Examensarbete
LITH-ITN-MT-EX--06/033--SE

Multiple scattering in participating media using Spherical Harmonics

Johan Åkesson

2006-06-05

Linköpings universitet
TEKNISKA HÖGSKOLAN

Department of Science and Technology
Linköpings Universitet
SE-601 74 Norrköping, Sweden

Institutionen för teknik och naturvetenskap
Linköpings Universitet
601 74 Norrköping

LITH-ITN-MT-EX--06/033--SE

Multiple scattering in participating media using Spherical Harmonics

Examensarbete utfört i medieteknik
vid Linköpings Tekniska Högskola, Campus Norrköping

Johan Åkesson

Handledare Doug Roble and Nafees Bin Zafar
Examinator Ken Museth

Norrköping 2006-06-05

Avdelning, Institution
Division, Department

Institutionen för teknik och naturvetenskap

Department of Science and Technology

Datum
Date

2006-06-05

Språk
Language

☐ Svenska/Swedish
☒ Engelska/English

☐ _______________

Rapporttyp
Report category

Examensarbete
☐ B-uppsats
☐ C-uppsats
☒ D-uppsats

☐ _______________

ISBN

ISRN **LITH-ITN-MT-EX--06/033--SE**

Serietitel och serienummer
Title of series, numbering

ISSN

URL för elektronisk version

Titel
Title

Multiple scattering in participating media using Spherical Harmonics

Författare
Author

Johan Åkesson

Sammanfattning
Abstract

In computer graphics, and in digital visual effects, many scenes contain natural phenomena that cannot be modelled with traditional solid object methods. Because of the complex interaction with light, rendering of these phenomenon has been an computationally demanding task and simplifications has to made. One of the most crippling simplifications has been to assume only single scattering of light.

This thesis presents a new method of approximating multiple scattering in volumes in a low frequency environment. The method provides a light independent solution by using a transfer function, which enables fast re-lighting. It uses a spherical Harmonics representation of the transfer function.

Nyckelord
Keyword

computer graphics, rendering, scattering, multiple scattering, volume rendering, spherical harmonic

Abstract

In computer graphics, and in digital visual effects, many scenes contain natural phenomena that cannot be modelled with traditional solid object methods. Because of the complex interaction with light, rendering of these phenomenon has been an computationally demanding task and simplifications has to made. One of the most crippling simplifications has been to assume only single scattering of light.

This thesis presents a new method of approximating multiple scattering in volumes in a low frequency environment. The method provides a light independent solution by using a transfer function, which enables fast re-lighting. It uses a Spherical Harmonics representation of the transfer function.

Acknowledgements

In loving memory of my grandfather Magni Johnsson, who passed away during my stay abroad. Magni was a great inspiration to me and a big reason to why I got started with computers. He will be greatly missed.

I would like to thank everyone at Digital Domain for making me feel welcome at the company, and especially my supervisors Doug Roble and Nafees Bin Zafar. You guys rock! A big thanks to Ken Museth, my academic supervisor, for selecting me for this internship and his encouragement, support and guidance.

I would also like to thank my parents, my sister, grandfather and grandmother for their support, encouragement and financial help, without whom this internship would not have been possible.

I would also like to acknowledge Peter-Pike Sloan at Microsoft Research, Dr. David Ebert at Purdue University and Dr. Ravi Ramamoorthi at Columbia University for their guidance and help in the spherical harmonic jungle.

Finally I would like to thank all the Swedes from Norrköping living in LA, you make us all look good over there, Kjell for encouragement and advice, Mattias and Sara, Emma, Mattias S and all my other friends in Sweden, my roommates in America Erick the inventor and Matt, and of course Måndagsklubben.

Contents

Chapter 1

Introduction

This thesis was a part of an internship at Digital Domain, a visual effects production house located in Los Angeles, USA. The company has been involved in many motion picture productions from *Titanic*, *Apollo 13*, *Fifth Element*, to more recent *I,Robot* and *Stealth* as well as numerous commercials. The company has a long history of digital effects production and development. In 1998 Doug Roble won an Technical Academy Award for the *track*, the companys computer vision tool, and in the following year the software package *Storm* earned the company and it's creator Alan Kapler yet another Technical Academy Award.

The rest of this chapter will serve as an introduction to the software used at Digital Domain, a general problem description and a presentation of the conventions used throughout this report. Chapter 2 will introduce the concept of Spherical Harmonics and show how this technique can be used for traditional rendering of geometry. Chapter 3 gives an insight in rendering of participating media. Chapter 4 describes the proposed method of Visibility scattering. Chapter 5 describes implementation details, chapter 6 presents some results and chapter 7 contains a discussion on the results.

The reader is assumed to have a general, but not in depth knowledge of calculus, signal processing and computer graphics.

1.1 Storm

Storm is a volume modeller and renderer developed at Digital Domain. It has proved itself in production environment and has been used to create visual effects for several films. Storm is highly optimized for use in visual effects production and uses many tricks to increase performance, which is of critical importance for software in a production pipeline.

Storm can be divided into two parts: modelling and rendering. Both parts act as plugins for a software package called *Houdini* by Side Effects Software[1]. This

[1] http://www.sidefx.com

makes Storm easy to operate for users familiar with Houdini's user-interface and workflow, as well as integrating it into the existing pipeline at the facility.

1.1.1 Modelling

This project did not deal with the modelling aspects of Storm but it is helpful to be familiar with how this process works when trying to understand the rendering step.

Storm uses Houdini's node based user interface, see figure 1.1, to make it possible for users to create volumes from within existing scenes used in other parts of the production. The process generally starts with the creation of *cloud points*. These are points which will control the shape of the voulume. The user can attach attributes recognized by Storm to these points which are used to drive the generation of density data. The next step is to create a *buffer node* which defines the name, transformation, resolution and extents of the volume grid in space. The buffer node also determines the buffer type. There are several types of buffers, but the opacity type is the only one within the scope of this report. The *noise node* provides controls for the noise used to generate the volume.

The cloud point, buffer and noise node is then connected to a *fill node*. There are different fill nodes depending on what look the user wants to create. The user can now add *light nodes* to the scene. The lighttypes supported by Storm are point based lights, as well as cone and directional lights. The fill node and the light node is connected to a *merge node*. This node is now ready to be rendered.

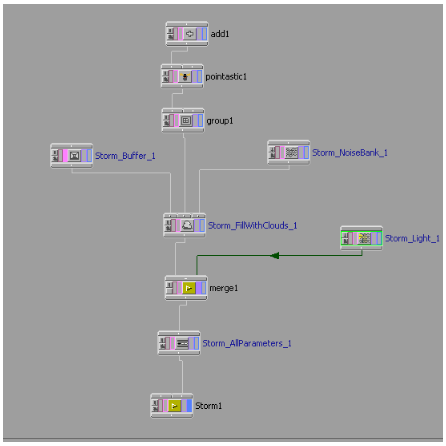


Figure 1.1. A basic Storm network displayed within Houdini.

1.1.2 Rendering

There are essentially two ways in which Storm can render a volume, although both modes uses the same underlying algorithm. The first is directly within the Houdini user interface. This mode is used primarily for previewing purposes. In order to render within Houdini the user needs to connect a *Storm parameters node* and a *Storm node* to the merge node described above. The other method of rendering is through Houdini's renderer Mantra. Mantra can be called from Houdini or used as a standalone program. The latter is important since it allows renderers to be launched in render farms without the extra process usage of the user interface. When launched in this way Storm acts as a shader-program, a program which evaluates the radiance for a ray, which is called from Mantra.

Storm uses a technique called *deep shadows*[9] to accelerate light calculations in volumes. This method calculates a map for every light source in the scene that effects the volume. The map holds information on the fraction of light that passes to certain depths.

Rendering is done by casting rays from the image plane which hits the volume. Ray marching is used to traverse the volume and the deep shadow maps are used to find the radiance at each location in the volume. Volume data is stored in slices to optimize memory access order. The data is also compressed using a run-length type encoding.

1.2 Problem description

One of the major weaknesses of Storms current rendering algorithm is that it only has support for discrete lightsources such as point lights or area lights. In many other parts of the pipeline high dynamic range environment maps[3] are used for lighting[2], which means that in order for lighting to be consistent on objects artist must manually approximate the environment using discrete light sources.

Another limitation is that Storm only takes single scattering into account. Light that travels through a volumetric effect would in real life bounce and scatter several times. When rendering a cloud for example, the lack of multiple scattering can manifest itself in dark or unlit areas on the 'backside' of the cloud, which would not appear for real clouds.

1.3 Convention

In this report the standard computer graphics coordinate system, where the z-axis denotes depth, will be used.

In mathematical and physics literature several different conventions has been used for spherical coordinate systems. Since spherical harmonics were first developed for use in physics it is has traditionally used the convention where ϕ is the angle in the x-y plane and θ is the angle from the z-axis. This convention will be used in this report and is displayed in figure 1.2. Note that this convention differs from the one used in mathematics. This gives the domains $0 \leq \theta \leq \pi$ and $0 \leq \phi < 2\pi$. All angles are measured in radians.

Throughout this report i as superscript denotes Spherical Harmonic coefficient, and i as subscript denotes incoming quantities.

1.4 Abbreviations

Table 1.1 shows abbreviations used in this report.

Abbreviation	Meaning
SH	Spherical Harmonics
LB	Lattice Boltzmann
BRDF	Bi-directional reflectance distribution function
PDE	Partial derivative equation
fftw	Fast Fourier Transform
HDR	High dynamic range

Table 1.1. Abbreviations used in this report.

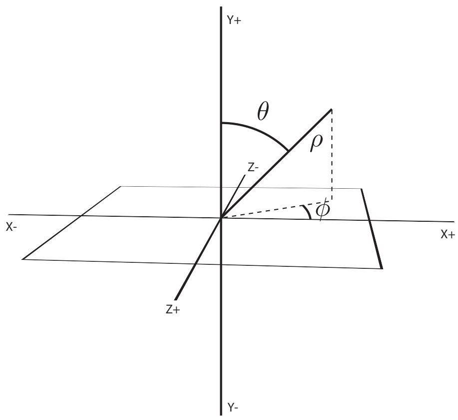


Figure 1.2. Coordinate system used in this report.

Chapter 2

Spherical Harmonics

Before dealing with volume rendering it is useful to first consider the rendering of solid surfaces using Spherical Harmonics (SH) lighting. This chapter serves as a introduction to SH and its applications in geometry rendering.

2.1 Definition

Formally SH is defined as a set of orthogonal functions that are the solution to Laplace's equation in spherical coordinates.

$$\nabla^2 = \frac{1}{r^2}\frac{\partial}{\partial r}(r^2\frac{\partial f}{\partial r}) + \frac{1}{r\sin\theta}\frac{\partial}{\partial\theta}(\sin\theta\frac{\partial f}{\partial\theta}) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2 f}{\partial^2\phi} = 0$$

Again the angles are defined on $0 \leq \theta \leq \pi$ and $0 \leq \phi < 2\pi$. Using separation of variables the functions can be expressed as

$$Y_m^l(\theta,\phi) = K_m^l e^{im\phi} P_m^l(\cos\theta) \tag{2.1}$$

where l and m are integer indexes with ranges from $l \geq 0$ and $l \geq m \geq -l$. l is sometimes referred to as the *band index*. For convenience l and m are sometimes expressed as a single integer index i

$$i = l(l+1) + m + 1$$

P_m^l is a family of *orthonormal basis functions* called *Associated Legendre Polynomials*. As opposed to regular Legendre Polynomials these return only real values. The orthonormal basis property means that if two basis functions are multiplied and integrated over the domain, the result will be 0 if they are different index, and 1 if they have the same index.

The mathematical definition of these polynomials turns out to be too computationally complex to use in computer implementions, instead a set of recurrence relationsships are used[15].

K_m^l is the normalization constant defined as

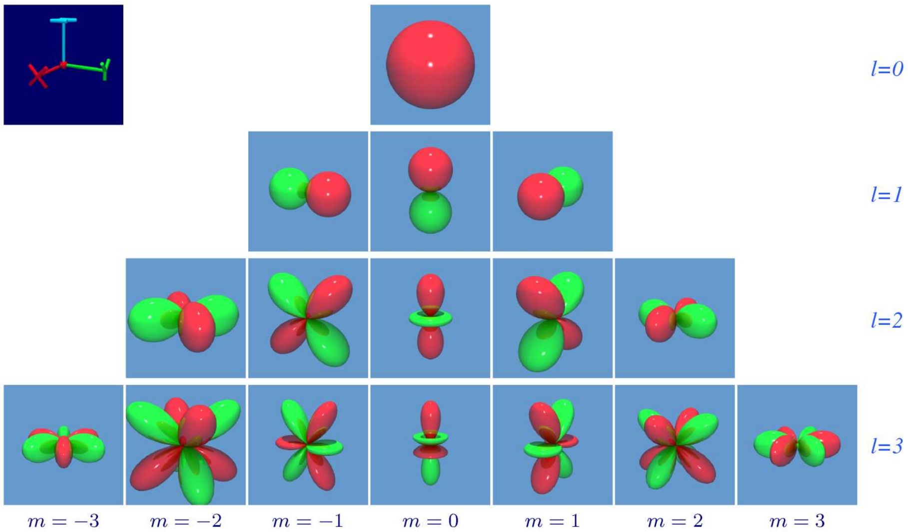


Figure 2.1. The first four bands of the Spherical Harmonic functions. The distance from the origin represents the value in each direction and the color represents the sign (green positive, red negative). Only the real parts of the functions are displayed. Images courtesy of Victor Luaña Cabal (victor@carbono.quimica.uniovi.es).

$$K_m^l = \sqrt{\frac{(2l+1)(l+|m|)!}{4\pi(l+|m|!}}$$

Figure 2.1 plots the functions for the first few bands.

2.1.1 Properties

From the orthonormal properties of SH functions follows that they can be used as linear basis functions, similar to the Fourier transform. The difference is that these basis functions are defined in a spherical space instead of a 1D space. Like with the Fourier transform, any signal can be projected onto these basis functions to produce a set of coefficients.

$$C_m^l = \int f(s) Y_m^l(s) ds \tag{2.2}$$

where s denotes a general spherical parametrization. The set of coefficients C_m^l is a description of the original function. An reconstruction can be found by

$$\tilde{f}(s) = \sum_{l=0}^{N-1} \sum_{m=-l}^{l} C_m^l Y_m^l(s)$$

where N is the number of bands used. The quality of the reconstruction depends on l, because the higher bands capture finer details of the original signal. It is often convenient to think about the band index as frequency, even though the functions are defined in the same space.

Another property of the SH transform which is important in computer graphics is that integrating two functions in normal space is equal to the dotproduct between in angular space.

$$\int f(s)g(s) = \sum_{i=0}^{N} f^i g^i$$

where f^i and g^i are the SH coefficients of the functions $f(s)$ and $g(s)$ and N is the number of coefficients.

A dot product is less computationally expensive in respect to an integration which is what has made SH popular in real-time computer graphics.

Unfortunately rotating SH coefficients is not trivial and has not been a focus for this thesis. This is however an active research area and a new fast method for rotating has recently been discovered[14].

2.2 Spherical harmonics for shading

The use of SH lighting has become an increasingly popular solution as the demand for realism in real-time graphic applications grows. It is used as a fast and efficient method of computing a global illumination solution. Traditionally this has been achieved by raytracing with Monte Carlo techniques[7] or radiosity[6], which requires costly integrations.

SH, first published over a century ago, has been used in quantum physics since the 1960's to describe angular momentum. Kajiya[8] was the first to describe its use with rendering when he used SH to solve the rendering equation, and more recently Sloan[12] used SH to achieve real-time performance. For global illumination rendering SH is attractive because of the fast evaluation of integrals.

2.2.1 The rendering equation

Raytracing involves evaluation of light scattering from a surface into the camera. The general equation for evaluating the radiance at a point is given by the *rendering equation*[7], in its full form defined as:

$$L(\vec{x}, \omega_o) = L_e(\vec{x}, \omega_o) + \int_S f_r(\vec{x}, \omega_i \to \omega_o) L(\vec{x}', \omega_i) G(\vec{x}, \vec{x}') V(\vec{x}, \vec{x}') d\omega_i \qquad (2.3)$$

where $\vec{x}$ a point on a surface, $\vec{x}'$ is a point on another surface visibile from $\vec{x}$, $L(\vec{x}, \omega_o)$ is the amount of light leaving the point in direction ω_o, L_e is the light emitted by the point, $f_r(\vec{x}, \omega_o, \omega_i)$ is the BRDF at the point, $L_i(\vec{x}, \omega_i)$ is the incoming light at the point from direction ω_i, $G(\vec{x}, \vec{x}')$ is the geometric relationship between $\vec{x}$ and $\vec{x}'$ and finally $V(\vec{x}, \vec{x}')$ which is a visibility test.

Using the projected solid angle equation 2.3 can be expressed as:

$$L(\vec{x}, \omega_o) = \int\limits_S f_r(\vec{x}, \omega_o, \omega_i) L_i(\vec{x}, \omega_i) H(\vec{x}, \omega_i) d\omega_i \tag{2.4}$$

where $H(\vec{x}, \omega_i)$ is the geometric or cosine term.

A simple scheme for lighting is *diffuse unshaded lighting*, where shadowing, self-shadowing and interreflections are disregarded. Geometry in the scene is assumed to be represented by polygonal meshes made up by vertices, faces and edges.

Each vertex is subjected to incoming light from a hemisphere Ω centered at the vertex with the pole pointing in the normal direction. In a diffuse lighting model the incoming light is reflected equally in all direction, which means that the BRDF function becomes a constant scalar. Equation 2.4 now simplifies to:

$$L_{DU}(\vec{x}) = \frac{\rho_x}{\pi} \int\limits_S L_i(\vec{x}\omega_i) max(\mathbf{N}_x \bullet \omega_i, 0) d\omega_i \tag{2.5}$$

where $\mathbf{N}_x$ is the normal at the point. ρ_x is the albedo, the ratio between the fraction of light that is absorbed and light that is scattered, defined as σ_a/σ_s. It is convenient to find a transfer function which expresses how a light environment will affect the point. The transfer function for diffuse unshadowed (DU) lighting, F_{DU}, can now be found by separating out the light source:

$$F_{DU} = max(\mathbf{N}_x \bullet \omega_i, 0) \tag{2.6}$$

To represent equation 2.6 in SH coefficients, using Monte Carlo technique, uniform random samples originated at the point are generated. The samples are run trough the transfer function, and inserted in the SH transform equation 2.2:

$$F^i_{DU} = \int\limits_\Omega max(\mathbf{N}_x \bullet \omega_i, 0) Y^i(\omega_i) \tag{2.7}$$

The result will be a set of coefficients, representing the diffuse unshaded transfer of light at the point.

To further extend the rendering equation self occlusion can be included. This process is also environment independent

2.2.2 Representing the environment

It is very convenient to separate the lighting environment from the rendering equation, however to evaluate the rendering equation with the transfer function in SH form, the environment must also be transformed into SH coefficients. The environment is assumed to be infinitely far away which means that all points receive the same amount of light from each direction respectively. The environment is sampled over all directions Ω with the function $L_i(\omega_i)$ and projected onto the SH functions:

$$E^i = \int_{\Omega} L_i(\omega_i) Y^i(\omega_i) \tag{2.8}$$

The set of environment coefficients, E^l_m, is shared by all points.

2.2.3 Rendering

To render the scene, rays are shot from the image plane and evaluated when intersecting a surface. Evaluation of the lighting at that point $\vec{x}$ is a simple matter of calculating the dot product of the coefficients for the transfer function F^i_{DU} and the coefficients for the environment E^i:

$$L(\vec{x}) = \sum_{i=0}^{N} F^i_{DU} E^i \tag{2.9}$$

Any number of coefficients can be used, but only a few is needed to capture the low frequency characteristics of those signals.

Chapter 3

Rendering of participating media

Rendering of a computer generated high quality image often involves raytracing. This process involves casting rays from an eye position which bounces off and interacts with density in the scene.

In the previous chapter rays are assumed to be travelling through a vacuum. For many scenes this assumption is adequate to produce a realistic looking image, however in many natural scenes where atmospheric effects are significant this assumption is not valid. Real light that travels through the air hits particles and is deflected from their original paths. Earths atmosphere for example if full of dust and other particles, which is why objects in a distance appears faded or blueish. We define *participating media* as a property of space which simulates those effects as light travels through it.

Participating media can be divided into two categories: *homogeneous* and *inhomogeneous*. In homogeneous media the optic properties does not change spatially or can be described by a function. An example of what can be considered homogeneous media is fog (over limited areas), whereas a cumulus cloud is highly inhomogeneous. The focus of this thesis is with inhomogeneous volumes even though homogenous examples will be used to ease testing of the method.

3.0.4 Previous work

Rendering of participating media is by no means a new problem in computer graphics. Kajiya was one of the first to describe the problem [8], and proposed an analytical solution. Interestingly Kajiya used SH to solve his PDE's, however it was not until the late 1990s that SH became popular in rendering.

More recently Stam [13] described multiple scattering as a diffusion process and solved it using the multigrid technique. Another popular technique is deep shadows[9] which uses maps to store transmittance values.

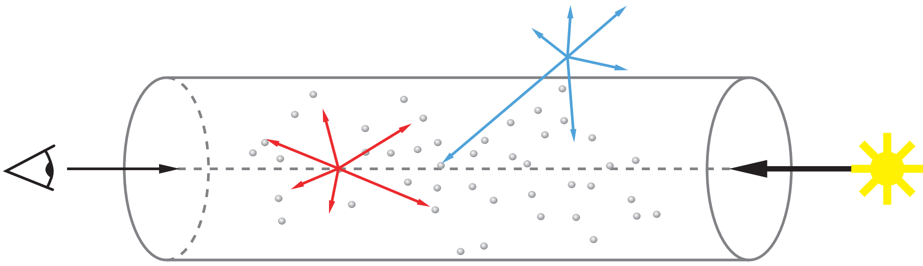


Figure 3.1. Illustration of the scattering effects light encounters while travelling from the right towards the eye through a cross-section of participating media. Out-scattering (red) reduces the rays intensity which makes it fainter to the observer. In-scattering (blue) is scatter event outside the crosssection and increases the intensity.

3.0.5 Grids

When dealing with inhomogeneous media in computer graphics a method is needed to represent it's spatially varying properties. In some cases this can be described with expressions, but most natural phenomena not uniform and can not be described with a simple function. Consider for example smoke rising from a fire, interacting with the air around it.

The discreet representation most often used is the *regular grid*, although many other representations exists, this is well suited for traversal, storage, etc. A regular grid is defined as regularly spaced lattice where density values are stored at each lattice point. The lattice points are sometimes referred to as *voxels*. Throughout this report regular grids with finite dimensions in space is assumed.

3.1 Light transport in participating media

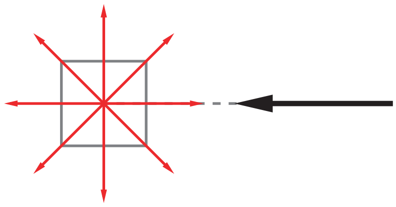
As light travels trough a participating medium it sometimes hits particles which causes the light to scatter or be absorbed. Certain media can also be assumed to emit light. All these effects are modelled by the *radiative transfer equation*[1], or some literature referred to as the *scatter equation*.

$$(\omega\nabla + \sigma_t(\vec{x}))L(\vec{x},\omega) = \sigma_s(\vec{x}) \int_S p(\omega,\omega')L(\vec{x},\omega')d\omega' + Q(\vec{x},\omega) \tag{3.1}$$

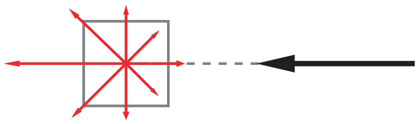
which describes the change of radiance in the medium at a point $\vec{x}$ where $L(\vec{x},\omega)$ is the radiance at the point, $p(\omega,\omega')$ is the phase function which is described further in the next section, and $Q(\vec{x},\omega)$ is the emission term. The optical properties of the medium is modelled with *scattering cross section* σ_s, *absorption cross section* σ_a, which together define *extinction cross section*

$$\sigma_t = \sigma_a + \sigma_s$$

These properties are functions of space and are usually modelled as a base value that is multiplied by the opacity at the point.



(a) Isotropic scattering.



(b) Isotropic scattering.

Figure 3.2. A ray of light enters a volume and scatters. In 3.2(a) isotropic scattering spreads the light equally in all directions. In 3.2(b) anisotropic scattering, in this example with strong forward scattering.

Equation 3.1 cannot be solved analytically except for in a few special cases[8].

3.1.1 Scattering

The scattering process is described by the *phase function*. In most natural media this is a function of angle between incoming and outgoing vector $\omega \angle \omega'$.

In its most basic form the phase function describes a uniform scatter event, where light is distributed equally in all directions. See figure 3.2(a). This is referred to as *isotropic scattering*. The phase function is in that case a scalar

$$p(\omega \rightarrow \omega') = \frac{1}{4\pi}$$

To model most natural media the isotropic scattering is not sufficient. This is especially true for media with high albedo as described in previous chapter, for example clouds or smoke. In these cases strong forward scattering is present. This means that more light is reflected forward, relative to ray's direction, than other directions. These scattering events has to be modelled with an angle dependent phase function. This is referred to as *anisotropic scattering*. Several analytical models for anisotropic scattering exists, most noteably Rayleigh and Mie scattering[11]. See figure 3.2(b) for an example.

Chapter 4

Visibility scattering

The method proposed in this chapter, named *visibility scattering*, has three important properties. It is *light independent*, in the sen ce that for a given volume a solution can be found that is valid for any light environment. The solution takes not only single scattering but also *multiple scattering* into account. Finally it enables *fast re-lighting* of volumes.

Visibility scattering makes use of the visibility transfer function, a powerful abstraction often used in signal processing. The transfer function is a non-physical unitless function. The reader is reminded not to confuse the transfer function with actual light density or energy.

4.1 Reduced incident intensity transfer function

The *reduced incident intensity transfer function* T_{ri}[13] expresses the fraction of light from the environment that penetrates the volume and reaches a point without being absorbed or out-scattered. In terms of the effects described in chapter 3 T_{ri} does not account for in-scattering or emission, only absorbtion and out-scattering. This is commonly referred to as 'single scattering'. The transfer function is parametrized in spherical coordinates.

To find the transfer function, which is calculated at every point in the volume, a set of random directions is generated. A ray is shot in each direction, ω. Each ray has an initial *transmittance* of 1. The transmittance is reduced by

$$T_{ri}(\vec{x_0}, \omega) = e^{-\int_0^d \sigma_t \rho(\vec{x_u}) du} \tag{4.1}$$

The ray marching is aborted if the transmittance reaches 0.

Equation 4.1 can be parametrized in spherical coordinates, which makes it possible to encode it with the Spherical Harmonic transform. A set of coefficients now represents the transfer function

$$T_{ri}^i = [T_{ri}^0, T_{ri}^1, T_{ri}^2 \dots T_{ri}^n] \tag{4.2}$$

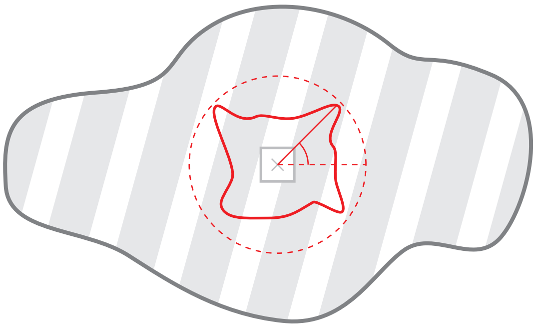


Figure 4.1. T_{ri}, shown in red, for a voxel in a volume. The function value is represented by the radius. The dotted circle represent the unit circle, i.e. full visibility.

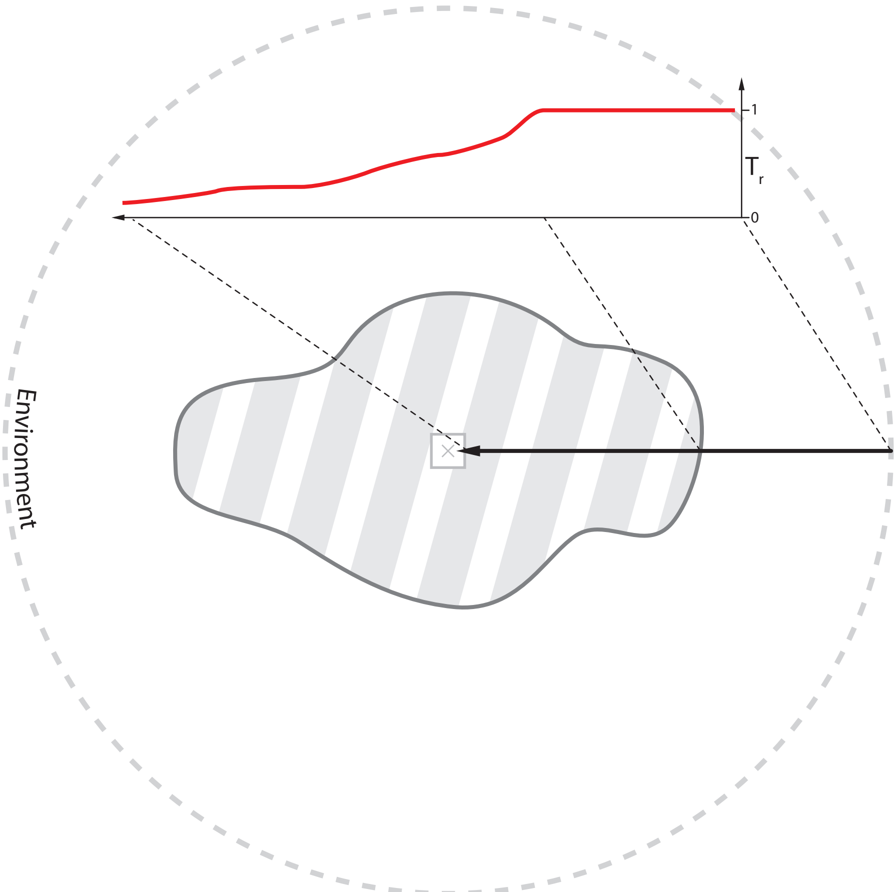


Figure 4.2. As a ray travels inside a volume its transmittance is reduced.

4.2 Multiple scattering

The reduced incident intensity transfer function only accounts for single scattering. In order to simulate multiple scattering a new scheme is proposed which will transport and scatter the transfer functions, retained in their SH form.

4.2.1 Setup and initialization

For every voxel in the volume a discrete set of directions, $\vec{d_i} \in \Re^3$, is defined (figure 4.3(a)), connecting the voxel to its neighbors. The number of directions can vary, but an even distribution between neighbors are desired to avoid directional weighting. In practice between 6 and 18 directions are used for 3D volume. Each direction contains a complete transfer function represented by SH coefficients. Furthermore a zero vector is used to collect all the light travelling through the voxel at each iteration.

Each direction is initialized (figure 4.3(b)) with the coefficients from equation 4.2 divided by the number of directions, which corresponds to an isotropic scatter event where the energy spreads equally in all directions.

4.2.2 Transport system

Like Lattice-Boltzmann Lighting (LB)[5], the proposed method simulates light transport by iterating distribution and collision events in all voxels. All operations are local to the voxel and it's nearest neighbors. However Visibility scattering differs from LB by the fact that it does not have boundary conditions nor does it continously inject light into the solution. This is because it deals with visibility and not light densities.

Distribution phase

The first phase in every iteration is the distribution phase. For every voxel, every direction is traversed and the corresponding transfer function is distributed to the neighboring voxel in that direction (figure 4.3(c)). The transfer function at a point $\vec{x}$ in direction $\vec{d_i}$ at iteration k is

$$T_k(\vec{x} + \vec{d_i}, \vec{d_i}) = T_{k-1}(\vec{x}, \vec{d_i}) \tag{4.3}$$

A subtle but integral part of the distribution system is that the direction of transport is maintained when distributing the transfer function.

In cases where a voxel does not have a neighbor in a direction, i.e. boundary voxels, the transfer function is considered to have left the system and is no longer a part of the simulation.

Collision phase

The second phase in the iteration process is the collision phase. In this phase the actual scatter events are simulated. This is done by again iterating through all directions in each destination voxel. At each direction a fraction of the transfer function is subtracted from the current direction and added to a final scatter transfer function calculated by

$$T_M(\vec{x}) = \sum T_k(\vec{x}, \vec{d_i})\sigma_s \rho(\vec{x}) \tag{4.4}$$

The scattered transfer function is then distributed evenly to all directions (figure 4.3(f)) by

$$T_k(\vec{x}, \vec{d_i}) = T_k(\vec{x}, \vec{d_i}) + \frac{T_M(\vec{x})}{N} \tag{4.5}$$

where N is the number of directions.

The effect of a scatter event will be that a part of the transfer function, according to the scatter coefficient and density of the voxel, will scatter and be redistributed, while the remaining part will continue travelling in the original direction.

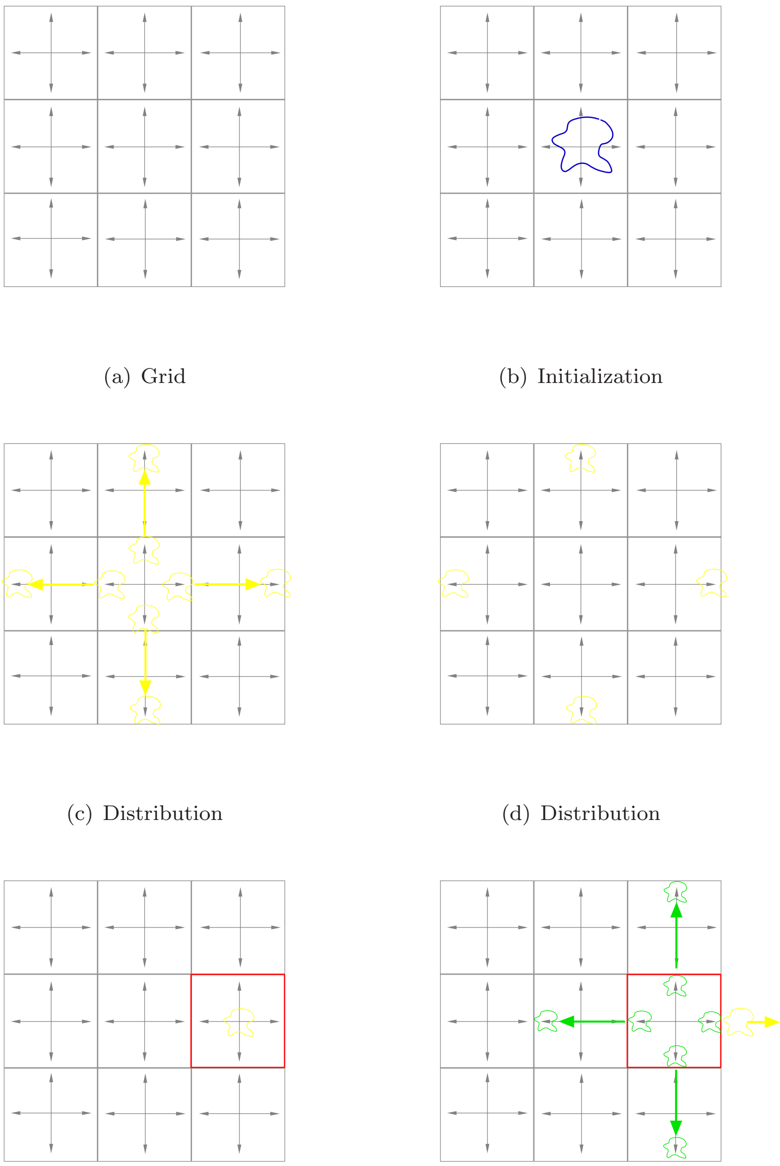
After a number of iterations the transfer functions that travels around and their contribution to the final transfer function will be smaller than a threshold value, solution converges.

Notice that it is necessary for the density values to be normalized. If they are not it is possible that light is added into the solution at every iteration.

4.3 Rendering

Rendering the volume has many similarities to rendering geometry shaded with SH. Identical to the algorithm described in chapter 3, a SH representation of the environment is used.

The process begins with casting rays from the image plane and tracing the rays through the volume with raymarching, front to back. At each point the radiance is evaluated by dot producting the transfer function coefficients T_M with the environment coefficients. The accumulated opacity of the ray is found by adding the opacities together. The raymarching is stopped when the opacity reaches one.



(a) Grid (b) Initialization

(c) Distribution (d) Distribution

(e) Collision (f) Collision

Figure 4.3. A complete iteration of the transport system.

Chapter 5

Implementation

5.1 Spherical Harmonics

Since SH was only recently gained popularity in computer graphics no widely adopted framework exists. There are some general implementations, mostly geared towards signal processing. Most noteably *SpharmonicKit*[1] and *spherepack*[2] which are C respectively Fortran based implementations. SpharmonicKit uses Fast Fourier Transform (fftw). None of these packages was considered suiteable for the purposes covered in this thesis, mainly due to their generality.

Instead a internal implementation of the Spherical Harmonic function was used. Although single precision floats has proved to be sufficient for rendering purposes, if very high bands are needed double precision should be used to avoid precision errors.

5.2 Test applications

Since Digital Domain had very little previous experience with SH a set of testapplication were set up.

5.2.1 Initial testing

The first application was developed to test the SH transform on images. The goal was to find out how many bands had to be used to get a reasonable representation, and explore different sampling methods.

The application takes an image, preferable in a high dynamic range floating point format such as Radiance HDR[3] or OpenEXR[4] and transforms it into a SH representation. The image is assumed to be in latlong format, that is it represents

[1] http://www.cs.dartmouth.edu/~geelong/sphere/
[2] http://www.cisl.ucar.edu/css/software/spherepack/
[3] http://radsite.lbl.gov/radiance/
[4] http://www.openexr.com

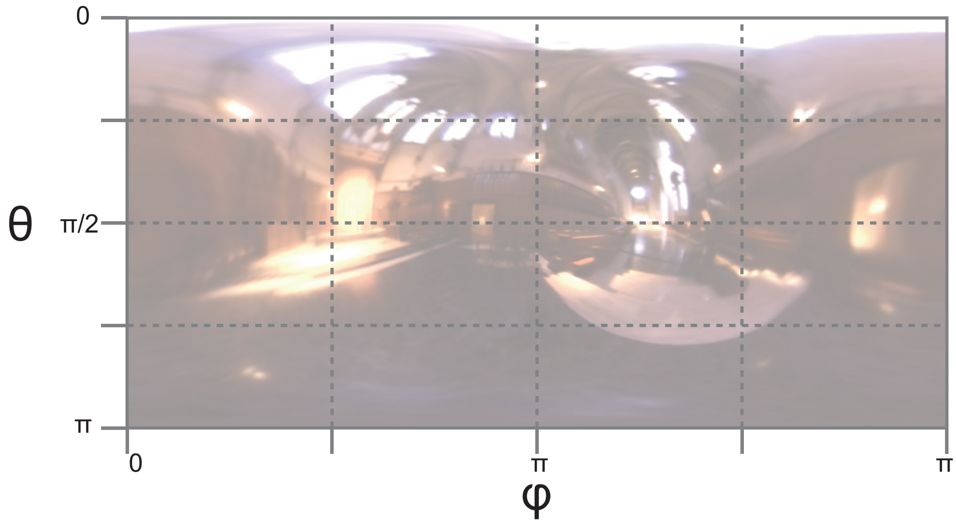


Figure 5.1. Environment image in latlong format

the view in all direction from a point. Figure 5.1 illustrates the mapping of the viewing angles.

The image I can be considered a discrete function parametrized by ϕ and θ which can be transformed using equation 2.2. A Monte Carlo technique is used to sample. Special consideration has to be taken when sampling to make sure that the distribution of the samples are uniform with respect to the spherical environment. With Monte Carlo sampling equation 2.2 becomes

$$I^i = \frac{4\pi}{N} \sum_{i=0}^{N} I(\phi, \theta) Y^i(\phi, \theta) \tag{5.1}$$

The set of coefficients is then returned to its function form again.

$$\tilde{I}(\phi, \theta) = \sum_{i=0}^{N} I^i Y^i(\phi, \theta) \tag{5.2}$$

The reconstructed image shows if some information is lost in the transformation. Figure 6.4 show this. Since it's not possible to use an infinite number of coefficients some high frequency components of the original signal is lost. The reconstruction can be considered a low frequency representation of the original signal. Figure 5.2 illustrates the phases in the program.

Important conclusions on the quality of the representation can also be drawn from this application. Figure 6.4 shows how the detail of the reconstruction increases as more bands are used in the transform. It also shows the importance of windowing.

Figure 5.2. Flowchart for the initial test program.

5.2.2 Renderer

When the properties of the SH transform had been explored an application was developed to test volume rendering, the focus of this project. Instead of directly trying to implement the new method into existing software a dedicated stand alone application was developed. This approach was chosen because it would allow for specialized datastructures and order of execution, and to minimize problems due to integration.

The raytracing engine developed used a number of existing libraries, such as Open Inventor[5], Boost[6], and a number of Digital Domains proprietary libraries. It was designed to take volume data produced by Storm, stored in the proprietary *vox* file format. It also takes camera and light animations and use those parameters to setup a scene.

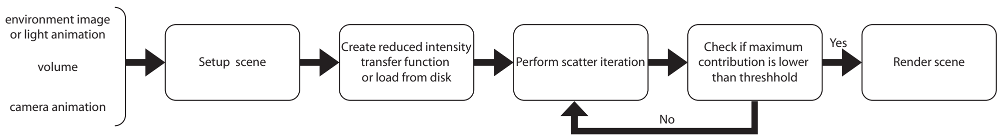


Figure 5.3. Flowchart for the renderer.

The renderer can be considered to go through three main phases. The first phase is acquiring the reduced incident intensity transfer function as described in previous chapters. Phase two simulates multiple scattering and finally in phase three the final image is rendered. Figure 5.3 illustrates these phases.

The renderer also has the ability to store and load transfer functions of a volume using a simple sequential file format. This function enables animations to be rendered very fast, because the transfer function only has to be calculated in the first frame, and is reused in all following frames. This requires that the volume is static between frames.

[5]http://oss.sgi.com/projects/inventor/
[6]http://www.boost.org/

Chapter 6

Results

Visibility scattering is a new method for volume rendering and a lot of research is needed before the true potential is revealed. The initial results however are encouraging.

Throughout the rest of this report three different volume data set are used. The first one is named *cloud*. This volume, generated in Storm, has an resolution of 204x173x321 density voxels. This data set was generated for the motion picture Stealth and is typical, both in resolution and complexity, for volumes used in production.

A second data set, called *block*, is simply uniform density filling the entire volume. This data set is useful for exploring the behavior of light in a more uniform volume because it is more intuitive how it should react. The resolution of the block data set is 300x300x300 density voxels.

The third data set called *torus*, also with resolution 300x300x300, is two solid rings intertwined. This shows the automatic self shadowing the visibility scattering method produces and also the shadow details possible to achieve with SH.

In order to allow for the solution to fit in memory, resolution of the transfer function in all three cases was 1/6:th of the density resolution. 10 SH bands where used when representing the transfer functions as well as the environment.

6.1 Multiple scattering

One of the main reasons for developing this method was to make it possible to render an image reasonably fast multiple scattering.

Figure 6.1 displays the difference between a rendering which only takes single scattering into account, using the reduced incident intensity transfer function directly, and one where multiple scattering was approximated.

Figure 6.1. Images rendered using single scattering (left) and multiple scattering (right). Both volumes are lit with the same environment map.

6.2 Performance

Rendering performance is a critical aspect for any system used in a visual effects pipeline. Efficient algorithms save valueable rendering time and faster feedback to artists which translates into economic efficiency. One of the most interesting challenges of this thesis work has been developing and evaluating the method with this goal in mind.

The visibility scattering method allows the user to greatly decrease rendering times by storing the transfer function, which can be reused in multiple frames and different scenes. Pre-process time, that is the time spent calculating the transfer function, is the dominant part of the algorithm whereas rendering is a relatively quick process. Table 6.1 shows rendering times. Note that even if time spent in pre-process differs between the data sets, the rendering times are similar.

The type of lighting environment or its complexity does not effect rendering times.

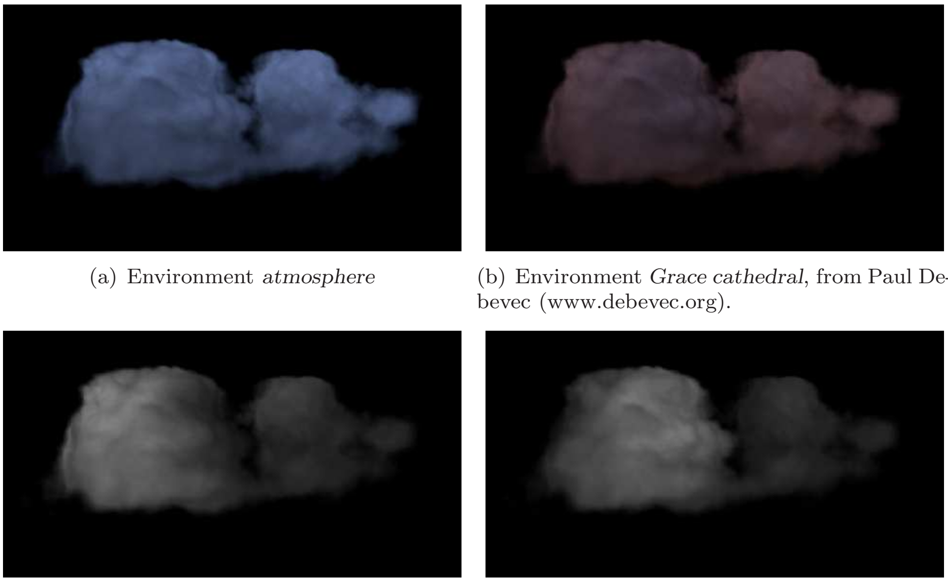
Data set	Resolution	Pre-process	Rendering
Cloud	204x173x321	60 min	5 min
Torus	300x300x300	45 min	4 min

Table 6.1. Mean rendering times for two of the data sets. Pre-processing is time spent finding the transfer functions. This has to be done only once.

Is should also be noted that many parts of the testing software which was not considered a focus for this project used non-optimized methods. It is likely that a lot of performance gains can be achieved by using known techniques for example accelerated ray-marching or optimizing data structures for fast access.

6.3 Fast re-lighting

The total separation of environment and transfer using transfer functions gives us a light independent solution. This enables fast re-lighting of the volume by reusing previously calculated transfer functions. Figure 6.3 shows the cloud data set lit by two different environments.



(a) Environment *atmosphere*

(b) Environment *Grace cathedral*, from Paul Debevec (www.debevec.org).

(c) Spotlight from side

(d) Spotlight from front

Figure 6.2. Cloud data set lit with different environments using the same transfer function.

6.4 Spherical Harmonics

Even though the use of Spherical Harmonics transform on the transfer function and the environment has limitations, for many cases it is an efficient and fast method.

An important observation was the effect of the number of sample rays used to create the initial reduced incident intensity transfer function. Figure 6.3 shows the effect of under sampling. Based on these results we found that at least 500 samples per voxel has to be used to get an accurate representation.

6.4.1 Bands

The number of bands that has to used to make an accurate representation obviously depends on the characteristics of the function and the requirements for the final image. In general we have found that around 10 bands, 110 coefficients, is sufficient to provide detail suitable for the high quality renderings used in visual effects production. Figure 6.4 shows an environment map encoded and reconstructed using different number of bands. The quality of the reconstruction might seem poor, but since the goal is to represent the lighting environment this is quite adequate.

6.4.2 Windowing

When using high bands an artifact called *ringing* can appear. This is an effect of the coefficients of the higher bands get too high values. One way to reduce this effect is by using windowing. This can be achieved by multiplying the coefficients with a windowing function, defined by

$$w_l = 0.54 + 0.46 * cos(\frac{\pi l}{Lmax}) \tag{6.1}$$

where w_l is the weighting factor for a band, l is the band number and $Lmax$ is the maximum band number used. Figure 6.5 shows a plot of function 6.1.

Although a host of windowing functions has been developed in the field of signal processing function 6.1 has proved to be sufficient in suppressing ringing for high frequency representations. Figure 6.4 shows the effect of applying a windowing function to the coefficients.

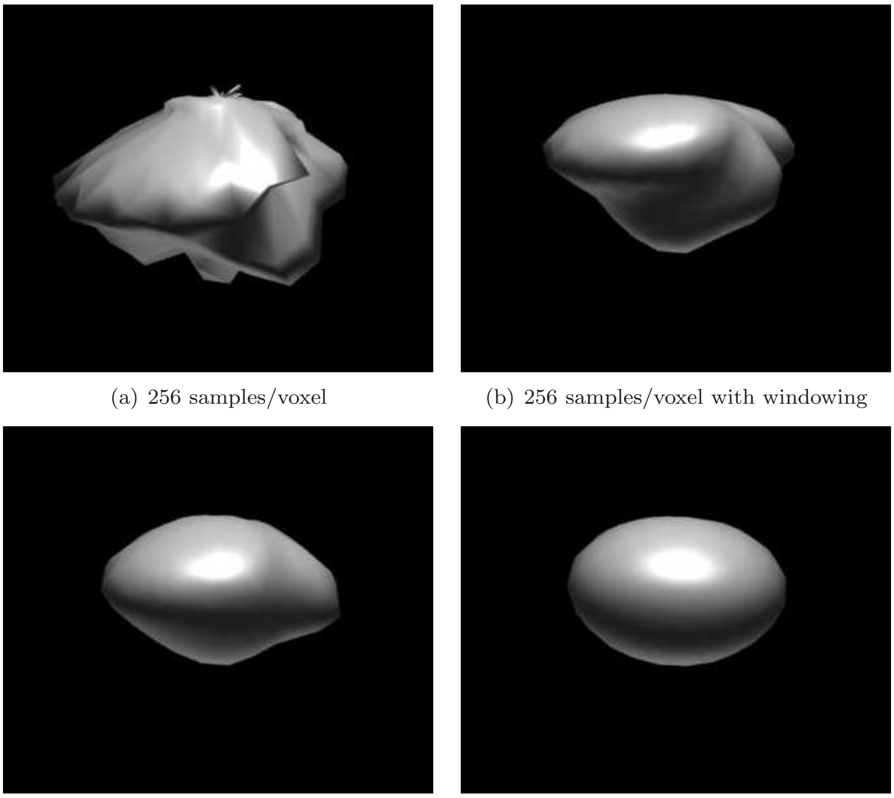
6.5 Issues

6.5.1 Poles

On rare occasions we have observed visual artifacts which seems to manifest themself at poles and along the equator. At the time of writing this report no explanation had been found, but most likely this is an effect of the SH transform. We suspect this because the artifacts appear near right angles to the coordinate system like in figure 6.6.

6.5.2 Intensity levels in reconstruction

We have found that it is hard to reproduce sharp peaks in the original signal with the SH transform due to the frequency limitation of equation 2.2. This effect appears for example when trying to represent an environment with a single bright area (such as the sun) in an otherwise dark surrounding. The reconstructed environment will display a bright spot in the same position as in the original but significantly less bright.

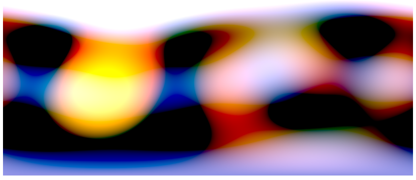


(a) 256 samples/voxel (b) 256 samples/voxel with windowing

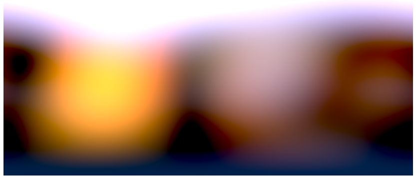
(c) 512 samples/voxel with windowing (d) 10000 samples/voxel with windowing

Figure 6.3. T_{ri} for a voxel using different number of sample rays per voxel. Notice that only 256 samples gives a somewhat noisy representation compared to 10000 samples which gives the most accurate representation.

(a) Original image



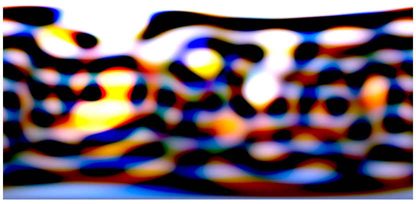
(b) Reconstructed using 5 bands



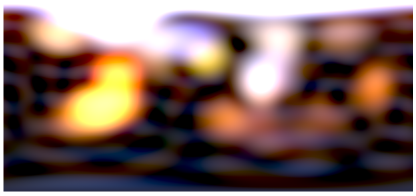
(c) Reconstructed using 5 bands with windowing

(d) Reconstructed using 10 bands

(e) Reconstructed using 10 bands with windowing



(f) Reconstructed using 15 bands



(g) Reconstructed using 15 bands with windowing

Figure 6.4. Environment latlong image 6.4(a) run through the test program, and the reconstructed images 6.4(b) to 6.4(g).

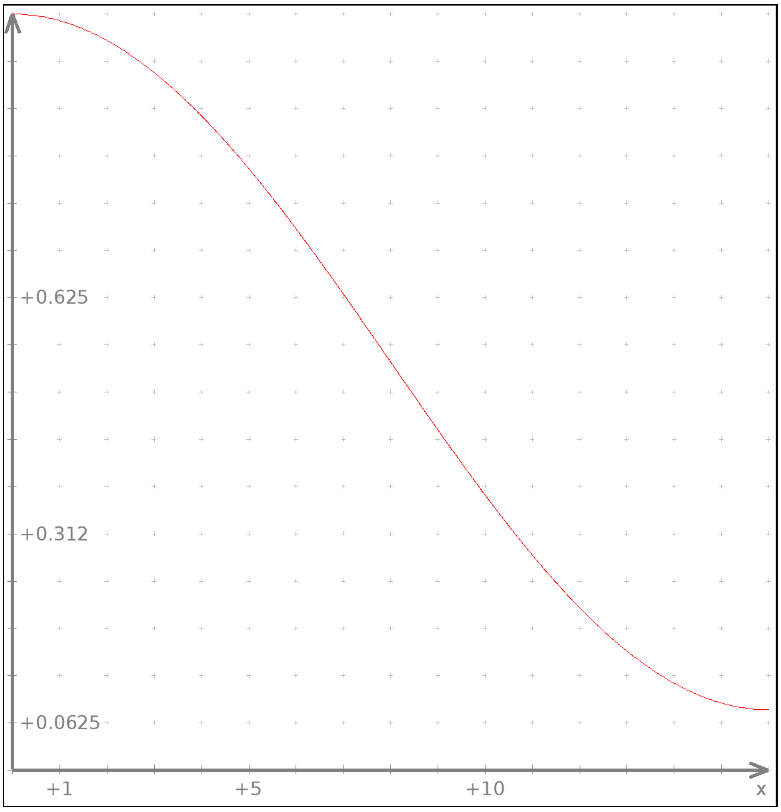


Figure 6.5. Plot of the windowing function with *Lmax* set to 15.

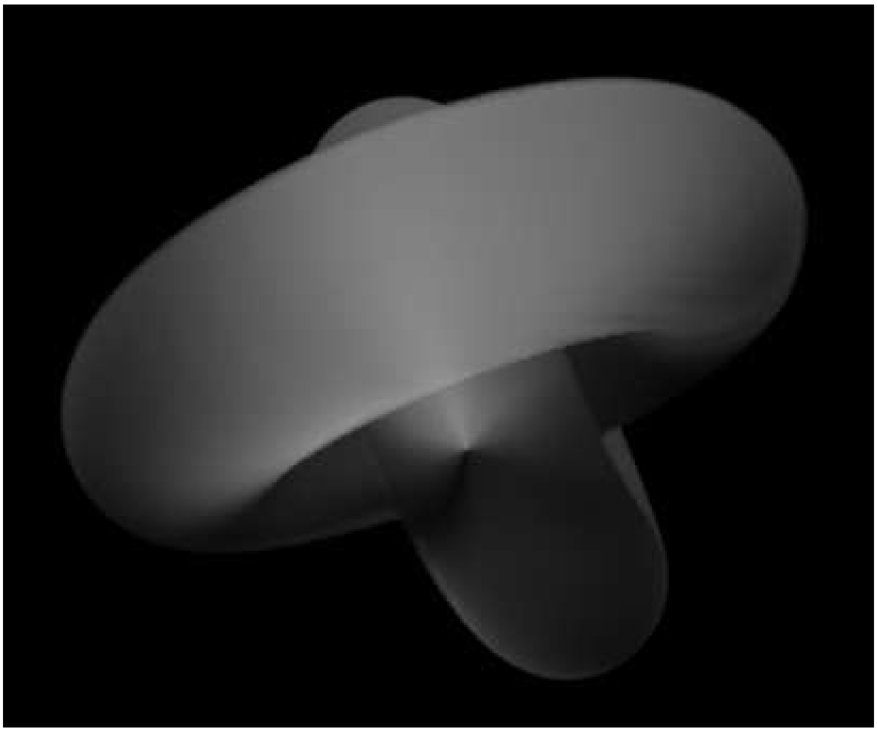


Figure 6.6. Example of artifact. In this case the location of the light source is directly behind the camera which corresponds to $\phi = 0, \theta = \pi/2$ in spherical coordinates.

Chapter 7

Discussion

Many methods where explored during the development of the Visibility scattering method. Visibility scattering borrows concepts from the Lattice-Boltzmann Lighting method[5], but differs in the fact that it does not assume anything about the lighting environment and that it distributes a complete set of coefficients rather than light densities.

Discrete Ordinate Method is a method developed for Atmospheric science used for simulating radiative transfer in clouds[4] which also has similarities to visibility scattering. This method uses a discrete number of transport directions but does not appear to use a transfer function.

7.1 Quality

Image quality is of course a property which is difficult to quantize and evaluate. In this case the end-users approval is the most important quality measurement tool. Artists who has seen the final rendered images describe a *global-illumination quality*, that is that the volume appears to be lit from all directions which gives a generally softer look. This has previously been hard to reproduce using only point lights.

This leads to the idea of combining SH lighting with previously used methods such as deep shadow. This would give the artist the best of two worlds, diffuse low frequency environment lighting, as well as local high frequency point lights.

It should also be mentioned that further testing is needed to find the optimal settings in terms of number of bands used, sampling rate, windowing functions and scatter coefficients.

7.2 Future work

There are several refinements and optimizations that could be applied to the proposed method.

7.2.1 Approximators

Storing all SH coefficients at every point is costly. Also volumes usually have areas where it is less important to sample the transfer function. These notions leads to the use of approximators. This allows us to sample only at the interesting locations and approximate everywhere else. However, the visibility scatter scheme presented in this report would become increasingly complex since couplings between sampling points no longer are uniform in distance or number of neighbors.

During the course of this thesis work, development of this technique was accomplished by Nafees Bin Zafar and a sketch[10] describing the results is scheduled to appear in the Siggraph conference in July-August 2006.

7.2.2 Phase function

Another interesting extension of the proposed method would be to include anisotropic scattering instead of isotropic. This would require that a phase function would be used to scatter the light in collision phase.

7.3 Conclusion

We have shown that we can use the transfer function, both as a efficient and fast representation as well as in the light transport simulation. With the visibility scattering method we get an light independent solution which include multiple scattering, that can be re-used and re-rendered quickly.

Bibliography

[1] James Arvo. Transfer equations in global illumination. In *SIGGRAPH '93 Course Notes*, volume 42, August 1993.

[2] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 189–198, July 1998.

[3] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 369–378, August 1997.

[4] K. F. Evans. The spherical harmonic discrete ordinate method for three-dimensional atmospheric radiative transfer. *Journal of Atmosferic Science*, 55:429–446, 1998.

[5] Robert Geist, Karl Rasche, James Westall, and Robert Schalkoff. Lattice-boltzmann lighting. In *Rendering Techniques 2004: 15th Eurographics Workshop on Rendering*, pages 355–362, June 2004.

[6] Donald P. Greenberg, Michael Cohen, and Kenneth E. Torrance. Radiosity: A method for computing global illumination. *The Visual Computer*, 2(5):291–297, September 1986.

[7] James T. Kajiya. The rendering equation. In *Computer Graphics (Proceedings of SIGGRAPH 86)*, volume 20, pages 143–150, August 1986.

[8] James T. Kajiya and Brian P. Von Herzen. Ray tracing volume densities. In *Computer Graphics (Proceedings of SIGGRAPH 84)*, volume 18, pages 165–174, July 1984.

[9] Tom Lokovic and Eric Veach. Deep shadow maps. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, pages 385–392, July 2000.

[10] Doug Roble Ken Museth Nafees bin Zafar, Johan Åkesson. In *Proceedings of SIGGRAPH 2006*, Computer Graphics Proceedings, Annual Conference Series, July 2006.

[11] Tomoyuki Nishita, Eihachiro Nakamae, and Yoshinori Dobashi. Display of clouds and snow taking into account multiple anisotropic scattering and sky light. In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 379–386, August 1996.

[12] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics*, 21(3):527–536, July 2002.

[13] Jos Stam. Multiple scattering as a diffusion process. In *Eurographics Rendering Workshop 1995*, pages 41–50, June 1995.

[14] Reiji Suda and Masayasu Takami. A fast spherical harmonics transform algorithm. *Mathematics of Computation*, 71(238):703–715, 2002.

[15] William Vettenling Brian Flannery William Press, Saul Teukolsky. *Numerical Recipes in C*. Press Syndicate of the University of Cambridge, 1992.