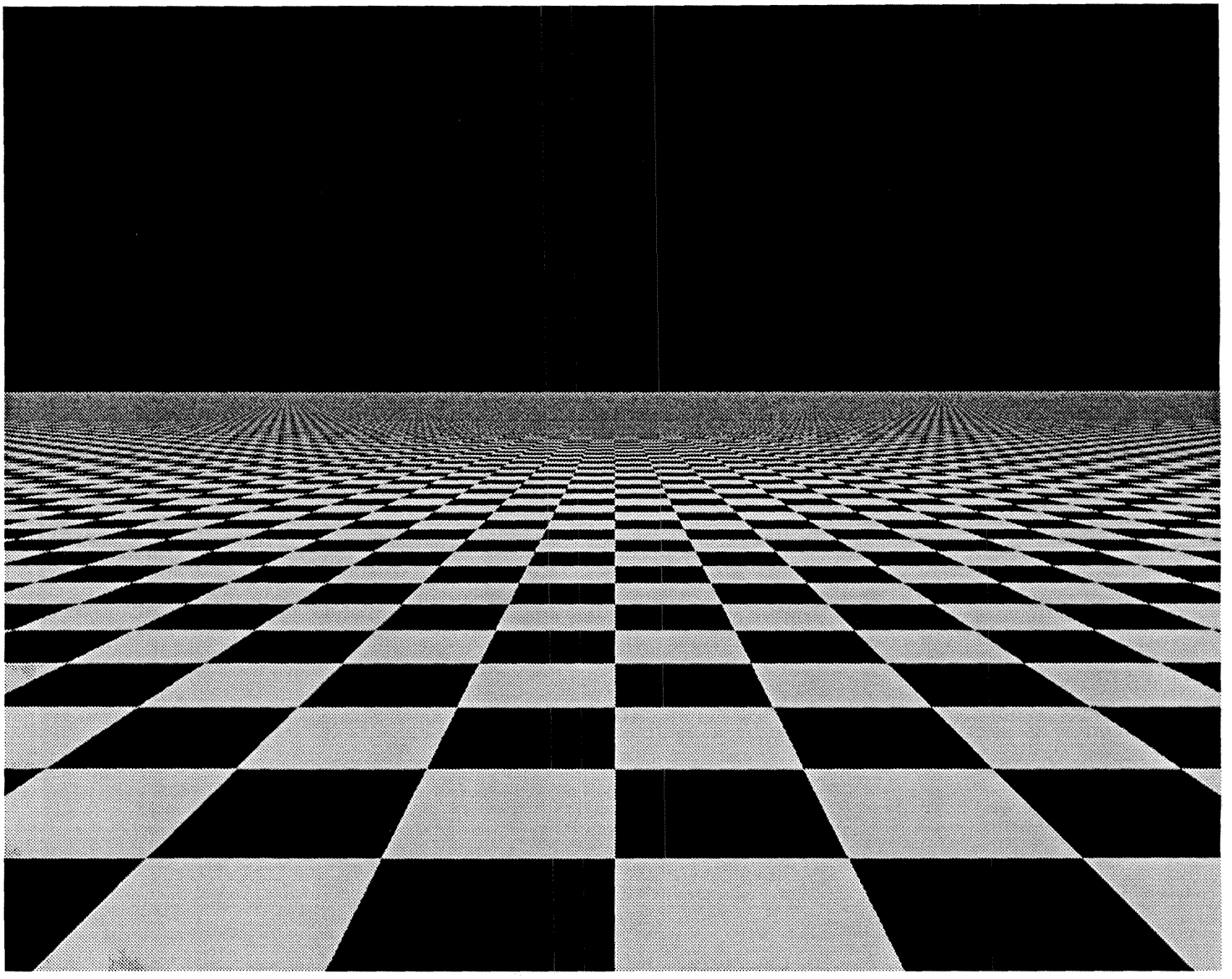


Figure 6.16: Well rendered infinite checkerboard. The infinite checkerboard when rendered with full-pixel jitter, 10×10 subsampling, and one sample per subpixel.

Chapter 7

Conclusion

The quest for realism in computer graphics is propelling model complexity beyond the point where all objects can be stored discretely. Eventually it will become necessary to convert geometric models into textures. This thesis presented several new methods towards that end.

This thesis has demonstrated that texels can realistically represent a range of soft objects. The texel primitive was explored in a series of three experiments, each moving further along the road from geometry to texture. The first experiment joined traditional computer graphics modeling with the texel primitive to create a Teddy bear. In the process, a fur texel was developed in conjunction with an appropriate bidirectional reflectance function (BDRF).

The second experiment left the realm of traditional primitive-based modeling: a complete computer graphics model of a swatch of cloth was created entirely within the confines of a single texel. A relaxation technique for weaving thread into a cloth model was presented.

The third experiment completed the trip from geometry to texture. A newly discovered phenomenon called the *texture threshold effect* was presented, which in essence, defines the point at which geometry turns into texture. Taking advantage of the effect, the BDRF of the cloth model was extracted. The resulting BDRF was

used to shade a polygonal model of a car seat which then took on the appearance of being upholstered.

The rest of this section will discuss points that do not fit well within the main body, and will present ideas for future work.

7.1 Discovery of the Texture Threshold Effect

The principle reason that the texture threshold effect has remained undiscovered until now is that the effect is only readily apparent given a set of cooperative circumstances. The following combination of circumstances is required to observe an effect similar to that in the sequence of images in Figure 5.1:

1. The light must be at infinity, lest it move relative to the eye as the camera pulls back;

2. the model must consist of an infinite plane, which is an uninteresting subject in most cases;

3. the infinite plane must have a texture or BDRF which is view-dependent. Otherwise, with the light at a fixed position, the plane would be forever shaded a constant color; and,

4. the camera must pass through the texture threshold.

These circumstances, while common individually, rarely occur at the same time. It is unlikely that someone would notice the texture threshold effect unless they were studying BDRF extraction techniques specifically.

7.2 Future Work

Fur

We would like to animate the Teddy bear, but the time required to compute each frame currently is prohibitive.

When considering possible objects to model with texels, a large collection of fuzzy objects comes immediately to mind, such as human hair, carpeting, and fields of wheat. This thesis provided a tool for representing such objects. As is typical in computer graphics, modeling objects for the first time often inspires additional modeling research. The mechanics of hair blowing in the wind, for example, has yet to be developed to the point that can be simulated. A good hair dynamics model, combined with texel rendering would make a very realistic animation.

Cloth Modeling and BDRF Extraction

Most cloth weaves can be produced by simply extending the relaxation process to two dimensions. There are two interesting aspects to the work of studying the BDRFs of various weaves. First, it would be nice to know how variations in the weave lead to variations in the BDRF. Different weaves have distinctive textures. If the BDRF extraction process is valid, then there will be differences. A library of cloth-like BDRFs could be created.

A more difficult task would be to discover all parameters necessary to match experiment with reality. If it becomes possible from a description of a weave to render an appearance that matches reality *for that weave*, then one could explore the appearance of types of cloth that have not yet been manufactured.

A much bigger problem, which has been only partially explored by researchers, precludes the widespread use of cloth in computer graphics imagery. The (macro-

scopic) drape of a piece of cloth subject to physical constraints has yet to be fully solved. Until it is possible to simulate the drape of cloth, a library of cloth BDRFs is of limited usefulness: upholstery and carpeting can be realistically modeled, but not clothing or curtains.

In addition to cloth, many other surfaces should be easy to model and would be useful to study, such as human hair or a field of grass or wheat. As with the drape of cloth, the problem of modeling the macroscopic geometry as it waves and blows in the wind has yet to be solved.

Sampling

In Chapter 6 the analysis of jittered subsampling demonstrates that a computer graphics renderer can control the amount of aliasing energy in an image. This result sufficiently allays concerns that the BDRF extraction process of Chapter 5 might suffer due to uncontrollable aliasing. The chapter did not, however, expand on the tradeoff that was made to control the energy: jittering sampling causes a large amount of noise to be added to the rendered image. An analysis of such noise which, at the very least, bounds the magnitude of the noise is left to future work.

7.3 Images and Source Code

The images in this thesis were created using a variety of programs, the state of which can be kindly termed "research" quality; they could never be wielded effectively in hands other than those of the author.

As the images in this thesis do not print/photocopy/microfilm well, the Postscript thesis, with the images embedded therein, are available via anonymous ftp from `ftp.cs.caltech.edu`. At the time of this writing, I can be reached via electronic mail at `tim@caltech.edu`.

Bibliography

[1] A.V. Balakrishnan. On the problem of time jitter in sampling. *IRE Transactions on Information Theory*, pages 226–236, April 1962.

[2] Alan H. Barr. Private communication., 1985.

[3] Frederick J. Beutler. Alias-free randomly timed sampling of stochastic processes. *IEEE Transactions on Information Theory*, IT-16(2):147–152, March 1970.

[4] James F. Blinn. Models of light reflection. *ACM SIGGRAPH '77 Conference Proceedings*, 11(2):286–292, 1977.

[5] James F. Blinn. Light reflection functions for simulation of clouds and dusty surfaces. *ACM SIGGRAPH '82 Conference Proceedings*, 16(3):21–29, 1982.

[6] William M. Brown. Sampling with random jitter. *J. Soc. Indust. Appl. Math.*, 11(2):460–473, June 1963.

[7] Brian Cabral, Nelson Max, and Rebecca Springmeyer. Bidirectional reflection functions from surface bump maps. *ACM SIGGRAPH '87 Conference Proceedings*, 21(4):273–281, 1987.

[8] Malcolm S. Casale and Edward L. Stanton. An overview of analytic solid modeling. *Computer Graphics and Applications*, 5(2):45–56, Februrary 1985.

[9] Robert Cook and Kenneth Torrance. A reflectance model for computer graphics. *ACM SIGGRAPH '81 Conference Proceedings*, 15(3):307–316, 1981.

[10] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Transactions on Graphics*, 5(1):51–72, January 1986.

[11] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. *ACM SIGGRAPH '84 Conference Proceedings*, 18(3):137–145, July 1984.

[12] C. Csuri, R. Hakathorn, R. Parent, W. Carlson, and M. Howard. Towards an interactive high visual complexity animation system. *Computer Graphics*, 16(3):289–299, 1979.

[13] Mark A. Dippé and Erling Henry Wold. Antialiasing through stochastic sampling. *ACM SIGGRAPH '85 Conference Proceedings*, pages 69–78, 1985.

[14] Robert Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. *ACM SIGGRAPH '88 Conference Proceedings*, 22(4):65–74, August 1988.

[15] S. Gabriel. private communication, 1985.

[16] Paul S. Heckbert. Filtering by repeated integration. *ACM SIGGRAPH '86 Conference Proceedings*, 20(4):315–321, 1986.

[17] James T. Kajiya. The rendering equation. *ACM SIGGRAPH '86 Conference Proceedings*, 20(4):143–150, August 1986.

[18] James T. Kajiya and Brian Von Herzen. Ray tracing volume densities. *ACM SIGGRAPH '84 Conference Proceedings*, 18(3):165–173, 1984.

[19] James T. Kajiya and Timothy L. Kay. Rendering fur. *ACM SIGGRAPH '89 Conference Proceedings*, 23(3):271–280, 1989.

[20] Elias Masry. Alias-free sampling: An alternative conceptualization and its applications. *IEEE Transactions on Information Theory*, It-24(3):317–324, May 1978.

[21] Nelson L. Max. Atmospheric illumination and shadows. *ACM SIGGRAPH '86 Conference Proceedings*, 20(4):117–124, 1986.

[22] Sachie Minato. Color and gloss. *J. of Color Science Association of Japan*, 4(1):29–30, 1979.

[23] T. Nishita, Y. Miyawaki, and E. Nakamae. A shading model for atmospheric scattering considering luminous intensity distribution of light sources. *ACM SIGGRAPH '87 Conference Proceedings*, 21(4):303–310, 1987.

[24] A. Norton, A. Rockwood, and P. Skolmoski. Clamping: A method of antialiasing textured surfaces by bandwidth limiting in object space,. *ACM SIGGRAPH '82 Conference Proceedings*, 16(3):1–8, 1982.

[25] Joseph P. Pavlovich and Thomas E. Tahan. *Computer Programming in BASIC.* Holden-Day, Inc., San Francisco, 1971.

[26] Ken Perlin. Hypertexture. *ACM SIGGRAPH '89 Conference Proceedings*, 23(3):253–262, 1989.

[27] M. David Potter and Bernard P. Corbman. *Textiles: Fiber to Fabric.* Gregg Division/McGraw-Hill Book Company, New York, 1967.

[28] William H. Press, Brian P. Flannery, Saul a. Teukolsky, and William T. Vetterling. *Numerical Recipes in C.* University Press, Cambridge, 1988.

[29] S. M. Rubin and T. Whitted. A 3-dimensional representation for fast rendering of complex scenes. *ACM SIGGRAPH '80 Conference Proceedings*, pages 110–116, 1980.

[30] Holly E. Rushmeyer and K. Torrance. The zonal method for calculating light intensities in the presence of participating medium. *ACM SIGGRAPH '87 Conference Proceedings*, 21(4):293–302, 1987.

[31] P. Sabella. A rendering algorithm for visualizing 3d scalar fields. *ACM SIGGRAPH '88 Conference Proceedings*, 22(4):51–58, August 1988.

[32] Harold S. Shapiro and Richard A. Silverman. Alias-free sampling of random noise. *J. Soc. Indust. Appl. Math*, 8(2), June 1960.

[33] John M. Snyder. *Generative Modeling: An Approach to High Level Shape Design for Computer Graphics and CAD.* PhD thesis, California Institute of Technology, 1991.

[34] Atsushi Takagi, Hitoshi Takaoka, Tetsuya Oshima, and Yoshinori Ogata. Accurate rendering technique based on colorimetric conception. *ACM SIGGRAPH '90 Conference Proceedings*, 24(4):263–272, August 1990.

[35] C. Upson and M. Keeler. Vbuffer: Visible volume rendering. *ACM SIGGRAPH '88 Conference Proceedings*, 22(4):69–65, August 1988.

[36] Jerry Weil. The synthesis of cloth objects. *ACM SIGGRAPH '86 Conference Proceedings*, 20(4):49–54, August 1986.