Estimating and Applying Dynamic Light Changes to Environment Maps in Real-time for use in Image Based Lighting

Laboratory for Computer Vision and Media Technology

Aalborg University · AAU 2005

Master Thesis

Group 1025 - Mikkel Sandberg Andersen and Tommy Jensen

FACULTY OF ENGINEERING AND SCIENCE
AALBORG UNIVERSITY

INSTITUTE OF ELECTRONIC SYSTEMS

TITLE:

Estimating and Applying Dynamic Light Changes to Environment Maps in Real-time for use in Image Based Lighting

TIME PERIOD:

10th semester,
February 2nd to June 2nd, 2005

PROJECT GROUP:
1025 - 2005 CVG10

GROUP MEMBERS:

Mikkel Sandberg Andersen

Tommy Jensen

SUPERVISOR:

Claus Brøndgaard Madsen

NO. OF COPIES: 8

NO. OF PAGES IN MAIN REPORT: 138

NO. OF PAGES IN APPENDICES: 24

NO. OF PAGES IN TOTAL: 162

ABSTRACT:

The purpose of this project is to implement a real-time Augmented Reality (AR) system and to explore how dynamical light changes in an outdoor environment can be handled within the system.
Relevant factors, including state of the art within the fields of inverse rendering, image re-lighting, light source estimation, and image based illumination are explored. A method using existing inverse rendering techniques is used to acquire diffuse surface reflectances in an offline procedure. The reflectances are used in an online procedure to estimate the illumination parameters of an outdoor scene. Using the surface reflectances and the estimated light parameters a re-lighted environment map is created on-the-fly to shade objects using the Irradiance Volume.
An AR framework has been developed. The framework is able to update an environment map, based on automatically estimated light parameters from partial low dynamic range image data. The re-lighted environment map is used to shade and the estimated light parameters to shadow arbitrary virtual objects.
The work presented provides an elegant method for estimating dynamically changing illumination parameters, without the need for calibration objects in the scene. It produces convincing results in handling dynamic illumination changes in an outdoor environment.

Synopsis

Formålet med dette projekt er at implementere et realtids Augmented Reality (AR) system og undersøge hvordan dynamiske lysændringer i et udendørs miljø kan håndteres i et sådan system.

Relevante faktorer, inklusive state of the art indenfor felter som invers rendering, billed genlysning, lys kilde estimering og billed baseret belysning, udforskes. Eksisterende invers renderings teknikker anvendes for at opnå diffuse overflade reflektanser for omgivelserne i en offline procedure. Disse informationer anvendes i en online procedure, der kan estimere belysningsparametrene for en udendørs-scene. Ved at anvende de estimerede lys parametre kan et environment map gen-lyses på frame-basis, og anvendes til at belyse objekter ved brug af et Irradiance Volume.

Et AR framework er blevet udviklet. Frameworket kan opdatere et environment map, baseret på automatisk estimerede lys parametre fra partiel Low Dynamic Range billed data. Det genlyste environment map anvendes sammen med de estimerede lys parametre til at belyse og skyggelægge et arbitrært virtuelt object.

Det præsenterede arbejde leverer en elegant metode til at estimere dynamisk ændrende belysnings parametre, uden at der er brug for kalibreringsobjekter i scenen. Der produceres overbevisende resultater i håndteringen af dynamiske lysændringer i et udendørs miljø.

Preface

This report is the master thesis of group 1025, 10th semester specialisation in Computer Vision and Graphics (CVG) 2005, Laboratory of Computer Vision and Media Technology, Aalborg University.

The target audiences for this report are people with an understanding of Computer Vision and Graphics, or people with knowledge or an interest in the field of light-estimation and real-time, image based, illumination and shadowing in Augmented Reality.

The report is comprised of four parts: Introduction, Methods, Results and Discussion, and Appendix. The appendix is placed last in the report.

Source references are on the Harvard-form, an example of this is: [Sherman and Craig, 2003]. Chapters and sections starting with a source reference is based on this source.

The group would like to thank Morten Friesgaaard Christensen for providing data for the masonry reflectance test.

Attached is a CD-ROM on which the following can be found:

- Report
- Source binaries
- Video material

Aalborg University, Denmark. June 2nd 2005.

Mikkel Sandberg Andersen

Tommy Jensen

Contents

I Introduction **11**

1 Introduction **13**

1.1 A short history of computer graphics 13

1.2 Virtual or Augmented reality? 13

1.3 Applications of Augmented Reality 15

2 Realistic illumination in Augmented Reality **17**

2.1 Illumination 17

2.2 Dynamic changes 19

2.3 Previous work 20

2.4 Summary 23

3 Defining the project **25**

3.1 The Aim of the Project 25

3.2 Technology 26

3.3 System Description 28

4 Results Preview **31**

II Methods **35**

5 Offline Data Acquisition **37**

5.1 Acquiring environment maps 37

5.2 Inverse rendering 37

5.3 Modelling scene irradiance 38

5.4 Summary 44

6 Online estimation of scene lighting **47**

6.1 Light representation 50

6.2 Rendering Equation 53

6.3 Specific Light Model 55

6.4 Sample point selection 57

6.5 Determining the sun's Position 65

6.6 Implementation 67

6.7 Test of Concept 70

7 Shadowing **75**

7.1 Methods 76
7.2 Shadow Volume 80
7.3 State of the Art 84
7.4 Implementation 85
8 Shading **89**
8.1 Global Illumination Shading 89
8.2 Shading algorithm 92
8.3 Environment map Re-lighting 92
8.4 Irradiance Volume 95
8.5 Radiance and Irradiance 95
8.6 Creation of an Irradiance Volume 96
8.7 Implementation 101
9 Augmentor System Implementation **107**
9.1 Map generation 107
9.2 Live Video and tracking Input 108
9.3 Separation of camera-positions 108
9.4 Graphical User Interface 109
III Results and Discussion **113**
10 Test **115**
10.1 Introduction 115
10.2 Test scenarios - Simulated Data 116
10.3 Test scenarios - Real World Data 121
11 Conclusion **131**
11.1 Aim of the project 131
11.2 Methods 131
11.3 Results 133
11.4 Discussion 134
11.5 Future work 135
Bibliography **136**
IV Appendix **141**
A Phong Shading **143**
B Testing the diffuse reflectance of regular masonry **147**

C Light estimation method verification test data tables **151**

D Calculating the Sun’s Position **155**

E Test Journals **159**

CONTENTS

Part I

Introduction

CHAPTER 1

Introduction

1.1 A short history of computer graphics

Computer graphics has seen an impressive development within the last few decades. The introduction of the personal computer has revolutionised the way computer graphics is thought of. Initially driven by an academic interest, real-world graphical applications, such as computer games, became wide spread during the late eighties and early nineties. The ground was set, when the first dedicated graphics accelerator chip was introduced in the mid nineties, providing extra processing power to create impressive graphics. The Graphics Processing Unit (GPU), a term coined by NVIDIA Corporation by analogy to the CPU, introduced a separate programming capability which initially was ably to carry out simple tasks, but has evolved significantly and still is evolving in the direction of a higher level of programmability.

The development of computer graphics has had two separate directions for some time. One direction working with - low resolution - real-time 3D animated graphics used in virtual reality applications, and one working with high quality photometric imagery e.g. as used by the film industry. Evidently, high quality is also desirable in real-time 3D animations but until recently both CPU and graphics hardware has not been able to provide sufficient computational power for the exhaustive complex computations needed. Although it is still not possible to implement some of the very high quality methods in Computer Generated Imagery (CGI), it is feasible that it is only a question few years before the two directions will converge.

1.2 Virtual or Augmented reality?

The term Virtual reality (VR), introduced by Jaron Lanier in 1989, describes an environment that is simulated by a computer. VR can be displayed in several ways. Projecting the 3D world onto a regular 2D screen or by providing a 3D view with either the use of stereoscopic goggles, so-called head mounted displays (HMD's) or passive/active stereo imaging (e.g. like the VR Media Lab 6-sided CAVE). VR usually provides some degree of manipulating the virtual world, either by the use of a standard keyboard or more advanced sensory devices such as a 3D mouse (e.g Wanda) or vision based gesture recognition.

Augmented Reality (AR) was introduced by [Wellner, 1993] in 1993 as the opposite of virtual reality. Instead of immersing the user into a purely synthetic world, the purpose of AR is to augment the real world with some sort of additional information, such as virtual objects. Milgram's Taxonomy for mixed reality [Milgram and Kishino, 1994] defines the following three axes.

- Extend of world knowledge (Tracking)
- Extend of presence (level of immersion)
- Reproduction Fidelity (Computer Graphics)

To visualise the relationship of AR and VR, [Milgram and Kishino, 1994] presents the relationship shown in Fig. 1.1.

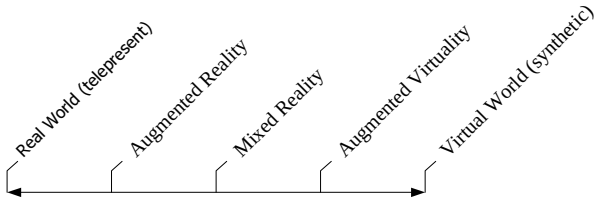


Figure 1.1: The relationship between the real world and the virtual world, as presented by [Milgram and Kishino, 1994].

The real-time branch of AR can, as well as VR, allow for manipulation of the virtual objects. AR finds application in many areas including non-real-time rendered movies and augmented still photos. In Figure 1.2 and Figure 1.3 examples of non real-time and real-time respectively Augmented Reality.

Figure 1.2: An example of non-real-time augmented reality used in the film adaptation of J.R.R. Tolkien's Lord of The Rings. (Image by New Line Cinema)

An additional element of AR, with respect to VR, is how to involve the real world. This can be done by the use of a Head Mounted Display with the ability to project an image into the visual path leading to the viewers eye. Another approach is to record the real world with a camera, adding the augmentation and displaying this on a 2D screen. The fact that AR seeks to merge the virtual objects with the real world, requires much attention in the area of simulating real world features like light and shadows, as to enable the virtual objects to seamlessly blend into the real world.

Figure 1.3: An example of real-time augmented reality. (Image by [Gibson et al., 2003])

Much research work in AR has been performed in the last decade, but for some reason the real-time branch of AR has not reached the same level of popularity as that of Virtual Reality. The reason for this can be found in both the tedious preparations and actual execution of an AR-system. Usually AR involves costly tracking and display equipment and a large amount of labour in producing the prerequisites such as a 3D model of the augmented area, setting up the augmented area with controlled lighting, etc. This, among other factors, is the reason AR systems have been restricted to limited systems most of which are run in research facilities.

1.3 Applications of Augmented Reality

The applications of Augmented Reality extends into many areas.

An area where 3D graphics, and in some cases AR, is already used extensively, is in architectural visualisation and urban planning. This provides users with the possibility of seeing a given construction in situ. This assists architects and urban planners to evaluate the consequences, with respect to e.g. the surroundings, of a given construction layout. An illustration hereof is shown in Fig. 1.4.

An emerging area is the use of AR in entertainment systems directed at information and education, areas known as info-tainment and edu-tainment. Imagine when visiting a historical site it will be possible to experience a reconstruction of a historical battle first-hand, or see how ruins looked in their glory days. Consider, for instance, an augmented reality system in the shape of the familiar pay-to-use-binoculars often located at attractions. This could bring new dimensions of interactivity and information to e.g. historical sites.

Figure 1.4: An example of Augmented reality in urban planning. (Image by [ARTHUR, 2004])

One of the fastest growing industries these days is the computer gaming industry. The industry has already embraced Virtual Reality, so AR seems to be the next natural evolutionary step in computer games when the technique has matured. With AR new levels of interactivity, feedback, and realism may be reached.

CHAPTER 2

Realistic illumination in Augmented Reality

Augmented Reality (AR) holds several challenges within the field of environment based illumination. This chapter will describe some of the specific areas this report is concerned with.

2.1 Illumination

Light is the basis of all real world vision, and also in virtual world vision, that is computer graphics. Light is a part of the complex area of quantum physics, that, although complex, is an important area to develop a basic understanding of the principles of light behaviour.

Light is electromagnetic radiation with a wavelength ranging from ultraviolet at 100 nm up to infrared at 1 mio. nm. In between lies the spectrum of light that is visible to the human eye, which ranges from about 380-770 nm.

Through time there have been many suggestions as how to describe the nature of light. The modern theory is the wave-particle duality, introduced by Einstein in the early 1900s, which is a consequence of quantum mechanics. This theory holds that light and matter simultaneously exhibits properties of waves and particles (photons). This means that some effects of light cam be explained through wave theory, e.g. reflection and diffraction, and other effects, e.g. absorption, due to its particle nature.

This in consequence means, that what the human eye see, is a result of emitted light particles (photons) interacting with the surrounding environment. In general, light leaves some light source, e.g. a lamp or the sun, reflected from many surfaces, and then finally reflected to our eyes, or onto the image plane of a camera.

Some of the effects that are caused by interaction between light and surfaces are listed here:

- Reflection
- Refraction
- Diffraction
- Dispersion
- Polarisation
- Coherence

- Scattering
- Shadowing

These are all relevant contributors to the lighting conditions of a given scene. They must therefore be modelled and applied when required. Obviously, taking into account, all aspects of the physical structure of a scene and the objects within the scene, requires both a great deal of information and also a great deal of computational power to process this information.

Within computer graphics, two ways of modelling illumination are often discussed, local and global illumination.

A simplified model, where the contribution from a light source is reflected directly from a surface is called "local illumination". So, for a local illumination model, the illumination of a surface is independent of the illumination of any other surface.

A "global illumination model" adds to the local model the light that is reflected from other surfaces to the current surface. A global illumination model is more comprehensive, more physically correct, and produces more realistic images.

There are several ways of deducing information of the lighting conditions in a given location. These methods are categorised as light source estimation methods. One commonly used approach to modelling scene illumination is image based lighting. This requires image data covering the entire augmented location, viewed from some central location. The image data is analysed to determine how to model the real scene illumination so that it can be recreated and used when lighting virtual objects. Image based lighting usually involves the steps shown in Figure 2.1.

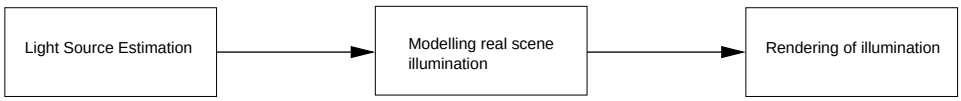


Figure 2.1: The process of image based lighting.

2.1.1 Environments

As described earlier, the illumination of any given point in a 3D scene is the result of a multitude of inter-reflections, refractions, etc. with all surfaces in the scene. The complexity of light transport varies from scene to scene. Two scenarios are considered, an indoor and an outdoor scenario. The indoor scene is typically far more complex than the outdoor scene. The cause for this is primarily the large number of significant local illumination sources, these can be both primary sources like lamps and windows, but also secondary sources like reflective surfaces. Outdoor scenes in relatively open areas are potentially much less complex. In many cases the reflective properties of surfaces in outdoor scenes can be considered to be very weakly specular, potentially diffuse, as will be explained later in this report. The illumination, during day time at least, most likely consists of one primary source, the sun, and a secondary, the sky dome and the environment. The position of the sun can be analytically determined, with

the knowledge of the day of year and time. An illustration of the effects of the two lighting scenarios is shown in Fig. 2.2.

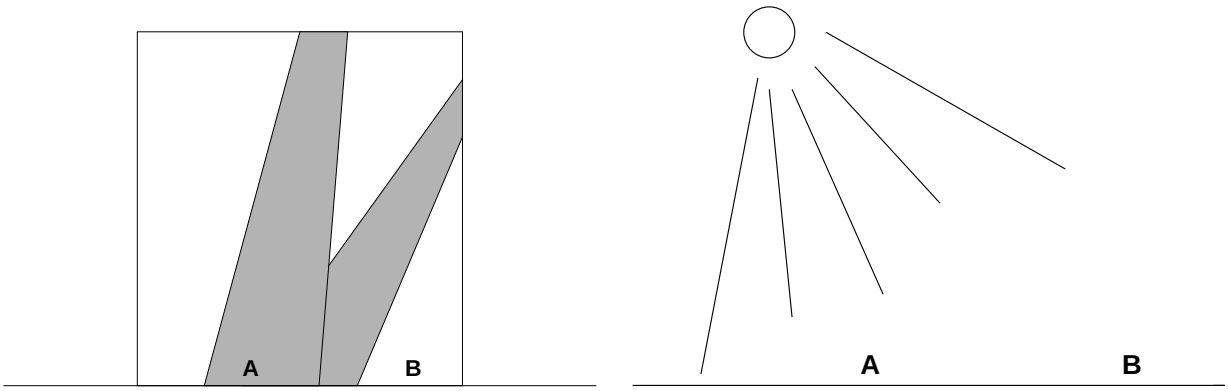


Figure 2.2: An illustration of two lighting scenarios. a) Indoor b) Outdoor

2.1.2 Image Based Lighting (IBL)

To be able to mimic the real world illumination in augmented environments it necessary to record and store the light information of the surrounding environment, in the scene that is to be augmented. A common method is Image Based Lighting (IBL) using image data covering all directions. This enables the computer to compute how light interacts with the augmented objects, but only at the exact time the environment was recorded. There are a number of approaches to capturing the surrounding environment and using it for light calculations. One method of obtaining all direction image data is the so-called light-probe approach, as introduced by Paul Debevec in [Debevec, 1998]. This involves taking a series of images at different exposures to create high dynamic range images (HDRI) of a highly reflective steel ball. This yields an image representing the surroundings. This and all other image based methods will capture the lighting conditions at a specific time and it will only be a fair representation in the immediate vicinity, both physically and time wise, of where the image was formed.

2.2 Dynamic changes

A well known problem when using IBL techniques for lighting virtual objects is that the image data covering all viewing angles represents the lighting conditions at a specific point of time during the course of a day. Outdoor image data recorded at 6 o'clock in the morning will not be representative of the lighting conditions at 3 o'clock in the afternoon, due to both the movement of the earth relative to the sun and potential weather phenomenas such as clouds blocking the sun. The same applies for indoor image data. In the indoor case, changes in light

may be caused by both artificial light being switched on and off, but also sunlight through windows.

2.2.1 Re-lighting images

Handling dynamic light changes with the use of Image Based Lighting (IBL) techniques require some sort of update mechanism, which is able to change the lighting of original captured image to reflect the current lighting conditions. Generally three things have to be known of the image that is to be re-lighted:

- 3D model of scene
- Reflective properties of scene surfaces
- New light parameters

With the knowledge of the three points above, the current light can be removed, and a new light scheme can be applied. For example the effect of the sun moving can be modelled.

There are a number of different methods for acquiring the extensive knowledge of surface properties required for re-lighting of an image. One is to physically measure the surface properties of all surfaces in the scene, yielding a so-called bidirectional reflectance distribution function (BRDF) for each surface. That is a practically impossible task of measuring each surface with a BRDF-measuring device.

Another approach, which is actually feasible, is to approximate the BRDF's using image data covering all viewing angles. The term, inverse rendering, addresses the problem of retrieving the reflective properties of surfaces in a given scene, based on the knowledge of a 3D geometric scene model and a light source estimation, based either on one or several images. This in terms means that the process of inverse rendering, is the process of retrieving the raw surface properties in a scene. The complexity of the inverse rendering problem depends on the accuracy that is desired. Inverse rendering with a global illumination (GI) lighting model is significantly more complex than with a local lighting model. Accordingly, high accuracy of the retrieved surface BRDF's will also complicate the process.

2.3 Previous work

This section explores previous work done by the authors of this project within the field of real-time augmented reality. Furthermore some of the relevant state of the art within the fields of inverse rendering, image re-lighting, light source estimation, and image based lighting is briefly described.

2.3.1 Real-time Augmented Reality

Source: [Andersen and Jensen, 2004]
The current project has its starting point in the 9th semester project *Real-time Augmented Reality* by the authors of this project. This section will offer a short description of the aim and the results of the 9th semester project.

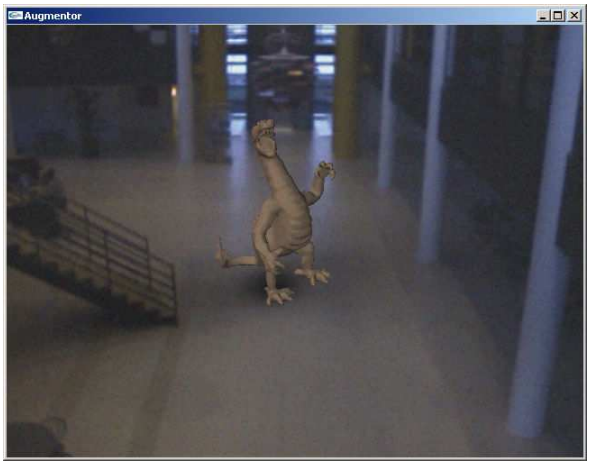
The goal of the project was to create a real-time Augmented Reality system. The objective of the system was to place virtual objects within a real scene, enabling these to interact with existing light, shadow, and occlusion. The user would be able to interact with both the viewing direction and the virtual objects.

In the project, the implementation of a real-time Augmented Reality system, was developed. The primary focus was on how to move towards photo-realistic rendering, enabling seamless blending of real world and virtual objects, in a real-time system. Two primary areas received an in-depth analysis, these were real-time global illumination and shadow casting of virtual objects, both image based methods.

The system developed for the project combined information obtained from a standard web-cam with orientation information from a hardware tracking unit, to enable virtual 3D objects to be placed into the video image. The tracking data enabled transformation of the virtual objects with respect to the orientation of the camera. The system handled occlusions between the real world and the virtual objects by the means of a pre-modelled 3D representation of the environment the application was augmenting. Illumination was done through the use of Irradiance volume as a global illumination approximation, and a radiosity based solution for shadow casting.

In Fig. 2.3 some of the results of the project are shown.

(a) A real ping pong ball, and a virtual shaded with a HDRI environment map.



(b) The dragon is shaded with a HDRI environment map.

Figure 2.3: Two images showing some of the results from the 9th semester project Real-time Augmented Reality.

2.3.2 Inverse rendering and re-lighting

Inverse Global Illumination: Recovering Reflectance Models of Real Scenes from Photographs

Source: [Yu et al., 1999]
The work presented in the paper encompasses gathering images from an environment, where 3D information of the environment is known. By analysing multiple images taken in the environment, where each point is photographed from multiple angles, the surface parameters of each point in the environment can be estimated. That is, the diffuse and specular BRDF for each point is approximated. When extracting the diffuse element for each point, all incident lights are taken into account, which assures that effects like colour bleed is apparent in the calculations. The specular part is extracted using local illumination taking only direct light sources into account.

When the surface parameters are known, a diffuse albedo map of the environment can be created. This process of extracting the various parameters using image data is called Inverse Global Illumination.

Recovering Photometric Properties Of Architectural Scenes From Photographs

Source: [Yu and Malik, 1998]
This paper describes an approach to generate computer rendered photo-realistic architectural scenes under novel lighting conditions, such as different times of day. The method uses a small set of images of the real scene, to achieve this. When determining the existing lighting conditions, the lighting model takes into account three main components, the radiance distribution from the two main sources, the sky and the sun, and also secondary illuminations from the surrounding landscape, i.e. the environment.

During modelling of illumination, a CIE all-weather sky luminance model is computed through interpolation of sky photographs taken a different times of day. Given knowledge of local time, the solar position is found from the latitude/longitude position on earth, and the day number of the year. Finally the landscape is modelled through a low resolution sphere environment map.

The method is able to produce relighted images, very similar to the real scenarios.

2.3.3 Light source estimation and image based lighting

Rendering synthetic objects into real scenes

Source: [Debevec, 1998]
This paper presents a method for using measured scene radiance and global illumination to add virtual objects to a scene with correct lighting. The method uses a high dynamic range image (HDRI) based model of the scene. The HDRI scene model is captured through photographing multiple exposures of a highly reflective metal sphere located at the same place as the virtual objects are to be placed. These low dynamic range exposures are assembled into a HDR image.

The method assumes that the image based scene model is distant, assumed to be unaffected by the virtual objects. A regular image is photographed, and constitutes the local scene, which can interact with the virtual objects. An estimated reflection model is computed for the local scene, so that is can catch shadows and receive reflected light from the new objects. Rendering is performed offline by a standard global illumination method.

Illumination Distribution from Shadows

Source: [Sato et al., 1999]
This paper presents a method for recovering illumination from the knowledge and shape of a real object. The method proposed, estimates illumination distribution of a real scene from image brightness observed on a real surface, with known reflectance parameters, in that scene. Specifically the illumination distribution is recovered from the radiance distribution inside shadows cast by an object with known geometry on to another object also with known geometry and reflectance. The method is able to reliably estimate of complex illumination environments by using the occlusion information of incoming light.

Rapid shadow generation in real-world lighting environments

Source: [Gibson et al., 2003]
This paper focuses on an algorithm for shadow generation at interactive frame rates well suited for Augmented Reality systems. The algorithm proposed uses consumer-level graphics hardware to render shadows cast by synthetic objects and a real lighting environment. A hierarchical shaft-based subdivision of line-space encoding of how each point is affected by light is described. This subdivision is then in turn used to detect which light sources are occluded by a synthetic object, removing the contribution from these sources. The results presented converges towards renderings that are subjectively similar to those obtained by using ray-tracing based differential rendering algorithms.

2.4 Summary

A major issue in all real-time AR systems, using image based illumination methods is how to maintain updated image data for the shading process. This is true for both very complex indoor scenarios, and also for simpler outdoor scenarios.

As can be seen from the papers described, a great deal of work has been done within the field of recovering surface parameters and relighting an image based on this. Examples of work within this field are both *Inverse Global Illumination* [Yu et al., 1999] and *Recovering photometric properties of architectural scenes from photographs* [Yu and Malik, 1998]. The field of light source estimation is also a well covered area, with most methods requiring some sort of calibration objects present in the scene. An example of such a method is described in *Illumination Distribution from Shadows* [Sato et al., 1999]. Furthermore, the area of real-time image based illumination has also received a great deal of attention within the last decade or

so, e.g. in *Rendering synthetic objects into real scene* [Debevec, 1998] and more recently in *Rapid Shadow Generation in Real-World Lighting Environments* [Gibson et al., 2003].

What we propose to do is to develop a method to keep data from an initial image data acquisition up-to-date with current illumination information. This will enable an AR system to shade virtual objects under changing illumination conditions, based only on an initial image acquisition. This will involve using techniques within the fields of inverse rendering, light estimation and re-lighting. Furthermore accelerated GI rendering techniques as well as shadowing are involved. An essential prerequisite for the system to work is, that the environment is an outdoor environment, where the illumination model can be approximated to one primary light source, the Sun, and an ambient term, which will describe reflections from the sky-dome, and landscape.

CHAPTER 3

Defining the project

This chapter aims to explain to the reader, the hypothesis of this project. Furthermore the technology, including both hardware and software descriptions and a general overview of the system are presented.

3.1 The Aim of the Project

The goal of this project is to implement a real-time Augmented Reality (AR) system. The objective of the system will be to place virtual objects in a real scene, enabling these to interact with existing light, shadow, and occlusion. The user will be able to interact with both the viewing direction and the virtual objects.

Amongst the several problem areas within the development of a real-time AR system, select areas will receive in-depth attention in this project, these areas are concentrated in the field of creating convincing, real-time, illumination in environments with dynamically changing illumination conditions. This area has been chosen, as it is an important part of seamlessly blending virtual objects with the real world.

A phenomenon within real-time illumination that currently receives much attention in research, is Global Illumination (GI) which is a general term for methods that approximates real world illumination. Real world illumination is not a trivial task. Complex interaction between not only direct but also reflected light and light scattered within materials requires both complex mathematical models of the world and processing of a large amount of data for each picture rendered.

To make real-time GI feasible, an approximation to the actual lighting conditions in the scene is required. Usually this can be accomplished by analysing a panoramic (360° x 180°) still image representing the entire surrounding environment, a so-called environment map. The drawback of using this method in an environment with dynamic light changes is that it represents the illumination at a given time. This project will present a method for dynamically updating an environment map, so as to be able to update the approximation to the lighting conditions.

The problem of handling dynamical light changes is divided into two major tasks. One will be maintaining the environment illumination information for the chosen light model. When an AR system is run in an arbitrary outdoor environment it must be able to respond to lighting changes during the course of a day. Thus there is a need for an autonomous method that is able to analyse images showing only parts of the entire scene, and derive the necessary light information for the scene from this partial scene data. The other major task will be rendering of the virtual objects, including shading and shadowing. Shading will be done through an Image Based Approach, based on the generated environment maps. Shadowing, which is an

important factor in computer graphics, as they provide the viewer with all important visual cues of object placements, will be done through the shadow volume technique.

In specific this project will explore whether a method for updating an spherical environment map real-time with the use of partial image information will be possible. This will make it possible to mimic dynamical light changes in an outdoor scene, in an augmented reality system using image based lighting (IBL).

The effort in this project will be in investigating the existing state of the art within the areas described in this section, and how to combine these to achieve a functioning system. The focus of this project will be on evaluating how these methods work in a collaborative system. Furthermore it will investigate how to implement an AR system that will function in a non-controlled outdoor environment with respect to dynamically changing light conditions.

3.2 Technology

The emphasis of this system is, as described earlier, on implementing a real-time system. Keeping the real-time demands in mind, when choosing both operating system, algorithms and code language is essential.

Real-time demands limits both the types of algorithms that can be used and the complexity of the scene we are able to augment. Some algorithms are too computationally heavy to be used in an online system, and are best fitted for offline rendering for e.g. movie effects.

The movie special effects industry adopted the concept of Augmented Reality relatively quickly. while the augmentations in movies often look very convincing, these results are not easily achieved. They are based on 3D models with very high level of detail, and often tracking and occlusions (match moving), are done manually frame by frame by accomplished professionals. The same is true for adjustment of the virtual lighting to match the real lighting. These techniques are not applicable for a real-time system.

The augmented reality system in this project seeks to find the balance between advanced relighting and illumination algorithms and a reasonable frame rate. The primary way of achieving this will be through extensive use of pre-calculated data where possible. This means that the system will be comprised of an offline and an online part.

The offline part contains tasks performed outside the system. The initial maps, including the albedo map, the diffuse surface mask, the weighted ambient map, and the normal map will be manually created with the use of a 3D modelling an rendering suite with a built-in GI (Global Illumination) renderer. These maps are described in further detail in chapter 6. The pre-computation of object shading will be carried out in the system.

The online part consists of two main tasks. The first task is the dynamic light estimation process, which monitors real scene illumination. It recomputes the environment map and shading data, if the current conditions differs, to a certain degree, from the last computation. The other task handles the rendering of scene graphics, shaded with the help of data computed by the first task.

Since the main idea of the project is to implement a system, that will function outside a research lab it has been decided that the target platform is standard consumer class workstations. the system is implemented in the object oriented programming language C++ using the MFC (Microsoft Foundation Classes) library for the graphical front-end, and will run on Microsoft Windows platform (Windows 2000 or XP). MATLAB will be used for camera calibration and 3D Studio MAX^{TM}for creating offline data used in the online process.

3.2.1 Hardware

The initial environment acquisition is performed through the method described in [Debevec, 1998]. A high resolution digital Single Lens Reflex (SLR) camera (Nikon D70) with a fixed 100 mm lens is used.

The video feed used in this augmentation process is captured through the use of a Canon A80 digital camera. This camera can be remotely triggered, and is suited for creating image sequences captured with a given time interval. Images are captured in 1024x768 pixel resolution, and down-sampled to a VGA sized (640x480) movie sequence.

For this system it is necessary to continuously determine the absolute orientation of the camera, as to match both the augmented objects and the 3D scene to the real world. Ideally this would be done using a precise hardware tracking unit. For this project it has been decided to use fixed camera positions and angles, for proof of concept, which makes a hardware tracking unit superfluous.

3.2.2 Graphics API

When choosing a graphics API, there are two major directions: Ray-tracer based or rasterization based. All major API's these days are based on rasterization rendering, hence only API's based on this technique will be considered.

Currently two major graphics API's exists: the Open Graphics Library (OpenGL) and Microsoft Direct3D. Direct3D is an integrated part of the DirextX API, proprietary property of Microsoft, for use on windows machines only. OpenGL was originally developed in 1992 by Silicon Graphics, as a descendant of an API known as Iris GL for UNIX. It was created as an open standard, and is available on many different platforms, including windows and Linux. With the release of DirectX8 came the concept of vertex and pixel programs (or shaders), which enables the programmer to replace certain parts of the rendering pipeline with custom code. OpenGL also implements vertex and pixel programs through extensions.

Summarising, there is no difference in what can be achieved with the two API's. Direct3D has a homogeneous hardware independent extension interface, while OpenGL does not. This means that depending on which extensions are used in OpenGL software will become more or less hardware dependent. For this project OpenGL has been chosen, since the authors have the most experience with this API.

3.3 System Description

The system designed for this project will consist of two main parts. An offline part, in which the prerequisite data is created, and an online part where the actual illumination updating and the rendering is carried out. In the following section, a principal outline of the system will be given, and then a rendering pipeline design proposal is presented.

3.3.1 Principle

The offline part in the method applied in this project, consists of several tasks. The first task is to acquire an environment map of the location that is to be augmented. This will usually be done through photographing a highly reflective metal sphere, which, after some processing, produces a $360^\circ \times 180^\circ$ image. Second task will be building an exact sized 3D model of the location, where at least all major surfaces are represented. Based on the 3D model and the Sun's position and radiant power, an irradiance map, which is a map describing how light is distributed in a scene, can be made in any standard rendering package, preferably one with GI capabilities.

From this initial data acquisition phase, an albedo map, describing the diffuse surface properties of the entire location, can be created by dividing the environment map by the irradiance map.

When the initial data acquisition and preparation is completed, the online part of the system can be used. The online part is where the actual light estimation, and updating of the environment map for illuminating virtual objects is carried out.

Basically a video feed, representing a small section of the environment is examined for illumination values, with the use of the albedo map. From this examination the illumination of the location is estimated. The estimation is used to re-light the entire environment map, which in turn is used for shading the objects in the scene with an image based lighting technique.

3.3.2 Rendering Pipeline

In Figures 3.1 and 3.2, a sequential view of the processing in the system are shown. The offline part is carried out with the use of readily available software, like HDR-Shop for assembling HDRI environment maps, and 3D Studio MAX for the initial GI irradiance map rendering. Figure 3.2 shows a sequential view of the tasks performed in on-line part the system.

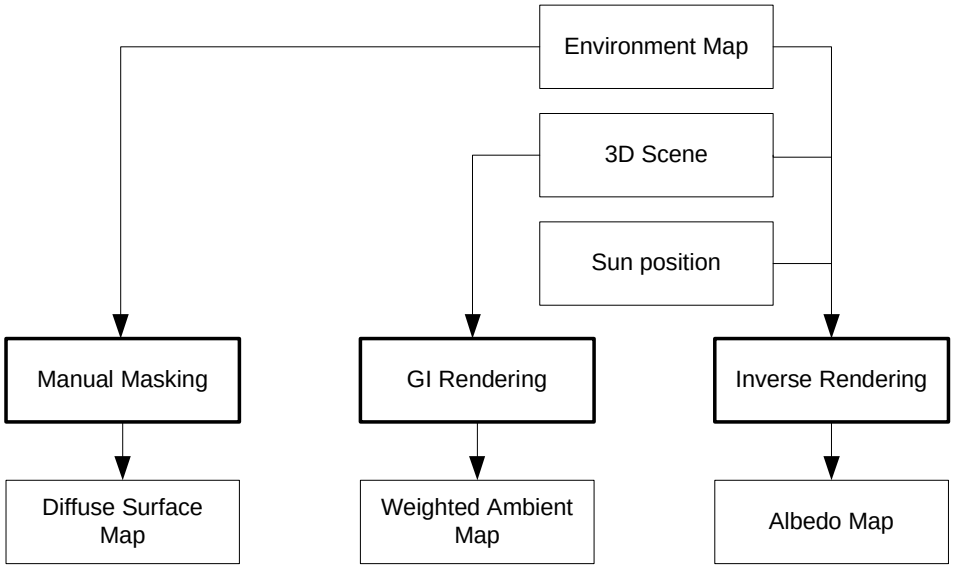


Figure 3.1: The offline process in the system. Tasks are marked with heavy border boxes.

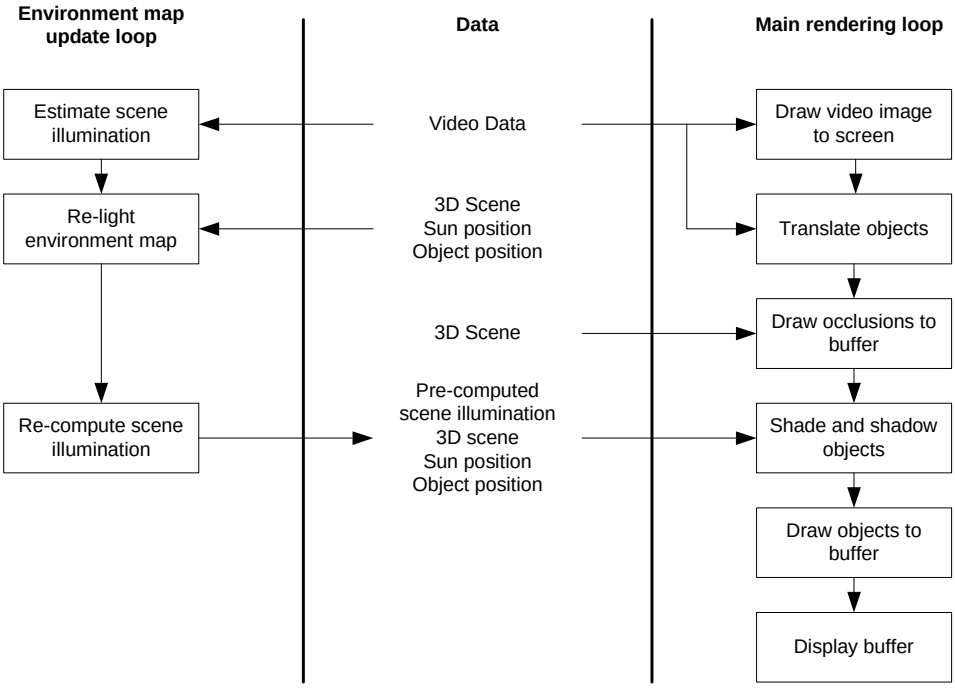


Figure 3.2: The online process in the system. Left column is the light estimation and environment updating loop. Middle column is the data available. Right column is the actual rendering loop.

CHAPTER 3. DEFINING THE PROJECT

CHAPTER 4

Results Preview

This chapter presents the results of the project, to give the reader a better understanding and overview of the project before the methods, used to create the images, presented in this chapter, are described in detail.

The aim of this project has been to explore whether a method for updating a spherical environment map real-time, with the use of partial image information, is possible. And to use this to enable mimicking of dynamical light changes in an outdoor scene in an augmented reality system using image based lighting (IBL).

The first step in rendering the objects for the scene, is to create an environment map of the scene. The environment map of the scene is then used to create an albedo environment map of the scene. An albedo environment map used in the project is seen in Figure 4.1.

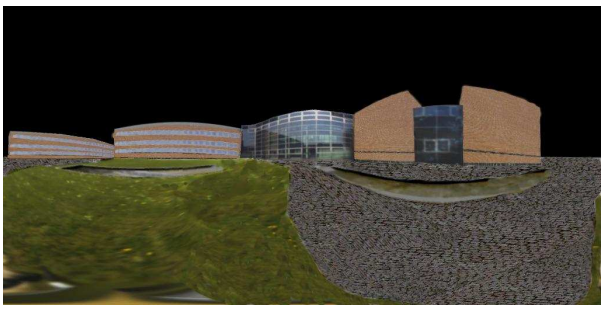


Figure 4.1: An environment albedo map of the scene used in the results.

The offline part of the system consists of determining the albedo environment map, creating geometry of the scene, and creating the diffuse reflection map, that indicates which surfaces in the scene can be considered diffuse. The online part uses this information to determine the light parameters of the scene, from an input image, which reveals partial information of the scene.

These estimated light parameters are used to re-light the albedo map, in order for to obtain a representation of how the environment is estimated to look in the current frame.

The online estimated environment map is used to shade a virtual object placed within the augmented scene. Along with a light addition from the sun and shadows cast from the object onto the real scene, this lets the virtual object blend into the real scene. This process is illustrated in Figure 4.2.

Figures 4.3 and 4.4 shows an excerpt of the images produced using the light estimation method presented in this project.

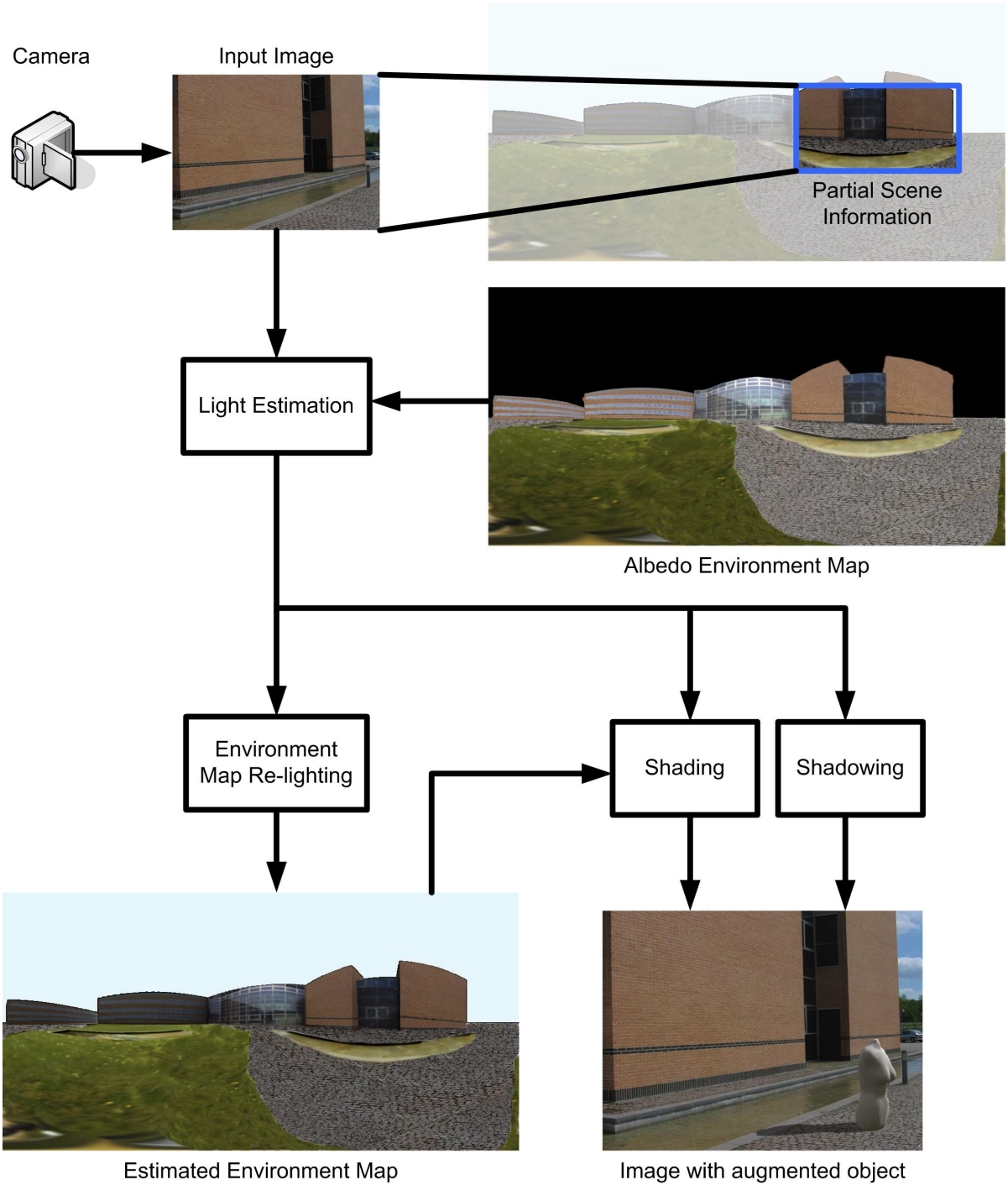
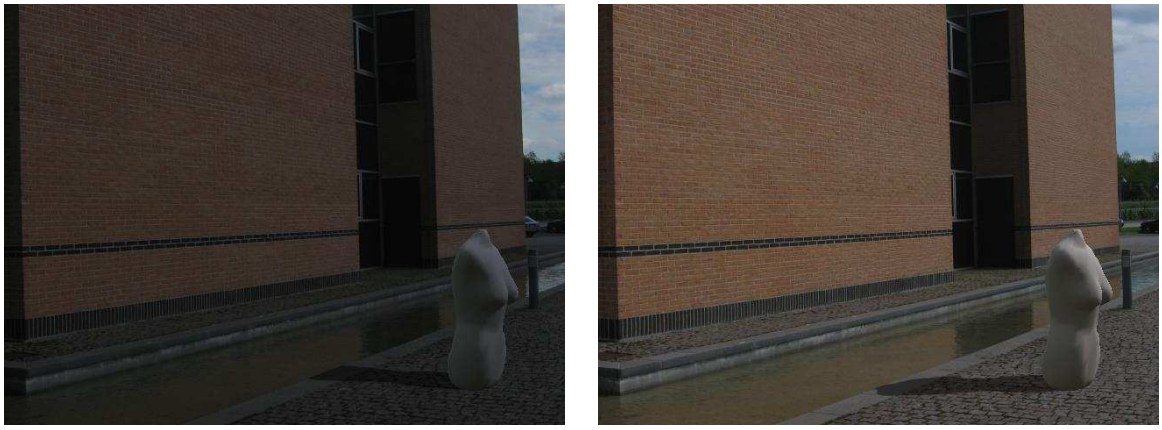


Figure 4.2: An illustration of the processing of one frame in the implemented system.

(a) *(b)*

Figure 4.3: Preview of the result. The object is shaded based on the light estimated for the frames.



(a) *(b)*

Figure 4.4: Preview of the result. The object is shaded based on the light estimated for the frames.

CHAPTER 4. RESULTS PREVIEW

Part II

Methods

CHAPTER 5

Offline Data Acquisition

This chapter deals with the offline data acquisition in the system. This includes exploring how to acquire environment maps and how to recover surface parameters throught inverse rendering.

5.1 Acquiring environment maps

[Debevec, 1998] has developed a method for creating environment maps using a so-called light probe. A light probe is typically a highly reflective metal sphere. The method is to take a series of different exposures, preferably in 1-stop increments, of the probe from three or more different angles. From these exposures, a high dynamic range image can be assembled, and the photographer and distortions masked out by composition images from more than one angle. The reflective sphere can the be cropped and transformed into either an angular map or a latitude longitude map. Paul Debevec has created a practical tool for this task, called HDRShop, which is used in this project.

5.2 Inverse rendering

Creating realistic images has been a major focus in computer graphics throughout its history. This has lead to several mathematical models and algorithms that can produce physically realistic images from knowledge of scene geometry, light positions and surface properties. These images accurately describe the physical quantities that would be measured from a real scene. Because these algorithms can predict real images, they can also be used in inverse problems to work backward from photographs to scene attributes, such as the illumination of a scene. This principle is called inverse rendering. The primary focus of this chapter will be how to acquire environment data, and how to extract usable surface properties from it.

In order to extract the light affecting object surfaces in an image, the following information must be obtained about the scene:

1. 3D scene information
2. Illumination source types and positions
3. Surface properties

Light distribution in a given scene can, depending on the applied light model, be very complex. Within the field of computer graphics, illumination models are often divided into two categories, local and global illumination models. In general, when light leaves a light source it travels to visible surfaces. If a surface is occluded from the light source it will be in shadow (either penumbra or umbra). The occlusion of a surface point from direct light sources in a scene does not mean that no light will arrive at the point. In the real world all surfaces are reflective to some degree and will contribute to the illumination of a scene. This in consequence means that the illumination of a given point in a scene is the result of both direct illumination and indirect illumination, e.g. light reflected by surfaces.

5.3 Modelling scene irradiance

One way to retrieve surface parameters from an environment image, is to generate the corresponding irradiance map. Irradiance is the radiometric term describing the received power per unit area. Thus the irradiance map is the difference between the surface reflectance, and a lit image. An example of a rendered image, with corresponding irradiance and albedo map is seen in Figure 5.1.

(a) *(b)* *(c)*

Figure 5.1: (a) Radiance map. (b) Irradiance map. (c) Albedo map.

The relation between the three images is, if pure diffuse surfaces are assumed, ideally:

$$\rho(u, v) \cdot H(u, v) = L(u, v) \tag{5.1}$$

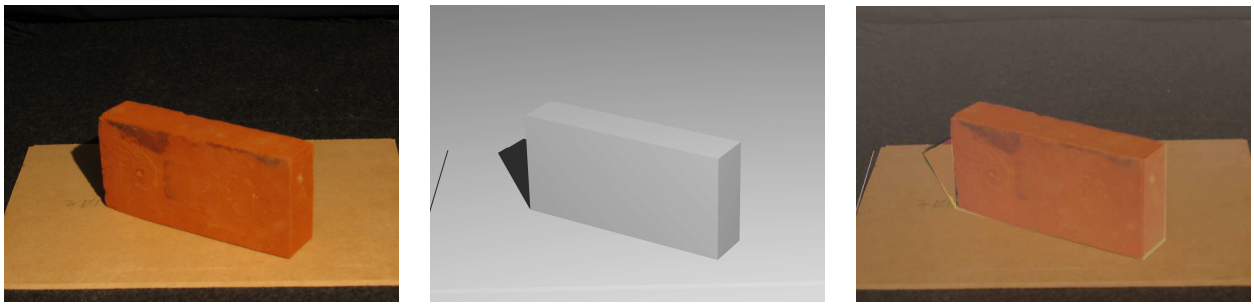
Where $\rho(u, v)$ is the albedo map, $H(u, v)$ is the irradiance map and $L(u, v)$ is the resulting radiance map. u and v are the pixel coordinates in the images.

The radiance map is the actual image of an environment. The purpose of inverse rendering is to obtain the albedo map, it therefore follows that given an image of an environment, a corresponding irradiance map is needed. The irradiance map may be estimated if a 3D-model of the environment and an approximated model of the light sources present in the scene.

Given a image, and a corresponding irradiance map, the albedo map may be calculated by rewriting equation 5.1 to:

$$\frac{L(u,v)}{H(u,v)} = \rho(u,v) \tag{5.2}$$

In practise, one way to calculate the irradiance map of an image, is to create a 3D representation of the environment. All surfaces are set to white and rendered with the same lighting conditions as the original image. The result is an irradiance map of the scene. This approach does not take effects like colour bleed into account, that would require multiple iterations.



(a) *(b)* *(c)*

Figure 5.2: (a) Radiance map. (b) Irradiance map. (c) Albedo map.

An example of an inverse local illumination rendering is seen in Figure 5.2, where a setup in a controlled environment with geometry, light source position, and radiance as known factors and albedo as the unknown. The example setup is a brick on a wood-plate, and one light bulb, approximating a point light source. The surrounding room is not perfectly non-reflective, so light bounces will, to some extent affect the colour of the shadow. From the shadow colour the ambient value is estimated, as the light bulb colour is estimated by photographing the light bulb at a fast exposure. The photograph of the setup is seen in Figure 5.2a. The resulting irradiance map from the constructed 3D model is seen in Figure 5.2b. These two images are applied to equation 5.2 to achieve the albedo map seen in Figure 5.2c. Due to imperfections between the real image and the reconstructed 3D model, the recovered albedo map has imperfections, as the real shadow is not cast exactly the same way as the virtual which shows at the edges. To make a perfect albedo map, the 3D model must be a perfect reconstruction of the real world.

The following sections explores inverse rendering using local illumination and global illumination methods.

5.3.1 Computing irradiance using local illumination

In local Illumination the distribution of light is calculated as the direct path from light sources to any given surface. This means light sources affect all surfaces that are directly visible to them, but effects such as light reflection of the lit surfaces and indirectly illuminating other surfaces, is not included in a local illumination model. To compensate for the lack of indirect

illumination by reflection, an ambient illumination term is added to the local illumination models.

In order to successfully make an inverse rendering using local illumination, three facts must be known about the surroundings:

1. The geometry of the environment
2. The position and colour of each light source
3. The ambient term of the surroundings

When this environment information has been established, it is possible to compute the irradiance map of the surroundings using a local illumination model. Using an image of the surroundings, equation 5.2 may be used to compute the albedo values of the various surfaces.

This approach assumes that all surfaces are diffuse, as the surface properties have not been estimated. If a term like specular is to be taken into account, all surface properties must be known. The system developed in this project is deployed for outdoor use. The assumption made in this regard, is that the environment used in the system consists of ground and buildings. The ground is either stone, concrete, dirt, or grass, while the buildings are primarily made of bricks. To avoid window and glass surfaces, which are highly specular, posing any problems these must be manually marked, so that the system will disregard these areas. The assumption is that both ground and building materials are diffuse surfaces. This assumption has been explored in appendix B on page 147, and proved to be justifiable.

Estimation of light sources and ambient term

In outdoor environments, the main light sources, during the day, are the sun and sky. The sun provides the most dominant light source at day, while the ambient term can mainly be determined from the colour of the sky, and to some extent the surrounding landscape.

There are several ways to determine the parameters of the sun as a light source. The position of the sun can be determined using a sunlight model as in [Preetham et al., 1999] if the time of day and the viewpoints longitude/latitude coordinates are known. The position of the sun may also be determined if a panoramic view, as in Figure 5.3 is available, with a visible sun. The area with the highest intensity in such a map is most likely the sun. If the sun is not visible as in 5.3 the the sky is the main light source, whereas the main light source is in fact ambient.

In a case where the main light source is “ambient”, a local illumination model is unsuitable to create irradiance maps, as the ambient term is merely a constant. In that case a global illumination model is the best suited to perform the inverse rendering.

To find the intensity of a light source, in this case the sun, there are two obvious ways to determine it’s colour. If the position of the light source is known, e.g. from a sunlight model, and the albedo value of a diffuse surface directly lit by the sun is known, Lambert’s cosine law may be used to determine the colour and intensity of the sun. In this calculation the ambient term must be known as well.

Figure 5.3: A panoramic view of an outdoor environment.

The ambient value can be determined by finding a uniform surface partially occluded from the sun. In this case subtracting the shadowed area from the directly lit equals direct light influence. If such surfaces are present in the scene, an equation system can be developed to determine the relation between direct and ambient terms needed to create an irradiance map.

Another way to determine the intensity of the sun, is to photograph it using a camera with an exposure time short enough to avoid the sun saturating pixels in the image. Few cameras are able to do this, which is why a neutral density filter can be used to reduce incoming light. If a High Dynamic Range (HDR) light probe image is acquired in this way it may be used to determine the direction and the colour/intensity of the sun at the same time.

A virtual scene has been setup in order to test how whether a local illumination inverse rendering will be applicable to extract an albedo map from a globally illuminated image.

The test scene is seen as a panorama from a central point in Figure 5.4. Real data, used in the system, will be in the same format. The lighting of the scene consists of two sources. The sunlight, which is seen on the edge of the image is the primary light source, which can be seen from the very dominant shadows it casts.

The remainder of light in the scene is provided by the skylight, which is why areas in shadow are not entirely black.

All surfaces are considered diffuse. The direction of the sunlight is determined from the suns position in the panorama image, the intensity is determined from the intensity in the image. The ambient term is found by comparing an area in shade to a similar area in direct light.

This information is used to render the irradiance map seen in Figure 5.5.

Applying these images to equation 5.2 yields the albedo map shown in Figure 5.6.

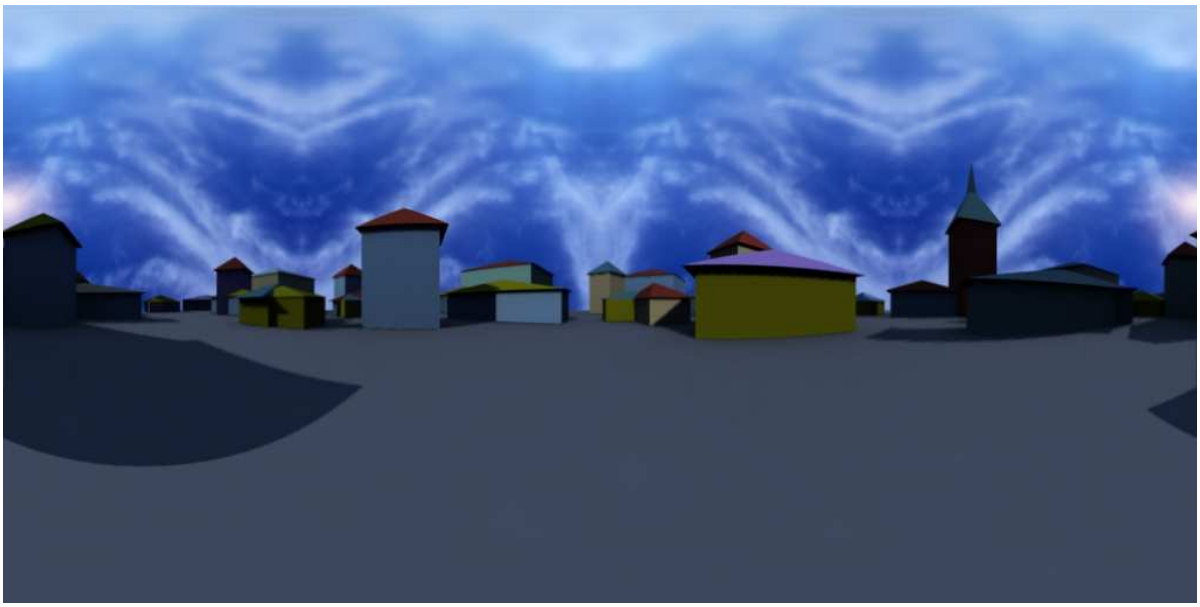


Figure 5.4: A panoramic view of a virtual test scene

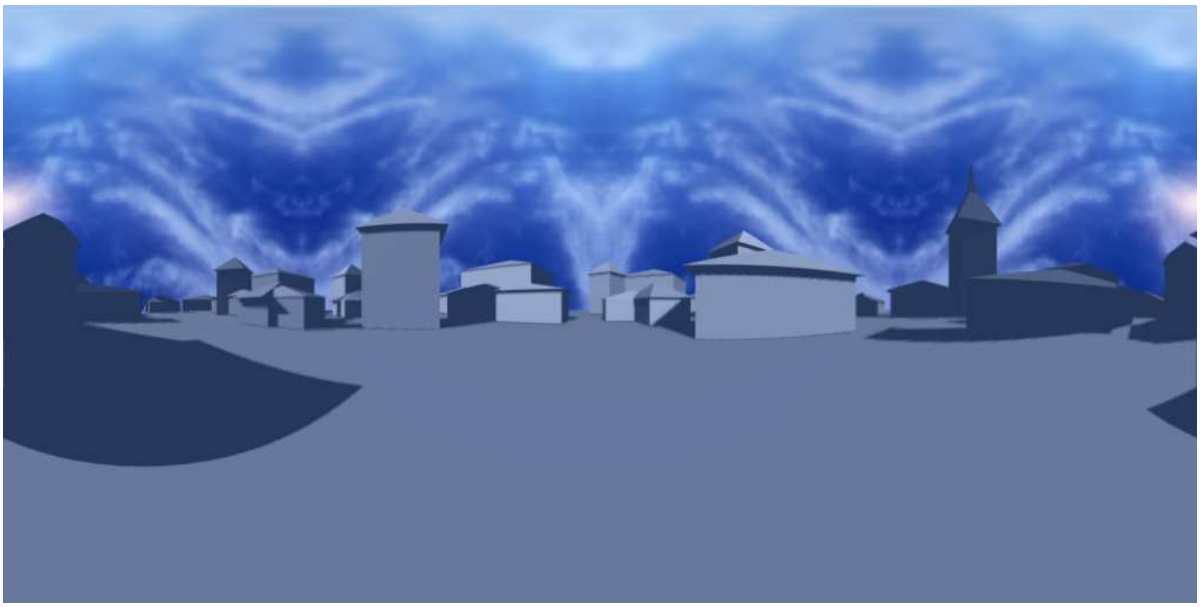


Figure 5.5: The rendered irradiance map using local illumination.

Figure 5.6: The calculated albedo map using local illumination.

5.3.2 Computing irradiance using Global Illumination

Using global illumination in inverse rendering requires two sets of data to be known, the geometry of the scene, and a radiance map taken from a central point in the scene.

The radiance map is used as a skylight, that lights the virtual scene geometry.

To obtain the irradiance map, the skylight is set to light the geometry with white surfaces. The result is an irradiance map, as seen in Figure 5.7.

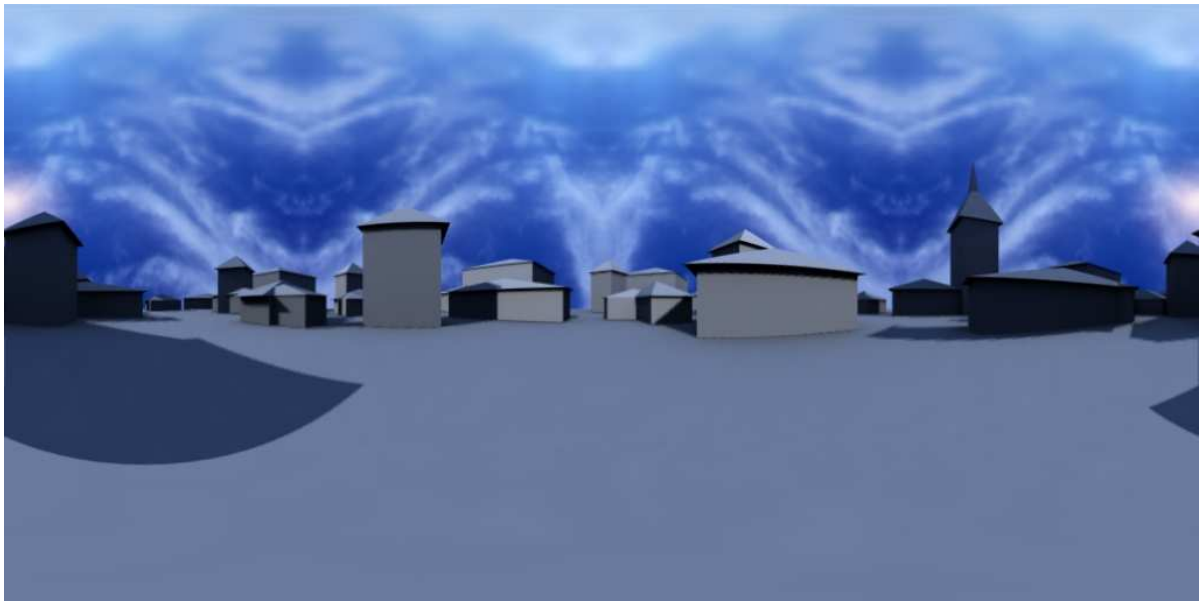


Figure 5.7: The estimated irradiance map using global illumination

Equation 5.2 is applied to the resulting irradiance map and radiance map, to obtain the albedo map. The albedo map of the simulated data using inverse global illumination rendering is shown in Figure 5.8.

Figure 5.8: The calculated albedo map using global illumination

If effects such as colour bleed between surfaces is apparent in the radiance map, an iterative inverse rendering process is required in order to remove this effect in the albedo map. When the first irradiance map approximation is obtained, and used to calculate the first approximation of an albedo map, the albedo map is then mapped to the surfaces in the scene. The scene is rendered again, using albedo map surface colours instead of the white colour. The first approximation of the albedo map is subtracted from the rendered image and the result is a second approximation of the irradiance map that is used to calculate the second approximation of the albedo map. This process is repeated until the differences from one iteration to the next becomes negligible.

For comparison, the real albedo map from the surfaces is seen in Figure 5.9.

5.4 Summary

The methods described in this chapter will be used in the offline initialisation phase of the system. It is essential that the surface reflectances, the albedo map, produced for use in the online light source estimation, is as accurate as possible. Keeping in mind that the offline phase has no time demands, a global illumination method is used for the inverse rendering, as it will produce the most accurate results.

Figure 5.9: The original (true) albedo map

CHAPTER 5. OFFLINE DATA ACQUISITION

CHAPTER 6

Online estimation of scene lighting

This chapter explores how scene parameters can be extracted from a live video feed real-time.

One purpose of the system developed through this project is to estimate the lighting of the surroundings in real-time, and light augmented virtual objects, super-imposed upon real imagery, accordingly. When the system is running in an arbitrary outdoor environment the lighting is changing over the course of a day. This creates the need for an autonomous method, that is able to derive the light information of the scene whenever it changes.

The information known to be available to the method before execution is limited to:

Albedo environment map: Created in the offline process, by performing inverse rendering on an environment map photographed in the scene. The albedo is the reflection coefficient of a surface.

3D model of the environment: An exact proportioned geometric representation of the environment that has been photographed.

LDR input images of the environment: During online execution the method will be provided with input images photographed by a camera. These images are provided in a Low Dynamic Range, which means that not all points in a scene photographed with the camera is usable due to a high contrast between different areas in the environment. The result of this would be that while some areas are within the dynamic range of the camera, other regions will be either saturated or under-exposed. A feature that is present in most cameras today is Automatic Gain Correction, which enables the camera to regulate the aperture size and exposure time of each image in order to get the best exposed image, which sets demands for a light estimation method to be robust in regards to this camera feature.

When updating an environment map from a dynamic lighted scene, several solutions may be considered. Some of which are listed below:

Light templates: If a set of environment maps is recorded from the scene, at different times of day, where the lighting changes, dependent on the weather and time of day. This set of environment maps are templates of different lighting conditions. To utilise these templates the system would have to analyse the images it is acquiring and determine which of the templates matches the lighting best, and switch to that environment map in the scene. When the scene lighting has changed enough to be visually different from the current used environment map, set by a certain selection criteria, the system chooses a new environment map that fits the lighting seen in the live input images best, and gradually changes the derived virtual lighting in the scene to this new setting.

Light probe: If the [Debevec, 1998] method is utilised, by having a reflective metal sphere in the scene, and a secondary camera that records it at all time. The image of the probe is used to extract light information, which is used to update the virtual lighting of the scene. While this method may be able to obtain HDR information from the light probe in the scene, it is a comprehensive setup to have two cameras and a stationary light probe.

Real-time EM update: It is possible to create one environment map of the scene, recorded from the centre of the scene. During execution of the system, the camera records the scene at various angles from a given position. These images from the camera is mapped to the appropriate areas of the environment map recorded in the offline process, and the environment map is thereby partially updated dynamically. The problem with this approach is that the camera is only able to photograph the scene in a low dynamic range (LDR), which means that certain light parameters may be lost due to this limitation.

Average difference: A simple approach to the problem would be to photograph an environment map of the entire scene offline. During execution of the system the intensity of the pixels in the input images is analysed. The intensity mean value will be used to evaluate how the current lighting differ from the lighting when the original environment map was created, by comparing the measured intensity mean to the corresponding mean value in the original environment map. The average intensity difference between the two lighted environments is found, and the original environment map is scaled by the scalar derived from the difference. All virtual objects derive their lighting from this environment map, and is thereby updated. This method will only work effectively if all light sources are static, and always have the same relation between their intensities. In almost any environments these limitations to the lighting does not apply.

Image analysis: A final approach is to analyse the LDR images provided by a camera and the derive light parameters of the scene from these. The derived light parameters are used to re-light the original environment map to correspond to the current lighting of the scene.

The advantages and disadvantages of the five approaches are summarised in table 6.1:

Method	**Advantages**	**Disadvantages**
Light templates	Simple Low amount of processing required	Limited to the number of different templates
Light probe	High accuracy HDR estimation	Impractical setup
Real-time EM update	Simple	LDR limits precision of light estimation
Average difference	Simple	Imprecise results if lights moves
Image analysis	Good approximation Not limited by LDR	Requires assumptions concerning: light model and surface properties. Requires much knowledge of scene

Table 6.1: Advantages and disadvantages for each proposed method

The latter method has been chosen, which is to analyse the LDR images provided by a camera and derive light parameters of the scene from these. The derived light parameters will then be used to re-light the original environment map to correspond to the current lighting of the scene.

The purpose of the method described in this chapter is to estimate scene illumination. Estimating the lighting of a scene must be done in accordance with a light model, which must be chosen. The light estimation then analyses the images of the scene, and estimates the unknown parameters of this light model, in order for it to approximate the lighting of the scene as closely as possible.

The system is set to work in an outdoor environment. This means that the direction of the sun, which is by far the dominant light source of an outdoor scene during daylight, can always be computed at any time of day, using a sunlight model, when the geographical coordinates of the scene is known.

When the factors in a light model are estimated, it is possible to re-light an environment map based on these information. As the sun is the dominant light source, it is equally important to estimate how clouded it is, as the clouds have impact on the intensity of the sunlight. If there are no clouds the sun shines brightly on every surface directly visible to it.

If the sky on the contrary is cloudy, the light from the sun is scattered in the clouds and the entire surface is lit by the light from the sunlit clouds. The two situations are shown in Figure 6.1. The method for estimating the scene lighting must be robust in regards to both situations.

(a) Clear sky, sunny.

(b) Overcast.

Figure 6.1: Example of two different weather situations, that both needs to be estimated correctly.

The method to estimate scene lighting from the video-feed must be able to perform its task within a confined time-frame. If the method is to effectively estimate dynamic light changes, ideally it will have to estimate the lighting of the scene on a per-frame basis, and, if necessary, change the lighting of the virtual objects from frame to frame.

If the method solely relies its estimation on the input from the image acquired, the method may not be very effective in all situations, as the camera may in some cases photograph an

area of the scene that does not reveal adequate information of the overall lighting of the scene. Therefore the method should be able to determine the usability of all information it gathers. If the camera photographs the sky, or a specular surface, the method should be able to filter the information it can not derive light information from. Furthermore if the camera films an area which is in shadow for a long period of time, the method should be able to establish that it can not derive information of the light source the area is shadowed from. In the situation that the input image only covers shadowed areas, the method should not try to estimate the light source, but rather rely on the last known estimation of the light source, until information becomes available again that the light can be estimated from.

The input images will be delivered to the light estimation method in a low dynamic range(LDR) format. This means, that there is a chance that certain areas of the images from the video feed are saturated. If there are saturated pixels in the images, the method must ignore these pixels, as they are not suitable for information acquisition. The occurrence of this situation is minimised if the cameras Automatic Gain Correction is enabled, but this instead sets demands for the adaptability of the light estimation method, as the lighting of the input images are changed actively by the camera to cover the highest possible range.

Another element that must be taken into consideration, when acquiring live video in a system as the one developed in this project, is that dynamic objects can occur in the scene, which includes people walking by, cars that are parked, all are elements, that may not appear on the original albedo environment map. If the method is to effectively extract the light parameters of the scene, it must be able to detect such anomalies in the supplied images and disregard them in the further light calculations.

6.1 Light representation

This section explores which light model may be used to represent the lighting of the scene.

When the light is estimated, it needs a representation. Two ways of representing the knowledge of an environment would be:

Environment map: The light changes is determined by updating an environment map of the surroundings, from which lighting of objects can be derived.

Light model: The lighting of the scene are fitted to the parameters of a mathematical light model.

The solution using an environment map is a low level representation of the scene lighting, as it is pixel-based, and does not hold any parametric representation of the lighting, though parameters can be derived from it through processing. The Light model approach is a high level representation of the environment, as it is a parametric model.

The approach chosen is to estimate the lighting of the scene based on a light model, as the higher level of representation makes it possible to determine the lighting of part of the environment not visible to the camera.

In the choice of light model some considerations must be taken to the accuracy in regards to reality:

Level of reality How close is the model to reality, is it physical correct?

Level of complexity: How complex is the model, is it possible to use it real-time?

The considerations of choosing the right light model is a trade-off between how realistic it will be possible to estimate the real scene lighting, and how much processing time is needed to perform this estimation. If a completely realistic model is used, it may be impossible to perform it real-time, while a simplified model may be fast, but not very close to reality.

When choosing a light model, the first choice is between using a local illumination or global illumination model to estimate the lighting. Normally the global illumination versus local illumination discussion is referred to the challenge of rendering a scene naturally, but is equally relevant in the inverse case, as global illumination is an approximation to real lighting, and thus if it may be reverse rendered the lighting could theoretically be almost perfectly estimated.

If a global illumination model is used to perform the light estimation, it would entail that all surfaces in the pixels sampled during a frame should be tested for incoming light over a hemisphere. This is a very costly process, and can be faulty if the hemisphere sampling is in low resolution, where powerful light sources are missed. The process can be sped up by pre-calculating the hemispheres in various points in the scene, this way the sampling in each hemisphere could be skipped during the real-time sampling.

Though this process can be optimised to using fewer samples and detecting the dominant light areas and take them into account, it is dubious that a global illumination model can be used in real-time light estimation. Furthermore such an implementation would generate a large set of equations with an equally large set of unknown variables, whereas the practicality of a global illumination solution is not possible to implement in a real-time application.

If a global illumination solution where to be implemented, another approach would be to surround the entire scene with a limited number of directional light sources, analyse the various surfaces, and develop an equation system to determine the light intensity of each source.

One model that uses this approach is the method presented in [Madsen et al., 2003]. This method addresses the problem of using IBL in real-time, by estimating the position and radiances for a small number of point light sources from the analysis of an environment map. The approach is to lower the number of light sources affecting the scene. That is, to approximate the complex omni-directional lighting environment with a small number of light sources, so the resulting virtual lighting resembles the real lighting of the scene.

The method analyses different exposures of the scene, and detects what areas are the most significant in radiance, and what their position and intensity/colour is. An example of how objects are illuminated with this method can be seen in Figure 6.2a.

The model used by [Madsen et al., 2003] is a global illumination approximation using local illumination. An advantage of this method is that using these environment light sources automatically ensures that surfaces are only lighted by those light sources visible to them, as

(a) Spheres illuminated by the [Madsen et al., 2003] method.

(b) The [Madsen et al., 2003] method used to create shadows in the scene.

Figure 6.2: The [Madsen et al., 2003] method fits point light sources to an environment map, and use these to shade objects and create their shadow.

opposed to the ambient term of local illumination, where a constant light is added to all surfaces.

A drawback is that for this method to work perfectly the entire environment map is needed. This problem can be worked around, by placing the environmental lights equally in all directions, and always update those that the camera points at during execution. The big problem of the method, in relation to dynamically updating of the lights, is that it involves a diffuse filtering of the environment map, which is a very slow process. Furthermore the use of this model would ideally require a HDR input, in order for it to estimate the intensity and colour for each environmental light source.

The approach may be altered to analyse all surfaces in the scene that are visible to the camera, and determine how various fixed environment light sources would have to be configured, in order for the different surfaces to look the way they do. This will create a set of equations in the number of samples obtained in the input images, and with as many variables as there are environmental light sources. This approach would circumvent the need for HDR input in the [Madsen et al., 2003] method.

Local illumination is a simplified approximation of real lighting capable of running at high speed. The effect of global illumination, where surfaces in shadow are illuminated by light reflected of the lit surfaces, is in local illumination simplified to an ambient term, that is a constant light addition to all surfaces.

Local illumination models simplifies the surface calculations to computing the radiance received from all point light sources in the scene. To implement a light estimation method using a local illumination model entails that the intensity of each light source is an unknown variable along with an ambient light term.

The pros and cons for each model is seen in table 6.2:

Method	**Advantages**	**Disadvantages**
Global illumination	Realistic result	Very heavy process
[Madsen et al., 2003]	Good results	Best with HDR
Local Illumination	Fast Simple	Less realistic result

Table 6.2: Advantages and disadvantages for each proposed method

A pure Global Illumination model is impossible, and considering the outdoor environment it would be dubious to apply an estimation of the parameters of such a model.

The method inspired by the [Madsen et al., 2003] method, placing a number of environmental light sources and determine how they would be affecting each point of the surfaces and thereby be enabling intensity estimation of each light source, is a possible model to use. The drawback of implementing this method is that many point samples of the intensity at various points in the scene would be required, depending on the number of directional light sources used. This raises the computational requirements as well as the number of required samples may mean that more than one image is needed to derive all parameters. The advantage of the approach is that it is possible to update the lighting of the virtual object almost instantly when the light source intensity is estimated.

The local illumination would require less estimation of light parameters, making it a faster method, but possibly inaccurate when the global illumination is taken into account. The advantage of the local illumination model, aside from less parameters, is that it fits the environment used in this project, an outdoor scene.

Outdoor scenes during daylight approximately equivalents a local illumination model with the sun being the only dominant light source. Therefore the light estimation will be based on a local illumination model.

6.2 Rendering Equation

The light of the scene must be fitted to a rendering equation, that describes the relation between the light emitted from the light source, and the light that each surface in a scene emits.

The radiance at a point $\vec{x}$ in a scene, is given by the rendering equation:

$$L_o(\vec{x}, \vec{\omega_o}) = \int_{\Omega_i} f(\vec{x}, \vec{\omega_i}, \vec{\omega_o}) \cdot L_i(\vec{x}, \vec{\omega_i}) \cdot \cos\theta_i \quad d\vec{\omega_i} \tag{6.1}$$

Where $\vec{\omega_o}$ is the directional vector from point $\vec{x}$, to the view position. $\vec{\omega_i}$ is the directional vector to an incoming light source. $f(\vec{x}, \vec{\omega_i}, \vec{\omega_o})$ is the Bidirectional Reflectance Distribution Function (BRDF) for a given surface. $L_i(\vec{x}, \vec{\omega_i})$ is the incoming radiance at $\vec{x}$ from direction $\vec{\omega_i}$. $cos\theta$ scales the incoming light by the angle between the normal of the surface and the incoming lights directional vector $\vec{\omega_i}$. The incoming directional vector $\vec{\omega_i}$ is integrated over the entire hemisphere of the surface at point $\vec{x}$.

In this section the focus is on how parameters of a light model may be derived from the shading of diffuse reflections in a scene. Diffuse reflections are a very good source of information of light, when the albedo reflection coefficient of a surface is known.

The BRDF for a diffuse, or Lambertian, surface is:

$$f(\vec{x}, \vec{\omega_i}, \vec{\omega_o}) = \frac{\rho_d(\vec{x})}{\pi} \tag{6.2}$$

$\rho_d(\vec{x})$ is the albedo of a surface at point $\vec{x}$. The albedo is divided by π in order to normalise the output of the diffuse BRDF.

In equation 6.1 all incoming lights to a point are taken into account, as in Global illumination. Rewriting the equation to correspond with local illumination, with N light sources yields:

$$L_o(\vec{x}) = \frac{\rho_d(\vec{x})}{\pi} \sum_{n=1}^{N} L_{i,n}(\vec{x}) \cdot \cos\theta_i \tag{6.3}$$

In order to derive the parameters of the light model it is assumed that the albedo ρ_d and $cos\theta_i$ is known. The albedo is obtained in the offline process, where an environment map is inverse rendered. The angle of the incoming light is assumed to be known, the reason for this will become apparent later.

The outgoing radiance from point $\vec{x}$, $L_o(\vec{x})$, is sampled from an input image of the scene. This input image is provided by a camera, which means that the outgoing radiance will not be directly compatible with the colour measured by the camera, as this returns a unit-less value. If the camera is assumed to be linear in it's response, the conversion from the camera-value to the radiance in the scene may be measured by a constant, which will be denoted as k. The radiance of the outgoing light is multiplied by k, which is written as the quantity $P(\vec{x})$ which denotes the pixel value:

$$P(\vec{x}) = k \cdot L_o(\vec{x}) \tag{6.4}$$

Applying equation 6.4 to equation 6.3 gives:

$$P(\vec{x}) = k \cdot \frac{\rho_d(\vec{x})}{\pi} \sum_{n=1}^{N} L_{i,n}(\vec{x}) \cdot \cos\theta_i \tag{6.5}$$

Furthermore $\cos\theta_i$ is split into two vectors, as it equals the dot product between a points normal $\vec{n}$ and the direction of the incoming light $\vec{l}$:

$$\cos\theta_i = (\vec{n}(\vec{x}) \bullet \vec{l}(\vec{x})) \tag{6.6}$$

Substituting $\cos\theta_i$ yields:

$$P(\vec{x}) = k \cdot \frac{\rho_d(\vec{x})}{\pi} \sum_{n=1}^{N} L_{i,n}(\vec{x}) \cdot (\vec{n}(\vec{x}) \bullet \vec{l}(\vec{x})) \tag{6.7}$$

With the constant k, the outgoing radiance is a known factor in the equation, as it now equals the pixel value $P(\vec{x})$. In equation 6.7 the light sources are positional light sources, and the direction from each point in the scene to each light source varies. If the light sources are assumed to be directional from an infinite distance, the light direction $\vec{l}(\vec{x})$ and incoming radiance $L_{i,n}(\vec{x})$ becomes independent of the position $\vec{x}$ and the equation can be modified to:

$$P(\vec{x}) = k \cdot \frac{\rho_d(\vec{x})}{\pi} \sum_{n=1}^{N} L_{i,n} \cdot (\vec{n}(\vec{x}) \bullet \vec{l}) \tag{6.8}$$

Assuming the incoming light direction is known, this leaves the incoming radiance from each light source $L_{i,n}$ as an unknown factor. This may be solved using a linear equation system, as seen in equation 6.9

$$\begin{bmatrix} P(\vec{x_1}) \\ P(\vec{x_2}) \\ \vdots \\ P(\vec{x_M}) \end{bmatrix} = \begin{bmatrix} \frac{\rho_d(\vec{x_1})}{\pi} \cdot (\vec{n}(\vec{x_1}) \bullet \vec{l_1}) & \frac{\rho_d(\vec{x_1})}{\pi} \cdot (\vec{n}(\vec{x_1}) \bullet \vec{l_2}) & \dots & \frac{\rho_d(\vec{x_1})}{\pi} \cdot (\vec{n}(\vec{x_1}) \bullet \vec{l_N}) \\ \frac{\rho_d(\vec{x_2})}{\pi} \cdot (\vec{n}(\vec{x_2}) \bullet \vec{l_1}) & \frac{\rho_d(\vec{x_2})}{\pi} \cdot (\vec{n}(\vec{x_2}) \bullet \vec{l_2}) & \dots & \frac{\rho_d(\vec{x_2})}{\pi} \cdot (\vec{n}(\vec{x_2}) \bullet \vec{l_N}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\rho_d(\vec{x_M})}{\pi} \cdot (\vec{n}(\vec{x_M}) \bullet \vec{l_1}) & \frac{\rho_d(\vec{x_M})}{\pi} \cdot (\vec{n}(\vec{x_M}) \bullet \vec{l_2}) & \dots & \frac{\rho_d(\vec{x_M})}{\pi} \cdot (\vec{n}(\vec{x_M}) \bullet \vec{l_N}) \end{bmatrix} \begin{bmatrix} k \cdot L_{i,1} \\ k \cdot L_{i,2} \\ \vdots \\ k \cdot L_{i,N} \end{bmatrix} \tag{6.9}$$

In equation 6.9 the number of light sources is denoted by N, while the number of samples is denoted by M. In order to derive the radiance value from every light source, the number of samples must exceed the number of light sources that needs to be estimated.

6.3 Specific Light Model

The outdoor scene may be approximated using a simpler light model, in which case there are assumptions that are made about the characteristics for such a scene:

One light source: The sun is assumed to be the only source of light within the scene. The large distance to the sun from the earth, means that the light can be considered parallel, and the sun light source is therefore assumed to be directional.

Ambient term: The lighting of the areas that are in shadow from the sun is modelled through by an ambient term.

Equation 6.7 is rewritten to fit these assumptions, and gives:

$$P(\vec{x}) = k \cdot \frac{\rho_d(\vec{x})}{\pi} \cdot \Big(L_a + L_d \cdot (\vec{n}(\vec{x}) \bullet \vec{l}) \Big) \tag{6.10}$$

Where L_a is the ambient radiance in the scene, and L_d is the direct radiance from the sun.

If the following information is known:

1. Image of the scene: $P(\vec{x})$
2. The light source direction: $\vec{l}$
3. The surface albedo: $\frac{\rho_d(\vec{x})}{\pi}$

Then it is possible to estimate the source radiant powers, L_a and L_d, with a scaling factor.

From equation 6.10 it is possible to make the following statement:

The image of the scene, and the surface albedo are available from the offline process as described in chapter **??**. The light source direction is dependent on the earths rotation around the sun. There are several daylight models, that handles this computation, and will be used to determine the light direction vector $\vec{l}$, and will be described in further detail later.

The ambient light, which is denoted as L_a in the light model, is normally a constant light contribution to the entire scene. The outcome of this simplification is that all points not lit by direct light are equally weighted, which is not consistent with reality. The light of every point in a scene is the result of an integration over the surface‘s entire hemisphere, meaning that points on a vertical wall has less visibility to the sky, than points on the horizontal ground, and thus has less light affecting them.

An example of this effect is seen in Figure 6.3, where two white diffuse surfaces are placed in a scene, lit from what in a local illumination model would be the equivalent the ambient term. The two surfaces are oriented horizontally and vertically, respectively. Measurements in the image shows that there is significantly more light affecting the horizontal surface, than the vertical. The horizontal surface has a mean pixel value of 188.5, where the vertical surface has a mean pixel value of 161.9. The reason for this difference is that the horizontal surface has visibility to the entire sky hemisphere, whereas the vertical only has visibility to half the sky hemisphere. A normal local illumination model does not take this into account, but it is a factor that needs attention if light is to be derived reliably from the images photographed of a scene.

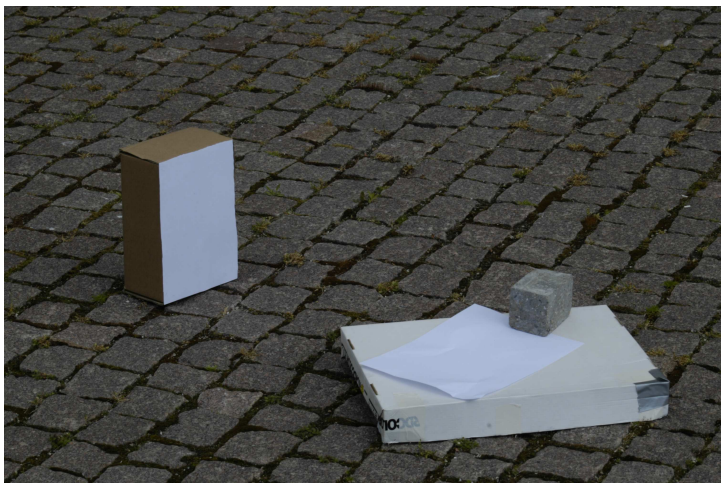


Figure 6.3: A vertical surface receives less indirect light from the sky hemisphere than a horizontal surface.

A way to handle this aspect of the lighting shadowed area in a scene, is to pre-calculate the hemisphere of various points in the environment. In the offline process the hemispheric light contribution to all points in the environment map can be calculated, which may be utilised in the online light estimation.

If the visibility hemisphere for every point on the environment map is pre-calculated in such a way that surfaces with full visibility has a value of 1 and points with no light has the value 0, the visibility hemisphere aspect can be included by using this value as a weight to the ambient light [Whitehurst, 2001].

If the ambient light weight is denoted as c, and the model in equation 6.10 is modified to use this ambient weight, the final rewritten equation:

$$P(\vec{x}) = k \cdot \frac{\rho_d(\vec{x})}{\pi} \cdot \left(c \cdot L_a + L_d \cdot (\vec{n}(\vec{x}) \bullet \vec{l}) \right) \tag{6.11}$$

Equation 6.11 is based on the following assumptions:

One Directional light: The sun is assumed to be the only source of light within the scene. The large distance to the sun from the earth, means that the light can be considered parallel, and the sun light source is therefore assumed to be directional.

Light Direction: As the direction from a point on Earth to the sun can be computed using a model, it is assumed that the direction to the one directional light source can be acquired using such a sun direction model.

Infinite Light Distance: The distance from the sun to the Earth is of a quantity where the inverse square law is insignificant with regards to the local distances used in the system.

Weighted Ambient term: The lighting of the areas that are in shadow from the sun is handled by an ambient term, that is weighted in accordance with sky-dome visibility.

Radiance measurable by a scalar: The Radiance is measurable by a scaling factor which denotes the scaling the camera applies to the radiance it measures from a scene. The camera is assumed to be linear.

The following section explores how the method can be applied to the image provided from a camera.

6.4 Sample point selection

When the method receives an input image, some pixels are more useful than others. There are many factors that determines whether a pixel is be useful or rejected as usable by the method:

1. Reflection parameters of the surface a pixel represents.
2. If the pixel is in an area that is undefinable as being shadowed or directly lit by the sun.

3. The appearance of dynamic objects within the scene.

4. If the pixel is either saturated or under-exposed.

These rejection parameters are explained in details in the following:

Reflection parameters: As mentioned earlier, a map is created in the offline process in which the user has specified which areas are considered lambertian reflectors, henceforth called the diffuse reflection map. Areas such as metals, windows on buildings and dynamic objects that may have appeared on the original image used to create the albedo map, like cars and people moving are all disregarded when the diffuse reflection map is created by the user. Furthermore it may be necessary to mask out the sky, as it is never the same, and can not be considered a diffuse reflector. All points in the environment map that are to be ignored are drawn into the "diffuse reflection environment map" the map is denoted as $\psi(\vec{x})$.
$\psi(\vec{x})$ is a binary map, where 1 means that the pixels in the area is considered a lambertian reflector, and 0 means that it is not considered a lambertian reflector.

Undefinable belonging: An aspect to be taken into account is that the pixels in the penumbra area of the shadows can not be considered a precise source of information, when the degree of shadow can not be measured. Therefore pixels which have differing shadow values in the shadow map, in a neighbourhood, with the size based on the imprecision of an optional tracking in the system, are ignored, to avoid misleading sampled data. This data may be saved in a penumbra map, is be generated for each input image, as the penumbra area will change during a day.
This map is denoted as $\kappa(\vec{x})$, and is binary, where pixels with the value 1 is pixels that is definitely either shadow or directly illuminated, while 0 valued pixels are those whose belonging can not be definitely determined.

Dynamic objects: Another aspect that will lead to rejection of certain samples is if the colour composition of the input pixel is significantly different from what has been recorded in the albedo environment map, a change that can not be explained by a lighting model. This should discard some of the changes caused by dynamic objects in the scene. This map is called the difference map, and is generated for each frame.
This map is denoted as $\Delta(\vec{x})$, and is binary, where 1 means that the pixel does not vary much from what is expected taking the last known light parameters into account, 0 means that the difference between the input and what can be explained by a lighting model exceeds a certain value.

Saturation: Saturated pixels in the input image is an artifact related to the fact that only LDR input is available, and saturated or under-exposed input is yet another rejection parameter. Saturated pixels are unusable for light derivation, as they do not give useful information of the intensity of the light source, other than it being powerful.
This map is called the saturation map, and is also generated for every input image to the method. The saturation map is denoted as $\delta(\vec{x})$, and is also binary, where 1 means that the pixel is neither saturated nor under-exposed, which is instead indicated with a 0.

These various aspects are combined into one map, denoting which pixels in the input map are to be ignored for further processing, this map may be called "the ignore map". The ignore map is created by logically AND'ing the diffuse reflection map, the penumbra map, the difference map and the saturation map.

The ignore map, $i(\vec{x})$, is obtained by:

$$i(\vec{x}) = \psi(\vec{x}) \wedge \kappa(\vec{x}) \wedge \Delta(\vec{x}) \wedge \delta(\vec{x}) \tag{6.12}$$

When the light estimation algorithm operates on images provided from a camera, information on which areas are in shadow is very useful. These areas are where ambient light can be derived, as the direct light does not have any direct influence on the light measured at this point. The shadows of an outdoor scene change during the course of a day, and can therefore not be pre-computed, which is why the shadows are present in the input image from the camera needs to be computed real-time per frame using geometry and sun direction information. If such a map is generated it can be fitted to the current view of the scene. The shadow map that is generated per frame is denoted as $s(\vec{x})$. The shadow map can be used as a binary map, that neutralises the part of the lighting model that adds light from the dominant light source, L_d.

In equation 6.11 $\vec{n}(\vec{x})$ is, at this point, still an unknown variable. The information of each surfaces normal can be derived from the geometry representing the environment. This may be done in a number of ways:

Normal derivation from depth information: If the depth buffer from the camera point is obtained it is possible to determine the surface normal of each pixel by comparing the depth information with its neighbouring pixels, and approximate the direction of the normal.

Normal derivation from ray-casting: When a geometric representation of the environment is available, it is possible to determine the normal of each point in an image, by casting a ray through the pixels in the image, and hitting the pixel's corresponding surface on the geometry. When this intersection point on a face of the environment is known, it is possible to interpolate the normal direction of the point from the normal of each vertex the face consists of.

Normal derivation from principal lighting: If a local illumination model is used to light the geometry, the colour returned from each point is a product of the reflectance of the surface, the power of the light that affected it, and the angle at which the light affected the surface. If the surfaces reflection is set to be purely diffuse with a uniform reflection parameter of 1 on each colour channel, and the power of the light is equally set to 1, the only variable left that affects the intensity of each pixel is the light angle, which is directly a product of the angle between the surface normal and the incoming light direction, this means that the angle is measurable from the intensity of each pixel in a rendering. This knowledge may be used to, in a very simple way extract the exact surface normal of any point visible from a virtual camera. If the geometry is lighted from the principal directions, only one component of the surface normal will react to each

> principal direction. With this in mind the surface normal of each point may be derived using only 2 render passes. The first render is set up by having a red directional light lighting from the $-x$ direction, a green light from the $-y$ direction, and a blue light from the $-z$ direction, in this render only the positive components of each surface reacts to the light, the positive x components of each surface reacts to the red colour exactly by the power of the positive $x$component. The second render is set up with the same lights, only from the opposite directions, so that only the negative components of each surface reacts to the light. The surface normal of each point in the scene is now obtained by subtracting the second image from the first image, each colour component now corresponds exactly to the normal components of each points. (Source: [Decaudin, 1996])

The method of deriving normal directions of each surface from depth information, is a method that may produce imprecise results, especially if surfaces are very curved and the normal direction is steep in regards to the direction to the camera. The method is dependent on the precision of the z-buffer in order to make a reasonable normal extraction. Furthermore the returned normal will represent the normal of the flat face spanned from 3 vertices, as the geometric properties of the 3D scene is what is measured with the depth information.

The normal for a point is usually extracted from an interpolation of the 3 corner vertices a face consists of. An analysis of the depth information will not be able to measure this interpolated normal, but will instead return the normal of the face. The method is best suited to extract normal information from objects that consists of planar surfaces, while objects with curvature, such as cylinders and spheres are less suited using the depth normal extraction method.

By casting rays into the scene, the exact normal may be derived from the geometry, though this is a needlessly complex approach when compared to the method, where the the surface normals are derived from rendering the environment under different lighting. The approach of lighting the entire scene from the principal directions in the three complimentary colours is used in the project, and it is used to create a normal map in the initialisation process, that corresponds the other maps used in the method. the normal map is denoted $\vec{n}(\vec{x})$

This brings the total amount of maps used in the light estimation to 9:

The maps that may be computed in the offline part are:

- Albedo environment map, $\rho_d(\vec{x})$.
  - **type:** Floating point
    - **Describes:** The diffuse reflection parameter for each colour channel
- Diffuse reflection environment map, $\psi(\vec{x})$.
  - **type:** Binary
    - **True:** Is diffuse reflection
    - **False:** Is not diffuse reflection
- Weighted ambient environment map, $c(\vec{x})$.

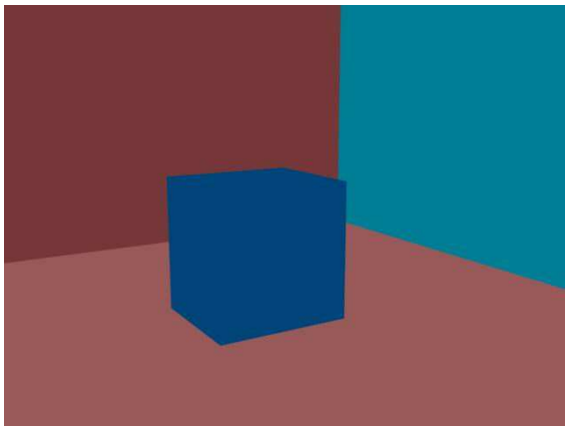
  - **type:** Floating point
    - **Range:** 0 - 1
      - **0:** Points where no part of sky hemisphere is visible
      - **1:** Points where entire hemisphere is visible

- Normal direction environment map, $\vec{n}(\vec{x})$.
  - **type:** Vector of 3 floats
    - **Components:** x, y, z
      - **Range:** $0 - 1$

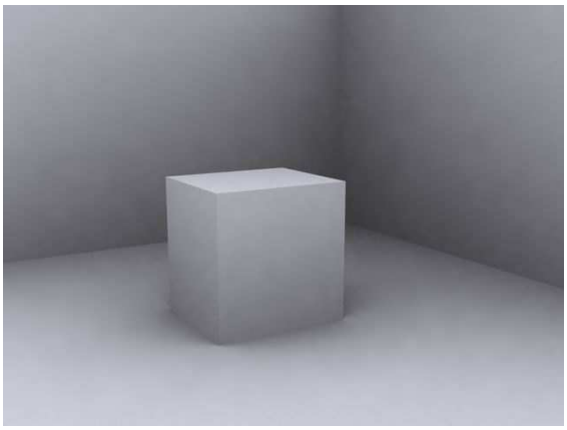
The maps that are produced real-time are:

- Live radiance input, $P(\vec{x})$.
  - **type:** 24 bit discreet image data, 8 bit per colour channel
- Shadow map, $s(\vec{x})$.
  - **type:** Binary
    - **True:** Is lighted by light source L_d
    - **False:** Is not lighted by light source L_d
- Penumbra map, $\kappa(\vec{x})$.
  - **type:** Binary
    - **True:** Can be determined to be either directly or ambient lit
    - **False:** Can not be determined to be either directly or ambient lit
- Difference map, $\Delta(\vec{x})$.
  - **type:** Binary
    - **True:** Colour composition does not match the albedo with variable scalar
    - **False:** Colour composition match the albedo with variable scalar
- Saturation map, $\delta(\vec{x})$.
  - **type:** Binary
    - **True:** Is neither saturated nor under-exposed
    - **False:** Is either saturated or under-exposed

In Figures 6.4, 6.5, 6.6, 6.7, 6.8 and 6.9 illustrations of the various map types used in the suggested light estimation method are shown, the diffuse reflection map is not illustrated, as all surfaces in the example are diffuse.

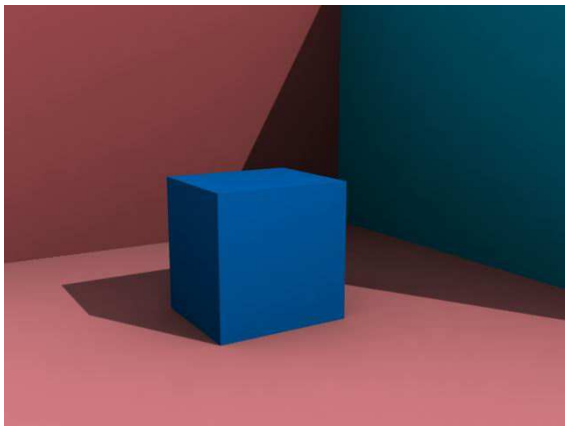


(a) Offline Albedo map, simulated

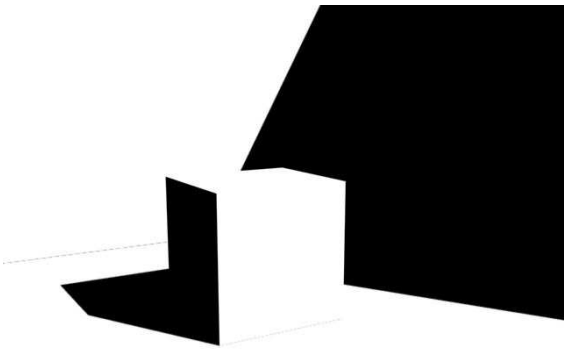


(b) Offline weighted ambient, simulated

Figure 6.4: A simulated scene, where the albedo (a) and weighted ambient (b) are generated in an offline process.

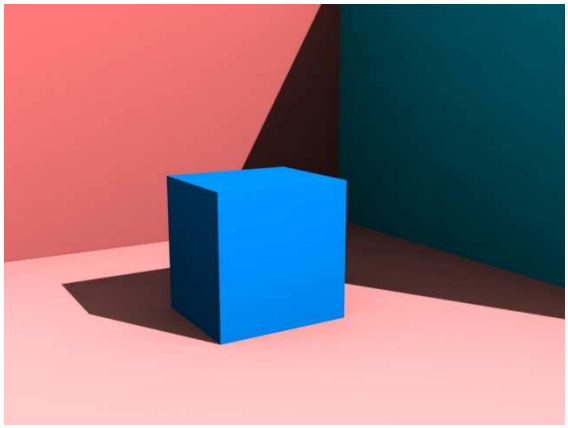


(a) Online live input, simulated

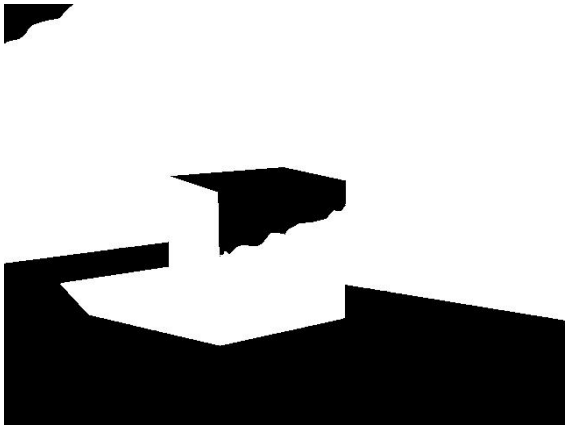


(b) Online shadow map, simulated

Figure 6.5: During execution of the system an online input (a) image is received for each frame, and a shadow map (b) is generated real-time.

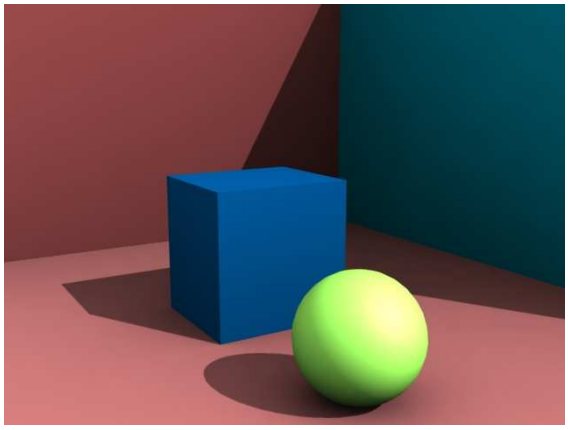


(a) Online live input, where elements are over exposed, and components of the reflected colour is saturated, simulated



(b) Online saturation map, simulated

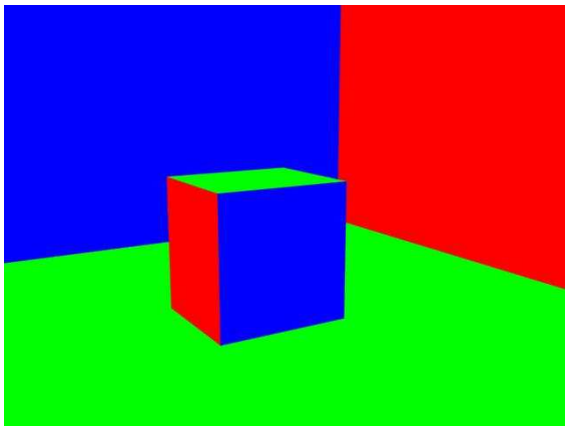
Figure 6.6: During execution of the system an online input (a) image is received for each frame, if this is contains saturated components, these will be rejected through the use of a saturation map (b) which is generated real-time.



(a) Online live input with dynamic moving object appearing in the scene, simulated

(b) Online difference map, simulated

Figure 6.7: During execution of the system an online input (a) image is received for each frame, and if pixels appear in the image, which does not hold the colour composition, as can be expected from the albedo map these pixels are rejected through the use of a difference map (b) which is generated real-time. The shadow of the dynamic object is not being rejected, as its colour composition equivalents the composition of its corresponding albedo.



(a) Surface normal map displaying the positive components of the normals, simulated

(b) Surface normal map displaying the negative components of the normals, simulated

Figure 6.8: The normal maps are used to extract the surfaces reflection angles, where red equals the x component of the normal, green the y component, and blue the z component.

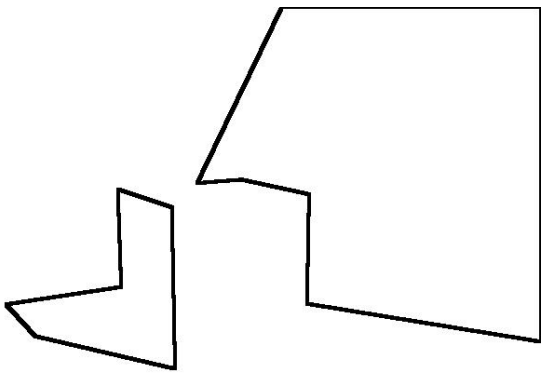


Figure 6.9: The penumbra map is obtained by finding the difference between a dilated and eroded shadow map.

The addition of the binary shadow map $s(\vec{x})$ makes up the final lighting equation, which is a modification of 6.11:

$$P(\vec{x}) = k \cdot \frac{\rho_d(\vec{x})}{\pi} \cdot \Big(c(\vec{x}) \cdot L_a + s(\vec{x}) \cdot L_d \cdot (\vec{n}(\vec{x}) \bullet \vec{l}) \Big) \tag{6.13}$$

The suggested method that will be used in this project, is to verify which pixels in the input image represents pixels in the albedo environment map, and are usable according to the "ignore map", as sample pixels. The resulting pixel colour is then set as $P(\vec{x})$ in equation 6.13.

To derive the light parameters of the scene several samples are taken in the input image. These samples are applied to the linear equation system in 6.14:

$$\begin{bmatrix} P(\vec{x_1}) \\ P(\vec{x_2}) \\ \vdots \\ P(\vec{x_M}) \end{bmatrix} = \begin{bmatrix} \frac{\rho_d(\vec{x_1})}{\pi} \cdot c(\vec{x_1}) & \frac{\rho_d(\vec{x_1})}{\pi} \cdot s(\vec{x_1}) \cdot (\vec{n}(\vec{x_1}) \bullet \vec{l}) \\ \frac{\rho_d(\vec{x_2})}{\pi} \cdot c(\vec{x_2}) & \frac{\rho_d(\vec{x_2})}{\pi} \cdot s(\vec{x_2}) \cdot (\vec{n}(\vec{x_2}) \bullet \vec{l}) \\ \vdots & \vdots \\ \frac{\rho_d(\vec{x_M})}{\pi} \cdot c(\vec{x_M}) & \frac{\rho_d(\vec{x_M})}{\pi} \cdot s(\vec{x_M}) \cdot (\vec{n}(\vec{x_M}) \bullet \vec{l}) \end{bmatrix} \begin{bmatrix} k \cdot L_a \\ k \cdot L_d \end{bmatrix} \tag{6.14}$$

As seen in equation 6.14 two samples would suffice in order to derive an estimate of L_a and L_d to a scaling factor. Samples are only added to the equation system if it passes the ignore test from equation 6.12. The vector $\vec{l}$ is a directional vector to the light source. In equation 6.14 vector $\vec{l}$ is the same for all points, because the sun can be assumed to be the same directional light source for all points within a close environment.

The following section describes how the vector of the direction to the sun can be determined from geographical and chronological information. With this information $\vec{l}$ in the above equation can be determined.

6.5 Determining the sun's Position

Sources: [Schlyter, 2005], [Giesen, 2005]
To be able to recreate the actual lighting conditions of a real outdoor scene in the virtual world, a sun model is needed to calculate the sun position.

Today it is common knowledge that the earth orbits the sun, and not vice versa. In the following calculations, however for simplicity the orbital elements in this section are valid for the sun's apparent orbit around the earth.

The find sun's position relative to the observer we define the term "observer". The observer is located at the centre of the *celestial sphere* with zenith above the head and the horizon North-East-South-West. The position of the sun, moon, or any other celestial body can be represented on the celestial sphere. An illustration of the terms is shown in Figure 6.10.

In astronomy the position is expressed in terms of altitude or elevation, h, and azimuth, α,

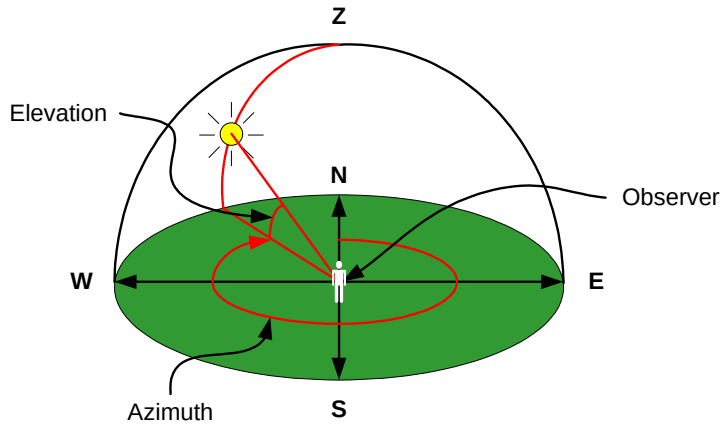


Figure 6.10: The sun position is measured relative to the celestian sphere at the observers location.

these correspond to the more widely used terms in mathematics, ϕ and θ. It is given that:

$$\phi = \pi - h \quad \epsilon \quad [0;\pi] \tag{6.15}$$

$$\theta = \alpha \quad \epsilon \quad [0;2\pi] \tag{6.16}$$

Azimuth angle is measured relative to North. Since sun position is measured relative to the observers position, the longitude and latitude of the observer is part of the equation. The other factor the current time, describing the both earth's orientation on the orbital plane around the sun and the self rotation. In Figure 6.11 the concept of longitude and latitude for determining position on earth is illustrated. earth The specific formulae for computing the azimuth and

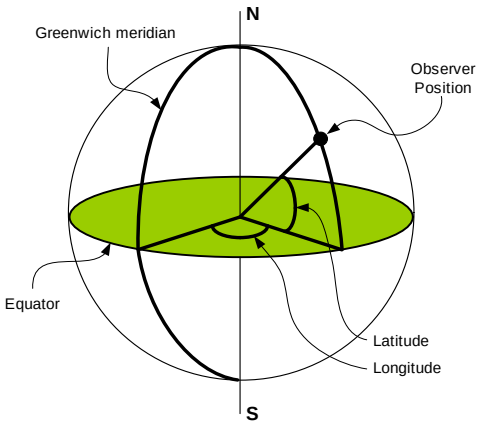


Figure 6.11: The observers position on earth is measured in latitude and longitude.

altitude angles of the sun are shown in appendix D. With the use of the computed azimuth and altitude angles, the Sun's orbit can easily be simulated in a computer program, as illustrated in Figure 6.12.

The following section discusses how the light estimation method will be implemented in the real-time augmented reality system Augmentor.

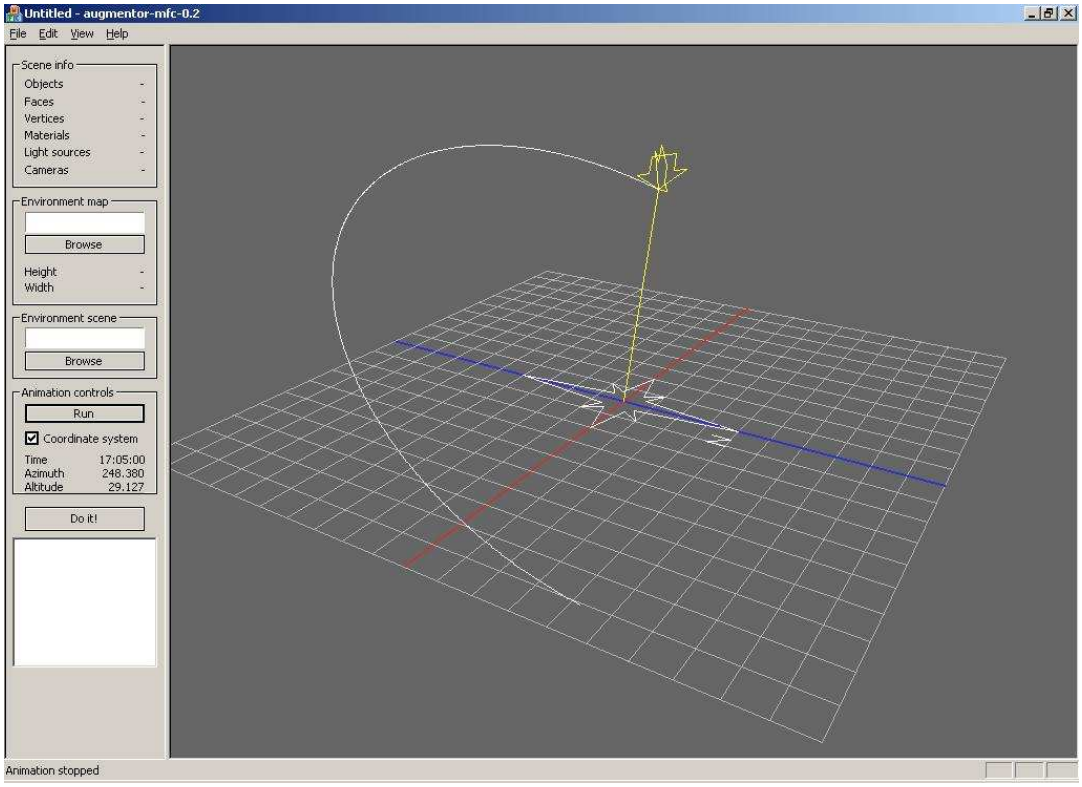


Figure 6.12: Augmentor displaying a simulation of the Sun path across the celestial sphere during a part of May 3rd 2005.

6.6 Implementation

During implementation there are certain practical issues that needs to be taken into account. Is it possible to perform a light derivation for each frame? Is it even necessary to execute the derivation for every frame, once per second, or even less? This section considers the various factors of implementing the light estimation method, described in this chapter.

6.6.1 Implementation Considerations

The first thing that must be decided is how often the light estimation algorithm is executed when the augmented reality system is running. Ideally the algorithm should be able to execute for every frame and process every pixel in the input image, but the confines of limited processing power entails that an analysis of such a data stream would be too demanding a task to perform real-time. In order for the method to perform well, and still make room for the rest of the system to perform various tasks, there are three suggestions on the implementation of the light estimation method:

- Process method for every frame, for as many pixels possible within the time frame.
- Process method on entire image every time a certain interval of frames are exceeded.

- Perform test on random points in the input image to the method to determine if they match the last known light parameters, if not, perform full light estimation on entire image.

If the method is executed once per frame, this automatically sets a time frame for how much time the method is allowed to use calculating the light parameters. The light estimation method must not use all the calculation time between frames, as it may be assumed that there are other tasks between frames requesting processing time, such as lighting the objects based on the results obtained from the light estimation method.

One way to solve the time problem would be for the light estimation method to be executed when all other tasks within a frame has been performed. This ensures the method may effectively use the remaining time, if it is structured to perform the computations progressively, taking one new sample of the input image per iteration, so the method may be exited when either a stop criteria or a time limit is met.

This has the drawback, that whatever light parameters that are estimated during that frame are not displayed until the following frame, meaning the lighting of the objects will at all times be at least 1 frame behind the the real lighting of the scene. If this delay will be noticeable when the system is running is questionable, it may be visible during rapid light changes in the scene, from one frame to the next.

Furthermore it should be considered if the light derivation should only rely its light derivation on the last frame it has analysed. If the last image grabbed from the camera only shows an area in shadow, the method has no means of estimating the direct light from the sun. If the light derivation only relies on the last frame used for light derivation, the method has no robustness towards the scenario that the camera for a given time only photographs surfaces that may be used to derive either direct or ambient light. Therefore the method must be able to use the last known estimation of either of the light parameters, if no new estimation is obtainable.

It may be advantageous to separate the derivation of the direct light and the ambient light in regard to the problem of having the camera facing either purely shadowed or lit areas. This way the method automatically collects samples and uses them to either ambient or direct light derivation. If there is not found sufficient samples of direct light during the analysis of the image, the samples will be applied to the linear equation system, but only the ambient part of the estimated parameters is updated, while the last known direct parameter is maintained.

If the method is executed, for instance, every 5 second, the method should have enough time to process the entire input image, but this sets a natural limitation of how fast the method can detect light changes and derive the necessary parameters in order to re-light the virtual objects.

The implementation of the method should instead analyse every frame lightly, and perform parameter derivation only if the analysis shows that the last estimated parameters no longer coincides with the lighting of the environment. This way the method should be able to quickly detect when the lighting of the scene changes and perform the parameter extraction when that happens.

To ensure that the method does not interfere with the processing in the rest of the system, it may be advantageous to execute the entire light estimation in a thread, that runs autonomously beside the rest of the tasks in the system. This has the advantage that the system is allowed to

perform the other time-demanding tasks, and the disadvantage that the lighting of the virtual objects may be behind the lighting of the rest of the scene, depending on the speed at which the light estimation derives the light parameters.

If the input image is blurred because of camera movement it is nearly impossible to check if the lighting of the image corresponds to the last known light parameters, not to mention that a full light derivation is as difficult because of the blur. This off course depends on the degree of motion blur in the image. Furthermore, if the system is connected to tracking equipment, the tracking can not be assumed to be synchronised with the camera frame rate, which means that during movement the tracking could be imprecise, aside from the regular imprecision of a given tracker, making a light parameter derivation further difficult. This means the implementation should only use images where the camera is still, or only has light movement.

6.6.2 Suggested Implementation

The above considerations about the implementation of the light estimation method, has resulted in the structure of the code shown in pseudo code 1.

Pseudo code 1 The light estimation method, performed autonomously.

1. Acquire current input image, its FOV and its orientation.
2. Generate shadow map $s(\vec{x})$, difference map, $\Delta(\vec{x})$, saturation map, $\delta(\vec{x})$, penumbra map, $\kappa(\vec{x})$, and derive ignore map.
3. Perform test of random points in image, compare to last estimated L_a and L_d.
4. If analysis matches last estimated lighting within an allowed deviation: Exit.
5. Else:
6. Determine which parameter needs to be updated, and if test showed this type of samples are present in input image.
7. For a set number of pixels: if ignore test accepts pixel, put it in sample stack.
8. Derive light parameters from samples.
9. If current ambient light, L_a, did not match the input: Update ambient light parameter.
10. If current direct light, L_d, did not match the input: Update direct light parameter.

The test applied in the first part of the execution of the light estimation method is performed by sampling the pixel value of various points in the input image. This value is compared with what the method guesses should be at that point. The method makes its guess by using the current light parameters and calculating how that point should look if lighted according to that, equation 6.13 is used to make the guess. The mean deviation between the guessed value and the sampled value is computed for all the input samples that passes the ignore test.

Before the mean deviation is calculated each sample is classified to be either an ambient or a direct light sample from the computed shadow-map. With this knowledge a mean deviation can be calculated for ambient and direct light, independently from each other, which makes it possible to determine if only one of the parameters needs update. These separated mean deviations, together with a count of how many test samples are classified to either direct or

ambient, can be used to determine if the input image is displaying only one type of the two sample-classes, and therefore only should update one of the two parameters.

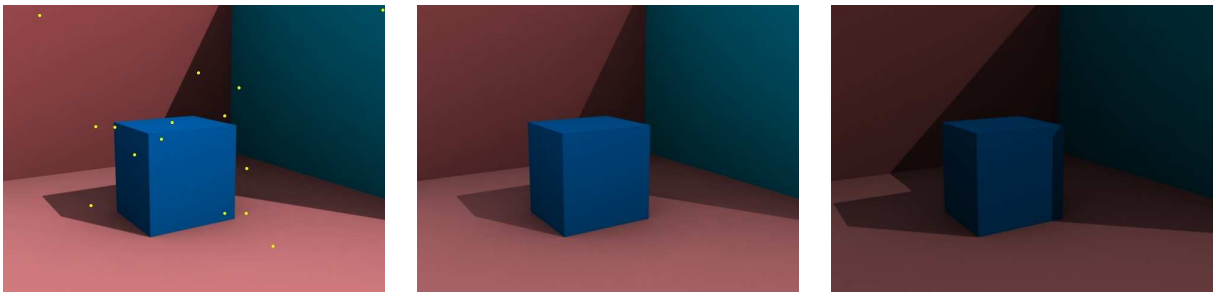
6.7 Test of Concept

In the following section the method is tested using simulated data, to verify if the suggested light estimation is sufficient to determine light parameters.

The simulated scene shown in Figures 6.4 to 6.9 is used to verify if the suggested light estimation method works on ideal data.

The test will include three images, that has the same motive, but are lighted differently. The light estimation method will be applied to the test images, and the resulting parameters will be checked if they match the changes applied to the light parameters between the images.

The images used in the test are seen in Figure 6.13.



(a) *(b)* *(c)*

Figure 6.13: The sample points are plotted into the simulated online input image (a), which matches the image shown in Figure 6.5a. Image (b) has half the power to the direct light, and image (c) then has half power to ambient and a changed light direction

The lighting of the scene consists of two parts:

- Direct light source
- Sky hemisphere

The test scene is rendered in Discreets 3D Studio MAX 7, using the light tracer configuration for advanced lighting.

Test Case 1

In Figure 6.13a the test image for the first case is seen with the various sample points, plotted with yellow dots.

In test case 1 the following configuration of the scene is applied

Direct light vector: $\begin{pmatrix} -0.37 \\ 0.657 \\ 0.657 \end{pmatrix}$

Sky light colour configuration: $\begin{pmatrix} 0.949 \\ 0.949 \\ 1 \end{pmatrix}$, Direct light source colour: $\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$,

These data are used in equation 6.14, where a computation yields the following estimation of the light parameters:

$k \cdot L_a = \begin{pmatrix} 2.983 \\ 2.974 \\ 3.116 \end{pmatrix}$ and $k \cdot L_d = \begin{pmatrix} 3.279 \\ 3.302 \\ 3.325 \end{pmatrix}$

If the parameters of $k \cdot L_a$ is measured against the source of the ambient light, the sky hemisphere, it is apparent that the estimated light matches the light from the sky hemisphere, by a scalar, as it can be seen that the ambient light is measured to be slightly more blue than the other colours, which was the same colour composition of the sky hemisphere.

Test Case 2

In Figure 6.13b the lighting parameters are changed and the test is performed again on the new image and the results are compared to determine if the method is detecting the change correctly.

The incoming lights direction and sky light colour is unchanged, while the direct light source is scaled to half power:

Sky light colour configuration: $\begin{pmatrix} 0.949 \\ 0.949 \\ 1 \end{pmatrix}$, Direct light source colour: $\begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$

A processing of the new data in equation 6.14 yields the following estimation of the light parameters:

$k \cdot L_a = \begin{pmatrix} 2.983 \\ 2.974 \\ 3.116 \end{pmatrix}$ and $k \cdot L_d = \begin{pmatrix} 1.64 \\ 1.59 \\ 1.612 \end{pmatrix}$

If the resulting light parameters for the analysis of image 2 is compared to the light parameters for image 1, it can be seen that the method measures the direct light changes in the radiant power correctly up to a scaling factor. When the light intensity of the source is scaled to half, a similar change is measurable in the derived light parameter.

Test Case 3

In the third image the light direction and the power of the ambient light is changed in the render.

In Figure 6.13c the third test image is seen. The sampling point remains the same.

The direct light source colour configuration remains at half power.

The incoming lights direction is changed to: $\begin{pmatrix} -0.592 \\ 0.5864 \\ 0.553 \end{pmatrix}$

The sky light colour configuration is scaled to half power, while the direct light remains at half power.

Sky light colour configuration: $\begin{pmatrix} 0.475 \\ 0.475 \\ 0.5 \end{pmatrix}$, Direct light source colour: $\begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$

A processing of the new data in equation 6.14 yields the following estimation of the light parameters:

$k \cdot L_a = \begin{pmatrix} 1.49 \\ 1.42 \\ 1.55 \end{pmatrix}$ and $k \cdot L_d = \begin{pmatrix} 1.653 \\ 1.76 \\ 1.665 \end{pmatrix}$

The parameters estimated in the third test confirms that the measurement of the ambient term matches the changes in the scene. Furthermore the test verifies that the direct light estimation is correct, as it roughly estimates the same scaled radiant power, as in test 2.

The data sampled at the various sample points can be seen in Appendix C.

Summary

The light estimation method presented in this chapter, is based the following assumptions:

- The lighting of outdoor scenes can be simplified to a local illumination model, with the sun as a single directional light source.
- The sky-dome visibility from all points in a scene can be precomputed into an ambient weight factor.
- The sun direction can be computed using a sunlight-model.
- The scene contains diffuse surfaces. Where masonry buildings are considered diffuse.

From the offline process, an albedo map, diffuse reflection map and geometry of the environment is provided.

With these information, the following information can be determined:

- The weighted ambient map, by intersection-testing all hemisphere directions at all mapping points in the scene.
- A shadow map calculated from sun direction and scene geometry.
- A normal map by lighting the scene from principal directions.

The light estimation methods mode of operation is, to sample a given set of points in an input image of the environment, and use the above information to establish the light parameters that lights the scene.

Each sample point is only used in the estimation if the following conditions are met:

- The sample point is present on the diffuse reflection map.
- The sample point is neither over-exposed nor under-exposed.
- The sample point is consistent with the albedo by a colour-uniform scalar.
- The sample point is at a given distance from a the edge of a shadow.

An aspect of the lighting of a given scene that has not been taken into account during the development of the system, are the powerful light reflections from highly reflective surfaces, like windows. When sunlight is reflected onto these surfaces they reflect onto the areas that is normally accepted as sample points. These sources of potential error can not be masked away by a map provided from an offline process, as the reflections in the scene are never static, but dependent on the suns motion on the sky-dome.

In order to take these reflections into account the method would need knowledge of which surfaces in the scene are highly reflective, and calculate the reflections they, from the sun direction, would cast on the rest of the environment. From this, the areas affected by the sun could be identified. The points in the scene at which these reflections are affecting the measured value are then written to another map that is taken into the ignore test. To generate the map with these reflection, the shadow volume technique could be considered, where volumes are created from all reflective surfaces in the reflection direction from the sun. Such a modification could be considered as a reflection volume.

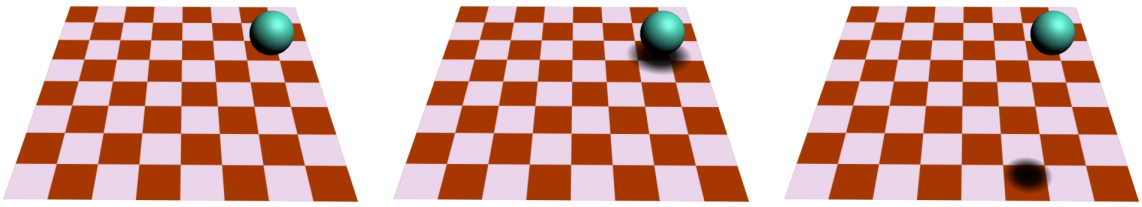
CHAPTER 6. ONLINE ESTIMATION OF SCENE LIGHTING

CHAPTER 7

Shadowing

In this chapter various shadow algorithms are presented. Section 7.1 briefly describes the most commonly used shadow algorithms. One of these are chosen for use in this project, and will be described in details in section 7.2

Shadows are an important element of realism in Augmented Reality. In 3D images shadows are essential to provide the viewer with visual cues about object placement. In Figure 7.1 the importance of shadows in providing a sense of object placement can be seen.



(a) *(b)* *(c)*

Figure 7.1: Three illustrations of shadow importance. The object has the same placement in all three figures. (a) Object with no shadows. Viewer has no sense of placement. (b) Object is distant from the viewer. (c) Object is closer to the viewer.

The terminology used in this section is illustrated in Figure 7.2 on the following page. Occluders are objects which cast shadows onto receivers. Umbra is a fully shadowed region and penumbra a partially shadowed region.

When dealing with real world shadows, many of the principles in shadow methods often used in real-time graphics will not appear realistic, because they are constructed for a limited number of ideal point light sources. Complex real world lighting scenarios are composed of many area light sources. Outdoor scenes are lit by both direct sunlight but also atmospherically scattered sunlight, which is a very large area light source. In indoor scenes walls, floors and windows all constitutes light sources. All of these factors result in both multiple shadows and shadows with soft edges.

Generating correct shadows calls for a detailed analysis of a given scene, determining which surfaces, both light sources and reflective surfaces, contribute to the irradiance at a given point in the scene.

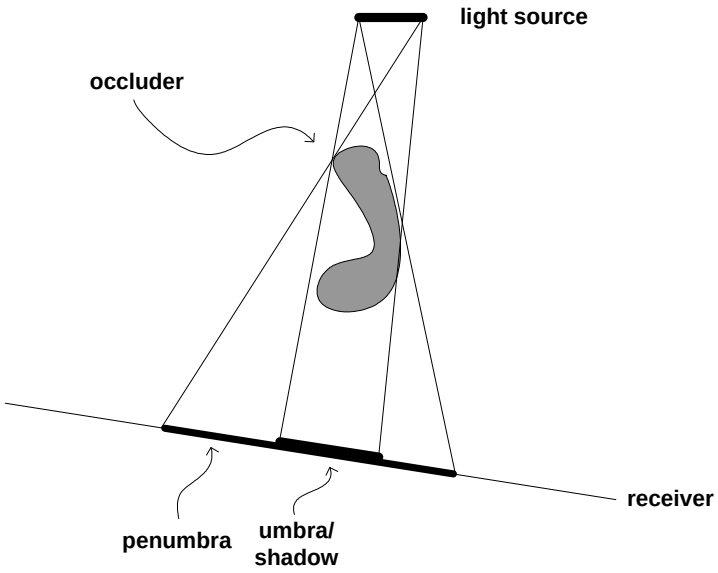


Figure 7.2: Shadow terminology: light source, occluder, receiver, shadow, umbra, and penumbra. (Image by [Akenine-Möller and Haines, 2002]

7.1 Methods

Shadows in a scene are areas that are in some way occluded from being illuminated by one or more light sources in the scene. The straight forward way to determine if an area is in shadow, would be to apply ray-tracing. This entails to shoot rays from every visible point in the scene to every light source in the scene, and test if the rays intersect any object. If the rays intersect objects, the point the ray was shot from is dampened by the factor the occluded light source added to its radiance value. This approach will produce correct shadows for point light sources, but it is a slow method, with limitations.

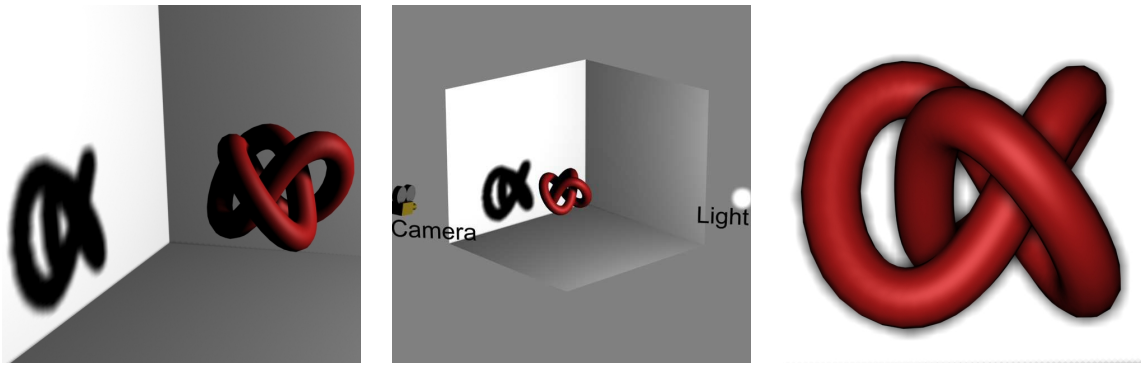
To achieve realistic shadows the concept of global illumination must be taken into account, where shadow areas are not only lit by the light sources visible from the shadowed surface, but also by the light reflections on other surfaces. This is particularly important in indoor scenes. In this project, an outdoor scene is being augmented, where the concept of local illumination is a acceptable approximation, as the sun is the dominant light source, and can be treated as a directional light source.

7.1.1 Shadow Mapping

Shadow Mapping is originally developed by Lance Williams [Williams, 1978]. Shadow Mapping creates realistic shadows cast by arbitrary geometric objects if sufficient shadow map resolution is given.

The basic idea in Shadow mapping is to put a camera in the position of the light source, and let it record depth information from that point of view. This depth information is then compared

to the depth information from the users viewpoint, and it is possible to ascertain which areas of the scene that is in shade, and which areas are not. This concept is illustrated in Figure 7.3.



(a) *(b)* *(c)*

Figure 7.3: The eye viewport (a), and light viewport (c). In shadow volume, all points on the surfaces, that are not visible from the lights point of view are in shadow.

The algorithm is simple. The scene is rendered from the point of view of the light source. The depth buffer of the render is stored, and projected onto the scene.Now the scene is rendered from the cameras point of view, and at each fragment, the projected depth texture is compared with the current depth buffer. If two compared depth values are the same the area is not in shadow.

The quality of the shadows created with the above approach, depends on the numerical precision of the depth values and the resolution the shadow maps are rendered with. The use of shadow maps forces the developer to make a trade-off between performance and shadow details, the higher details/shadow resolution, the lower performance.

Given infinite resolution and precision the shadow mapping method would produce perfect shadows, but practically this is not the case, and shadow mapping has artifact problems, which is seen as areas that are incorrectly shadowed or has jagged shadows.

7.1.2 Shadow Volume

Source: [Akenine-Möller and Haines, 2002]
Shadow volumes is one of the most commonly used real-time shadow casting algorithms today.

The idea in shadow volumes is to cast shadows onto arbitrary objects by use of a volume spanned from the objects silhouette. From a given point in space, which is the light source, a volume is created by finding the silhouette of the object that is to cast a shadow, and extruding a 3D volume from the light source through the silhouette points, towards infinity. The part of the volume that is on the other side of the light occluding object than the light, is a volume of the shadow, the object will cast. The volumes themselves are purely theoretical, and thus invisible to the camera in the scene.

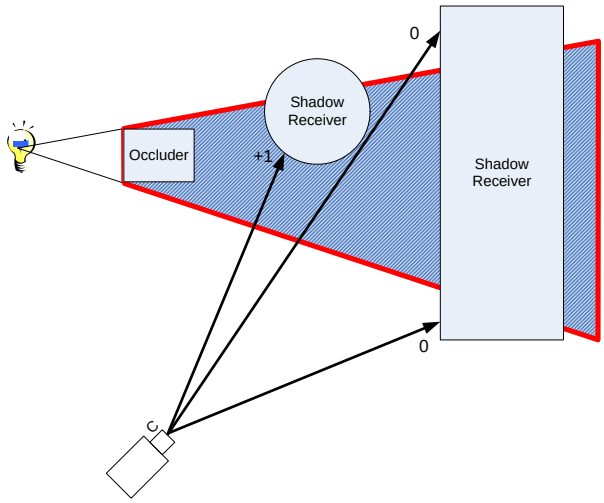


Figure 7.4: The Occluder object is creating a shadow volume, the areas in shadow are those surfaces inside the volume, which can be determined with the stencil buffer.

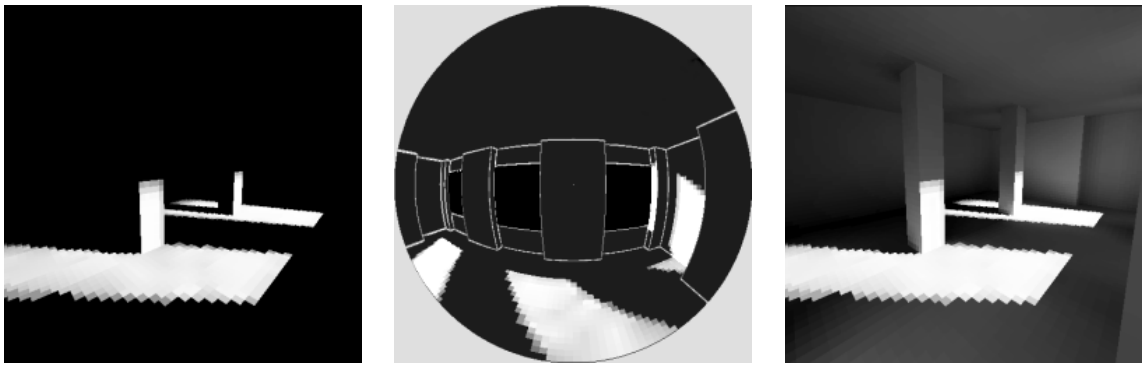
The implementation of the Shadow Volume method is to follow a ray through each pixel until the ray hits visible objects on the screen. While the ray is on its way to this object, a counter is incremented each time it crosses a front facing face of a shadow volume, meaning the counter is incremented each time the ray goes into a shadow. Each time the ray crosses a back facing face of the volume, the counter is in the same way decremented, to signify that the ray is outside a given shadow. When the ray hits the object to be displayed, the counter is checked. If the number is greater than zero, the pixel is in shadow, and otherwise it is not. An illustration of the Shadow Volume is seen in Figure 7.4

7.1.3 Radiosity

Source: [Elias, 2003], [Akenine-Möller and Haines, 2002], [Nettle, 1999a], and [Nettle, 1999b]. Radiosity is a global illumination technique, that is used to determine the light distribution in a scene. The classic radiosity method subdivides an entire scene into small patches, and computes the form factor between all surfaces. The form factor describes how much light is distributed between the various patches in the scene. The radiosity is calculated by setting all the patches up in a matrix, that has as many rows and columns as there are patches, making the radiosity an n^2 problem.

The basic principle in radiosity is to remove the distinction between objects and light sources. All surfaces in radiosity is considered light sources, and all patches are lit by the patches that are visible to them.

There are several methods that can be applied to radiosity in order to accelerate the radiosity calculations. One of these is to render the view from all patches, and let these renders dictate the illumination value of each patch. This is done in several passes, where the light is distributed around the scene for each pass. This is illustrated in Figure 7.5, where a scene of patches is lit by the light visible from each patches viewpoint.



(a) *(b)* *(c)*

Figure 7.5: (a) A scene lit after 1st pass. (b)The view from a unlit patch after 1st pass. (c) The scene lit after 2nd pass. The incident light on the patches after 1st pass lets otherwise occluded patches see light, and the rest of the scene is already lit in the 2nd pass. [Elias, 2003]

7.1.4 Discussion

The application of the system developed in this project is outdoor scenes. This means that the type of shadow casting needed is a sharp shadow, as the shadows cast by the sun.

The algorithm used in the system should also be able to perform well with a high frame rate.

Furthermore, the sun can be approximated into being a directional light source, when the scene is confined to a local area, so the algorithm must be able to make parallel projected shadows.

The system should support objects that may be moved, and at the same time a light source that moves, as the sun.

This sums up to the following requirements to the shadow algorithm:

1. Sharp shadows
2. Fast
3. Directional light support
4. Movement support

In table 7.1 the advantages and disadvantages for each method is seen.

Method	Advantages	Disadvantages
Radiosity	Realistic light for diffuse surface Can be accelerated in hardware	Slow Bad point source handling Bad handling of shiny surfaces
Shadow Volume	Modifiable to soft shadows Very sharp shadows Self-shadowing	Requires closed meshes Mesh-computations per-frame
Shadow Mapping	Sharp shadows Fast Modifiable to soft shadows	Prone to visual artefact's

Table 7.1: Problems and advantages for the various shadow algorithms.

In a system where sharp shadows are needed, and there is one dominant light source, Radiosity is not the obvious choice. If radiosity is to support sharp shadows and movable objects at the same time, it would require either a very fine patch-subdivided mesh of the entire scene, or a very smart adaptive real-time subdivision of the mesh in the contrast area between the umbra and fully lit area. In both cases it would require massive calculations each time an object is moved. An advantage is that if the sky including the sun is modelled as a hemisphere, the shadowing of the objects will be very realistic, but at the expense of performance.

Shadow Mapping and Shadow Volume are the obvious choices when hard shadows are to be simulated. Where the speed of the Shadow Volume is very much dependent upon the size of the mesh that casts a shadow, as it processes the entire mesh each time it creates a shadow volume. The shadow mapping however is not dependent directly on the polygon count of the mesh, only in the speed at which the graphics card is able to draw the scene objects.

Movement of the object does not require re-computation of the volume, because the light source is assumed directional. This is at the same time a factor that eases the computation of the shadow volume, as it is normally dependent on finding the vector for each vertex to the light source.

Shadow Mapping is prone to visual artefact's, which require extra processing to avoid. Another issue is that the shadow mapping requires a high resolution to avoid shadows with a jagged look. One way to avoid it is to have a camera following each object in the scene, but this creates problems if these objects moves close to each other.

Shadow Volume is a flexible solution, requiring some per frame computations, which can be accelerated. Shadow Volume meets the set requirements, and has been chosen for shadow casting in the system. The following section explores Shadow Volume in further details and describes how it has been implemented in the project.

7.2 Shadow Volume

Source: [Heidmann, 1991], [Carmack, 2000], [Surdulescu, 2003] and [Beam, 2004]
When using the Shadow Volume method, there are a couple of challenges to meet. Before the

shadow volume can be generated for an arbitrary object, there are certain processes needing completion.

A shadow volume is generated by first detecting the edges in a mesh, that bounds two faces with differing facing in regards to the light direction, i.e. the edges between front-facing and back-facing faces in the mesh. These edges are the silhouette edges of the object.

This detection is performed real-time each time the shadow volume is generated. The generation of the shadow volume is performed by extruding the edges towards infinity. If the edge spanned between vertex v_{23} and v_{42} is detected as a silhouette edge, and the the position of the light is l the extrusion of the edge to a volume, would be a quad with the following coordinates:

$$Q = \{v_{23}, v_{42}, (v_{42} - l) \cdot k, (v_{23} - l) \cdot k\} \tag{7.1}$$

An illustration of equation 7.1 is shown in Figure 7.6, and a volume extruded from the silhouette edges of a model is shown in Figure 7.7

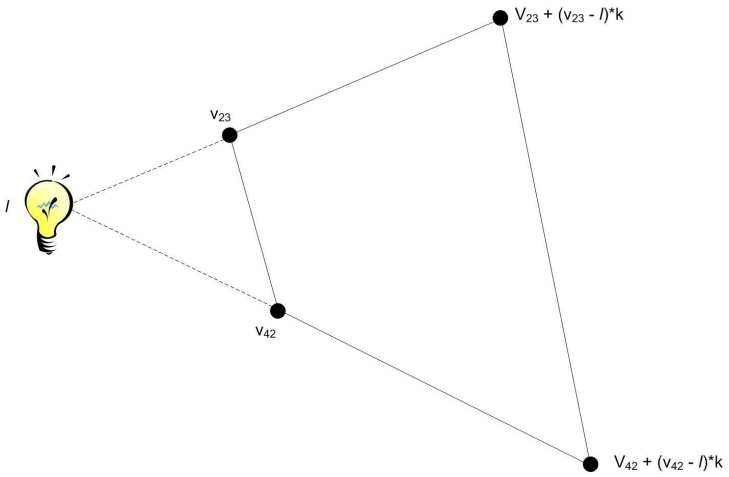


Figure 7.6: The Quad Q is generated by edge extrusion in the opposite direction of the light source l, and scaled by the constant k

In order to detect all silhouette edges real-time from an objects mesh, a tree of all vertices, faces and edges must be generated in order to perform the detection.

It is also important that there are no holes in the mesh. When a 3D editor writes a mesh to a file, there can be several vertices positioned in the same point, with differing normal direction. These vertices must be welded in order to speed up the silhouette detection, and since the normal of the faces are the only normal data usable in the shadow volume, this process does not discard important information.

When all faces and vertices have been setup in a tree, all edges between the vertices are detected. The edge contains information of which vertices it is spanned between, and which faces it is bounding, these information are important later, when the silhouette edges are detected.

In the online process, the edges that marks the silhouette of the object, from the light sources point of view, must be identified. When these edges are known, they are extruded to infinity to

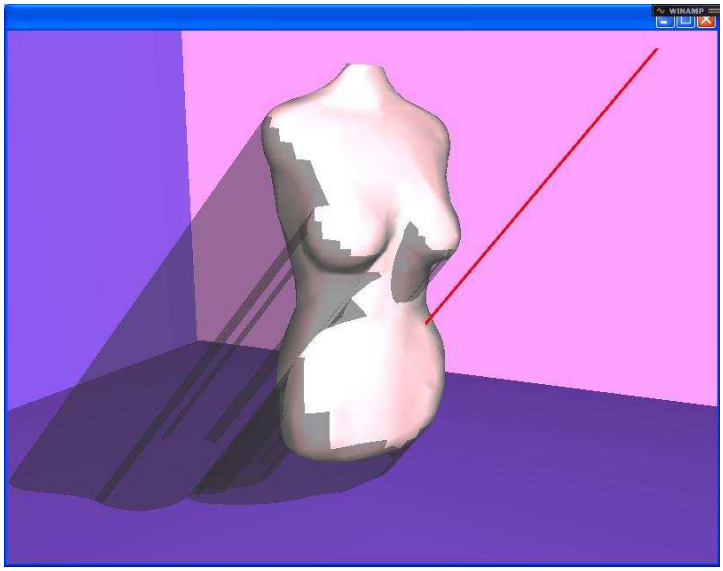


Figure 7.7: A shadow volume extruded from the silhouette edges of a model.

create the volume. All rasterised fragments appearing inside the volume is dampened to create shadows in those areas.

As the welding of vertices and edge identification is during the offline process, these are not tasks that need to be optimised, to the same degree the online tasks. The silhouette extraction is the task where the greatest performance gain can be achieved through optimisations, as the straight forward way to perform the task would be to loop through all faces in the mesh and determine if they are front facing or back facing. If this process could be optimised, the shadow volume would be able to perform faster, and support bigger meshes.

The common way to implement shadow volumes in order to accelerate the process, is to make use of the stencil buffer, present on any common 3D accelerator. Stencil test is an extra function closely coupled with the depth test. The stencil adds additional non-displayable bit planes to determine the stencil value of a pixel. The stencil value can be used in the same way as the depth value to accept or reject rasterized fragments. When stencilling is enabled, the stencil buffer is updated by comparing the stencil value of each pixel to a set reference stencil value. A pixel is only rendered if the stencil test passes.

The method to determine if a point is inside the shadow volume, is, As earlier mentioned, to shoot a ray from the viewpoint to the point, and count how many times a volume is entered, and the number of times a volume is exited. These enter and exit counts are performed until the first rasterized fragment is reached, that is, the first visual surface.

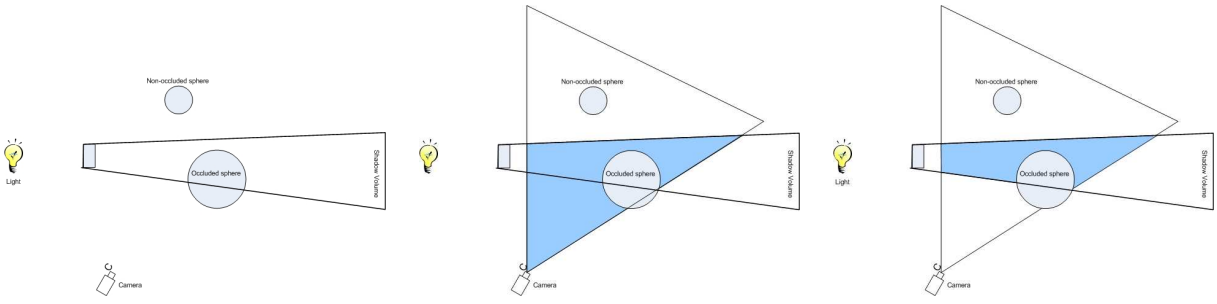
In 1991 [Heidmann, 1991] developed an algorithm, called the "Z-pass" algorithm, that made use of the stencil buffer to keep track of the enters and exits, in and out of a shadow volume. The traditional shadow volume method using the stencil capabilities requests that the scene is first rendered only to the depth buffer, this state of the depth buffer is then locked, so no further depth writing is possible. The next step is to set up the stencil comparison function to always pass, which allows the shadow volumes with respect to the various light sources to be rendered.

The rendering of the shadows is performed by first rendering the front faces of the shadow volume, where the stencil function set up in such a way that the stencil values in the buffer is incremented where both the stencil and the depth test passes, this is the counting of the *enters* into the volume. The second pass is to render the back faces of the volume, and decrement the value in the stencil buffer in those points where both the depth and stencil test passes, which equivalent the counting of the *exits* out of the volume.

The resulting stencil buffer contains zero in the spots where no volumes are present, or the view ray has entered, but also exited a shadow volume. The stencil buffer contains the value 1 or more in the points where the the view ray did not exit all volumes it entered.

The contents of the stencil buffer may now be used to dampen the points in shadow according to the power the light source that is now occluded added to the value of each point. If more than one light source is present in the scene, the procedure is performed for each light source, and the resulting shadow areas are dampened according to how much each light source occluded from the point added to it.

The algorithm is illustrated in Figure 7.8.



(a) *(b)* *(c)*

Figure 7.8: The shadow volume algorithm, a scene with 2 objects, one occluded by the volume (a). In the first pass (b), one back border of the shadow volume is detected, and in the second pass (c), the front border is detected.

The Z-pass algorithm has the structure seen in pseudo code 2:

The above procedure works well as long the camera in the scene is outside the area enclosed by the shadow volume. If the camera moves inside the volume, the pass that processed the faces closest to the eye fails, as they can not be seen from inside the volume.

This shortcoming is addressed by using a different version of the shadow volume algorithm called the "Z-fail" algorithm, which is also known as "Carmack's Reverse", as [Carmack, 2000] developed it. The "Z-fail" algorithm is similar to the Z-pass algorithm, except for the second and third step, as seen in pseudo code 3.

The difference between the two approaches is that in the Z-pass algorithm the lit/shadowed status of the various point of the scene is computed from the camera towards the back of the scene. In Z-fail the scene is computed going from the back of the scene towards the camera, which is indifferent to the situation where shadow volume faces are clipped by the

Pseudo code 2 The Z-pass algorithm.

1. Render scene with current light source turned off. This renders the colour buffer as if it is all in shadow from the current light source, as well as writing the right depth values to the depth buffer.
2. Disable colour buffer and depth buffer writing.
3. Render all shadow volume front faces, increment stencil buffer for all points passing depth test.
4. Render all shadow volume back faces, decrement stencil buffer for all points passing depth test.
5. Enable colour buffer and depth buffer writing.
6. Turn on light, and render scene. Colour writes are only allowed at stencil buffer points with the value 0.

Pseudo code 3 The Z-fail algorithm.

1. Render scene with current light source turned off. This renders the colour buffer as if it is all in shadow from the current light source, as well as writing the right depth values to the depth buffer.
2. Disable colour buffer and depth buffer writing.
3. Render all shadow volume back faces, increment stencil buffer for all points failing depth test.
4. Render all shadow volume front faces, decrement stencil buffer for all points failing depth test.
5. Enable colour buffer and depth buffer writing.
6. Turn on light, and render scene. Colour writes are only allowed at stencil buffer points with the value 0.

near frustum viewing plane. Furthermore the Z-fail algorithm requires that the shadow volume is closed, which means that additional polygons must be added in the top and the bottom of the volume from the outline defined by the shadow volume calculations. This means that the Z-fail algorithm requires more polygons drawn to the screen, and it therefore is a slower algorithm than the Z-pass. The normal approach is to detect when there is a possibility that the camera may enter a volume, and in that case switch between the two drawing algorithms, for speed purposes.

In this project the Z-fail algorithm has been chosen as drawing algorithm as it, though a bit slower, always draws the shadows correctly.

7.3 State of the Art

This subsection explores which methods may be used to optimise the silhouette extraction process that is executed for each frame the light has moved.

Fast Backface Culling Using Normal Masks

Source: [Zhang and Hoff(III), 1997]
The paper by [Zhang and Hoff(III), 1997], "Fast Backface Culling Using Normal Masks", introduced a new method to, in the render routine, quickly filter away the faces that are back facing in regards to the camera viewpoint. The operation is performed by adding one logical operation test per polygon processed. The method introduces a normal mask where each bit is associated with a cluster of normal directions. The methods mode of operation is to test the facing of the clusters that all faces has been added to, the clusters that fails the test and are considered as backfacing will be encoded into the normal mask, so the polygons that belongs to these clusters are filtered during the draw operation.

Efficient Perspective-Accurate Silhouette Computation

Source: [Barequet et al., 2001]
The [Barequet et al., 2001] method consists of setting up view parallel planes through the object. A preprocessing of the planes representing all the faces in the geometry ensures that all faces may be classified as silhouette quickly from a query generated from the current view dependent plane. The orientation of the different faces intersected is detected, and taking the view direction into account(View query plane) returns which edges are silhouettes.

Hardware-Determined Feature Edges

Source: [McGuire and Hughes, 2004]
[McGuire and Hughes, 2004] suggested that by converting global/per-edge computations to local/per-vertex computations a feature edge detection algorithm running entirely in hardware can be implemented. This is used to create screen space contours and end caps. An application of this is hardware-based silhouette extraction, which can be used to generate the volumes used in the shadow volume method. The drawback of the method is that the size of the vertex program used to implement the method makes it slow down the current generation GPU, with a resulting substantial loss of rendering speed.

7.4 Implementation

The methods described in the previous State of the art section may be used to accelerate the detection of the silhouette edges in the shadow-casting mesh.

7.4.1 Offline Processing

While the [McGuire and Hughes, 2004] method is the fastest, it requires a next-generation graphics accelerator in order to give a performance boost.

The [Barequet et al., 2001] method produces a good approximation to the silhouette of an object which may be extruded to produce the shadow volume.

The [Zhang and Hoff(III), 1997] method gives a measurable performance boost of up to 200%, and can, with minimum alterisation, be made to filter away all clusters of faces that are certain to be either front facing or back facing in regards to the light source from silhouette computations.

In the implementation of the shadow volume the [Zhang and Hoff(III), 1997] method will be used in a modified version to quickly determine during volume creation which edges of the volume that is to be extruded.

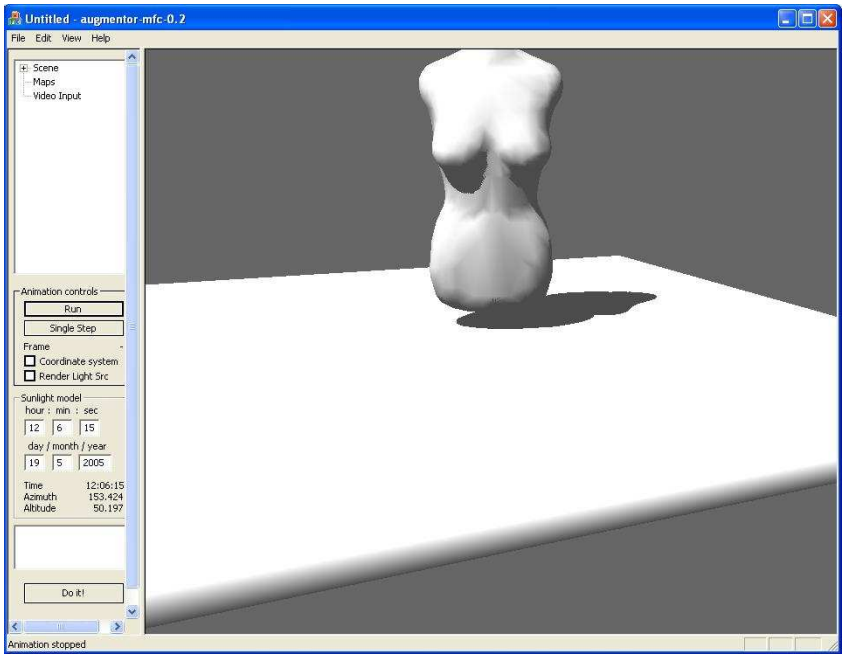


Figure 7.9: The Shadow Volume implemented in the augmentor framework, where the shadow casting is set to match the sun position delivered from a sun-model.

In the initialisation process of the shadow volume, the mesh that is to cast shadows is processed to determine all edges in the mesh, and which faces they are neighbouring.

This generates an edge-list, face-list and a vertex-list. The edge-list contains information of which two faces it is between, and which two vertices it is spanned by.

The preparation of accelerating the silhouette detection is done by segmenting the unit sphere into clusters of different directions, and test all faces to which cluster they belong. A cluster-list is generated that contains information of which faces has been detected as belonging to which cluster. This concludes the mesh-preparation for the shadow volume.

7.4.2 Online Processing

A cluster represents a set of directions. When the shadow volume must be generated during the drawing of each frame, the clusters are tested if their direction four corner directions, which

represents the extremes of the directions the cluster represents, are either back facing or front facing. If the corner directions of the cluster are all determined to be either back facing or front facing none of the faces in the cluster needs to be tested for having silhouette edges.

The clusters whose corner directions are tested to not all be either back or front facing, contains faces where some has edges that are silhouette edges. The edges of the faces contained in the clusters classified as silhouette clusters are then tested to be silhouette edges, and a silhouette edge list is generated.

All the edges contained in the silhouette list are extruded to infinity(in this case infinity is defined by the distance to the far clipping plane), and the volume is capped in both ends.

Another aspect that is taken into account in the generation of the shadow volume, is that the light that casts the shadow is assumed to be a directional light source. This fact can be used to optimise the silhouette detection, as the silhouette is extracted through parallel projection, instead of the traditional perspective projection that is usually used when casting shadows from a point light source. The use of directional light simplifies the facing test for each face, as the light vector remains the same for all faces in the mesh, and therefore does not need recomputing for every face.

When the shadow is cast from the virtual object on the virtual ground of the augmented scene, the shadow is cast on geometry that is not visible. The geometry represents the scene that the virtual object is displayed on top of. In order for the shadow to realistically cast a virtual shadow in the real environment, the geometry that represents the video input is rendered only to the depth buffer, while the real image is rendered to the colour buffer instead of the geometry. When the render pass where shadow is to be displayed the video image is dampened so it represents the last measured ambient value.

In Figure 7.9 the implementation of the shadow volume is seen, where the shadow casting is set to match the sun position delivered from a sun-model.

Summary

Shadows are in the context of this project used for the following purposes:

- Shadow casting from virtual objects put into a real scene on real geometry.
- Shadow casting from environment geometry, used to generate the shadow map for the light estimation and environment map re-lighting parts of the system.

To generate shadows, the shadow volume method has been applied, where the implementation uses the Z-fail algorithm to ensure the shadows are view-independent. A clustering technique usually applied to culling problems has been utilised in order to speed up the online shadow volume generation.

The implemented shadow volume is used to cast shadows from both the virtual objects that are augmented into the scene, and to cast shadows from the virtual environment geometry that represents the environment of the real scene. The shadows generated from the virtual

environment are used to determine which parts of the real scene are in shadow and what is in direct illumination from the sun. This information is used in the light estimation part of the system as well as in the environment map re-lighting part.

CHAPTER 8

Shading

This chapter discusses how the light parameters measured in the Light Estimation, may be used to shade objects. A method will be presented, and described.

The system developed through this project estimates the lighting of a real scene, through analysis of images of the scene. The result of this light estimation is an ambient and a direct light parameter. These parameters can be used to shade an arbitrary object using a given local illumination model. Using a local illumination approach to shade an object rules out the possibility to let the lighting of the object be shaded by light bounces on the surrounding geometry, which is an important aspect of lighting objects realistically.

Simulating the properties of Global Illumination is a non-trivial task, in non-real-time and especially in real-time. One way to simulate Global Illumination in real-time is to obtain an environment map of the surroundings from the position of the object, and calculate how the lighting from the environment influences the lighting of the object in different directions, by hemisphere integration on all pixels in the environment map. The result of this computation is an irradiance environment map, that can be used to shade an object real-time by a look-up in the irradiance environment map.

In this project, the approach is to re-light an environment albedo map, using the light parameters obtained from the light estimation method.

The resulting environment map is used in conjunction with an irradiance volume, that can approximate the light affecting an object at different places in an environment.

The following section describes Global Illumination, and how shading is performed using a global illumination model.

8.1 Global Illumination Shading

In global illumination methods the problems associated with local illumination are addressed. Classic ray-tracers simulates light reflecting only once off each diffuse surface, with global illumination the many reflections of light bounces around a scene are simulated, to create a more realistic virtual light.

Each object in a ray traced scene must be lit by a light source for it to be visible, in a globally illuminated scene an object may be lit simply by its surroundings.

When light hits a surface, some of the light is absorbed, and some is reflected. The surface properties of the object determines how much light is reflected. Surfaces that have been illuminated acts as light sources themselves. Classic ray-tracing can not be used to simulate

global illumination, because ray-tracing only handles point light sources. So to perform global illumination with classic ray-tracing each emitting surface must be subdivided into an infinite amount of point lights. This means that diffuse surfaces becomes an infinite number of point lights when illuminated, and illuminating other surfaces from those point lights becomes difficult.

8.1.1 Bidirectional Reflectance Distribution Function

Source: [Akenine-Möller and Haines, 2002]
A Bidirectional Reflectance Distribution Function (BRDF) is a function that describes how light is reflected from a surface, it is used to describe material properties. Input to the function is incoming and outgoing azimuth and elevation angles, as measured on the surface. The wavelength of the incoming light is another input, which is equivalent to the colour of the light.

The BRDF returns a unit-less value that signifies the amount of energy reflected in the outgoing direction, based on the incoming direction.

The BRDF is a relation between incoming and outgoing radiance, but does not explain how materials physically interacts with light.

An important property of the BRDF is the *Helmholtz reciprocity*, which states that the input and output angles can be switched, and the function value will remain the same.

The BRDF is an abstraction that describes how light interacts with a surface, but it is also an approximation of a more general equation, the Bidirectional Surface Scattering Reflectance Distribution Function (BSSRDF). The BRDF does not include the scattering of light within the surface, which can be seen in e. g. marble. BSSRDF includes these properties, but only handles reflections of light, not transmission of light through the surface. This can be handled with two BRDF's and two BTDFs (T for Transmittance), by defining one of each to the two sides of the material, which makes the BSDF (S for Scattering).

With a BRDF and an incoming radiance distribution, the reflectance equation gives the outgoing radiance for a given viewing direction, relative to the surface. This is done by integrating the incoming radiance from all directions on the hemisphere on the surface:

$$L(\theta_o, \phi_o) = \int\int_{\Omega} f(\theta_o, \phi_o, \theta_i, \phi_i) L(\theta_i, \phi_i) cos(\theta_i) d\sigma(\theta_i, \phi_i) \tag{8.1}$$

Subscript i and o are the incoming and outgoing (view) directions. The L function is the radiance travelling in a given direction, and f is the BRDF. The integrals indicates an integration over the local hemisphere Ω for the surface. The utilisation of the equation is, for all directions on the surface, to determine the incoming radiance, multiply it with the BRDF for the incoming and outgoing directions, scale it by the incoming angle to the surface, and integrate. The result of this operation is what radiance is seen for a given viewing direction.

The Lambertian BRDF is the ideal diffuse, which can be used to model the appearance of rough surfaces, with the characteristic that they reflect light equally in all directions, see Figure 8.1.

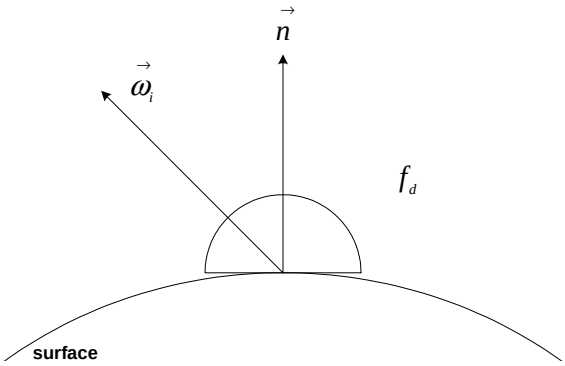


Figure 8.1: The Lambertian BRDF. Surfaces with this BRDF scatters light equally in all directions.

The Lambertian BRDF is given by:

$$f_d(\vec{\omega_i}, \vec{\omega_o}) = \frac{\rho_d}{\pi} \tag{8.2}$$

This equation is independent of both $\vec{\omega_i}$ and $\vec{\omega_o}$, the constant ρ_d is the reflectance, or albedo, and ranges from 0 to 1.

The counterpart to the diffuse BRDF is perfect specular BRDF. The ideal specular BRDF models perfectly smooth surfaces. This surface reflects all light in exactly the exact mirror direction, as in Figure 8.2.

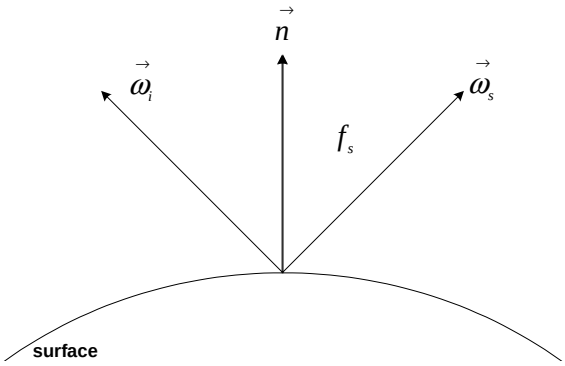


Figure 8.2: The perfect specular BRDF. Surfaces with this BRDF scatters light only in the mirror direction

The specular BRDF is given by:

$$f_s(\vec{\omega_i}, \vec{\omega_o}) = \begin{cases} \rho_s & \vec{\omega_r} = \vec{\omega_o} \\ 0 & \vec{\omega_r} \neq \vec{\omega_o} \end{cases} \tag{8.3}$$

where ρ_s is the specular reflection (in the range 0 to 1).

8.2 Shading algorithm

In this project a Global Illumination shading method will be used to shade a virtual object set in a real scene. To accomplish the task of merging the real and the virtual parts of the image, the real image must affect the virtual object, and the virtual object must affect the real scene.

The virtual object must be shaded according to its surroundings in the image. The virtual object and real objects must also affect each other with their shadows, for the illusion to be convincing.

The lighting of the surroundings are calculated, and the light parameters are stored in an environment map, through an environment map re-lighting process.

To shade objects from an environment map various methods may be applied. A popular approach is to convert the environment map, to an irradiance map,like [Debevec, 1998], and use the obtained irradiance map as the diffuse shading of an object. This approach does not take into account that if an object is moved around in the environment, the lighting of the object changes depending on its position. An example is that if an object is moved close to a coloured wall, the colour of the wall will bleed onto the lighting of the object. The [Debevec, 1998] method does not handle this aspect, as the basic assumption of the method is that the light present in the environment map, or radiance map as it is called in the method, is all equally far away.

An algorithm that supports movement in a scene, that can change the effects surroundings light

Irradiance Volume is usually used as a supplement to global illumination algorithms. The Irradiance Volume method approximates the lighting of an entire virtual scene in advance. The method then analyses the scene, and shades objects accordingly. This is a very fast method to shade objects based on their surroundings.

The lighting of the real scene is measured, by creating an environment map of the surroundings. The irradiance volume is modified to sample the radiance from such an environment map, and used to shade the virtual objects in the scene, so they match the lighting of the surroundings.

The following section discuss how the light parameters obtained from the light estimation part of the project are translated into an environment map for a given scene, afterwards the Irradiance Volume is examined, and the modification to use environment maps is discussed.

8.3 Environment map Re-lighting

When an environment map has to be re-lighted for a given scene, there are various ways to do it.

One way would be to map the albedo map to the geometry of the scene, and set up the rendering library to shade the scene using the direct and ambient parameters estimated from the images of the scene. This scene is then rendered to an environment map.

Another way is to assign an albedo value of 1 to all surfaces of the environment geometry, and shade the environment using the aforementioned light parameters. This is rendered to an environment map, and represents an irradiance environment map.

Both of these methods has the disadvantage that if the light parameters are of a certain power, the power they will affect the scene with may result in an environment map that is over-exposed. This can be rectified by rendering the environment maps in different powers of the light sources, and assembled afterwards to an environment map with a high dynamic range.

A method that will not be limited by the Low dynamic range rendering capabilities of a real-time graphics library, is to use the rendering equation of the shading model used to estimate the lighting of the scene. By rendering the environment using the same model as used in the estimation will arguably also lead to the best reconstruction of the lighting of the environment. The rendering using this model is performed per-pixel, and does not use the rendering capabilities of a graphics accelerator.

The shading model used to estimate the lighting of the scene is:

$$P(\vec{x}) = k \cdot \frac{\rho_d(\vec{x})}{\pi} \cdot \left(c(\vec{x}) \cdot L_a + s(\vec{x}) \cdot L_d \cdot (\vec{n}(\vec{x}) \bullet \vec{l}) \right) \tag{8.4}$$

$P(\vec{x})$ is the pixel value resulting of the model, the value equivalents the radiant power at various points in the scene scaled by a factor. The constant k is the scaling factor to all radiant energies, and is linked to the light parameters estimated from the light estimation method.

If the estimated direct and ambient light parameters are denoted, P_d and P_a, respectively, they are denoted by the following parameters in equation 8.4:

$$P_a = k \cdot L_a \tag{8.5}$$

$$P_d = k \cdot L_d \tag{8.6}$$

whereas the light model can be written as:

$$P(\vec{x}) = \frac{\rho_d(\vec{x})}{\pi} \cdot \left(c(\vec{x}) \cdot P_a + s(\vec{x}) \cdot P_d \cdot (\vec{n}(\vec{x}) \bullet \vec{l}) \right) \tag{8.7}$$

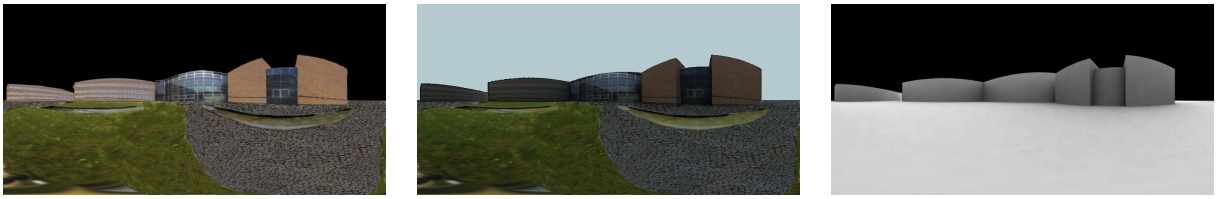
$\rho_d(\vec{x})$ is the albedo at point $\vec{x}$ in the scene, while $c(\vec{x})$ is the weighted ambient term at point $\vec{x}$. Both of these parameters are stored in environment maps, that are also used by the light estimation method.

$s(\vec{x})$ is the shadow map that can be calculated real-time by the shadow volume. The shadows of the environment cast by use of the shadow volume are rendered to an environment map, where points in shadow are black, and points in direct light are white.

$\vec{n}(\vec{x})$ is the surface normal at point $\vec{x}$ in the scene. The surface normal is obtained by rendering a normal environment map, using principal direction directional colouring.

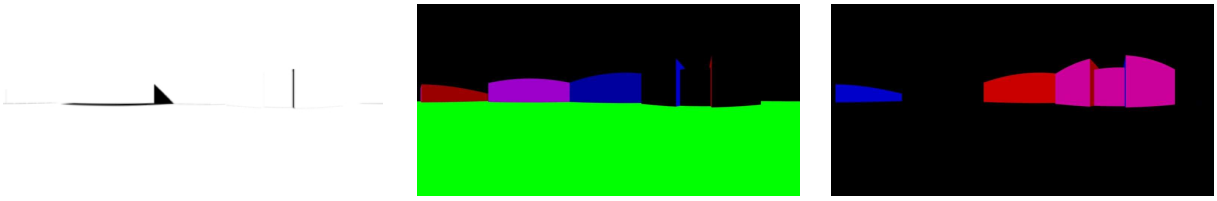
$\vec{l}$ is the light direction computed by the sunlight-model.

The various maps used in the re-lighting of environment maps can be seen in Figures 8.3 and 8.4



(a) *(b)* *(c)*

Figure 8.3: An albedo environment map (a), is used in the creation of a new environment map (b) representing the estimated light parameter values. (c) displays a weighted ambient environment map.



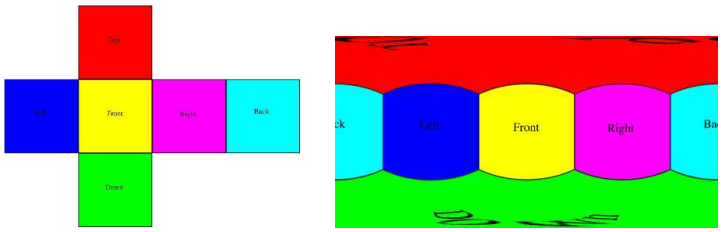
(a) *(b)* *(c)*

Figure 8.4: The shadow (a), positive directions (b) and negative directions (c) environment map.

A disadvantage of re-lighting an environment map with the shading model from equation 8.7 is that surfaces that originally where non-diffuse are re-lighted as if they are. But as non-diffuse surfaces are most likely view-dependent, a re-lighting using their reflectance function, would entail, that the environment map would have to be re-lighted every time the object, that is shaded by the environment map, is moved within the environment. As there is no reflectance function known for the non-diffuse surfaces in the scene, they are in the re-lighting phase assumed to be diffuse.

When re-lighting an environment map, the sky is a factor that can not be recreated if elements like clouds and the colour of the sky is to be recreated. The assumption made in the re-lighting of the the sky, is that the ambient value estimated in the scene represents the mean intensity of the light affecting the scene from the sky. Based on this assumption, the sky-colour is during re-lighting set to the estimated ambient value.

When creating a shadow environment map, as seen in Figure 8.4, real-time, an approach is to use the shadow volume algorithm, used in the system, to cast shadows from the environment geometry onto itself, rendering the shadowed areas as black, an the lighted areas as white. To convert the shadow images, that is rendered real-time in OpenGL, to an environment map, a virtual camera can be set up in the point in the virtual scene, from where the environment map, that is to be re-lighted, was recorded in the real scene. From this point, the camera renders six square images, one for every principal direction, with a 90 degree field of view. These six images covers every direction from the camera point. To obtain a spherical representation of the environment, a cubic to spherical conversion is applied to the six images. An example is seen in Figure 8.5.



(a) *(b)*

Figure 8.5: Example of how the mapping from (a) cubic to (b) spherical environment representation. Image by [Bourke, 2002]

The remainder of this chapter explores the irradiance volume, and how it may be modified to use environment maps.

8.4 Irradiance Volume

Source: [Greger et al., 1998]
A realistic image synthesis has always been a major goal in the field of computer graphics. One such realistic synthesis is the concept of global illumination, as opposed to local illumination. Traditionally global illumination accounts for light interaction between surfaces in an environment, so both direct and indirect light sources are taken into account. These global illumination algorithms provide realistic effects but often at the price of computational expense.

This section explores the Global Illumination approximation algorithm called Irradiance Volume. Irradiance volume is a volumetric representation for the global illumination within a space based on the radiometric quantity irradiance. The irradiance volume provides a realistic approximation to global illumination without the amount of computation needed with traditional global illumination algorithms. The basic assumption of the irradiance volume is that the objects the irradiance volume shades, do not themselves affect the lighting of the scene.

The irradiance volume furthermore assumes that the visual appearance of a scene with diffuse surfaces can be computed from the reflectance and irradiance at the surfaces.

The method analyses the surrounding scene, samples it in various key points in the scene, and approximates how the lighting conditions are in those points. When a volume of such samples has been created, a random object can be shaded with a look-up in the volume.

8.5 Radiance and Irradiance

Radiance is the density of light energy flowing through a given point in a given direction. The luminance is the point viewed from a certain direction. The radiance in all directions at a given point **x** makes a directional function called the radiance distribution function (RDF). There is

a RDF at every point in space, which is why radiance is a five-dimensional quantity defined over all points and directions, two directional and three spatial.

The reflective properties and the incident radiance of a surface determines its outgoing radiance. The incident radiance of the surface is defined by hemisphere of incoming directions called the field-radiance function (FRF).

If a surface is diffuse then its radiance is defined by the terms of the surface irradiance, H:

$$H = \int L_f(\omega) cos\theta \quad d\omega \tag{8.8}$$

L_f is the field radiance incident from direction ω, and θ is the angle between ω and the normal vector $\tilde{\mathbf{n}}$ of the surface.

If the radiance from all incoming directions to one point is sampled, the irradiance for all possible orientations from that point can be computed. These irradiances can be plotted as a radial function. This is the irradiance distribution function (IDF) for a given point. This IDF can be computed for every 3D point in a given scene. A complete IDF is not practically possible, as it would demand sampling in a very high amount of directions and points, and the computational requirements are equally high. For high efficiency, the IDF is approximated.

If the IDF is approximated within the space enclosed by an environment in a form that can be evaluated quickly, the costly explicit evaluation of the IDF, for each time an object moves, can be avoided. This approximation is the irradiance volume, and the assumption is that it is possible to approximate the IDF with enough accuracy to avoid visual artifacts.

The simple way to approximate the function is to evaluate a finite set of points, and use interpolation to determine the function between them.

8.6 Creation of an Irradiance Volume

Irradiance volumes must be used in conjunction with an existing lighting solution, as it samples the radiance in the lit environment. Radiosity could be one of such solutions. To build the irradiance volume, the possibility to sample radiance at all points in all directions must exist. For a geometric environment built of polygons, the sampling can be achieved with ray-casting techniques.

The traditional implementation of the irradiance volume, involves a grid of samples being spanned in the environment in which objects are to move. The sampling is performed uniformly in the directional dimensions, since the IDF in a point is always continuous. The spatial dimensions are a bi-level grid for adaptive sampling, the reason for this, is that speed is the primary concern rather than accuracy.

The pseudo code for the computation of the grid is shown in pseudo code 4.

The subdivision of cells containing geometry, is to alleviate the most likely source of errors in the irradiance interpolation, since most discontinuities are caused by interruptions of light

Pseudo code 4 The creation of an irradiance volume grid

1. Subdivide the bounding box of the scene into a regular grid.
2. For any grid cell that contains geometry, subdivide into a finer grid.
3. At each vertex in the grid, calculate the irradiance distribution function.

flow by surfaces. The bi-level grid allows low density sampling in open areas, while a higher sampling is possible in regions with geometry.

The first-level grid is a $3 \times 3 \times 3$ grid that is built within the environment. Any grid cell containing geometry is then subdivided into a second-level $3 \times 3 \times 3$ cell. The regular grid outer vertices must be placed slightly within the boundaries of the environment so they do not intersect pieces of geometry. An illustration of a grid/cell/sample structure is seen in Figure 8.6.

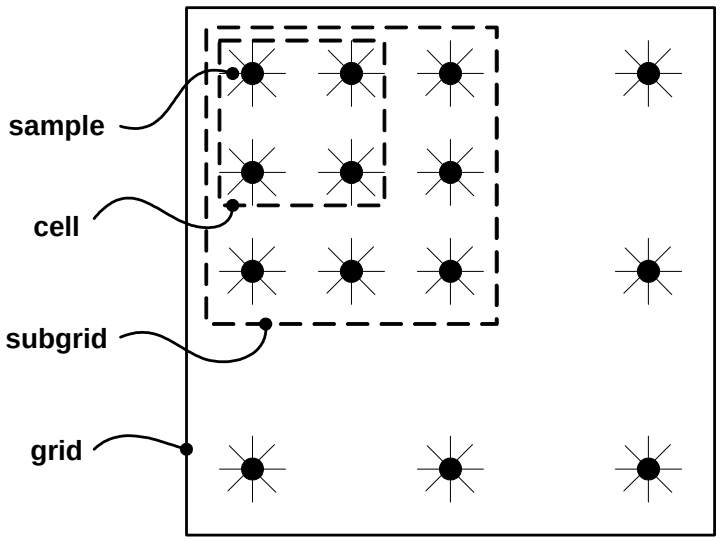


Figure 8.6: A grid is the overall structure of the storage of the irradiance volume. A grid contains cells. Each cell can either contain samples at its corners, or a new sub-grid-level, which contains sub-cells.

To approximate the IDF at an arbitrary point in a scene, the procedure seen in pseudo code 5 is :

Pseudo code 5 The creation of an irradiance volume sample

1. Given a directional sampling resolution, choose a set of directions over the sphere of directions in which to sample the radiance.
2. In each chosen direction, determine and store radiance.
3. Use the stored radiance to compute the irradiance distribution function.

This is performed for each grid vertex, to form the irradiance volume.

To gather the radiance viewed from a point in space, the radiance is point sampled in a set of directions over the spherical set of directions $S^2(\theta, \phi)$. Each sample defines a region of constant radiance over S^2.

To represent the sphere of directions S^2, [Shirley and Chiu, 1994] provides a method to map a unit square to a hemisphere, preserving the relative area of regions on the square. With this procedure, all directions represent an equal area on the hemisphere, which gives a mathematical advantage in the later calculations. This representation is better than the traditional longitude-latitude mapping, because mapped areas have better aspect ratio. The actual implementation will include two such hemispheres for each sample, to create one complete sphere. An illustration of the hemispheres that are stored as a square map is seen in Figure 8.7.

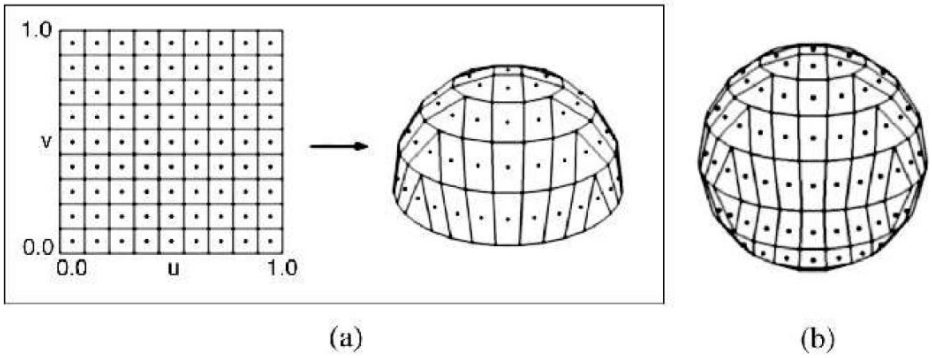


Figure 8.7: (a) shows the hemispherical mapping of a square map. (b) shows a sphere created from two of such hemispheres. (Image by [Shirley and Chiu, 1994])

The radiance $L(\mathbf{x}, \omega)$ is computed by casting a ray from point $\mathbf{x}$ in direction ω and evaluate the first surface, that is hit. If the surface is diffuse, the radiance is gathered from the surface, if not, the ray is reflected until a diffuse surface is found. A radiance sample can be seen in Figure 8.8 on the next pagea.

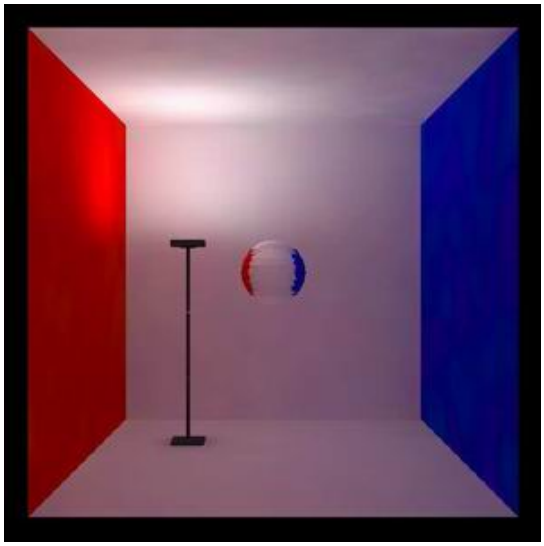
The number of samples on the hemisphere, that are needed to create a good approximation to the IDF at a point depends on the surroundings. A scene containing many objects may require a high sampling rate to avoid that certain important features of the scene is not missed.

When all radiance values have been gathered at a point, they will be used to calculate the irradiance for each sampling direction on the sphere. The irradiance and the radiance is calculated for the same resolution and the same directions. Irradiance is a smoother function than radiance, which means that a high sampling rate is not always needed, depending on the surroundings. In Figure 8.8b the radiance sample from Figure 8.8a is seen converted to an irradiance sample.

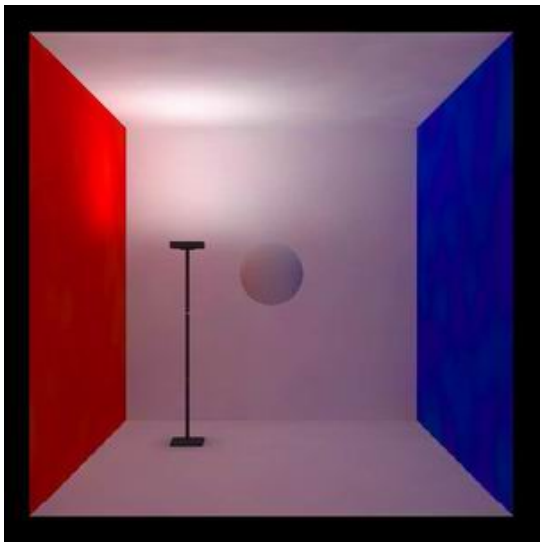
For a given direction the irradiance is computed for a hypothetical surface whose normal vector points in that direction. With a set of directional samples $\{\omega_1, \ldots, \omega_n\}$ sampled at a given point $\mathbf{x}$ the irradiance in direction ω is calculated using equation 8.9:

$$H(\omega) \approx \sum_{i=1}^{N} L(\omega_i) max(0, \omega_i \cdot \omega) \Delta\omega_i \tag{8.9}$$

where L is the radiance in direction ω_i from point $\mathbf{x}$, $\Delta\omega_i$ is the solid angle of the cone each ω_i represents, that is, each direction per sample represents a surface area on the sphere of



(a) The radiance of the Cornell Box sampled from the centre of the box



(b) The radiance sample converted to an irradiance sample

Figure 8.8: A sample in the middle of a geometric model as radiance and irradiance respectively. (Image by [Greger et al., 1998])

directions, this area becomes a cone when projected from the centre. N is the number of directional samples, on the sphere.

The sphere of directions is subdivided into regions of equal size, it follows that $\Delta\omega_i \approx \frac{4\pi}{N}$ and equation 8.10 emerges:

$$H(\omega) \approx \frac{4\pi}{N} \sum_{i=1}^{N} L(\omega_i) max(0, \omega_i \cdot \omega) \tag{8.10}$$

Equation 8.10 is calculated for a number of directions at the sample point in order to obtain a set of directional irradiance samples. The irradiance is computed in each direction by summing the cosine-weighted contribution from each radiance sample on the hemisphere oriented in direction ω. The approximate irradiance form an approximation of the IDF at a given point in space.

The resulting IDF can, like a radiance sphere, be displayed as an irradiance sphere.

In an implementation the Irradiance Volume is represented as three distinct data structures: samples, cells and grids.

- **Samples:** Contains directional irradiance values for a particular point.
- **Cells:** Representing a box in space, bounded by eight samples, one at each corner. Can contain a grid to form a bi-level structure.
- **Grids:** 3D array of cells.

When the irradiance volume is queried for the IDF at a given point between the samples, the following steps shown in Figure 6 are taken:

Pseudo code 6 Querying the Irradiance Volume

1. Calculate which cell in the first-level grid the point belongs to.
2. If cell contains a grid, calculate which cell in the second-level grid in which the point is contained.
3. Find which data value in the cell's samples corresponds to ω.
4. Given the points position within the cell and the eight corner sample values, interpolate to get the irradiance.

A completed irradiance volume can be seen in Figure 8.9.

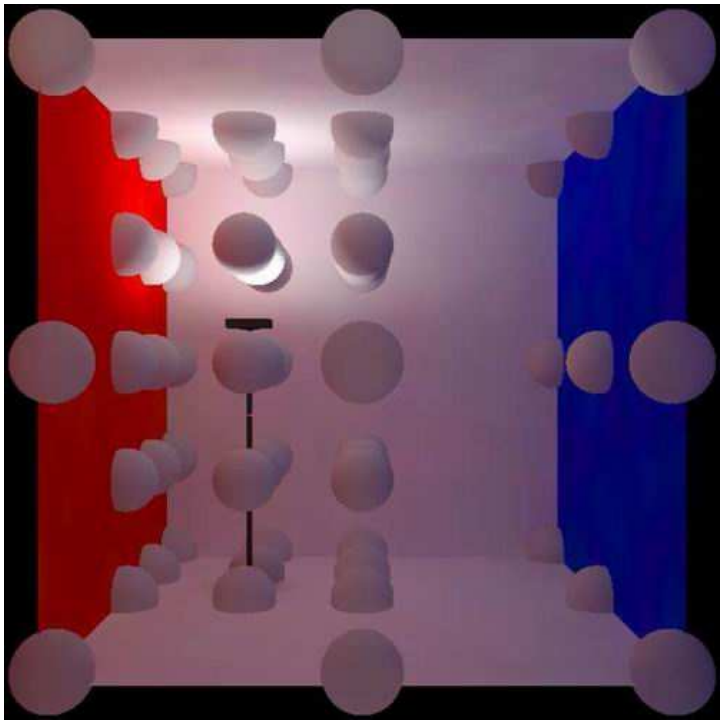


Figure 8.9: A completed Irradiance Volume. (Image by [Greger et al., 1998])

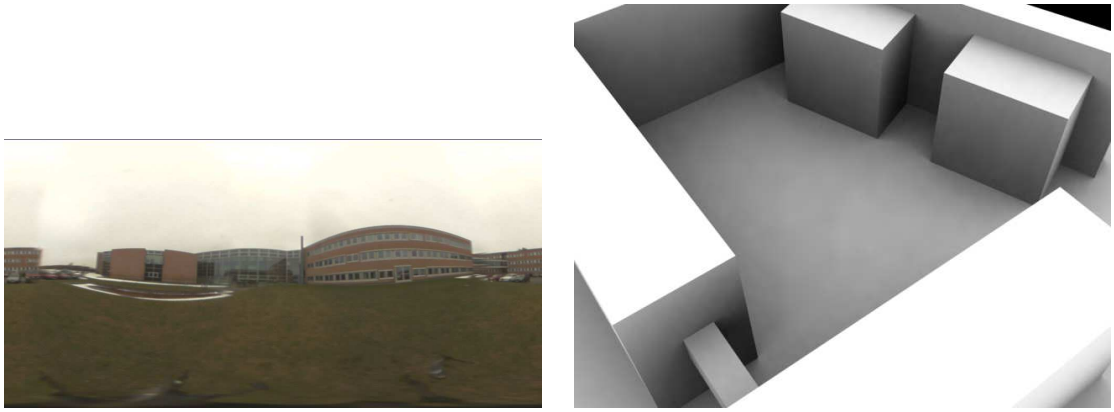
The process of finding which cell containing a given point is simple, as the bounding box of the volume is known. To find the value corresponding a direction ω, the hemispherical mapping described by [Shirley and Chiu, 1994] must be inverted. This will convert the direction to a (u, v) coordinate in the rectangular array describing the sample. This is used in the query where a simple check is performed on ω to determine which of the two hemispheres of a sample it belongs to. Then the inverse mapping of the hemisphere is used to find out which pixel in the irradiance map the direction lies within. This value is obtained from all eight surrounding samples, and trilinear interpolation is used to find the irradiance in direction ω at a given point.

8.7 Implementation

In the system developed for this project the implementation of the Irradiance volume function is generally as described in this chapter. With the exception that the radiance is given by an environment map instead of the traditional use of Irradiance volume, where it functions in conjunction with a method, like radiosity, that has already computed the radiance in the various parts of the scenes.

The irradiance volume created for the system takes as input a simple 3D model of a given scene, the 3D model created, has a minimal set of faces that still describes the main walls of the environment of the scene. An environment map of the scene is generated by the re-lighting part of the system every time a change in the scaled radiance is detected by the light estimation part of the system.

An example environment that has been used in this project is seen in Figure 8.10a, where a corresponding 3D model is seen in Figure 8.10b.



(a) *(b)*

Figure 8.10: Environment map of the example scene, and its corresponding 3D environment geometry model.

In order to speed up the creation of the irradiance volume, the scene seen in the example is a simplified version of the real environment, but the algorithm is able to handle a scene with arbitrary geometry. If more detailed and correct representation of the real scene is desired for the irradiance volume, it is possible to do, simply by adding more geometry in scene.

The Irradiance volumes secondary input, the bounding volume, is an axis aligned bounding box (AABB), the bounding volume defines the areas in which the virtual objects, that are to receive shading from the irradiance volume, are allowed to move. This is also the bounds in which the irradiance volume will be set up. The grid of the irradiance volume will start in each corner of the AABB, and subdivide each cell in the volume according to how many sub-levels the user has decided the irradiance volume should contain for the given scene.

The structure of the volume is constructed, as suggested by [Greger et al., 1998], by three

entities, grids, cells, and samples. To construct the volume, these entities follows two rules:

1. A grid always contains 8 cells.
2. A cell contains either a new grid or 8 samples.

From these rules an irradiance volume grid is created from the bounding boxes given by the user. The volume is first created by making a grid in the area confined by the bounding box. This first level of the grid will contain 8 cells.

Each cell is evaluated, and if it contains geometry, the cell is set to contain a grid, which again contains 8 new cells. This new grid is the next level of the volume. This process is repeated for as many iterations needed until it reaches a stop criteria. When the stop criteria has been met, all cells not containing grids, is set to contain 8 samples.

There are two types of samples used in the irradiance volume:

Positional sample A Positional sample is the corner samples in the Grid/Cell/sample structure.

Directional sample Directional Samples are contained within each positional sample.

When the system creates the radiance samples in the environment, it shoots rays out into the model of the environment model, and the first surface that is hit, is used to calculate the corresponding longitude-latitude coordinate on the environment map of the model. The returned value from the environment map is used as the radiance for that direction. The irradiance volume in this system handles one environment map, and as such there will be areas of the scene, that are not described by the environment map because of occlusions.

If the intersection test does not hit any geometry, it is assumed that the sky is hit, and the direction in which the ray was cast is read directly from the environment map into the power of the radiance samples. When the system shoots rays for each sample, it finds the 3D position of the intersection. If the 3D position from where the environment map for the scene was recorded is subtracted from the intersections 3D position, the result is a vector, that can be converted to a pixel coordinate on the environment map. This process is performed for each found intersection point. The process does not test if there is geometry between the intersection point and the environment map recording point, in which case the radiance found in the intersection point is not correct.

For each sample taken in the environment, each direction is integrated over a hemisphere of directions, which mean that flaws in one directional radiance sample has little effect on the final irradiance sample, given most of the sampled directional radiances are correct.

When the radiance has been gathered for each sampling point in the scene, each radiance sample is converted into an irradiance sample. This is a very computational intensive process, if there are many samples in the system. Given that each sample has the same number of directional radiance samples, and each directional radiance sample is oriented equally from sample to sample, a look-up-table can be created, that handles the $\omega_i \cdot \omega$ calculation once. If all radiance samples have the same mapping dimensions, this part of equation 8.10 is always

the same for each sample, and so it can be pre-calculated once, and used by the system in the creation of the irradiance samples. The creation of irradiance samples now only needs to multiply each pre-calculated direction with their corresponding radiance sample, and the irradiance volume is created.

This concludes the initialisation of the irradiance volume. When an object within the confinements of the irradiance volume is to be shaded, a query is sent to the volume, that searches the structure to find the sample for the objects particular 3D position. The search in the structure is as described by [Greger et al., 1998] to start at the top-level, determine which cell contains the queried 3D position. If this cell contains a grid, the search descends one level and searches the new grid. Once a cell containing the 3D position has been found, the 8 samples of the cell are interpolated with regards to the queried 3D position, from the 8 positional samples at each corner of the cell.

Then the 4 directional samples, that are closest to the requested directional sample, the vertex normal that is to be shaded, are found. The sample colour is then found by interpolating between the 4 directional samples and the 8 positional samples. The shading of the vertices of an object using the irradiance sample is currently utilising the diffuse Lambertian BRDF.

The implementation of the irradiance volume has shown, that it is important, to have more than one level before starting to subdivide the cells containing geometry. If the difference between the sizes of two adjacent cells is big the lighting of a sample has a tendency to "pop" when moving from one cell to another. The problem is particularly clear in movement between adjacent cells of different size, that are close to a surface. If the Irradiance Volume is initialised to contain at least 2 levels of sub-grids the problem however becomes invisible in the shading.

A general issue of the irradiance volume is that if there is a small but very dominant light source in the environment, such as the sun, it is possible that not all irradiance samples "catches" the light source during sampling, as there are only a finite number of directional samples in each positional sample, that has to cover the entire environment. One way to minimise this possible error, is to increase the directional samples per irradiance sample. This however increases the processing time. But the processing time is mainly used on converting the radiance samples to irradiance samples. With this in mind, the super-sampling method could be applied, where a high amount of directional radiance samples are created for each positional sample, and afterwards averaged to a smaller hemisphere map. This way more points in the environment are hit, and the possibility to find the light sources are increased.

A solution used in the system to ensure that the shading of the object always takes the sun into account is to omit the sun from the environment map during re-lighting, and add the sun during the shading of each vertex in an object. This ensures that the most significant light source in the scene is present in all samples. The environment map that shades the object through the irradiance volume then acts as the ambient light to the object, while the sun that is added during shading acts as the direct light source as in a local illumination model. The radiant power of the direct light is the direct light power that is determined by the light estimation.

The difference between this method and a normal local illumination model, is that the shading of the sides of an object that is not in direct illumination is no longer a uniform colour, but in fact is shaded from light reflections from the environment. An advantage of adding the direct lighting of the object during the vertex processing of the object is if the method is combined

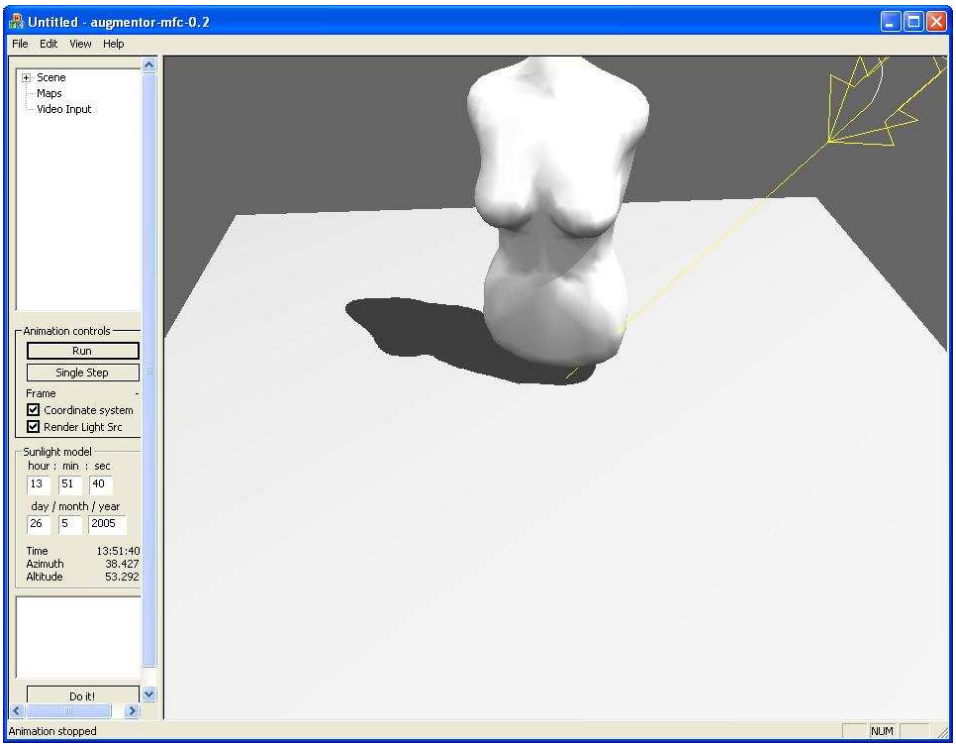


Figure 8.11: Augmentor displaying an object shaded by the irradiance volume, where self-shadowing is applied by the use of a shadow volume. The environment map used to shade the object is seen in Figure 8.10a

with a shadow algorithm that supports self-shadowing.

The shading of the irradiance volume is dependent of the shadows from the direct light source, which is not possible if the direct lighting is added to all samples in the irradiance volume during sample-generation. An example of this is seen in Figure 8.11, where the object is shaded using the irradiance volume. The environment map used is the one seen in Figure 8.10a, the direct light direction is provided from a sunlight-model.

The construction of the irradiance volume is performed in an offline process, where the geometry is processed, and the points in which the corner samples are placed are determined. At this point the content of the various samples are not found, the points on the environment map that each directional sample in each positional sample is precomputed at this point. This information is stored within each positional sample, so that when the environment map is updated from the image re-lighting part of the system, the information is used to quickly assign the colour values of the various directional samples. When the proper colour values from the re-lighted environment map has been copied to each sample, every sample is converted to an irradiance sample using a precomputed look-up-table to quickly integrate each directional sample over their hemisphere. This process is repeated every time the environment map is updated, and the storage of the environment map information ensures that no intersection testing into the environment is necessary in order to update the irradiance volume.

Summary

The estimated light parameters are used to shade the augmented virtual object. In this project, the shading is performed by re-lighting an environment map, in order for it to approximate the estimated current lighting of the scene.

During the re-lighting of the environment maps, the following assumptions are made:

- The re-lighting of the environment map assumes that all surfaces in the environment are diffuse.
- When re-lighting the sky in the environment map, the radiant power of the sky is assumed to be the estimated ambient value.

When the estimated environment map has been created, the irradiance volume is used to compute how the lighting of the surroundings affects the shading of a virtual object at a given point in the scene.

The shading of the object, based on the environment, is equivalent to the ambient term of a local illumination model. The difference from a traditional local illumination model, is that the shading obtained through the environment approximates global illumination in the scene.

The estimated direct light parameter is not used to position a virtual sun in the re-lighted environment map. The direct light value is added during the vertex processing of the virtual object for two reasons:

- If the estimated scaled direct radiance value was drawn into the virtual environment map as a sun, it would not be present in all irradiance samples in the volume, as each positional sample only samples the environment in a finite set of directions.
- It is possible to turn the shading from the direct light off, which is advantageous, when used in conjunction with a shadowing algorithm, as this allows self-shadowing of an object.

CHAPTER 9

Augmentor System Implementation

This chapter describes how the various methods described in the previous chapters are assembled into a collaborative system. The system, named Augmentor, is implemented as to execute the methods presented in a real-time augmented reality system.

The system, that has been developed through the project, is an augmented reality framework, created for an outdoor environment. The system is able to analyse images of a scene, of which it has a priori knowledge, and estimate the light parameters that equates the lighting of the real scene. These parameters are used to realistically light inserted virtual objects in the real scene.

The Augmentor system is comprised of various parts:

Offline Preparations: Before the system can be executed, data about the scene that is to be augmented must be acquired in an offline process.

Online Light Estimation: The online light estimation uses the data acquired in the offline part, to estimate the parameters of a local illumination model.

Environment Map Re-lighting: The estimated light parameters are used to create an environment map from the data acquired in the offline process.

Shadowing: The estimated light parameters are used to cast a shadow from the virtual augmented object onto the background image, using the shadow volume technique.

Shading: The direct light parameter and the re-lighted environment map are used to shade the virtual augmented object, using an irradiance volume. The shadow volume algorithm is used to create self-shadowing on the virtual object.

The previous chapters describing the methods, contained descriptions of how the methods were designed. In the various parts of the system there are minor deviations between the intended design and the implementation state at project deadline. The main reasons for the differences has been to ease the workload in implementing the various methods, as will be described for each of the following parts.

9.1 Map generation

In the design of the final system the only user provided data should provide in from the offline data acquisition, is the geometry of the surroundings, an albedo environment map, and the diffuse reflection map.

In the current implementation of the system, the additional data that must be acquired in the offline phase from the user is:

- Weighted ambient environment map
- Normal environment map

The decision to let these maps be created in the offline phase, was to keep focus of the project on the light estimation method presented through the project. Creating these maps based on geometry is a non-trivial task, and can be performed by standard 3D packages.

9.2 Live Video and tracking Input

The intention of the Augmentor framework is to be able to connect a camera and a tracker to the system and augment a given scene, by super-imposing virtual objects on top of the live video stream. The data from the tracker makes sure that the pose of the virtual camera and the real camera match.

In the implementation of the Augmentor system, video input is a pre-recorded video input, with no tracking required, while tracking is not used in the system. The reason for these de-limitations is practicality, they allow the system to execute on data that are readily available from a video-file, as opposed to a live input that can only be executed during day, when the weather is appropriate, and when a computer has been setup in the outdoor environment that is to be augmented.

9.3 Separation of camera-positions

The method presented in this project is to analyse images taken from a camera, and compare these images with an albedo environment map, that has been created in the offline process. This comparison requires that the environment map is photographed from the same position in the scene that the camera is recording the input images to the online light estimation during execution of the system.

During the implementation of the system, practical constraints entailed that the online camera recording position was moved from the position from where the environment map was recorded. These constraints induced the need for further maps provided from the offline process, as the online recorded images can no longer be compared to the environment map.

The result is that an albedo map, normal map and weighted ambient map must be created in the offline process to correspond the new camera pose. Another consequence is that camera movement is not allowed as there is no longer an environment map to compare the input to, this is already a de-limitation as camera tracking is not supported by the implemented system.

The resulting light parameters from the light estimation are used to re-light an environment map of the scene, with care taken to make the environment albedo map and the camera view albedo map match intensity and colour-wise.

9.4 Graphical User Interface

To ease the use of the system, as well as providing an effective tool for developing the methods in the system, Augmentor has been equipped with a Graphical User Interface.

In Figure 9.1 a screenshot of the Augmentor GUI is shown.

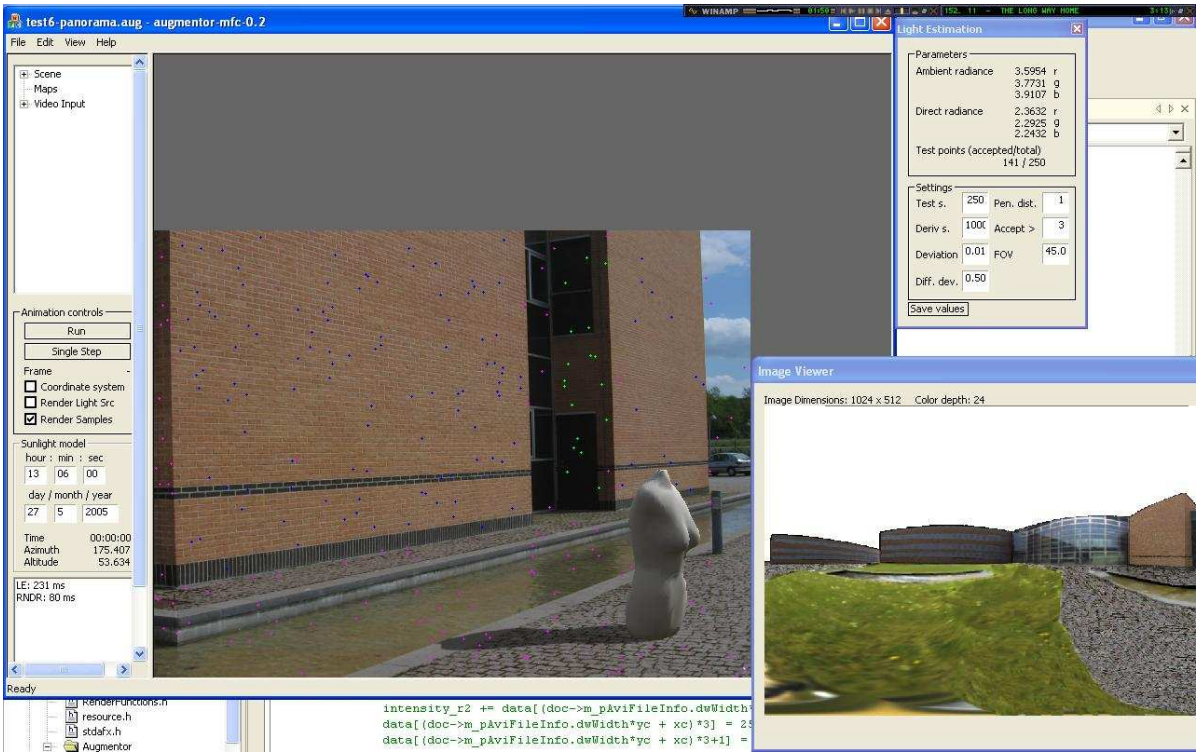


Figure 9.1: The Graphical User Interface for the Augmentor system.

The GUI contains several options:

File Tree

The user is provided with information of the files that are loaded into the framework through a file tree, that displays vertex/face/material information for the loaded 3D model, size and length information if video is loaded.

Animation Control

The user is enabled to control the speed at which the augmented scene is displayed when video input is enabled. Furthermore the user can turn the display of coordinate system, light source (sun) direction and sample points for the light estimation method.

Sunlight model

The user can manually set the date and time for the sun light model, to make sure the sun light model corresponds with the time at which the video was recorded.

Display Window

The display window is where the augmented objects are drawn onto the video image. In Figure 9.1 several coloured dots are seen on the video feed. These are the sample points at which the light estimation method tests if the last estimated light parameters correspond to the lighting of the current input image. The colours of the dots indicates if they are either identified as ambient or direct light samples, blue indicates a direct light sample, where green represents ambient light samples. Purple dots indicates samples that are rejected.

Light Estimation Dialog Box

To enable the user to regulate the various parameters the light estimation use in its sampling of the image, a dialog box with Light Estimation controls is provided.

The box displays the estimated light values, and has the following controls:

Test samples: The number of samples in the image where the light parameters are tested if they match the lighting of the scene. This test is used to determine if a light parameter derivation.

Derivation samples: The number of samples that is taken in the image every time the light parameters are to be re-estimated.

Deviation: The mean deviation allowed between the expected intensity at the test sample points and the actual measure at the sample point.

Difference deviation: One of the tests performed on all pixels is the difference test, where the pixel is tested if it matches the colour composition of the original albedo for the surface. The scale between the albedo and the measured intensity is computed for each colour channel. The difference deviation parameter indicates how much the scale of the colour channels are allowed to deviate from each other before the pixel is rejected as a pixel that possibly displays a dynamic object.

Penumbra Distance: The penumbra distance indicates how close to the edge of a shadow pixels are accepted as reliable samples.

Accept pixels: If the test only identifies a certain amount of either direct or ambient samples, the accept parameter indicates if this amount of samples is enough to determine that the mean deviation detected is reliable. The more sample points the more reliable the deviation must be.

Field of View: Indicates the field of view of the camera the input images are photographed with. This parameter is currently unused, as it relates to the case where the camera point and environment map centre point is the same. The parameter is then used to calculate which area in the environment map is currently being viewed in the scene by the camera.

Re-lighted Image viewer

Allows the user to view the estimated environment map of the scene, as it is re-lighted. The environment map is used for the ambient shading of the virtual object in the scene.

CHAPTER 9. AUGMENTOR SYSTEM IMPLEMENTATION

Part III

Results and Discussion

CHAPTER 10

Test

This chapter will describe the test cases and scenarios used for validation of the techniques and algorithms used in this project.

10.1 Introduction

Due to the many part elements of the system developed for this project, the testing regime will involve, test on both simulated and real data, first testing part elements separately and the finally a collective acceptance test, involving all part elements. The tests will be performed on both simulated input data and on real world data.

There are two main parts in the system, an offline and an online part. The offline part involves acquiring data for the online part. This includes the 3D models and maps used, and if relevant a prerecorded input video stream. Since all elements of the offline part are acquired through standard programs available, no tests will be performed directly on the offline part.

The online part is where all computations and rendering operations are performed. This is where the tests of this chapter will be concentrated. The part tests includes the following cases:

- Simulated data
  - Estimation of direct and ambient radiance on an animated image sequence
  - Reconstructing illumination in an image series from estimated illumination values
  - Comparative test with commercial rendering software
- Real world data
  - Estimating direct and ambient radiance in a time lapse movie sequence
  - Shading objects from an environment map
  - Shading objects from an environment map based on a time lapse movie sequence

As indicated in the implementation description, in Chapter 9, the system implementation is an offline version, supporting prerecorded video input from a fixed camera position and angle. This design choice was made since the tests will require input images from periods of several hours, which makes it practical to record a time lapse movie sequence for use in test purposes, thereby avoiding running the system for several hours.

The purpose of the tests will be to verify the functionality of the implemented system. This involves determining whether or not the system is eventually able to augment an outdoor scene, adjusting the illumination according to sun movement, and sun occlusions by e.g. clouds.

Throughout this chapter, when the term estimated radiance is used, it is synonymous with the scaled radiance estimated by the system.

10.1.1 Hardware

All tests are performed on the same hardware platform. The machine configuration is as follows:

- AMD Athlon XP1800 CPU
- 512 Mb DDR (PC2100) RAM
- GeForce 3 TI200 64Mb Graphics adaptor
- Microsoft Windows XP professional

10.2 Test scenarios - Simulated Data

The advantage of the tests performed with the simulated data is that a ground truth can be established for comparing with the values generated by the system developed for this project. Furthermore, the quality of the a priori data needed is fully controllable.

10.2.1 Estimation of direct and ambient radiance on an animated image sequence

Purpose

The purpose of this test is to verify that the light estimation algorithm is able to compute both direct and ambient radiances in a series of computer rendered, simulated environments. This will allow for an objective verification of the computed radiances.

The test is performed in the Augmentor framework with the use of a 100 image sequence rendered with the commercial 3D software package, 3D Studio MAX. For all images the direct and ambient radiances are known, and they will be compared to the values computed by the light estimation algorithm in Augmentor. A excerpt of the image sequence is shown in Figure 10.1. A test with moving dynamic objects is also performed, as to verify that the system will exclude all dynamic objects from the light estimation. A excerpt of the image sequence used in this test is shown in Figure 10.2.
A larger excerpt of the images tested is placed in Appendix E.0.2 on page 159.

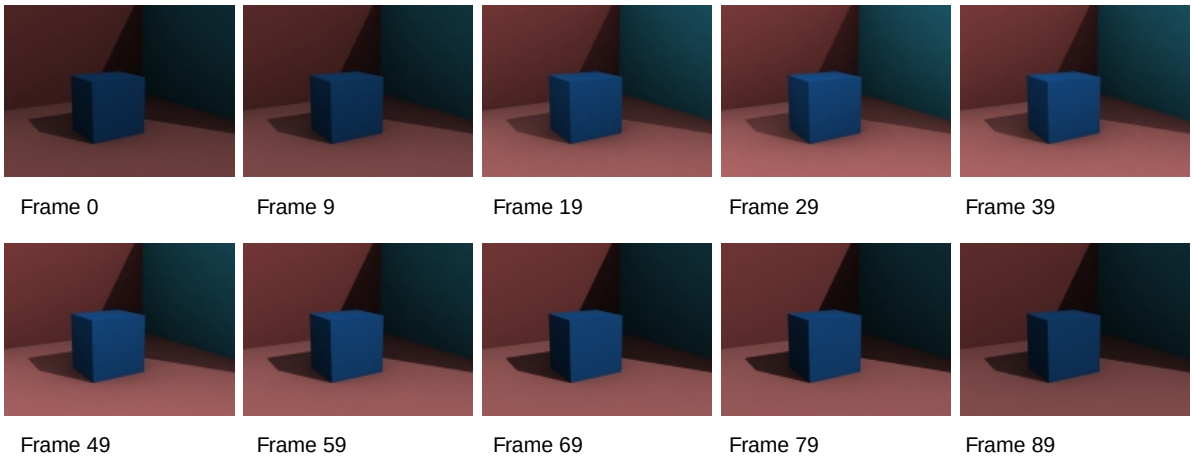


Figure 10.1: An excerpt of the input image series used for the test.

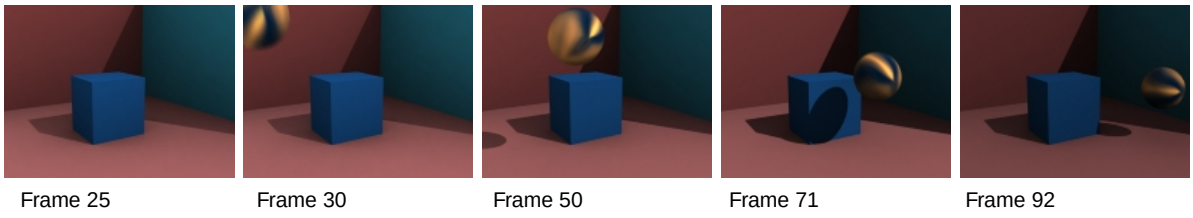


Figure 10.2: An excerpt of the input image series used with a dynamic object.

For this test to be successful the following main parts of the system must be functioning correctly:

- Shadow volumes
- Light estimation

Results

This test was performed on simulated data. This implies that all circumstances are ideal, meaning we have a perfect match of 3D scene and input image, albedo and shadow map are perfectly accurate and all surfaces perfectly diffuse. Shown in Figure 10.2.1 is a graph with a comparison of the radiances in the original rendered input frames and the per-frame estimated radiances. As can be seen in Figure 10.2.1, which is the case with no dynamic objects, the system is able to estimate both direct and ambient radiances up to a scaling factor with good accuracy.

Figure 10.2.1 shows the estimated radiances in the same scene but with a dynamic object present. When the object enters the scene around frame 30 the curves show that ambient is underestimated and direct radiance is overestimated. This is seems to be caused because the system momentarily samples the dynamic object as a directly illuminated surface. The

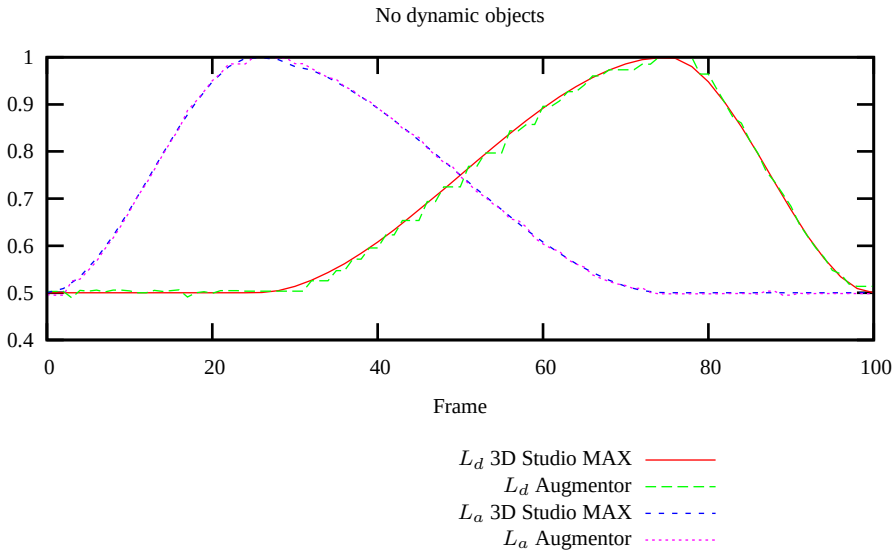


Figure 10.3: A graph showing the values for direct and ambient radiances from the original 3D Studio MAX rendering and the values estimated by Augmentor. All values are normalised to 1. No dynamic objects are present.

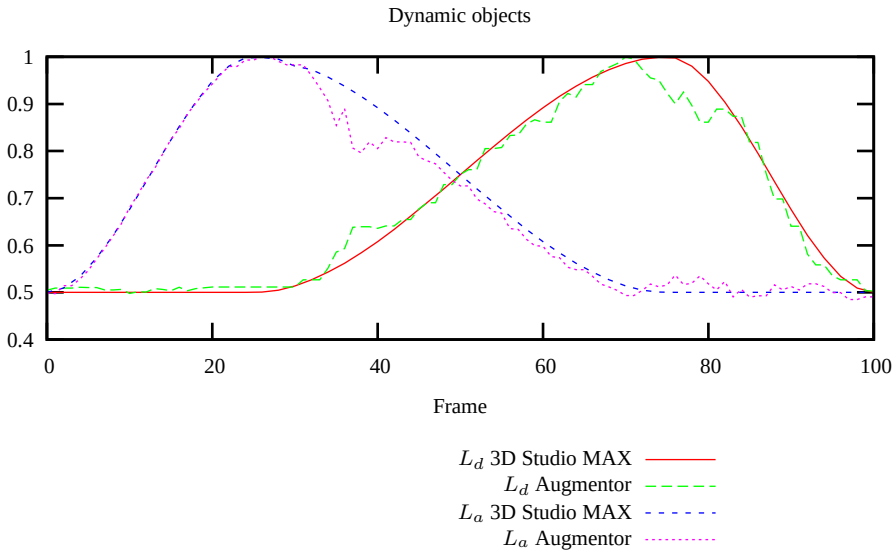


Figure 10.4: A graph showing the values for direct and ambient radiances from the original 3D Studio MAX rendering and the values estimated by Augmentor. All values are normalised to 1. Dynamic objects are present.

problem is only present for a few frames, and the size of the error is relatively low. The underestimation of direct radiance around frame 70 occurs because of the shadow cast by the dynamic object.

The system parts involved in this test are primarily the shadow map generator and the light estimation, an therefore we can conclude that these produce very reliable results on simulated data.

10.2.2 Reconstructing illumination in an image series from estimated illumination values

Purpose

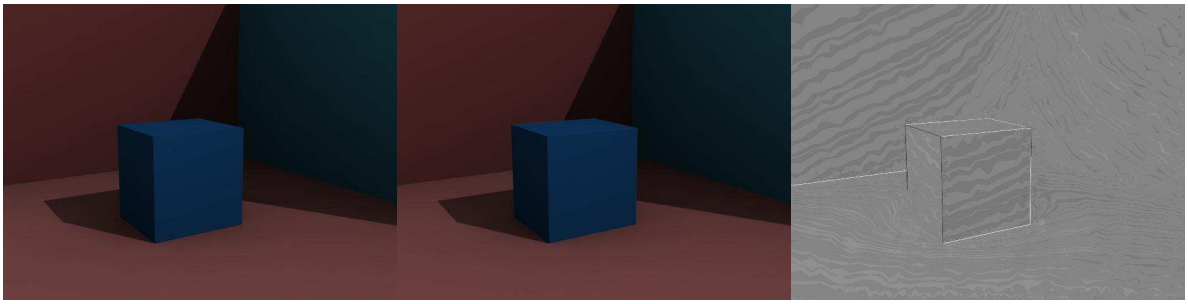
The purpose of this test is to verify whether the light estimation algorithm is able to estimate scene illumination parameters accurately, and reconstruct original light parameters based on the estimation. A selected few frames of the image sequence, used in the first test, are reconstructed using the estimated direct and ambient radiance in conjunction with the albedo map.

For this test to be successful the following main parts of the system must be functioning correctly:

- Shadow volumes
- Light estimation
- Environment map re-lighter

Results

The four re-lighted frames indicates that the estimated radiances are very reliable on simulated data. The difference images computed for all reconstructed frame shows low mean pixel errors with the worst case being 0.962 and a standard deviation of 3.4. Besides confirming the results of the previous test, this test also verifies the functionality of the environment re-lighter algorithm. Figure 10.5 shows one re-lighted frame. All the tested frames are shown in Appendix E.0.3 on page 160.



Original (Frame 0)	Reconstructed	Difference
L_d	L_d	Mean error: -0.119
L_a	L_a	Std. dev.: 3.4

Figure 10.5: Reconstruction of the original illumination conditions based on estimated values by Augmentor.

10.2.3 Comparative test with commercial rendering software

Purpose

The purpose of this test is to compare the results of an object rendered into a virtual scene by a non-real-time commercial rendering software with an object rendered into the same scene by the real-time system developed for this project, Augmentor. The scene is rendered in 3D Studio MAX with the female body sculpture as the augmented object. For the augmented scene, an environment without the augmented object is rendered in 3D Studio Max and used as input for Augmentor. The system uses the same female body sculpture as augmented object. This test involves all parts of the system, and can be considered as an acceptance test. By visual comparison it is observed whether the system produces images comparative to the ones produced by the commercial rendering software.

For this test to be successful the following main parts of the system must be functioning correctly:

- Shadow volumes
- Light estimation
- Environment map re-lighter
- Irradiance volume

Results

For this test, two sample images of the same object was rendered in both 3D Studio MAX and in Augmentor. The images are shown i Figure 10.6.

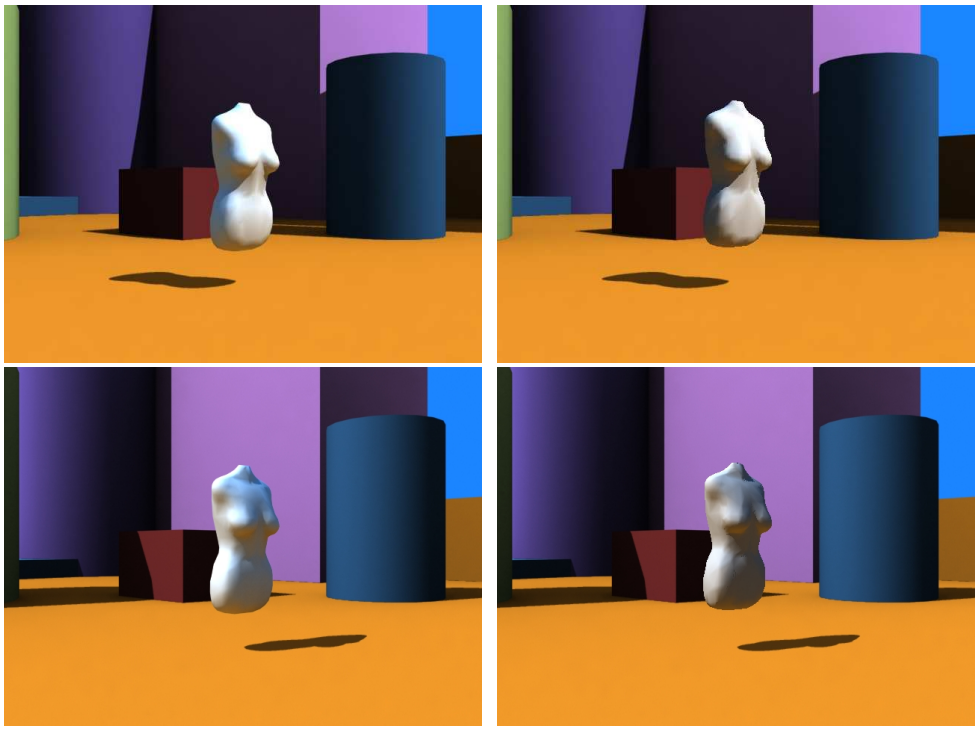


Figure 10.6: Results of the comparative test on simulated data. Left column shows the images generated by 3D Studio MAX, and the right column those generated by Augmentor.

The two images rendered by 3D Studio MAX took 5m 20s per frame, whereas the rendering time per frame in Augmentor is just below 1 second. The images show that Augmentor does yield images comparable to those of an offline GI renderer like 3D Studio MAX. For this to be the case, Augmentor is able to correctly estimate scene radiances, and re-light the environment map with these values. Furthermore the system is able to shade and shadow the object based on the re-lighted environment map.

10.3 Test scenarios - Real World Data

The purpose of the test performed with real world data is to verify the general functionality of the system developed for this project. The system is tested with two different real data sets. Both two-hour time lapse sequences with a frame interval of 20 seconds, as to cover as many illumination scenarios as possible.

10.3.1 Estimating direct and ambient radiance in a time lapse movie sequence

Purpose

The purpose of this test is to verify the functionality of the system when used with real world data.

This test corresponds to the first test on simulated data. A 300 frame image sequence, taken with a Canon A80 standard digital camera is used. The images were acquired from a fixed position every 20th second during a 2 hour period on Sunday May 15th (11.00-13.00), yielding approximately 360 frames. The image sequence shows the corner of a brick office building with one side in direct light and one in shadow. During the sequence both sides becomes shadowed. An excerpt of the image sequence is shown in Figure 10.7 For this test to be

Figure 10.7: An excerpt of the input image series used for the test.

successful the following main parts of the system must be functioning correctly:

- Sunlight model
- Shadow volumes
- Light estimation
- Environment map re-lighter

Results

This test was performed on real world data, and therefore there is no ground truth available as to determine whether the system is able to estimate radiances correctly. In order to render the results probable, this test is two-fold. First, the ambient and direct radiances for the image sequence were estimated. The estimated values are shown in Figures 10.3.1 and 10.3.1. It may seem counter intuitive that the direct radiance rises during the two hour period. The rise is caused by the sun altitude approaching zenith, which increases the incident radiance. This is barely visible to the human eye, when inspecting the image sequence. Figure 10.3.1 shows that the average intensity of a directly lit 20x20 area, indicated in Figure 10.11, in the input images does in fact rise. This coincides with the rise in pixel intensity for the surface of a virtual plane, illuminated with the systems illumination model, inserted into the same place.

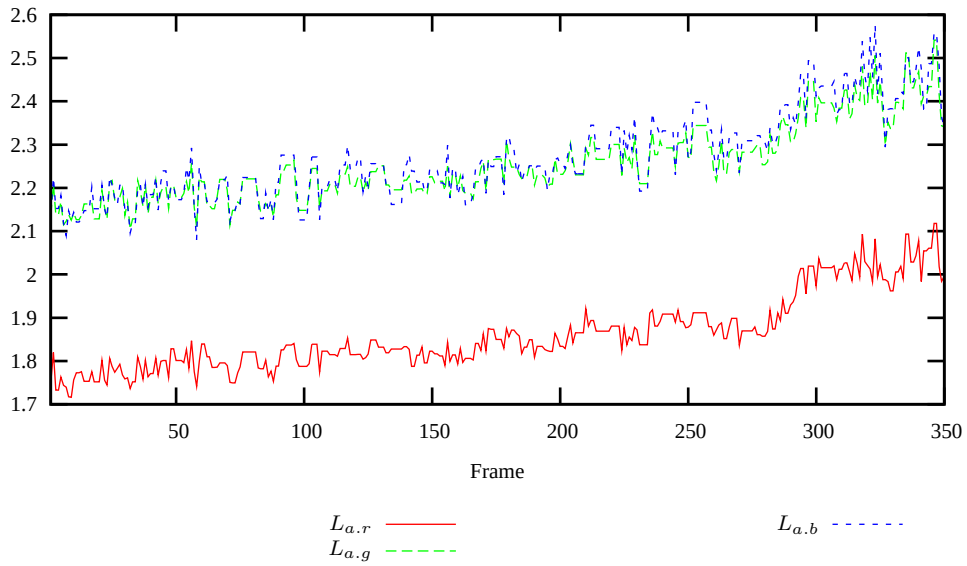


Figure 10.8: A graph showing the values for ambient radiance values estimated by Augmentor. No dynamic objects are present.

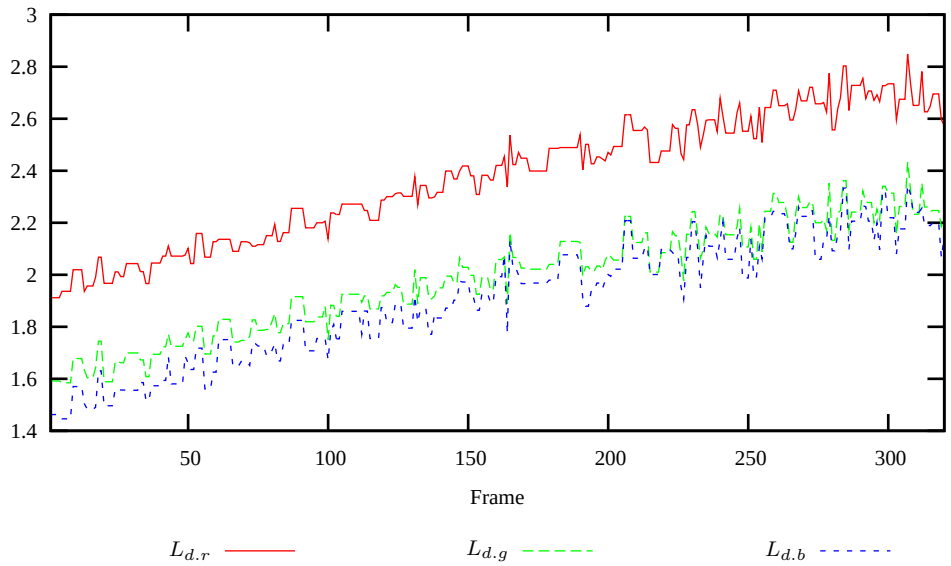


Figure 10.9: A graph showing the values for direct radiance values estimated by Augmentor. No dynamic objects are present.

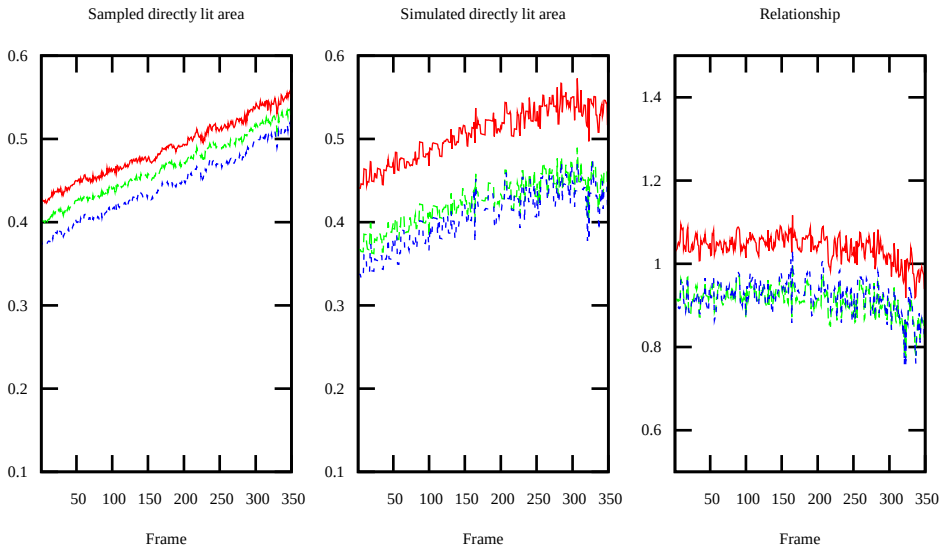


Figure 10.10: On the left, a graph showing the average pixel intensities sampled in a directly lit area, in the middle the simulated pixel intensities in a directly lit area, and on the right the relationship of the two curves. Pixel intensities are normalised to 1.

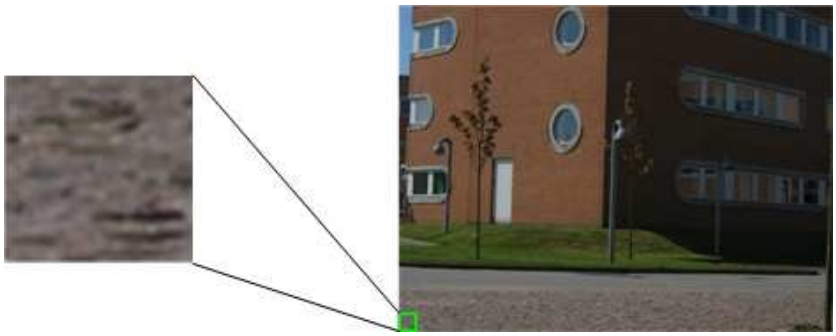


Figure 10.11: The marked area is where the directly lit pixels are sampled.

The graph shown in Figure 10.3.1 is a verification of the light model used. The verification consists of sampling the average pixel intensity in a 20x20 area for each frame, plotting these intensities in a graph.

Second, a “virtual surface”, inserted at the same location, is lit with the light model using the estimated parameters, the average pixel intensities are plotted in a graph. The two graphs are compared by simple division. If the light model is correct, the two graphs will exhibit a constant relationship. This is the case except for the last 40 frames, where the simulated surface has a decreasing intensity. Probable causes for this may be either erroneous calibration of the scene with regards to the sunlight, or that the surface sampled exhibits other properties than diffuse. This will have to be further explored.

Original (Frame 0)
$L_a = 2.13, 2.42, 2.45$
$L_d = 2.09, 1.79, 1.66$

Reconstructed

Difference
Mean error: 3.34
Std. dev.: 11.45

Figure 10.12: Reconstruction of the original illumination conditions based on estimated values by Augmentor.

Based on the results of this test, it can be concluded, that both the sunlight model shadow volumes are functioning correctly. The light estimation and re-lighting parts are also working according to specifications. Figure 10.12 shows a frame where the estimated radiances are used to reconstruct the illumination. The error is within acceptable limits, and is caused both by the fact that the albedo map used is blurred but also that non-diffuse surfaces are re-lighted as a diffuse surface.

10.3.2 Shading objects from an environment map based on a time lapse movie sequence

Purpose

This test corresponds to the last of the test on simulated data.

The purpose of this test is to verify that the system will function correctly, i.e. shade and shadow the virtual object correctly, during the course of a day. The test should illustrate that the sun's movement is accounted for and also that clouds occluding the sun are also handled.

For this test an environment map of the location illustrated in 10.17, is used. Furthermore a time lapse recording, from a known, fixed, view direction, taken on Friday 27th of May 2005 from 13:46 to 15:46, with a frame interval of 20 seconds and a resolution of 640x480 pixels is used. This yields approximately 360 frames. The time-lapse sequence was made on a semi-cloudy day, which gives ample room to test whether the system will adapt to lighting changes. For the test, the image sequence is used as input for the system, and it is observed whether the shading and shadow of the virtual object are consistent with that of the objects in the scene. An excerpt of the image sequence is shown in Figure 10.17.

Figure 10.13: An excerpt of the input image series used for the test.

For this test to be successful all the main parts of the system must be functioning correctly:

- Sunlight model
- Shadow volumes
- Light estimation
- Environment map re-lighter
- Irradiance volume

Results

Shown in the two graphs, Figure 10.14 and 10.15, are the estimated radiances during the entire image sequence.

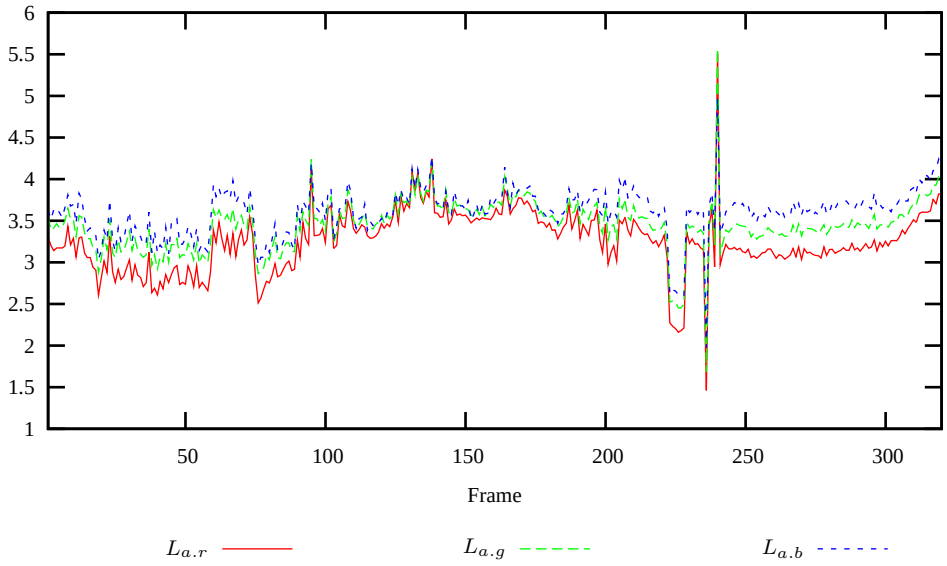


Figure 10.14: A graph showing the values for ambient radiance values estimated by Augmentor in the outdoor scene.

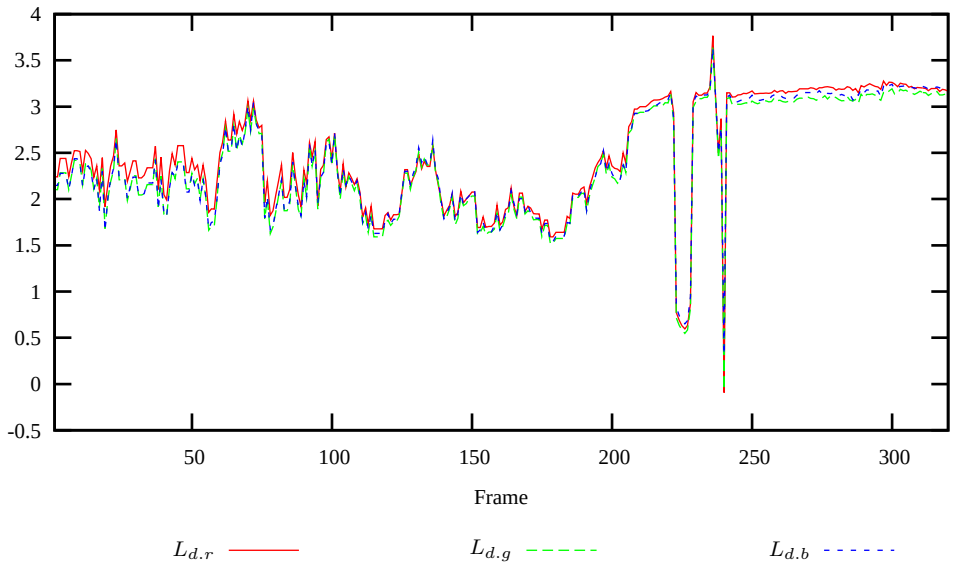


Figure 10.15: A graph showing the values for direct radiance values estimated by Augmentor in the outdoor scene.

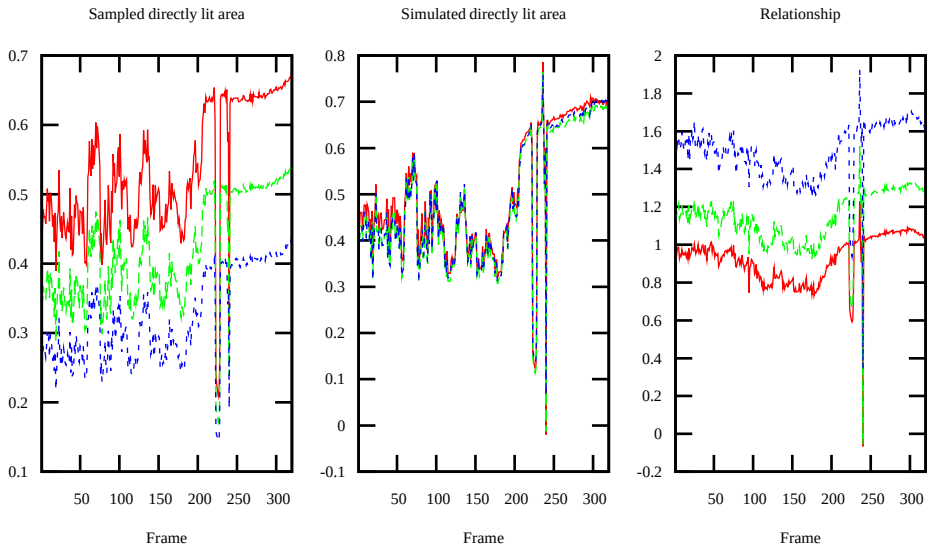


Figure 10.16: On the left, a graph showing the average pixel intensities sampled in a directly lit area, in the middle the simulated pixel intensities in a directly lit area, and on the right the relationship of the two curves. Average pixel intensities are normalised to 1.

As can be seen from the graphs in Figures 10.14 and 10.15, the radiance, varies during the course of the two hour period. The drops in direct radiance around frame 224-229 and 241 where large clouds occludes the sun are very pronounced.

The graph shown in Figure 10.3.2 illustrated the verification of the light model used. The verification consists of sampling the average pixel intensity in a 20x20 area for each frame, plotting these intensities in a graph.

Second, a “virtual surface”, inserted at the same location, is lit with the light model using the estimated parameters, the average pixel intensities are plotted in a graph. The two graphs are compared by simple division. If the light model is correct, the two graphs will exhibit a constant relationship. This test shows that the relationship of the estimated curves has a deviation of approximately 20% at times. There are several possible explanations, of which one is miscalibration of the orientation of the scene with regards to the sunlight model, and the normal maps used. This will of course have to be further explored. Though the error may seem high, it does not appear to any noticeable degree on the augmented objects.

In Figure 10.17 four augmented frames, with corresponding environment maps are shown. The environment maps have been re-lighted from the estimated illumination parameters.

An AVI movie file of the full sequence can be found on the accompanying CD in the directory verb|media/|.

Augmented frame (Frame 11)

Environment map

$L_a = (3.2891, 3.5130, 3.6953)$

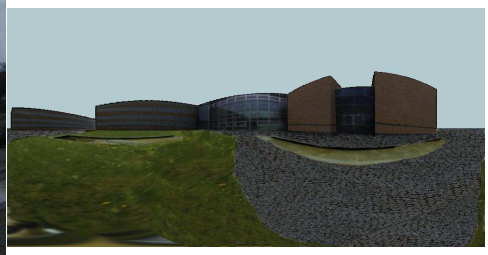
$L_d = (2.5150, 2.3868, 2.3910)$

Augmented frame (Frame 90)

Environment map

$L_a = (2.9180, 3.1296, 3.1581)$

$L_d = (1.9205, 1.8013, 1.8222)$



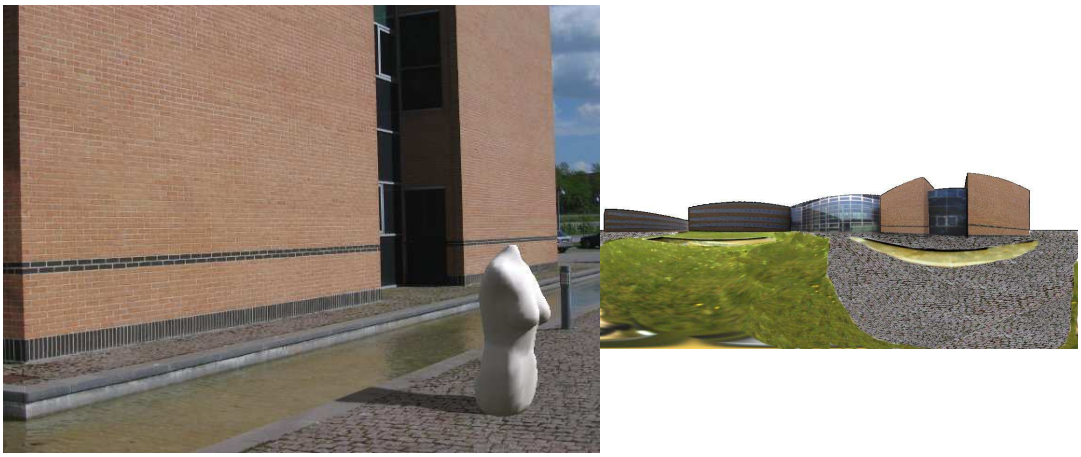
Augmented frame (Frame 229)

Environment map

$L_a = (2.2102, 2.5008, 2.5944)$

$L_d = (0.8676, 0.8175, 0.9460)$

Figure 10.17: Results of frames augmented by Augmentor.



Augmented frame (Frame 272)

$L_a = (3.1623, 3.4059, 3.6752)$

$L_d = (3.2046, 3.0896, 3.1524)$

Environment map

Figure 10.18: Results of frames augmented by Augmentor.

The images in Figure 10.17 and 10.18 clearly illustrates the capabilities of Augmentor. The four frames have different lighting conditions, to which the system has adapted the environment map, and correctly shaded and shadowed the virtual objects. The estimates for the light model are derived from the existing diffuse surfaces in the scene. All tasks are performed at a frame rate a little below 1 frame per second. Table 10.1 shows the average times for the tasks performed per-frame.

Process	Avg. time
Light estimation	190 ms
Re-lighting	710 ms
Irradiance volume update	<10 ms
Rendering	<10 ms

Table 10.1: Average processing times in Augmentor.

One frame per second is well below what is normally considered interactive frame rates. As shown in table 10.1, the majority of the frame time is used for the re-lighting algorithm. This is mainly caused by the fact that for this test, environment maps used were 1024x512 pixel, which is most likely a substantially higher resolution than what is actually needed for believable shading of objects.

CHAPTER 11

Conclusion

This chapter concludes the "Estimating and Applying Dynamic Light Changes to Environment Maps in Real-time for use in Image Based Lighting" master thesis in the Computer Vision and Graphic specialisation at Laboratory of Computer Vision and Media Technology, Aalborg University.

11.1 Aim of the project

The aim of this project has been to explore whether a method for updating a spherical environment map real-time, with the use of partial image information, is possible. And to use this to enable mimicking of dynamical light changes in an outdoor scene in an augmented reality system using image based lighting (IBL).

11.2 Methods

Among the several problem areas within the development of a real-time Augmented Reality (AR) system, a few areas have received in-depth attention in this project, these areas are concentrated in the field of creating convincing illumination in real-time in environments with dynamically changing illumination condition. This area was chosen, as it is an important part of enabling seamless blending of virtual objects with the real world. To realise an adaptive AR system that will function in locations with dynamically changing illumination conditions, methods for estimating illumination form surfaces with known properties have been explored.

The project consists of two main parts. An offline part, which entails all data acquisition and preparation hereof, and an online part in which the actual environment map updating and rendering takes place. The following two sections gives an overview of the tasks in the system.

11.2.1 The offline part

The offline part contains the preparation tasks needed to run the online part. Firstly an environment map is acquired for the location to be augmented. This is done through the method described by [Debevec, 1998], which involves taking images of a highly reflective metal sphere, and assembling a HDRI image.

The environment map is used as the basis for creating two maps used by the online part. One is an albedo map, describing the reflective properties of diffuse (Lambertian) surfaces. Since

not all surfaces may be diffuse, a second, binary map is made by the user, which defines the diffuse surfaces in the scene. The method applied in the project assumes that a scene contains diffuse surfaces. This has been shown in the project to be a fair assumption when dealing with regular masonry buildings.

Furthermore knowledge of the scene geometry must be known. An exact size 3D model of the location is required. From this model and knowledge of the sun position and radiance (derived from the environment map) an irradiance map, describing the illumination distribution in the scene geometry, can be produced. The irradiance map is rendered with a GI rendering package (3D Studio MAX was used).

With an irradiance map and an environment map a so called inverse rendering is performed simply by dividing the environment map by the irradiance map. This yields an albedo map, describing the diffuse surface properties in the scene.

As a last detail, a so-called ambient weight map is rendered in the scene geometry. This map is used as an extension of the ambient term in the local illumination model used for the light estimation in the online part. The ambient weight term, or ambient occlusion map as others have called it, describes how much of the sky-dome any given surface in the scene is able to "see".

11.2.2 The online part

The online part is where the actual light estimation and rendering takes place.

As mentioned in the description of the offline part, the system uses diffuse surfaces for estimating the illumination parameters. The focus in this project has been on augmenting an outdoor scene, this design decision allows for the use of a simplified light model, a local illumination model with one primary light source, i.e. the sun, and an ambient term representing the sky-dome and surrounding landscape. The method applied takes into account only diffuse (Lambertian) surfaces. To provide sufficient knowledge for the light estimation to work, the sun position must be known. This is accomplished though a sunlight model, which will compute the sun's azimuth and altitude angles.

With this information readily available, a linear equation system with two unknowns can be developed and solved by standard least squares method. The two unknown quantities being the direct and the ambient radiance.

The estimated radiance values are used to re-light the environment map used for shading the virtual objects by the irradiance volume. From knowledge of the albedo values for all surfaces, the scene geometry and the light model with the estimated parameters, an updated environment map is generated for every frame.

The Irradiance Volume is a volumetric representation of the global illumination within a space based on the radiometric quantity irradiance. The irradiance volume provides a realistic approximation without the amount of calculations needed with traditional global illumination algorithms. This approximated irradiance is utilised to shade the virtual objects within the scene. The irradiance volume used in this project has been modified, as to ensure that the dominant light source, i.e. the Sun, will not be missed during the environment sampling. This

is accomplished by maintaining two volumes. One representing the ambient illumination term, and the other the sun. The volume representing the ambient term can furthermore be used in conjunction with the chosen shadow algorithm, shadow volumes, to create very realistic self shadowing. As to provide the viewer with all important visual cues to object placement, a shadowing algorithm is used. Shadow volumes are furthermore used to cast realistic shadows on the surrounding environment.

11.3 Results

The method presented in this project is able to update an environment map based on partial low dynamic range (LDR) image data of the complete scene. The method uses diffuse surfaces in a scene to estimate parameters of a light model used to re-light an environment map. For the estimation to be successful, both directly illuminated areas and shadowed areas must be visible in the partial image data. The re-lighted environment map is used, through image based lighting, to realistically shade arbitrary virtual objects in the scene.

To assist in verification of the methods suggested, an Augmented Reality (AR) framework has been developed. It contains a graphical user interface with the possibility to manage both the files loaded into the system, but also parameters of the algorithms, e.g. the sunlight model and light estimation. The framework is based on Microsoft Foundation Classes (MFC) and the rendering API used is OpenGL.

To verify the functionality of the system the testing regime applied, involved testing with both simulated data and real world data. The advantage of testing with simulated data is that it is possible to obtain a ground truth, regarding both the light parameters, but also of the augmented object.

Three test on simulated data were performed. Firstly a test to verify that the light estimation method is able to correctly estimate ambient and direct radiance in an environment where all conditions are perfect. The test showed that the light estimation was able to produce very reliable estimates of radiances on simulated data. To further illustrate the accuracy of the estimated values and to verify the functionality of the image-relighting algorithm, the illumination was reconstructed from the albedo map and estimated radiances in four arbitrary frames. The test yielded accurately reconstructed images.

The last test on simulated data was a comparative test. The conclusion was that the system is able to produce augmentations, taking into account a dynamically changing environment, similar to those of an offline GI rendering engine, with a static environment, at a fraction of the rendering time. The average per-frame time for 3D Studio MAX being 5 min 20 s and, after initialisation, below 1 s for Augmentor.

To verify the methods actual applicability to real-world situations, two tests were performed on real data. These are the real-data counterparts of the simulated tests. In the first test, the estimated radiances for an image sequence were computed and plotted into graphs. The system estimated that the direct radiance increased during the time the images were captured. To verify that this increase was in keeping with the light model, average pixel intensities were

measured on a selected directly lit area in the image, and on a virtual plane, illuminated by the light model, placed at the same location. The sampled and virtual surfaces shows the same average intensity increase, with some incorrect results at the end of the sequence. This seems to be caused by a system calibration error, but will have to be further explored to establish an explanation.

The final test was a system wide test, involving all system parts, in augmenting virtual objects into a real scene. For this test, a virtual female torso was successfully augmented into a time lapse image sequence, captured from a known, fixed, viewing direction. To verify the light model, average pixel intensities in a selected area were also sampled in this scene. A comparison with the simulated intensities, showed a deviation during the sequence, where the intensities are underestimated. Again, a calibration error, is a probable cause for this error. In particular for the last test sequence, another contributor could be a proportionally smaller area in shadow. This could impose an error on the estimation of the ambient radiance. Further investigation into this area is necessary to establish an explanation. Despite of the error, no noticeable effect is observed on the augmented image sequence, which implies that the method has a certain degree of robustness in less than ideal environments.

11.4 Discussion

When moving towards photorealism within the field of Augmented Reality (AR), one must keep in mind, that AR, as opposed Virtual Reality (VR), co-exists with the real world. Consequently, to maintain the illusion that virtual objects are a part of the real world, phenomenas inherent to the real world must be considered. This project has focused on one major factor in supporting this illusion; illumination. The initial goal, to shade virtual objects correctly, in an outdoor environment with both predictable and unpredictable dynamic changes, has been researched through this report.

The research has been put into practise in a functioning AR framework, which is able to shade and shadow virtual objects, with good fidelity, in an outdoor location. The framework estimates the parameters of a local illumination method, which in turn is used for re-lighting an environment map. The environment map is basis for an image based lighting technique.

The method presented in this report provides an elegant method for estimating illumination parameters in an AR system, without he use of any calibration objects present in the scene. The method is able to use existing geometry to update light parameters, leaving the scene unmodified. The principle of handling dynamic light changes in an Augmented Reality (AR) system by updating light parameters as presented, is unequalled in existing research, like for instance [Sato et al., 1999] and [Gibson et al., 2003].

The consequence of using the method presented is that some preparation is needed. The requirements of a priori knowledge of a given location is directly linked to the fact that no modifications are made in the location. That trade-off means that a series of conditions must be met.

The augmented scene must be outdoor, and scene geometry must be known. Furthermore two

maps must be created, an albedo map, describing scene surface properties and a diffuse surface mask, defining diffuse surfaces in the scene. Lastly the location and time must be known. The authors of this projects believe that the requirements do not present a substantial increase in the workload otherwise required when using static image based illumination.

Currently the prototype system is able to perform a full augmentation, from light estimation, through environment map relighting, to actual shading and shadowing at just below 1 frames per second. This frame rate is not satisfactory in a real-time system, and the cause is mainly found in both the light estimation, and environment map relighting computation. The time of relighting can be improved substantially by using a lower resolution environment map than used in the tests. Furthermore, it is the belief of the authors, that most of the algorithms used can be optimised considerably, thereby making the method feasible for a real-time Augmented Reality system.

In a “online” AR system, a fixed view direction, as employed in this project is clearly not desirable. The methods presented here are directly applicable to a system involving independent view direction, as recording camera orientation with the use of a hardware tracking system has several well explored solutions. With the use of a fixed camera position, optical angle encoders can provide more than sufficient tracking resolution.

The system developed through this project is purely a prototype, used for proof of concept. It is, nevertheless, feasible that the methods developed, are very usable in a production type systems. These systems could vary from offline rendering systems for movie effects, and architectural visualisations to real-time applications such as Augmented Reality info-tainment systems placed at e.g. historical sites.

11.5 Future work

This section will discuss some of the possible improvements and additions to the work presented in this project.

As described earlier, the method presented in this project, works under the assumption, that the partial image input data supplied contains diffuse surfaces in both direct light and in shadow. This is necessary for a valid estimation of light parameters. In that context, exploring a “flight envelope” principle, where the system is able to detect erroneous input, i.e. views without shadows and vice versa, and in some way indicate to the user which viewing directions the system can be can safely used.

A drawback of the current implementation of the system is the number of initial maps needed. A method for creating the larger part of these maps internally in the system would ease the use of the system, and enable full two degrees of freedom in orientation. The number of initial maps could be reduced to only two maps, the albedo map, and the diffuse surface map. The generation of all other maps, including normal maps and weighted ambient map, are done through methods well suited for implementation in an integrated offline part of the system.

Throughout this work, it has been a general assumption, that the scene surfaces consists mainly of diffuse (Lambertian) materials, but the method will extend nicely into more complex sur-

faces, provided a parametric description (BRDF) is available.

Testing has shown, that the estimates provided by the system have a tendency to be prone to a certain amount of noise. This noise is in some cases visible on the images produced. The cause of this noise can, in part, be attributed to the test data used. Images captured with a time interval of 20 seconds exhibits a certain degree of discontinuities in illumination, which is a major contributor of noise. To counteract noise, different filtering algorithms may be explored. Filter types from the simple, low pass types to more advanced predictive types. Furthermore, sample point selection could be altered to a statistically based selection, instead of a random seed for each sample.

Any graphics API will contain specific methods for improving rendering speed. OpenGL is no exception. Through the use of, amongst other things, the very fast RAM, and the powerful processors (GPU's) present on modern graphic card, rendering speed may be improved substantially.

Bibliography

[Akenine-Möller and Haines, 2002] Akenine-Möller, T. and Haines, E. (2002). *Real-Time Rendering*. A K Peters Ltd, 2 edition.

[Andersen and Jensen, 2004] Andersen, M. S. and Jensen, T. (2004). Real-time augmented reality. Technical report, Aalborg University - Laboratory of Computer Vision and Media Technology. `http://www.control.auc.dk/~04gr922/pdf/report.pdf`.

[ARTHUR, 2004] ARTHUR (2004). Augmented round table for architecture and urban planning. `http://www.fit.fraunhofer.de/projekte/arthur/index_en.xml`.

[Barequet et al., 2001] Barequet, G., Duncan, C. A., Goodrich, M. T., Kumar, S., and Pop, M. (2001). Efficient perspective-accurate silhouette computation. *Proceedings of the seventeenth annual symposium on Computational geometry*, pages 60–68.

[Beam, 2004] Beam, J. (2004). Tutorial - stenciled shadow volumes in opengl. `http://www.3ddrome.com/article/shadowvolumes.php`.

[Bourke, 2002] Bourke, P. (2002). Converting to and from 6 cubic environment maps and a spherical map. `http://astronomy.swin.edu.au/~pbourke/projection/spheretexture/`.

[Carmack, 2000] Carmack, J. (2000). Unpublished correspondence. `developer.nvidia.com/docs/IO/1317/ATT/CarmackOnShadowVolumes.txt`.

[Debevec, 1998] Debevec, P. (1998). Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography.

[Decaudin, 1996] Decaudin, P. (1996). Cartoon looking rendering of 3D scenes. Research Report 2919, INRIA.

[Elias, 2003] Elias, H. (2003). Radiosity. `http://freespace.virgin.net/hugo.elias/radiosity/radiosity.htm`.

[Gibson et al., 2003] Gibson, S., Cook, J., Howard, T., and Hubbold, R. (2003). Rapid shadow generation in real-world lighting environments. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 219–229, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

[Giesen, 2005] Giesen, J. (2005). Basics of positional astronomy. `http://www.jgiesen.de/SME/details/basics/`.

[Greger et al., 1998] Greger, G., Shirley, P., Hubbard, P. M., and Greenberg, D. P. (1998). The irradiance volume. *IEEE Computer Graphics and Applications*, 18(2):32–43.

[Heidmann, 1991] Heidmann, T. (1991). Real shadows real time. *IRIS Universe*, (18):28–31.

[Madsen et al., 2003] Madsen, C. B., Sørensen, M. K. D., and Vittrup, M. (2003). Estimating positions and radiances of a small number of light sources for real-time image-based lighting. In *Proceedings: Annual Conference of the European Association for Computer Graphics, EUROGRAPHICS 2003, Granada, Spain*, pages 37 – 44.

[McGuire and Hughes, 2004] McGuire, M. and Hughes, J. F. (2004). Hardware-determined feature edges. In *Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, pages 35–147. ACM Press.

[Milgram and Kishino, 1994] Milgram, P. and Kishino, F. (1994). Taxonomy of mixed reality visual displays. *IEICE Transactions on Information and Systems*, Vol.E77-D Issue.12(1321-1329):9.

[Nettle, 1999a] Nettle, P. (1999a). Radiosity in english. `http://www.fluidstudios.com`.

[Nettle, 1999b] Nettle, P. (1999b). Radiosity in english ii: Form factor calculations. `http://www.fluidstudios.com`.

[Preetham et al., 1999] Preetham, A. J., Shirley, P., and Smits, B. (1999). A practical analytic model for daylight. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 91–100. ACM Press/Adddison-Wesley Publishing Co.

[Sato et al., 1999] Sato, I., Sato, Y., and Ikeuchi, K. (1999). Illumination distribution from shadows. In *Proceedings: CVPR99*, pages 306–312.

[Schlyter, 2005] Schlyter, P. (2005). Computing planetary positions - a tutorial with worked examples. `http://www.stjarnhimlen.se/comp/tutorial.html`.

[Shirley and Chiu, 1994] Shirley, P. and Chiu, K. (1994). Motes on adaptive quadrature on the hemisphere.

[Slater et al., 2001] Slater, Steed, and Chrysanthou (2001). *Computer Graphics And Virtual Environments: From Realism To Real-Time*. ADDISON-WESLEY, 1 edition.

[Surdulescu, 2003] Surdulescu, R. (2003). Cg shadow volumes. `http://www.gamedev.net/reference/articles/article1990.asp`.

[Wellner, 1993] Wellner, P. (1993). Interacting with paper on the DigitalDesk. *Communications of the ACM*, 36(7):86–97.

[Whitehurst, 2001] Whitehurst, A. (2001). Depth map based ambient occlusion lighting. `http://www.andrew-whitehurst.net/amb_occlude.html`.

[Williams, 1978] Williams, L. (1978). Casting curved shadows on curved surfaces. *Proceedings of the 5th annual conference on Computer Graphics and interactive techniques*, pages 270–274.

[Yu et al., 1999] Yu, Y., Debevec, P., Malik, J., and Hawkins, T. (1999). Inverse global illumination: recovering reflectance models of real scenes from photographs. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 215–224, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Yu and Malik, 1998] Yu, Y. and Malik, J. (1998). Recovering photometric properties of architectural scenes from photographs. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 207–217, New York, NY, USA. ACM Press.

[Zhang and Hoff(III), 1997] Zhang, H. and Hoff(III), K. E. (1997). Fast backface culling using normal masks.

BIBLIOGRAPHY

Part IV

Appendix

APPENDIX A

Phong Shading

Source: [Slater et al., 2001]
This appendix contains a description of the Phong shading model, that approximates how light affects 3D surfaces.

In computer graphics the ideal shading of surfaces is by utilising a global illumination model, to calculate all lighting in a scene. Global illumination calculates influences from all light sources in all directions at all points of a 3D scene.

Local illumination

A way to simplify illumination calculations is by only calculating the most significant influencing lights, as the peripheral lights has little effect on the illumination. This is called local illumination.

The Phong Shading Model is a local illumination model.

The shading of each 3D surface, is divided into three categories:

Diffuse reflection: Diffuse reflection is the direct illumination of a surface, see figure A.1a.

Specular reflection: Specular reflection controls how a surface shines when lit with direct light, see figure A.1b.

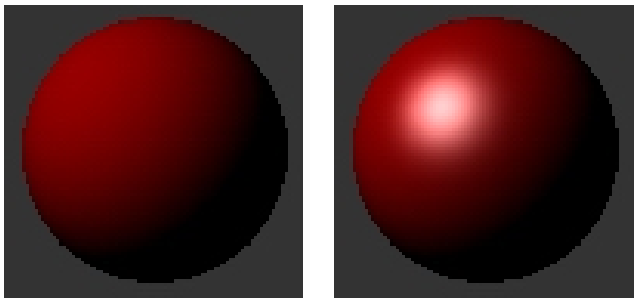
Ambient reflection: Ambient reflection controls the illumination of surfaces not affected by light.

To calculate the total light reflection in a scene the three reflections must be calculated for each point.

Diffuse reflection

Diffuse reflection may be calculated with the following equation. The factors are illustrated in figure A.2 on the next page:

$$I_{r,d} = k_d \cdot I_i \cdot \cos D \tag{A.1}$$



(a) *(b)*

Figure A.1: (a)Diffuse reflection is the direct illumination of a surface, which makes the surface visible. (b) Specular reflection controls how the surface shines on surfaces with direct light.

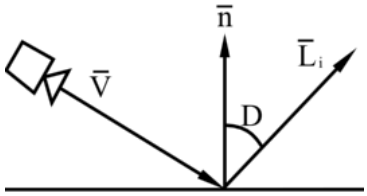


Figure A.2: The diffuse reflection of the Lambert Reflection Model.

where:
$I_{r,d}$: The resulting intensity value for each edge voxel.
k_d: Diffuse reflection coefficient. One for each colour channel.
I_i: Intensity of the incoming light.
D: The angle between the normal vector and the incoming light vector.
$\vec{n}$: The normal vector to the surface.
$\vec{L_i}$: The vector for the incoming light.
$\vec{V}$: The viewing vector.

The diffuse calculations are performed for each colour channel denoted by k_d. The calculations are performed with normalised values, ranging between 0 and 1. With normalised vectors, $\cos D$ may be calculated with the following equation:

$$\cos D = \vec{n} \cdot \vec{L_i} \tag{A.2}$$

Specular reflection

Specular reflection may be calculated with the following equation. The factors are illustrated in figure A.3 on the facing page:

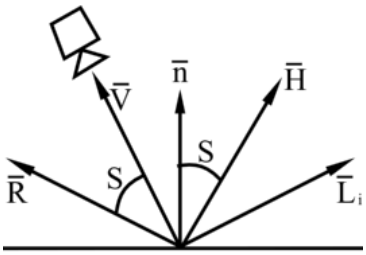


Figure A.3: The specular reflection of the Lambert Reflection Model.

$$I_{r,s} = k_s \cdot Ii \cdot (\cos S)^m \tag{A.3}$$

with:
k_s: Specular reflection coefficient.
I_i: Intensity of the incoming light.
S: The angle between the perfect reflection angle and the viewing angle.
m: Shininess-factor. Ranging from 1 and to values above. The higher the value, the more concentrated the specular reflection will be.
$\vec{L_i}$: The angle for the incoming light.
$\vec{H}$: The angle between the viewing angle and the incoming light angle.
$\vec{R}$: The perfect light reflection angle.

These are the same calculations for each colour channels.
The $\cos S$ may be replaced by:

$$\cos S = \vec{V} \bullet \vec{R} = \vec{n} \bullet \vec{H}$$

Where

$$\vec{H} = \vec{V} + \vec{L_i}$$

By applying the latter replacement the calculations may be simplified, as all factors $\vec{n}$, $\vec{V}$ and $\vec{L_i}$ are known.

Ambient reflection

The ambient reflection is the simplest of the three, as it does not incorporate angle calculations. Ambient is the light that affects all points.

The ambient reflection I_a may be calculated with:

$$I_{r,a} = k_a \cdot I_a \tag{A.4}$$

where:
k_a: Ambient reflection coefficient
I_b: Ambient light intensity

A.0.1 Total reflection

All factors adds to the total reflection like:

$$I_r = I_{r,a} + I_{r,d} + I_{r,s} \tag{A.5}$$

If there are several light sources within the scene, they will have to be calculated one at a time, and the partial results summed at the end. Light sources has no effect on the ambient reflection, hence this can be omitted of the illumination calculations.

$$I_r = k_a + I_a + \sum_{j=1}^{N}(k_d \cdot I_{i,j} \cdot (\vec{n} \bullet L_{i,j})) + \sum_{j=1}^{N}(k_s \cdot I_{i,j} \cdot (\vec{n} \bullet (\vec{V} + \vec{L_{i,j}}))^m) \tag{A.6}$$

This can be reduced to:

$$I_r = k_a + I_a + \sum_{j=1}^{N}(I_{i,j} \cdot (k_d \cdot (\vec{n} \bullet \vec{L_{i,j}}) + k_s \cdot (\vec{n} \bullet (\vec{V} + \vec{L_{i,j}}))^m)) \tag{A.7}$$

The summations ranges from 1 to N, with N denoting the number of light sources.

APPENDIX B

Testing the diffuse reflectance of regular masonry

The purpose of this appendix is to investigate if the assumption that regular masonry buildings reflectance distribution function is the Lambertian reflectance is a fair assumption. The test object is a building brick, that will be lighted in a controlled environment from different angles and tested against how the the light reflected from the brick is expected to reflect if the surface is diffuse.

Purpose

The purpose of this test is to verify or deny the assumption that regular masonry present in most any architectural structures is somewhat diffuse, and therefore is usable in the light estimation method developed in the course of the project.

Test setup

A masonry brick is placed in a controlled environment, and one light source is used to illuminate the setup.

The light source is static, and lights the brick from the same angle as the camera records the brick. The environment consists of completely black surfaces, as to minimise reflected light affecting the brick illumination.

The brick is rotated for every image, starting with one surface a 0 degrees with regards to the viewing angle, and ending at 90 degrees.

The values measured on the surface of the brick is compared to what is expected from an object with a diffuse reflectance.

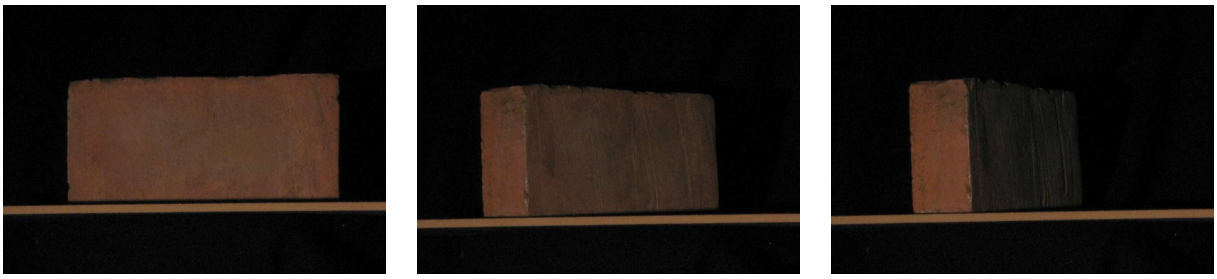
The Lambertian part of the Phong Shading model is used for calculating the expected light measured at a diffuse surface.

Results

The test is performed with a circular light placed in the camera position. The brick is photographed at various poses, the reflected light from the brick is measured, and compared to a correspondent Lambertian reflection. The results are seen in table B.1, and an excerpt of the images used are seen in Figure B.1.

Angle	Measured Intensity	Simulated	Measured Normalised
0	64.47	1	1
10	63.78	0.985	0.989
20	61.57	0.94	0.955
30	56.45	0.866	0.876
40	51.14	0.766	0.793
50	42.82	0.643	0.664
60	34.76	0.5	0.539
65	29.29	0.423	0.454
70	24.08	0.342	0.374
75	17.95	0.259	0.278
80	11.41	0.174	0.177
85	4.11	0.087	0.064
90	0.49	0.0	0.008

Table B.1: The results of the brick reflectance test.



(a) *(b)* *(c)*

Figure B.1: An excerpt of the brick photographed at various angles, 0 degrees(a), 60 degrees (b) and 80 degrees (c).

The normalised intensities of the measurements seen in table B.1, are plotted to a graph in Figure B.2.

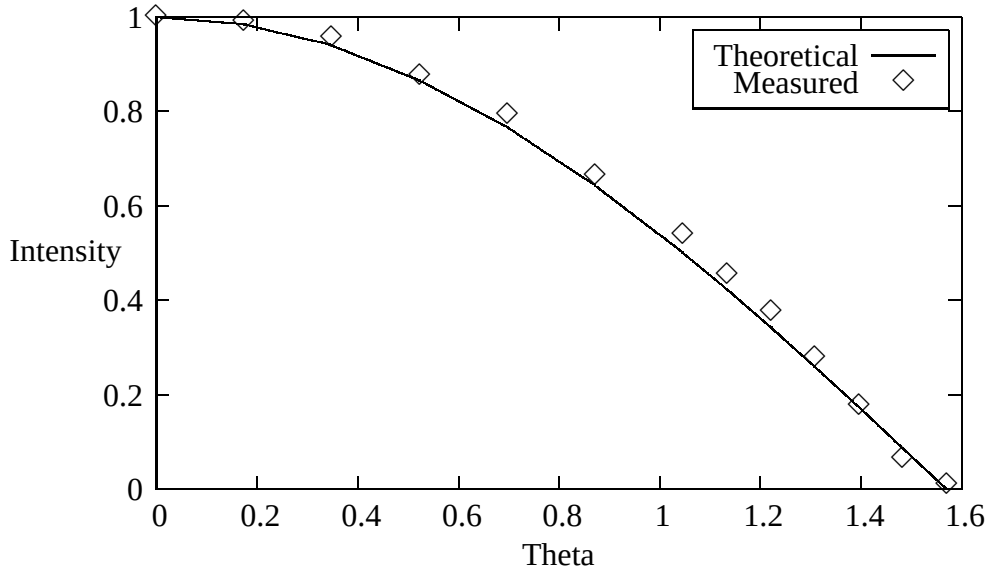


Figure B.2: The results of the brick reflectance test.

Discussion

The results indicate that the reflectance of the brick is not directly compatible with the diffuse reflectance function, as the measured reflecting light from the brick is slightly more powerful than the expected light using Lamberts cosine law. An explanation of this behaviour is that the light used in the setup is not a point light source, as is the assumption of the Lambertian reflection function. This means that the angle at which the brick is affected by the light is widened. Another aspect, that is a source to deviations is the roughness of the brick, which entails that more light will be reflected from some parts of the measured surface, than if it was completely planar. Taking these differences between the real setup and the theoretical into account, the reflectance of the brick can be said to be a diffuse reflector.

This test is not a complete test that ultimately verifies if masonry is diffuse, such a test would entail testing the surface from many angles with the light being positioned at as many different angles, this process can be performed using a Goniometer. The results of this test hints that the reflectance of the brick is in the tested setup acting as a diffuse reflector.

APPENDIX B. TESTING THE DIFFUSE REFLECTANCE OF REGULAR MASONRY

APPENDIX C

Light estimation method verification test data tables

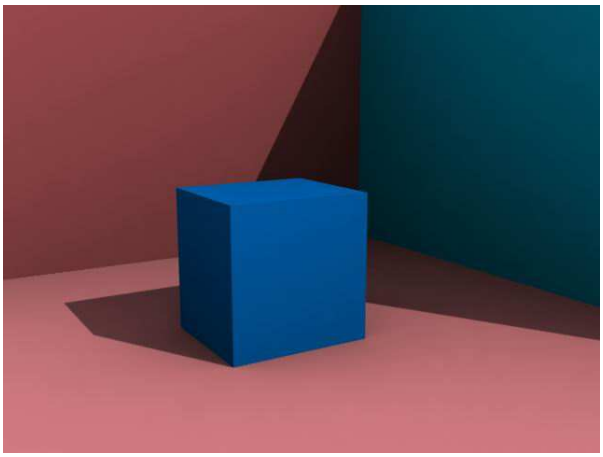


Figure C.1: The sample points are plotted into the simulated online input image.

Table C.1 displays the data collected at the various sample points.

Sample	**Online** P	**Albedo** ρ_d	**Weight** c	**Shadow** s	**Normal** $\vec{n}$
1	$\frac{146}{255}, \frac{70}{255}, \frac{72}{255}$	$\frac{117}{255}, \frac{56}{255}, \frac{56}{255}$	$\frac{179}{255}$	1	-1,0,0
2	$\frac{116}{255}, \frac{55}{255}, \frac{57}{255}$	$\frac{117}{255}, \frac{56}{255}, \frac{56}{255}$	$\frac{110}{255}$	1	-1,0,0
3	$\frac{44}{255}, \frac{21}{255}, \frac{22}{255}$	$\frac{117}{255}, \frac{56}{255}, \frac{56}{255}$	$\frac{101}{255}$	0	-1,0,0
4	$\frac{0}{255}, \frac{38}{255}, \frac{48}{255}$	$\frac{0}{255}, \frac{126}{255}, \frac{150}{255}$	$\frac{82}{255}$	0	0,0,-1
5	$\frac{54}{255}, \frac{32}{255}, \frac{33}{255}$	$\frac{152}{255}, \frac{89}{255}, \frac{89}{255}$	$\frac{96}{255}$	0	0,1,0
6	$\frac{193}{255}, \frac{113}{255}, \frac{116}{255}$	$\frac{152}{255}, \frac{89}{255}, \frac{89}{255}$	$\frac{186}{255}$	1	0,1,0
7	$\frac{82}{255}, \frac{48}{255}, \frac{50}{255}$	$\frac{152}{255}, \frac{89}{255}, \frac{89}{255}$	$\frac{144}{255}$	0	0,1,0
8	$\frac{0}{255}, \frac{79}{255}, \frac{99}{255}$	$\frac{0}{255}, \frac{126}{255}, \frac{150}{255}$	$\frac{168}{255}$	0	0,0,-1
9	$\frac{0}{255}, \frac{90}{255}, \frac{164}{255}$	$\frac{0}{255}, \frac{69}{255}, \frac{122}{255}$	$\frac{197}{255}$	1	0,1,0
10	$\frac{0}{255}, \frac{39}{255}, \frac{73}{255}$	$\frac{0}{255}, \frac{69}{255}, \frac{122}{255}$	$\frac{154}{255}$	0	0,0,-1
11	$\frac{0}{255}, \frac{40}{255}, \frac{74}{255}$	$\frac{0}{255}, \frac{69}{255}, \frac{122}{255}$	$\frac{156}{255}$	0	0,0,-1
12	$\frac{0}{255}, \frac{70}{255}, \frac{128}{255}$	$\frac{0}{255}, \frac{69}{255}, \frac{122}{255}$	$\frac{118}{255}$	1	-1,0,0
13	$\frac{0}{255}, \frac{85}{255}, \frac{154}{255}$	$\frac{0}{255}, \frac{69}{255}, \frac{122}{255}$	$\frac{174}{255}$	1	-1,0,0
14	$\frac{31}{255}, \frac{14}{255}, \frac{15}{255}$	$\frac{117}{255}, \frac{56}{255}, \frac{56}{255}$	$\frac{71}{255}$	0	-1,0,0
15	$\frac{158}{255}, \frac{92}{255}, \frac{94}{255}$	$\frac{152}{255}, \frac{89}{255}, \frac{89}{255}$	$\frac{125}{255}$	1	0,1,0

Table C.1: Data for test image 1

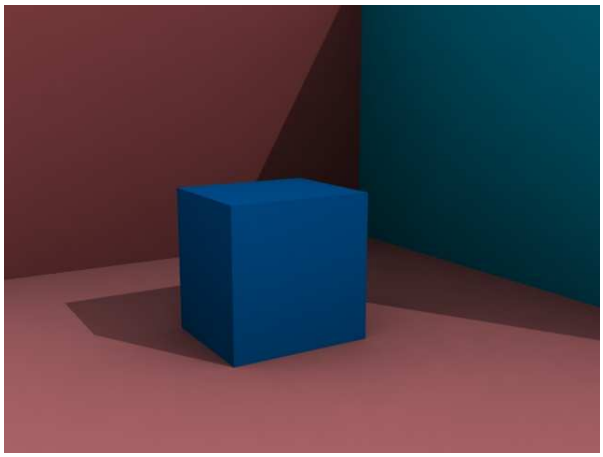


Figure C.2: The second test image. Direct light reduced to half.

Table C.2 displays the data collected in the second image, at the same sample points as the first.

Sample	**Online** P	**Albedo** ρ_d	**Weight** c	**Shadow** s	**Normal** $\vec{n}$
1	$\frac{112}{255}, \frac{53}{255}, \frac{55}{255}$	$\frac{117}{255}, \frac{56}{255}, \frac{56}{255}$	$\frac{179}{255}$	1	-1,0,0
2	$\frac{82}{255}, \frac{39}{255}, \frac{40}{255}$	$\frac{117}{255}, \frac{56}{255}, \frac{56}{255}$	$\frac{110}{255}$	1	-1,0,0
3	$\frac{44}{255}, \frac{21}{255}, \frac{22}{255}$	$\frac{117}{255}, \frac{56}{255}, \frac{56}{255}$	$\frac{101}{255}$	0	-1,0,0
4	$\frac{0}{255}, \frac{38}{255}, \frac{48}{255}$	$\frac{0}{255}, \frac{126}{255}, \frac{150}{255}$	$\frac{82}{255}$	0	0,0,-1
5	$\frac{54}{255}, \frac{32}{255}, \frac{33}{255}$	$\frac{152}{255}, \frac{89}{255}, \frac{89}{255}$	$\frac{96}{255}$	0	0,1,0
6	$\frac{149}{255}, \frac{87}{255}, \frac{90}{255}$	$\frac{152}{255}, \frac{89}{255}, \frac{89}{255}$	$\frac{186}{255}$	1	0,1,0
7	$\frac{82}{255}, \frac{48}{255}, \frac{50}{255}$	$\frac{152}{255}, \frac{89}{255}, \frac{89}{255}$	$\frac{144}{255}$	0	0,1,0
8	$\frac{0}{255}, \frac{79}{255}, \frac{99}{255}$	$\frac{0}{255}, \frac{126}{255}, \frac{150}{255}$	$\frac{168}{255}$	0	0,0,-1
9	$\frac{0}{255}, \frac{70}{255}, \frac{129}{255}$	$\frac{0}{255}, \frac{69}{255}, \frac{122}{255}$	$\frac{197}{255}$	1	0,1,0
10	$\frac{0}{255}, \frac{39}{255}, \frac{73}{255}$	$\frac{0}{255}, \frac{69}{255}, \frac{122}{255}$	$\frac{154}{255}$	0	0,0,-1
11	$\frac{0}{255}, \frac{40}{255}, \frac{74}{255}$	$\frac{0}{255}, \frac{69}{255}, \frac{122}{255}$	$\frac{156}{255}$	0	0,0,-1
12	$\frac{0}{255}, \frac{50}{255}, \frac{92}{255}$	$\frac{0}{255}, \frac{69}{255}, \frac{122}{255}$	$\frac{118}{255}$	1	-1,0,0
13	$\frac{0}{255}, \frac{65}{255}, \frac{119}{255}$	$\frac{0}{255}, \frac{69}{255}, \frac{122}{255}$	$\frac{174}{255}$	1	-1,0,0
14	$\frac{31}{255}, \frac{14}{255}, \frac{15}{255}$	$\frac{117}{255}, \frac{56}{255}, \frac{56}{255}$	$\frac{71}{255}$	0	-1,0,0
15	$\frac{114}{255}, \frac{67}{255}, \frac{69}{255}$	$\frac{152}{255}, \frac{89}{255}, \frac{89}{255}$	$\frac{125}{255}$	1	0,1,0

Table C.2: Data for test image 2

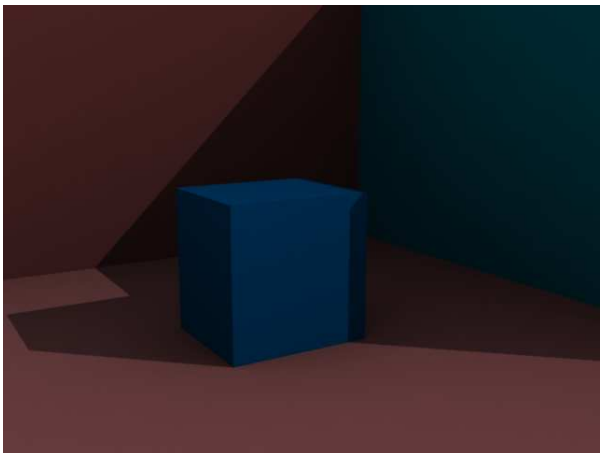


Figure C.3: The third test image. Changed light direction, and ambient light reduced to half.

Sample	**Online** P	**Albedo** ρ_d	**Weight** c	**Shadow** s	**Normal** $\vec{n}$
1	$\frac{67}{255}, \frac{32}{255}, \frac{33}{255}$	$\frac{117}{255}, \frac{56}{255}, \frac{56}{255}$	$\frac{179}{255}$	1	-1,0,0
2	$\frac{52}{255}, \frac{25}{255}, \frac{25}{255}$	$\frac{117}{255}, \frac{56}{255}, \frac{56}{255}$	$\frac{110}{255}$	1	-1,0,0
3	$\frac{22}{255}, \frac{10}{255}, \frac{11}{255}$	$\frac{117}{255}, \frac{56}{255}, \frac{56}{255}$	$\frac{101}{255}$	0	-1,0,0
4	$\frac{0}{255}, \frac{19}{255}, \frac{24}{255}$	$\frac{0}{255}, \frac{126}{255}, \frac{150}{255}$	$\frac{82}{255}$	0	0,0,-1
5	$\frac{27}{255}, \frac{16}{255}, \frac{16}{255}$	$\frac{152}{255}, \frac{89}{255}, \frac{89}{255}$	$\frac{96}{255}$	0	0,1,0
6	$\frac{92}{255}, \frac{53}{255}, \frac{55}{255}$	$\frac{152}{255}, \frac{89}{255}, \frac{89}{255}$	$\frac{186}{255}$	1	0,1,0
7	$\frac{41}{255}, \frac{24}{255}, \frac{25}{255}$	$\frac{152}{255}, \frac{89}{255}, \frac{89}{255}$	$\frac{144}{255}$	0	0,1,0
8	$\frac{0}{255}, \frac{39}{255}, \frac{49}{255}$	$\frac{0}{255}, \frac{126}{255}, \frac{150}{255}$	$\frac{168}{255}$	0	0,0,-1
9	$\frac{0}{255}, \frac{43}{255}, \frac{78}{255}$	$\frac{0}{255}, \frac{69}{255}, \frac{122}{255}$	$\frac{197}{255}$	1	0,1,0
10	$\frac{0}{255}, \frac{19}{255}, \frac{37}{255}$	$\frac{0}{255}, \frac{69}{255}, \frac{122}{255}$	$\frac{154}{255}$	0	0,0,-1
11	$\frac{0}{255}, \frac{20}{255}, \frac{37}{255}$	$\frac{0}{255}, \frac{69}{255}, \frac{122}{255}$	$\frac{156}{255}$	0	0,0,-1
12	$\frac{0}{255}, \frac{15}{255}, \frac{28}{255}$	$\frac{0}{255}, \frac{69}{255}, \frac{122}{255}$	$\frac{118}{255}$	0	-1,0,0
13	$\frac{0}{255}, \frac{39}{255}, \frac{71}{255}$	$\frac{0}{255}, \frac{69}{255}, \frac{122}{255}$	$\frac{174}{255}$	1	-1,0,0
14	$\frac{15}{255}, \frac{7}{255}, \frac{7}{255}$	$\frac{117}{255}, \frac{56}{255}, \frac{56}{255}$	$\frac{71}{255}$	0	-1,0,0
15	$\frac{35}{255}, \frac{20}{255}, \frac{21}{255}$	$\frac{152}{255}, \frac{89}{255}, \frac{89}{255}$	$\frac{125}{255}$	0	0,1,0

Table C.3: Test data for the third image

APPENDIX C. LIGHT ESTIMATION METHOD VERIFICATION TEST DATA TABLES

APPENDIX D

Calculating the Sun's Position

This appendix describes the formulae used for computing the Sun's position on the sky.

Source: [Schlyter, 2005]
Today it is common knowledge that the earth orbits the sun, and not vice versa. In the following calculations, however for simplicity the orbital elements in this section are valid for the sun's apparent orbit around the earth.

The time scale used in the following calculations is the "day number" from 2000 Jan 0.0 which equals 1999 Dec 31.0, it is denoted, J_{2000} in the following sections.

First step is to calculate the longitude of perhelion, w, which is the longitude, relative to the orbital point where the earth is closest to the sun. Furthermore, the orbital eccentricity, e, which is a measure of how much the orbit deviates from e perfect circle, and the mean anomaly, M, which is the time since the last crossing of perhelion, are calculated.

$$w = 282.9404^\circ + 4.70935 \cdot 10^{-5\circ} \cdot J_{2000} \quad \text{(D.1)}$$
$$e = 0.016709 - 1.151 \cdot 10^{-9} \cdot J_{2000} \quad \text{(D.2)}$$
$$M = 356.0470^\circ + 0.9856002585^\circ \cdot J_{2000} \quad \text{(D.3)}$$

Next step is to find the axial inclination in relation to the orbital plane, this is called the obliquity of the ecliptic

$$obliquity = 23.4393^\circ - 3.563 \cdot 10^{-5\circ} \cdot J_{2000} \quad \text{(D.4)}$$

The Sun's mean longitude, L, is calculated:

$$L = w + M \quad \text{(D.5)}$$

A first approximation to the eccentric anomaly is calculated,

$$E = M \cdot e \cdot sin(M) \cdot (1 + e \cdot cos(M)) \quad \text{(D.6)}$$

The Sun's rectangular coordinates, (x_{sun}, y_{sun}), in the plane of the ecliptic with the X axis pointing towards the perhelion, are calculated and converted to distance and true anomaly, r, υ. True anomaly is one of six Keplerian elements, which locates a satellite on an orbit. True anomaly is the angular distance from a satellite from perhelion, as seen from the centre of the Earth.

$$x_{sun} = r \cdot cos(V) = cos(E) - e \quad \text{(D.7)}$$
$$y_{sun} = r \cdot sin(V) = sin(E) \cdot \sqrt{1 - e} \quad \text{(D.8)}$$

$$r = \sqrt{x + y} \quad \text{(D.9)}$$
$$\upsilon = arctan\left(\tfrac{y}{x}\right) \quad \text{(D.10)}$$

From this the longitude of the Sun is calculated:

$$long_{sun} = \upsilon + w \tag{D.11}$$

Given these results we compute the Sun's ecliptic rectangular coordinates, these are then rotated to equatorial coordinates and then the Sun's Right Ascension, α, and Declination, δ are computed.

$$\begin{aligned} x_{ec.sun} &= r \cdot cos(long_{sun}) && \text{(D.12)} \\ y_{ec.sun} &= r \cdot sin(long_{sun}) && \text{(D.13)} \\ z_{ec.sun} &= 0 && \text{(D.14)} \end{aligned}$$

Rotate it by the obliquity:

$$\begin{aligned} x_{equatorial} &= x_{ec.sun} && \text{(D.15)} \\ y_{equatorial} &= y_{ec.sun} \cdot cos(obliquity) + z_{ec.sun} \cdot sin(obliquity) && \text{(D.16)} \\ z_{equatorial} &= y_{ec.sun} \cdot sin(obliquity) + z_{ec.sun} \cdot cos(obliquity) && \text{(D.17)} \end{aligned}$$

These are converted to Right Ascension, α, and Declination, δ. Right Ascension is a celestial coordinate corresponding to longitude in the equatorial celestial coordinate system. The null coordinate being defined bu the vernal equinox, which is one of the two points where the celestial equator intersects the ecliptic. Declination is, similarly the celestial coordinate corresponding to latitude.

$$\begin{aligned} \alpha &= arctan\left(\frac{y_{equatorial}}{x_{equatorial}}\right) && \text{(D.18)} \\ \delta &= arcsin\left(\frac{z_{equatorial}}{r}\right) && \text{(D.19)} \end{aligned}$$

From these computations we can compute the altitude and azimuth of the Sun at a given location on Earth. This involves computing the Sidereal Time. Sidereal Time is time measured as the apparent motion of fixed stars around the Earth, in opposition to solar time in which a day equals 24 hours, a sidereal day equals 23 hours and 56 minutes.

$$sidereal = GMST_0 + \text{UTC} + \frac{\text{Longitude}}{15} \tag{D.20}$$

Where $GMST_0$ is the sidereal time at the Greenwich meridian at 00:00, and UTC is the Universal Time Coordinated, which is equal to Greenwich time. Longitude is the terrestrial longitude in degrees, where western longitude is negative and eastern positive.

$GMST_0$ is calculated from the Sun's mean longitude, L

$$GMST_0 = \frac{L + 180^\circ}{15} \tag{D.21}$$

Next the hour angle of the sun is computed. An object's hour angle indicates how much sidereal time has passed since the object was on the local meridian (i.e. in the south). It is

also the angular distance between the object and the meridian, measured in hours (1 hour = 15 degrees).

$$HA_{sun} = sidereal - \alpha \tag{D.22}$$

The Sun's hour angle is converted to a rectangular coordinate system with the X-axis pointing towards the celestial equator in the south, the Y-axis to the horizon in the west, and the Z-axis to the north celestial pole.

$$\begin{aligned} x_{HA} &= cos(HA_{sun}) \cdot cos(\delta) && \text{(D.23)} \\ y_{HA} &= sin(HA_{sun}) \cdot cos(\delta) && \text{(D.24)} \\ z_{HA} &= sin(\delta) && \text{(D.25)} \end{aligned}$$

This coordinate system is again rotated along an east-west going axis so that the Z-axis points to the zenith.

$$\begin{aligned} x_{HA.rot} &= x_{HA} \cdot sin(\text{Latitude}) - z_{HA} \cdot cos(\text{Latitude}) && \text{(D.26)} \\ y_{HA.rot} &= y_{HA} && \text{(D.27)} \\ z_{HA.rot} &= x_{HA} \cdot cos(\text{Latitude}) - z_{HA} \cdot cos(\text{Latitude}) && \text{(D.28)} \end{aligned}$$

From this we can compute the azimuth and altitude of the sun.

$$\begin{aligned} azimuth &= arctan\left(\frac{y_{HA.rot}}{x_{HA.rot}}\right) + 180^\circ && \text{(D.29)} \\ altitude &= arcsin\left(z_{HA.rot}\right) && \text{(D.30)} \\ &= arctan\left(\frac{z_{HA.rot}}{\sqrt{x_{HA.rot}^2 + y_{HA.rot}^2}}\right) && \text{(D.31)} \end{aligned}$$

APPENDIX D. CALCULATING THE SUN'S POSITION

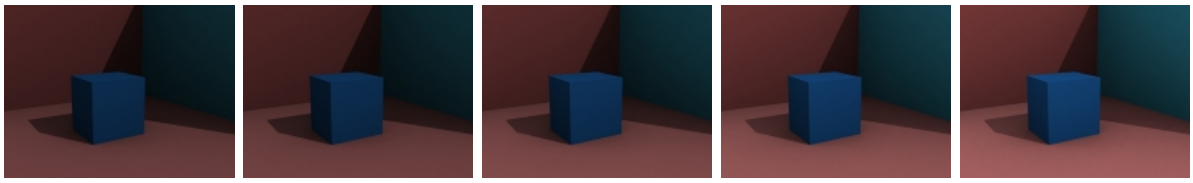
APPENDIX E

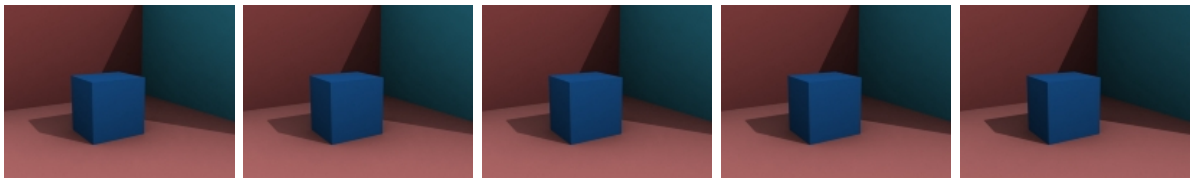
Test Journals

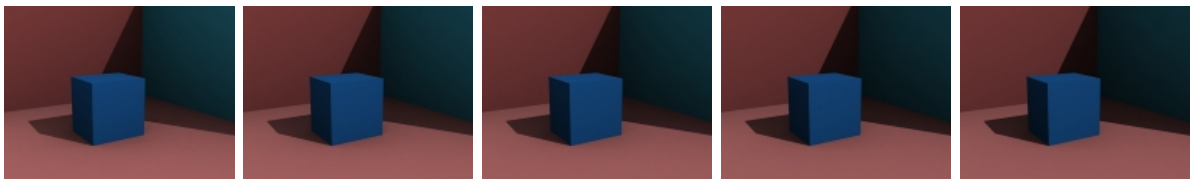
Simulated data

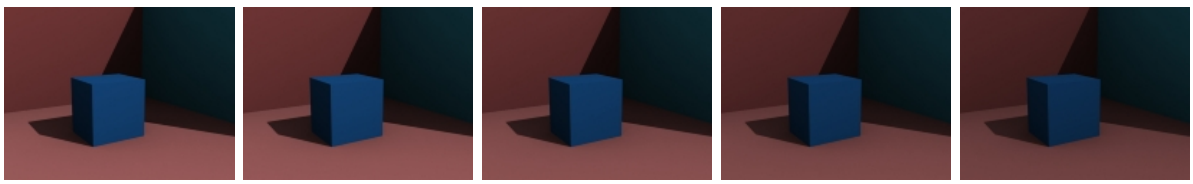
E.0.2 Estimation of direct and ambient radiance on animated image sequence

For the full data set in both AVI and JPEG formats, please see the accompanying CD, in the directory `/testdata/test1/`.

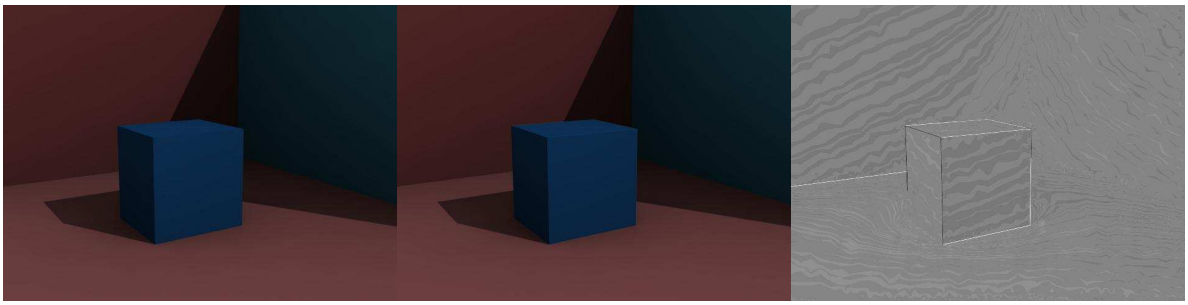

Frame 0 Frame 5 Frame 10 Frame 15 Frame 20


Frame 25 Frame 30 Frame 35 Frame 40 Frame 45


Frame 50 Frame 55 Frame 60 Frame 65 Frame 70


Frame 75 Frame 80 Frame 85 Frame 90 Frame 95

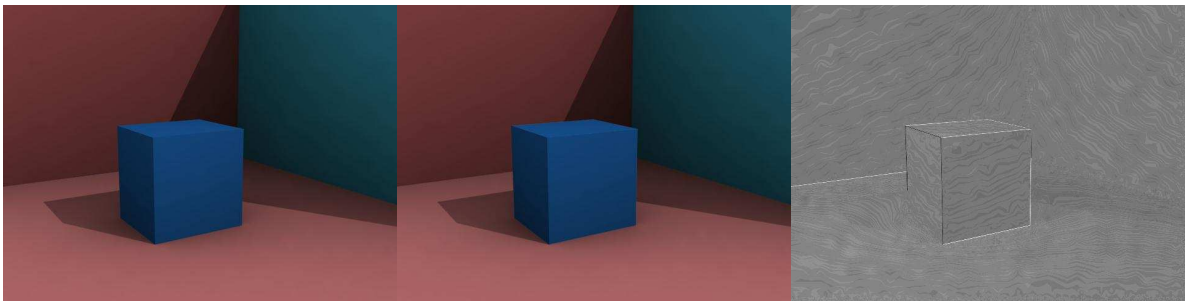
E.0.3 Reconstructing illumination in an image series from estimated illumination values



Original (Frame 0) | Reconstructed | Difference

L_d | L_d | Mean error: -0.119

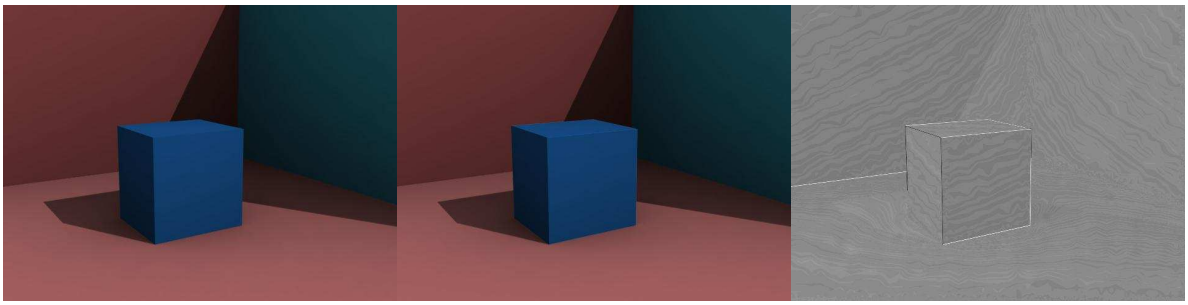
L_a | L_a | Std. dev.: 3.4



Original (Frame 25) | Reconstructed | Difference

L_d | L_d | Mean error: 0.274

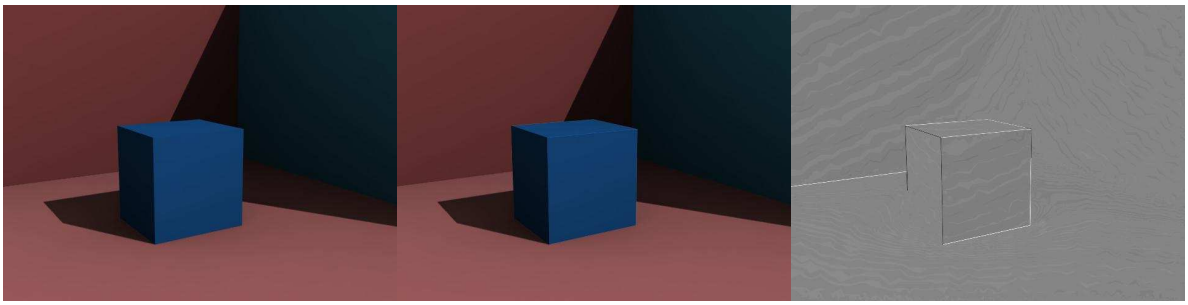
L_a | L_a | Std. dev.: 4.6



Original (Frame 50) | Reconstructed | Difference

L_d | L_d | Mean error: 0.962

L_a | L_a | Std. dev.: 2.8



Original (Frame 75) | Reconstructed | Difference

L_d | L_d | Mean error: -0.23

L_a | L_a | Std. dev.: 3.7

E.0.4 Estimating direct and ambient radiance in a time lapse movie sequence

Frame 10 Frame 20 Frame 30 Frame 40 Frame 50

Frame 60 Frame 70 Frame 80 Frame 90 Frame 100

Frame 110 Frame 120 Frame 130 Frame 140 Frame 150

Frame 160 Frame 170 Frame 180 Frame 190 Frame 200

Frame 210 Frame 220 Frame 230 Frame 240 Frame 250

Frame 260 Frame 270 Frame 280 Frame 290 Frame 300

Frame 310 Frame 320 Frame 330 Frame 340 Frame 350

E.0.5 Shading objects from an environment map based on a time lapse movie sequence


Frame 0 Frame 10 Frame 20 Frame 30 Frame 40


Frame 50 Frame 60 Frame 70 Frame 80 Frame 90


Frame 100 Frame 110 Frame 120 Frame 130 Frame 140


Frame 150 Frame 160 Frame 170 Frame 180 Frame 190


Frame 200 Frame 210 Frame 220 Frame 230 Frame 240


Frame 250 Frame 260 Frame 270 Frame 280 Frame 290


Frame 300 Frame 310 Frame 320