Virtual Phototgraphic Relighting

Jakob Skov-Pedersen

June 13, 2005

AALBORG UNIVERSITY

Department of Computer Science

Title: Virtual Phototgraphic Relighting

Theme: Interactive computer graphics and computer vision

Project period: September 2nd 2004 - June 13th, 2005

Term: F9-10S 2004-2005

Group: d541a/d641a

Author:
Jakob Skov-Pedersen

Supervisors:
Claus Brøndgaard Madsen
Olav Bangsø

Number printed: 7

Report pages: 53

Appendix pages: 7

Total page count: 61

Ended: June 13th, 2005

Abstract:

This thesis explores relighting of a single photograph using a computer.
The method explored is a three step process. The first step attempts to reproduce the original illumination, which is then used to neutralize the illumination of the photograph. The modulated photograph now represent the texture of the scene.
After neutralization of the original lighting a new illumination is computed, and the final step modulates the texture photograph using this new illumination.
Input to the system is the geometry of the scene, and an High Dynamic Range(HDR)photo.
The first step is processed using a global illumination simulation. This simulator can be in the form of a ray tracer or a radiosity solver. Several approaches to handling lighting is tried in this step.
Finally a new illumination is rendered using graphics hardware, and used to modulate the texture photo in real-time. The user can then interactively modify light-sources.

AALBORG UNIVERSITET

Institut for Datalogi

Titel: Virtual Phototgraphic Relighting

Tema: Interaktiv computer grafik og computer vision

Projektperiode: 2. september 2004 - 13. juni 2005

Semester: F9-10S 2003

Gruppe: d541a/d641a

Forfatter:
Jakob Skov-Pedersen

Vejledere:
Claus Brøndgaard Madsen
Olav Bangsø

Oplagstal: 7

Rapportens sideantal: 53

Appendiks sideantal: 7

Total sideantal: 61

Afsluttet: 13. juni 2005

Synopsis:

Dette speciale omhandler relighting af et fotografi via en computer. Den brugte metode har tre steps.
Det første step forsøger at beregne den originale belysning, som derefter er brugt til at fjerne belysningen i billedet. Det modulerede billede indeholder nu kun overfladeegenskaberne fra scenen.
Efter Den originale belysning er fjernet beregner det sidste step en ny belysning og kombinerer den og overfladeegenskaberne til et nyt billede med en ny belysning.
Systemets input er scenes geometri og et High Dynamic Range(HDR) foto. Det første step bliver beregnet via Global Illumination simulering i form af en raytracer eller radiosity solver. Der er brugt flere forskellige metoder til at håndtere belysningen i dette step.
Den nye belysning er beregnet i real-time ved hjælp af grafikkortet. Brugeren kan derfor interaktivt ændre på lys-kildernes placering.

Preface

This thesis is the result of a project running from september 2nd, 2004 to june 13th. 2005. It is submitted for approval for the degree of M. Sc. In engineering, software.

The purpose of the master thesis is to independently complete a project detailing empirical and/or theoretic treatment of problems in connection with the chosen theme on a scientiffic level. The theme for this project is "Interactive computer graphics and computer vision".

While working with the project detailed in this thesis I was visiting University Of California, San Diego. I would like to thank Henrik Wann Jensen, and the staff of the pixel lab at UCSD for help and ideas during my visit. Further I would like to thank my girlfriend Jeanette for help in finishing this thesis, and for putting up with the long work hours.

In the report all literature references have been composed with author and year in square brackets - e.g. [Kaj86]. References to books also has a page range for easy location. The position of the reference determines which specific part the reference relates to. If the reference is positioned before a period, it relates to the previous sentence, otherwise if the reference is positioned after a period the reference related to the previous section.

The enclosed CD contains source code for the software developed during this project. Also for reference a pdf of the report along with copies of all the images is also included.

The focus of the report is the analysis and design of the problem in preference to the implementation, according to the study regulations.

Aalborg, June 13th, 2005

Jakob Skov-Pedersen

Contents

Chapter 1

Introduction

This chapter provides an introduction to relighting and mixed reality, which is the main theme of the project. In addition important aspects that tie into relighting are introduced.

1.1 Motivation

Relighting is the process of modifying image data by modification of the illumination parameters. Using relighting to alter photographs was originally used in darkrooms by shadowing parts of the picture for periods during exposure of a copy [Wil]. This technique allows lightning or darkening areas of a photograph to make sure everything is within visible contrast. In this context darkening is also referred to as dodging, and lightening as burning. This technique has since been extended for use in digital systems, which will usually give much finer control over the process .

The dodge and burn processes give the artist the ability to lighten or darken parts of the image, but they will not account for the physical nature of light. This means that the artist must have some knowledge of how to alter the image so that the image is still believable to the viewer. Because of this the method is mostly used as a retouching technique to bring out details which where not otherwise visible. If what is desired is a total alteration of the lighting scheme for instance adding a spotlight, then it becomes more important that the methods adheres to light physical nature. Here the computer comes more to play, since it can be used to simulate the way light interacts with a scene through renderings, it can be used to alter the lighting scheme of a scene in a physically based manner.

Physically based relighting has applications in e.g. lighting scheme design, filming, or augmented reality. When designing a lighting scheme for a house it is beneficial to use photos of the house, and then realistically insert different lamps. In filming, and augmented reality relighting is often used to ensure seamless integration of virtual and real elements.

Using a computer for relighting falls within the sphere of mixed reality, which encompass methods that involve both real and computer synthesized elements.

1.2 Mixed reality

Mixed reality is used to refer to any visualization combining computer synthesized graphics, and real-world recorded imagery. The ultimate goal of mixed reality systems is to make the integration of synthetic and real data so seamless that it is impossible to tell what is real and what is synthetic. Altering of illumination in a photograph using a computer is an area of this field.

1.2.1 The mixed reality continuum

A common taxonomy used for classifying mixed reality methods is the mixed reality continuum [MK94]. Figure 1.1 shows a simplified scale for the continuum, where the two extreme cases are totally real and totally synthetic. Towards the synthetic end of the spectrum methods are often referred to as augmented virtuality. Methods nearer the real end are commonly referred to as augmented reality.

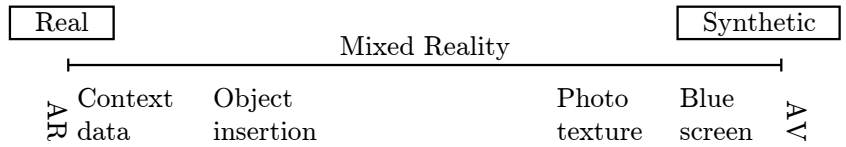


Figure 1.1: The mixed reality continuum. Left side is augmented reality, right is augmented virtuality.

An example of *augmented virtuality* (AV) is such methods as using photographs for textures in a virtual environment. Such a world will be predominantly synthetic, but some surfaces will be extracted from recorded real data. Another example is blue-screen work where acting is recorded an introduced to a synthetic world.

Under *augmented reality* (AR) are such systems as insertion of a virtual object into a real scene, as often used in the film industry. Or superimposing of contextual data as e.g. stock quotes in a newscast. Generally these effects use predominantly the data recorded, and make additions or modulations. Photographic relighting using a computer also belongs to this category, since it is mainly a modulation of input data.

1.2.2 Relighting

Photographic relighting is the general process of taking one image, and transforming this image into another by altering the composition of lights. There are three major categories of relighting methods.

Empirical relighting: The user empirically lightens and darkens an image in select regions to achieve the desired look of the image. See also section 1.1. This type of relighting also encompass relighting where a non-physical model is used either for analysis or computing new lighting. For example [OCDD01] assume repetitive surfacetextures, and thus separate anything that is not part of the pattern as the initial illumination.

Physically based relighting: Using the physical properties of the scene a new illumination is solved by the computer. This attempts to simulate what would have happened had the

original photograph been taken with the different lighting. See also section 1.1. Research that have followed this approach: [YDMH99], [LDR00], and [GHH01].

Interpolative relighting: By taking a series of pictures with different lighting conditions, and perhaps different views, it is possible to produce new images using an interpolation between the recorded data. This method will combine the input images so as to produce images with new lighting conditions.

This thesis is predominantly about physically based photographic relighting, and as such the remainder will treat relighting as only being physically based. In a physically based relighting situation the image is relighted while accounting for the physical properties of the original scene in the motive.

The physical properties of the scene must either be recorded when the image is captured, or inferred by the relighting program. The image recording process will place some constraints on what data can be inferred, i.e. inferring some data will exclude other data from being inferred. For example if only a single photograph is captured, then it is not possible to distinguish between a specular highlight and a white spot on the surface of the object. Because of this either multiple images are needed or some constraint must be placed on surfaces, e.g. any surface must have homogeneous material parameters, or all surfaces are considered diffuse.

The general process used for physically based relighting can be separated into the phases seen in figure 1.2.

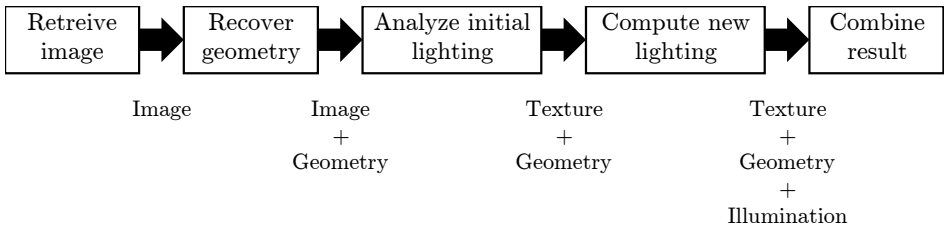


Figure 1.2: Block diagram for a general relighting system.

Retrieve Image. During this phase the image is retrieved from the motive. It could e.g. be recorded using a digital camera, or scanned from negatives or prints. Most relighting systems requires high dynamic range images because the relative itensity of each pixel is important. This process is well understood and treated by [DM97].

Recover geometry. In the second phase the geometry is recovered either from the recorded images[DTM96], or by a separate process such as a geometry scanner[Wikc]. This phase is beyond the scope of this thesis, and the geometry must thus be supplied by the user.

Analyze initial lighting. Here the distribution of light in the scene is analyzed and calculated. This is treated in chapter 3 on page 21 and 4.

Compute new lighting. Using a renderer a new distribution of light in the scene is calculated. This is treated in chapter 5 on page 43.

Combine result. Finally the geometry, the old lighting, the new lighting, and the original photograph is combined to yield a new image in which the lighting conditions have been altered. Some methods have this step implicitly included in the other steps for efficiency or convenience reasons. This is treated as a compositing problem[Wikb] in chapter 5 on page 43.

1.3 Image based methods

Image based methods use a 2d grid of values as a seed for a computation. In computer graphics image based methods exist both for modeling, lighting, and rendering.

Most mixed reality systems employ some form of image based methods. This comes from the mixed reality being a combination of captured data and synthetic data. Also whenever an image based method is applied to real data, the result will be within mixed reality. A such these concepts are tightly bound.

When rendering realistic computer graphics, three properties of the scene must be described (see figure 1.3). The first property is the geometry in the scene, this is a 3d description of the objects represented. Because objects are only visible when being illuminated by some light source the second property is a description of lightsources in the scene. Lastly the reason light illuminates an object, is because it interacts with the material of the object. Therefore a description is also required of how the material interacts with light.

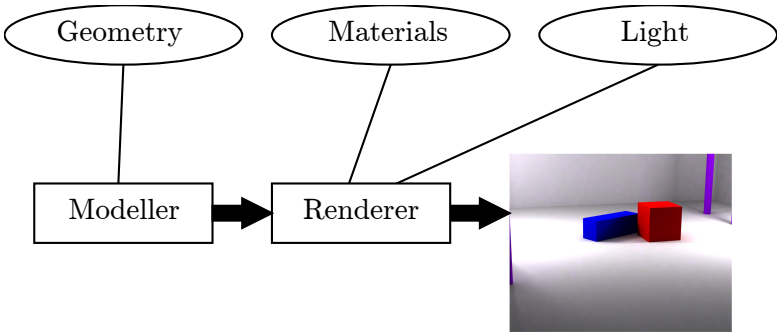


Figure 1.3: A renderer and the three types of input data.

It can be quite a challenge for a user to provide all the data required for synthesizing a realistic image. Image based methods help this problem by using photographic images to provide some of this data. A way to classify image based methods is by which of the three data-requirements they fulfill.

Image based modeling is used to recover geometry and texture from one or more images of a scene. For example [DTM96] uses a sparse set of architectural photographs to recover models with textures of buildings.

Image based lighting is using an image to illuminate an object in a scene. An often used method is capturing a light-probe by photographing a sphere. This light probe can tell something about the light arriving at a point from all directions. By combining the light probe

with the material properties of the object, a very realistic effect can be simulated. This method is often used when seamlessly composing synthetic objects into real scenes. This method is the basis of [Deb98].

Image based rendering methods aim at capturing all three data categories from images. This approach is e.g. used in interpolative relighting systems, where a series of images with different illumination is used to interpolate the new illumination as e.g. in [MDA02].

1.3.1 Image based methods and relighting

Because relighting always starts with an image, which requires an alteration of the lighting, it quite naturally relies upon image based methods. Figure 1.4 shows how image based methods can possibly be integrated into a relighting system. The lower text shows methods that can be applied to the relighting steps.

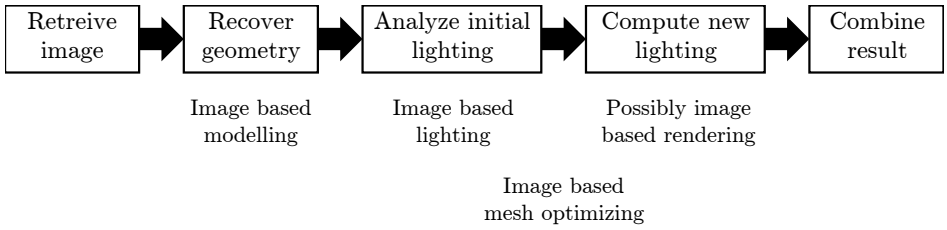


Figure 1.4: Image based methods in a relighting system.

It is possible to use Image based modeling as a method for recovering the geometry of the scene, and image based lighting for analysing the initial lighting. After finding the initial lighting Image based mesh optimation can be used to optimize the recovered mesh for better results. Image based rendering is a method for computing new lighting of the scene. This shows that Image based methods are naturaly intregrated in a relighting system.

1.4 Illumination

An illumination is what arises when a light-source is introduced to a scene. The light source will illuminate the scene, and surfaces in the scene becomes lit. The steady state distribution of light in a scene is the illumination distribution. An illumination distribution is defined by a number of light sources and a scene with which the sources interact.

In a physically based relighting system two illumination distributions are of special importance. The first one is the initial illumination distribution, which is formed by the lights affecting the scene as it was acquired, is usually analyzed from the original image which is desired re-lit. The second important illumination distribution is the new illumination distribution, which is found to reflect some modification of light sources affecting the scene.

Illumination systems often differentiate two forms of illumination, local and global illumination. Local illumination is the direct result of a number of light sources shining on a single geometric point in space. The remaining geometry is possibly used for shadows but nothing

else. Global illumination systems also account for light that hits one surface, and bounces onto another.

A renderer can be used to simulate the illumination distribution. The concept of a renderer has been shown to accurately reproduce a synthetic representation of the steady state distribution of light in a scene.

1.4.1 Renderers

A renderer is used to produce an image of a scene from a camera view. A scene has three important properties: The geometric description, material properties, and light-sources. In photographic relighting the type of renderer that is desired, is a renderer which will produce images based on physical properties. Some common physically based renderers are:

- Radiosity solvers
- Ray tracers
- Pipeline engines

Radiosity renderers are based on the mathematical finite element method. The scene is split into a number of finite elements, and an averaged illumination distribution is calculated for each element. By using elements which have a well defined relationship with regard to the property analyzed the solution may be evaluated at discrete intervals instead of as a continuous calculation over the scene. In computer illumination problems a patch is used as the basic element. A patch is a flat surface in 3d space with three or more edges. The solution obtained from a radiosity renderer is the outgoing illumination of each patch in any direction, this means that only view independent effects can be simulated. The view independence also means that two different camera definitions can utilize the same solution. The geometric input for a radiosity renderer must be in the form of a mesh of patches.

Ray tracers attempt to simulate the way rays of light is bounced in a scene. A ray tracer will trace a ray from the camera, and when this ray hits something determine how this point is affected by the lights in the scene. A local illumination ray tracer will directly calculate how each light source affects the illumination at that point, while a global illumination ray tracer attempts to determine also how illumination bouncing of other surfaces affect the current point, usually through a stochastic sampling. Ray tracers will function with any type of geometry where the intersection with a line in 3d space can be calculated. Unlike radiosity ray tracing is view dependent, and as such will allow effects like e.g. mirrors.

Pipeline engines are used for rendering graphics interactively to a computer screen. Pipeline engines use a mesh usually made from triangles. The basic components of the input mesh are called primitives. Primitives are processed one by one by the pipeline, and as such only local illumination is rendered. Because of the simplicity of the pipeline render engine it is used in real-time graphics. Also the design makes sure that it can be implemented smartly in graphics hardware, and as such allows acceleration. While many important behavioral aspects of light are not simulated with a pipeline engine, its speed enables interactive evaluation of the illumination, and as such is used when visually designing 3d scenes.

Many renderers are also based on a combination of these basic techniques. For instance some games precompute a radiosity solution for its view independent properties, and use this as a basis for illumination the scene during play.

For any renderer the complexity of the geometry will affect the time needed to render an image. Opposite of this some methods will produce more accurate solutions if the complexity of the mesh. Generally it is desirable to keep the complexity of the mesh as low as possible while still attaining a solution that is sufficiently accurate. Also using a method such as radiosity the requirements for accuracy will be different in different parts of the image.

1.4.2 Re-meshing for increased renderer accuracy

As per the last section methods such as radiosity use a mesh which must be sufficiently complex as to represent the required accuracy. However since regions where there is a shift between light and shadow require increased accuracy with such systems, the mesh used is initially created rough, and adaptively subdivided to fit requirements where needed.

Shadowing in the geometry must be paid specifically attention to when subdividing the mesh. A shadow will result in a discontinuity in the illumination solution.

A method for refining a mesh is hierarchical subdivision[HSA91]. Here, for each patch, the error introduced by the patch size is evaluated, and if the error exceeds some user threshold the mesh is divided into a number of new patches. Subdivided patches are stored using a tree hierarchy, where each level represents a subdivision. Figure 1.5 shows a hierarchical subdivision of a quad patch, and the corresponding tree structure. When computing the solution evaluations can be performed at any level in the hierarchy to achieve the most efficient calculations.

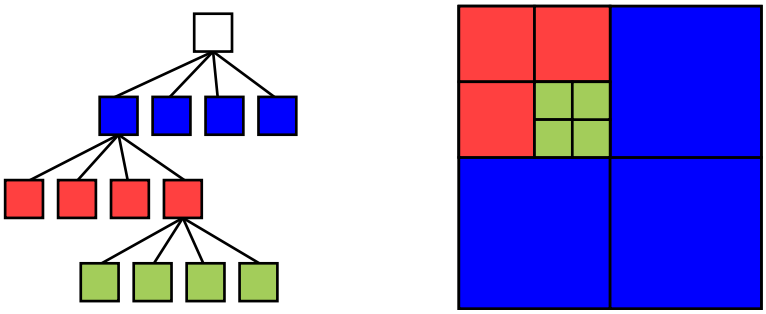


Figure 1.5: A heirarchical subdivision and the corresponding data tree.

A situation that arises in some relighting systems is only using a single viewpoint, and disregarding unseen geometry. In such situations it is desirable to only increase the complexity of the meshing that is visible, since the rest will have little effect on the outcome. Further some methods expect the acquired geometry to be incomplete, and as such will disregard any non-visible geometry. For such methods it is possible to clip the mesh before refining it to further decrease the amount of needed work.

1.4.3 Image based meshing

When refining a mesh for which there is a recorded image - a situation that arises often in mixed reality systems - it is possible to use this image to control the refining of the mesh. This method becomes an image based meshing.

The advantage of image based meshing is that a solution is already known in the recorded image. This solution will give hints as to where discontinuities in the illumination arise as a result of e.g. shadows. When attempting to use a mesh-based method to analyze the illumination in the recorded image, it is important that the mesh is fitted to the discontinuities in the image.

Image based meshing works by reading data from the image, and on the basis of that determine whether to subdivide the geometry. The geometry is adaptively refined where the image indicates a need for increased accuracy. This means the mesh can be refined to something resembling discontinuity meshing from section 1.4.2, but without the high complexity. This is possible because prior knowledge exists about the placement of shadows.

1.5 Problem description

The project is centered around relighting of a photograph. The input is a geometric model of the scene, the photograph, and a number of light probes at different locations. Since material parameters are complicated to retrieve the system will note require any such description. Materials will therefore be treated as being purely diffuse.

Because many renders are dependent on the resolution of the mesh, this project is also about producing a mesh that will allow for a good renderings in connection with photographic relighting. For this end image based methods for refining a mesh will be studied.

To facilitate interactive lighting design using the relighting method the project address real time rendering techniques especially in combination with computing a new illumination solution for relighting a photograph.

Chapter 2

Initial Analysis

This chapter starts with a treatment of Computer Generated Imagery, because it lies as a basis for virtual photographic relighting. After this the relighting process is defined.

2.1 3D computer visualization

Humans are accustomed to observe and manipulate three dimensional objects. This means for instance that if a human is to comprehend and contextualize complicated datasets a very effective visualization method is 3D renderings.

An example of the effectiveness of visualization in three dimensions is architectural design. Architectural drawings are often produced i 2d. While such drawings are good at conveying exactly how the house is to be built, it is very hard e.g. to discern whether a room feels spacious, or whether enough light enters the room through the window. Since such questions are three dimensional in their nature, they are best analyzed through such a representation.

2.1.1 Trends in 3D graphics

Engineering science relies heavily on 3D visualization systems to convey new designs and ideas. CAD systems have become the defacto method for designing new components.

A large part of the current drive for research in 3D graphics comes from the entertainment industry. At one end the video and movie industry is pushing for more realism in offline rendering systems. At the other end the gaming industry is pushing for more possibilities with interactive hardware renderings.

In the beginning of computer graphics these to industry sectors were pushing opposite directions, and creating two separate branches of research, namely photo-realistic and real-time, see figure 2.1 on the following page. However as algorithms have improved today, these two branches are starting to merge. This means that technologies are starting to migrate between the two branches. In the future these branches may even converge to a single Accelerated interactive realism branch.

Both of these branches of computer graphics influence relighting systems. A physically relighting system has two separate rendering stages, the first one is an offline step which

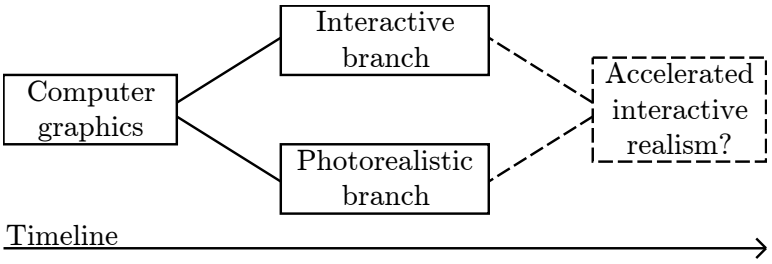


Figure 2.1: Branching in 3D graphics research.

requires a renderer which is as physically correct as possible. The other rendering stage is when computing a new lighting. This stage relies on simultaneous user interaction, and as such follows the real-time branch.

2.1.2 Interactive graphics: The rendering pipeline

A interactive rendering pipeline is composed of the following main stages [AMH02]:

Application → Geometry → Rasterizer

Most current implementations of the rendering pipeline have the geometry and rasterizer implemented in dedicated acceleration hardware. Acceleration hardware is either fixed-function or programmable, where the latter gives the user algorithmic control over the geometry and rasterrizing stages.

Setting up the scene (Application stage)

The application stage encompass what is performed by the application the user is interacting with. The application will setup the viewpoint, and send the geometry to the remaining sections of the pipeline. The geometry must be split into simple units for the remainder of the pipeline to accept. A simple geometry unit is called a primitive. Primitives are sent through the pipeline one by one to be rendered.

Preparing for viewing (Geometry stage)

The geometry stage will receive primitives from the application, and process these so the geometry becomes ready to be displayed on the screen. The geometry is transformed from the model-space of the application stage into screen-space needed for rasterizing. Also the geometry is clipped to fit within the desired view. Most calculations that are performed per vertex in the geometry is performed in this step. Fixed-function real-time implementations of the pipeline will calculate the lighting per vertex in this step as well. Later more powerful programmable pipeline implementations have enabled per pixel lighting which delays this calculation till the rasterizer.

Drawing to screen (Rasterizer stage)

This stage will handle drawing of each pixel occupied by a polygon in the frame-buffer of the rendered image. The value of each pixel is calculated individually as polygons are rendered. In the fixed-function pipeline values are simply interpolations of the values calculated per-vertex in the geometry stage. A programmable pipeline will allow application-defined calculations for each pixel, for instance lighting calculations.

2.1.3 Realistic synthesis: Photo-realistic renderers

Any physically based renderer tries to solve the rendering equation. The rendering equation describes the nature of light transport, i.e. how to find light leaving a surface from the light hitting the surface. It was derived in [Kaj86] as a common factor in most previous image synthesis work. The physical framework for describing the rendering equation comes from radiometry, which is the study of measuring light [Ash].

A basic component of light measurement is *radiant flux* (Φ). This is a measure of light flowing, and is energy over time (eqn. (2.1)).

$$\Phi = \frac{dQ}{dt} \tag{2.1}$$

Having a small surface area the radiant flux becomes radiant flux density, which is radiant flux per unit area. Figure 2.2 shows the radiant flux density leaving a surface (eqn. (2.3)), which is called *radiant exitance* (M). The radiant flux density arriving at a surface (eqn. (2.2)) is called *irradiance* (E).

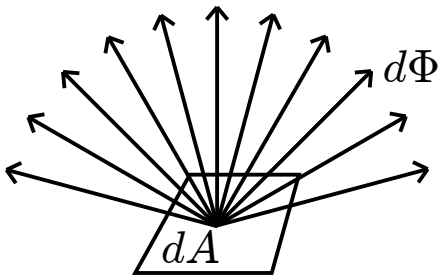


Figure 2.2: Radiant The radiant flux ($d\Phi$) leaving a surface area (dA).

$$E = \frac{d\Phi}{dA} \tag{2.2}$$

$$M = \frac{d\Phi}{dA} \tag{2.3}$$

The light leaving or arriving at a surface point does so on a hemisphere formed above the surface. This means that all the light arriving at a surface point is exactly the light that hits the area of this hemisphere. Therefore measuring all the radiant flux inbound on the surface is done by integrating over the area of the hemisphere. For this purpose a portion of the directions is called a *solid angle*, and is shown in figure 2.3 on the following page as $d\theta$.

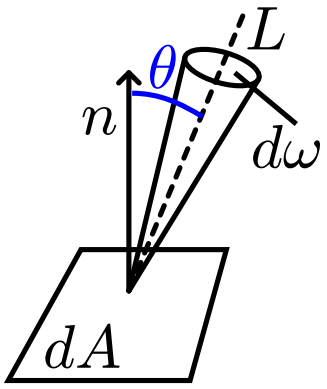


Figure 2.3: Geometric definition of radiance.

The radiant flux to or from a surface area through a solid angle is called *radiance* (L). This quantity is calculated as in eqn. (2.4) and shown in figure 2.3.

$$L = \frac{d^2\Phi}{dA(d\omega cos\theta)} \tag{2.4}$$

The rendering equation details the transport of light. The light leaving or arriving at a surface is the radiance. So the rendering equation details the relation between radiance leaving the surface and radiance arriving at the surface. The exitant radiance is a combination of the incoming radiance and the emitted radiance (eqn. (2.5) and (2.6)).

$$Outgoing\ radiance = emitted\ radiance + reflected\ radiance \tag{2.5}$$

$$L_o = L_e + L_r \tag{2.6}$$

The reflected radiance is depending on the surfaces ability to reflect light. To this end the Bidirectional Reflectance Distribution Function(BRDF) is used. The BRDF is a relation between incoming light from one direction, and outgoing light. More description of the BRDF is in section 2.2.2 on the facing page. For now it is just the function $f_r(x, \omega_i, \omega_o)$. By integrating over all incoming directions (Ω) *the rendering equation* for a particular outgoing direction (ω) becomes as in eqn. (2.7).

$$L_o(x,\omega) = L_e(x,\omega) + \int_\Omega f_r(x,\omega',\omega)L_i(x,\omega')(\omega' \cdot n)d\omega' \tag{2.7}$$

Classification of light-paths

To categorize the capabilities of rendering systems [Hec90] details a system for describing a light-path. A light-path details how light travels from a light source to the eye (or camera). In the system these are denoted by respectively an L and an E. Objects in the scene can result in either a predominantly diffuse reflection (D) or a predominantly specular reflection (S). E.g. seeing a diffuse sphere in a mirror results in this light-path: "LDSE".

By combining the light-path system with regular expressions it is possible to detail the capabilities of a particular rendering system. For example a system capable of rendering mirrors will support LDS*E paths, and a system capable of rendering caustics supports LS*DE paths. In this context the most general renderer capable of rendering any illumination will support L(D|S)*E paths. A "*" means any number of the preceding element, and a "|" means either the left or the right element.

2.2 Local illumination

Local illumination describes the light that is reflected at a point directly from a lightsource. The shading in such a system is only dependent on the lightsources and the surface parameters of items in the scene. Even though the shading is independent of other geometry, visibility cannot be since shadows are by nature a global effect in the scene.

2.2.1 Direct illumination

Direct illumination is the local response of a lightsource bouncing off a surface. Only single bounces are possible under this model, and lightpaths supported are thus L(S|D)?E.

Direct illumination is often simulated by defining a number of lightsource objects, and for each visible surface in the simulated image calculate how much each of these lightsources contribute to the overall shading of this point.

2.2.2 Surface parameters

When simulating light interactiion with an object, it is necessary to qantify in what measure the light is reflected of the objects surface. This is in most computer graphics represented by a function called the BRDF. The BRDF is a function of the incomming illumination direction, the exitant illumination direction, and the surface point, see eqn. 2.8. [OHHM02, pp. 15–44]

$$f_r(x, \omega_i, \omega_o) = \frac{dL(x, \omega_o)}{dE(x, \omega_i)} \tag{2.8}$$

Different meterials reflect light in different ways which affect the apearence of the materials. Some materials are diffuse while others are specular or something inbetween. Diffuse surfaces reflect light uniformely in all directions, while specular surfaces reflect light mostly in directions which are close to the mirror direction of the incomming direction with respect to the surface normal.

Lambertian reflection is the simplest surface model. It describes the view independent properties of most surfaces. The best way to observe teh lambertian property is to look at a diffuse cylinder. The light reflected upon such a cylinder is most intense where the surface recieves the most direct light. In fact labertian reflection is determined by the dot-product of the surface normal, and the vector towards the light. Thus the side of the cylinder closest to the light-source reflects the most light. Lambertian reflection is also often referred to as diffuse reflection, because of its soft gradual appearance. The dot-product of lambertian reflection is

already built into the rendering equation, meaning the BRDF for a diffuse reflector is constant for a particular point (2.9)

$$f_r(x) = \pi(x) \tag{2.9}$$

Phong reflection. The phong model builds upon Lambertian reflection by adding a specular lobe. The model is not based on any physical principle, but is rather designed to look nice. Because the model is very simple many realtime systems use a derivation of this model.

2.3 Global illumination

When rendering synthetic images many effects cannot be simulated by looking only at a single geometry point in space such as when simulating local illumination. Visualization systems that simulates such phenomena are refered to as global illumination renderers.

2.3.1 Diffuse indirect illumination

Diffuse indirect illumiantion is the result of a simplification of the full rendering equation. By assuming that all surfaces reflect light diffusely, radiance measure can be reduced to radiosity, which just represents diffusely reflected illumiantion. The light-paths simulated are of the form LD*E.

Radiosity is view-independent, and as such a point in space is transmitting the same amount of light to any other point visible. Thus radiosity can be defined for a patch as opposed to being defined for points like radiance. this means the rendering equation can be simplified to the radiosity formulation in eqn. (2.10). This defines teh radiosity leaving a patch x in any direction is the emittance (B_e) of x plus the light that is recieved from all other patches and reflected. V is the visibility function between patches x and x', and G is the geometric relationship.

$$B(x) = B_e(x) + \frac{\rho(x)}{\pi} \int_S B(x')V(x,x')G(x,x')dA' \tag{2.10}$$

Making the radiosity equation work over a finite number of surfaces, it can be reformulated as the linear system of equations in eqn. (2.11).

$$B_i = E_i + \rho_i \sum_j^n F_{ij} B_j \tag{2.11}$$

F_{ij} is called the form factor, and represents how much light can be transportet from one patch to another. It is defined as in eqn. (2.12).

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{V(x,x')G(x,x')}{\pi} dA_j dA_i \tag{2.12}$$

Since the formfactor is a nontrivial integral, many radiosity renderers simply sample it by sending random rays between the two patches.

2.3.2 Full global illumination

A full global illumiantion renderer simulates general light-paths of the form L(D—S)*E. Therefore the light can bounce of any number of surfaces travelling from a light-source to the eye.

2.4 Virtual Photographics Relighting

Virtual Photographics Relighting(VPR)is the process of altering the illumination in a recorded photograph using a computer.

This process has applications in e.g. lighting design, or common illumination systems[1]. Current methods fall in three categories: Inverse rendering, Relighting after light removal[JL04], and Illumination interpolation.

The following sections are a description of methods within each category.

2.4.1 Inverse rendering

Inverse rendering methods tries to fit the original data to a rendering model. Such systems return the material parameters of surfaces in the scene, which can be used to render new images.

The inverse global illumination method from [YDMH99] uses the ward BRDF [War92] as model basis. The specialized rendering equation in eqn. (2.13), which is a generalisation of ward brdf, is solve to find diffuse albedo (ρ_d), specular reflectance (ρ_s), and specular roughness (α).

$$L_{C_v P_i} = E_{C_v P_i} + \rho_d \sum_j L_{P_i A_j} F_{P_i A_j} + \rho_s \sum_j L_{P_i A_j} K(\alpha, \Theta)_{C_v P_i A_j} \tag{2.13}$$

This represents the radiance seen by a camera C_v from a point P_i. A_j is another patch in the scene, F is the analytical point-to-patch form-factor, and K is the specular term of the ward brdf. Using a combination of iterative refinement and least sqares fitting the parameters (ρ_d, ρ_s, and α) are recovered. The iterative part of the system is used to find the effect of specularities on the global illumination solution.

By making reasonable assumptions about specular highligts this system can recover diffuse albedo texture maps from a relatively small number of images.

Inverse rendering from a single image. Another approach is that of [BG01] and [BG02]. Here only a single image is required to recover the material properties of the scene. Geometry and placements of lights are supplied by the user. The methods uses a guessing approach, where eacch surface is initially considered purely diffuse, and an error in rendering compared to the original image is calculated. While the error exceeds some limit the model is progressively made more complicated.

Again the materials are fitted to the ward BRDF model. A generic global illumination renderer is used to compute the image used for error estimation. Because only a single image is used the constraint from section 1.2.2 on page 2 that not all preperties can be recovered

[1] Systems that attempt to match the illumination of a physical scene, and inserted synthetic objects

from a single applies. This means either surfaces have a complicated BRDF and homogenous material properties, or they have a diffuse BRDF and an albedo map.

2.4.2 Relighting after light removal

By removing illumiantion from an image the resulting image contains the diffuse albedo maps of the geometry. Methods in this category try to produce a plausible new illumiantion by using these albedo maps.

The light removal process will analyze the inital illumiantion with some assumptions. Methods can make assumptions on either the transport of light, or the properties of a texture. An example of an assumption on the transport of light is only diffuse reflection occurs. A property of a texture might be that the texture consists of a spacially repeating pattern.

Interactive virtual relighting. In [LDR00] a radiosity renderer is used to remove the lighting from the original images leaving unoccluded illumination textures (textures without the effect of shadows). The radiosity is based on hierarchical refinement [HSA91], but uses a texture driven subdivision scheme. To relight the image a radiosity calculation is performed for the desired change, and the unoccluded textures are modulated by this value.

(a) (b) (c)

Figure 2.4: (a) Original texture. (b) The resulting texture T_{inter}, with real occluded illumination removed, mapped onto the geometry of the real scene. (c) The final texture T_{final} after the texture-based correction (source: [LDR00]).

The use of a radiosity modulation means the acuracy of the initial illumination calculated is not crucial, only that any change to the illumination is consitently based on the original.

The hierarchical data structure is a tree where nodes are patches, and lower hierarchy nodes are subdivisions. The structure also maintains links between branches to signify radiance transfer. The radiosity algorithm used is a version extended to facilitate interactive updates [DS97]. When moving an object the only part of the hierarchy which needs to be updated is the part that the moved object mevs through. That means that by maintaing a shaft for each link in the hierarchy, and testing for intersection of this shaft and the move, it is possible to determine which parts of the hierarchy is to be updated. For a further treatment of hierarchical datastructure in radiosity see section 2.5.1 on page 18.

Because of the use of radiosity surfaces are considered diffuse.

Radiance modulation. This method has been proposed by [ML04]. It is a three-step process; where first step is rendering a radiance image using what is known about the geometry and lighting of the scene. And the second step creating a modulation radiance image which reflects a new illumination distribution. The last step is applying modulating the original image using

these two radiance images. When computing the radiance images all surfaces are perfectly white diffuse reflectors. Usign purely white refectors is possible since only the modulation of the initial and relighting radiance is needed.

Rendering the two radiance images can be performed by any available renderer, however most scenes should use a global illumination renderer to compute the initial illumiantion. The global illumination effect of colorbleeding is not regarded because of the white reflector assumption.

Texture-illuminance decoupling The approach for creating textures without illumination in [OCDD01] is based on empirical texture filtering. The assumption is that small scale changes (high spatial frequency changes) in the image are because of the texture, while large scale (low special frequency) changes are from lighting. This assumption comes from shadows often looking like big splotches in an image, while texture is fine details.

The separation is made using a plane spatial highpass filter. The filter compensates for geometric perspective to make sure that what is considered large/small scale is dependant on distance to viewer.

2.4.3 Illumination interpolation

By capturing multiple basis images of the same scene under different illumination, this can serve as a space in which new illumination distributions can be interpolated.

The free-form light stage. In [MDA02] the basis images are captured by pointing a fixed camera at an object, and manually holding a lightsource above the object in different positions while taking the pictures. To interpolate bewtween the different illuminations the position of the lightsource for each image must be known. The estimation of light-source positions is performed by plpacing four diffusely white spheres around the object. The shading of these spheres is used to determine light-source orientation.

To interpolate the lighting, an angular Vorini diagram [Wei] is used to segment the hemisphere of incomming light according to the light-source positions. Now the lightprobe, that is to be used for a relighting, is sampled in accordance with this diagram. Figure 2.5 shows how the angular voroni diagram is produced from the estimated light-source positions.

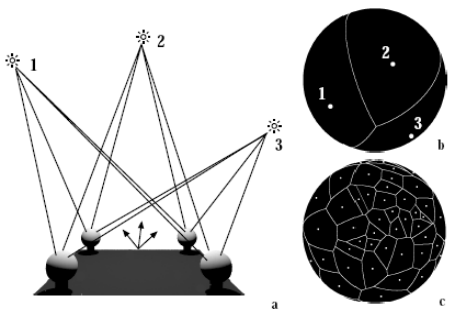


Figure 2.5: Construction of the Voronoi diagram in on the hemisphere, based on the estimated illuminant directions (source: [MDA02]).

2.5 Mesh subdivision

When calculating the initial lighting environment with a mesh-based technique, it is essential that the mesh used reflect discontinuities in the original radiance. Mesh-based techniques such as radiosity will only compute irradiance at discrete points determined by the underlying mesh. Since shadow boundaries will produce sharp changes in the initial radiance, but not in the geometry, some method is required to subdivide the geometry into a mesh, which will capture these edges.

There are two goals for the resolution of the mesh, first the total resolution of the mesh will impact the performance of any calculations performed. And second, where the incomming light changes for a surface (e.g. near shadow boundaries) it is essential that the resolution is high to allow an accurate solution. A method othen used in the area of computer science to balance these two goals is to adatively refine the solution locally where increased acuracy is required.

A very popular method for subdividing meshes in radiosity systems is hierarchical refinement, and many other methods are based upon this method.

2.5.1 Hierarchical refinement

Hierarchical refinement radiosity builds a tree hierarchy of the patches in the scene to be rendered, where each layer represents a subdivision of the mesh. An element in the mesh is normally called a patch in hierarchical subdivision radiosity.

Light transport in the hierarchy can be performed on either of the levels. That is patches that are far away will use the top level of the subdivision, so as to decrease the number of calculations. To attain this property the structure maintains a number of links across the hierarchy, which will singnify paths which radiosity can be transferred.

The algorithm starts with top-level links between full patches. Then for each link the form-factor is estimated. If the patches interact such that an estimation of the formfactors is not possible the patches are split, and the links moved a level down. A form-factor can beestimated when the distance between to patches is large compared to their sizes.

[HSA91]

2.5.2 Texture based refinement

An adaptation of hierarchical refinement radiosity specifically for use in relighting system is the scheme used by [LDR00]. A relighting system has prior knowledge about the radiance distribution in the form of a solution, which is the initial image. Therefore the approach is driven by the captured image.

Instead of looking at the form-factor to determine if a patch needs to be subdivided, the idea is to look at the original image. The idea is that neighboring patches with similar color in the image should also have similar visibility, and neighboring patches with dissimilar color should also have dissimilar visibilty. When determining visibility the system looks at visibility towards a few number of light-sources. Also a patch which is partially visible - one where there is a shadow boundary - is subdivided.

This method must have some way to determine the similarity of colors. In [LDR00] the colors are compared in CIELAB [GW02, pp. 322–323] color-space by their euclidean color

distance.

2.6 Interactive illumination design

When making lighting design it is important to know what light sources are available and how to use them. Examples of light sources are: spotlights, point lights, sphere lights and area lights. Spotlights are used to add focus on important items in the scene. Point lights, e.g. non-diffused thread bulbs, can be used for lighting larger items like tables or other furniture. Sphere lights, e.g. normal diffused light bulbs are good at hanging from the ceiling and lighting the entire room with a soft light. Area lights, like tube lights, work well when an even lighting is wanted over a defined area. For the light designer it is important to make all these light sources come together for the best lighting design.

Chapter 3

Analyzing initial illumination

Common illumination methods call for some analysis of the original illumination distribution in the seeding image. In this chapter the illumination analysis approaches is described

3.1 Model analysis

Generally analysis of captured data involves relating it to some modelanalysis!model.

When using computers to analyze a problem, the model is a mathematical formulation of the systems behavior. Examples are newtons equations for analyzing a falling object, or the rendering equation for analyzing illumination problems.

3.1.1 Analysis methods

When analysing captured data there are a number of approaches. *numerical methods* uses the computers ability to process numbers for finding approximate numerical solutions. A *simulation* can be applied to simplified data, and then the difference recorded data and the simulation is the result. Generally the difference between the two is a conceptual difference of how the problem is attacked.

3.1.2 Numerical analysis

Numerical analysis tries to fit the recorded data to the model. Fitting is performed by extracting values for model-parameters, which will yeild the recorded data. For example in [YDMH99] the model that the data is fitted against is the ward BRDF[War92]. The fitting is performed by least squares estimation. This tries to minimize an error estimate by tuning the desired parameters.

3.1.3 Simulation

In a simulation scenario, a simulation is started with starting conditions as close to what produced the recorded data as possible. The idea is then that the difference between the recorded data and the simulation result is a result of the unknown parameters of the problem. The data used to run the simulation is called the context for the analysis. The context could be things like scene geometry, or light-probes.

This is the method used in [LDR00] and [ML04]. Both uses what is known about the context to simulate an estimate of the conditions in the captured data. This estimate is used to neutralize the original conditions, so that new effects can be applied.

The next sections works with methods for performing this simulation.

3.2 A radiosity simulation

A radiosity renderer simulates the difuse inter-reflection in a scene. This means a radiosity renderer only works on diffuse bounces of lights in a scene. Radiosity renderers as such try to solve the radiosity equation from section 2.3.1 on page 14. A radiosity renderer will separate the scene into a finite number of elements. Light is transported between elements as per eqn. (3.1). The formfactor (F_{ij}) describes the actual geometric relationship between two elements, and also the visibility between them.

$$B_i = E_i + \rho_i \sum_j^n F_{ij} B_j \tag{3.1}$$

Because the calculation of the form-factors, and because the linear system becomes quite large many radiosity systems now attempt to iteratively solve the linear system without explicitly storing the matrix. Such methods use the idea of keeping calculation on a mesh-level granularity, but uses sampling instead of a linear system to come up with a solution. On such radiosity renderer is a hemicube gathering renderer.

3.2.1 Hemicube gathering

The hemicube algorithm was originally devised to calculate the formfactors of a radiosity system. However it lends itself to creating altered radiosity methods. The idea of the hemicube algorithm is to start by selecting a patch. Now for this patch all other surfaces are rendered to a hemicube resting on top of the current patch. This combines the geometric and visibility of the form-factors in a simple rendering. If the surfaces are rendered to the hemicube using the current solutions, this gives an estimate for the amount of light hitting the current surface.

By iteratively gathering a new solution from the previous this algorithm progressively converges to a solution. Now because the hemicube is just rendered as any other rendering of patches it is posssible to account for texturing, just by applying the texture to the surface. [Eli]

It is important to use High Dynamic Range(HDR) renderings when rendering the hemicube because the difference between e.g. a lamp, and a shoadow. Thus the hemicube henderings will us HDR to ensure that no illumination is lost.

The hemicube rendering is just a simple rendering of the patches in the scene from a special viewport. Thus a way to accelerate radiosity calculations is to use hardware to perform the

rendering.

3.2.2 Using hardware for hemicube rendering

Using hardware to render a radiosity solution without textures is normally done by rendering patch indices to the framebuffer, and then after a readback count the number of pixels with each index [Eli]. The reason for doing this is because the framebuffer does not allow for HDR floating point renderings. This way the hardware will not support textures in the renderer.

Newer graphical hardware has been extended to function with floating point framebuffers [Nvi]. This means that the index method is not necessarily needed, and textures can be rendered like any onther primitives.

3.3 A raytracer simulation

A raytracer works by tracing from the viewer, and trying to determine what the user sees. A raytracer works in the reverse direction of the light flowing, and as such attempts to determine where the illumiantion seen by the viewer might have come from. [PH04, pp. 4–16]

A raytracer can simulate global illumination effects by stochastically sampling directions from each surface point that needs to be rendered. This is called the monte-carlo method for solving the rendering equation. [DBB03, pp. 105–139]

Raytracers are not bound by geometry complexity nor material complexity, and can give enough time solve any global illumination effects.

3.4 Light probes in simulation

Light-probes are an important image-based tool used in relighting and common illumination systems. By capturing an image of a small mirror reflective sphere this will show light from all incoming directions at the location of the sphere. This is a good tool in relighting systems based on simulation analysis. This comes from the fact that it can capture a very complicated illumination including both intensity and color.

Using a light-probe in the rendering process involves sampling the image, and multiplying by the BRDF of the surface point.

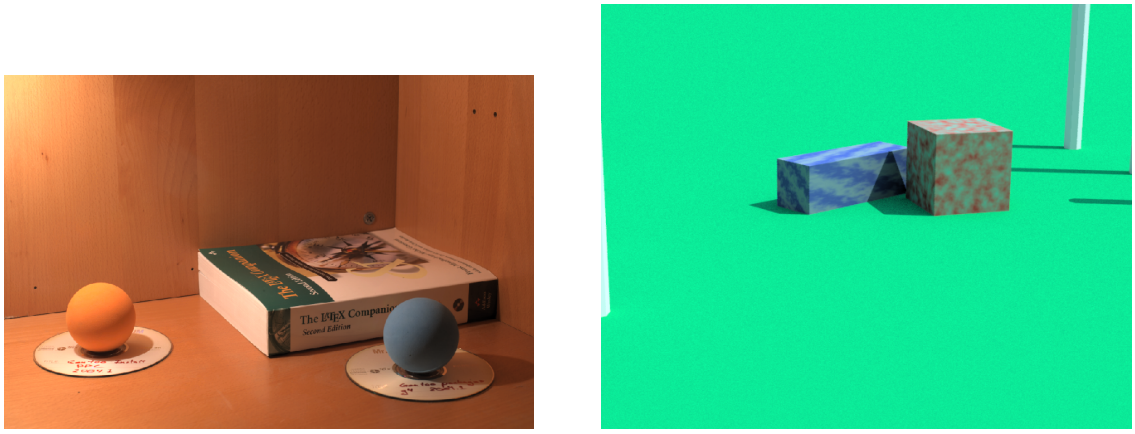
With the hemicube gatherer described in section 3.2.1 on the preceding page it is possible to use light-probes for radiosity renderers. And ray-tracers can trivially be made to render with light-probes. This means using light-probes is possible with both renderer types.

The hemicube radiosity gatherer will treat a light-probe as a texture that is to be applied to the background. While a ray-tracer will usually handle light-probes by sending a number of random sampler-rays towards the probe.

3.5 Experiments

Two experiments are performed, one with a virtual scene, and one with a synthetic scene. The real scene consists of a book and two rubber balls in a bookcase. The synthetic is an outdoor

scene with two blocks, between two rows of columns. The two initial images are shown in figure 3.1.

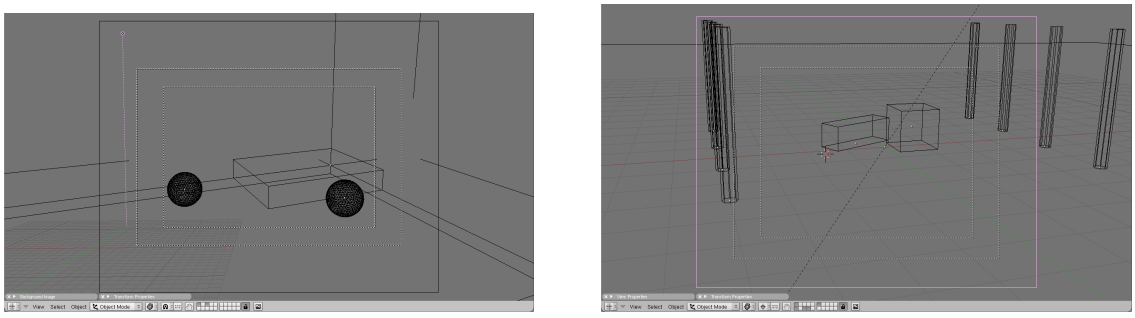


(a) The bookcase real scene.

(b) The outdoor virtual scene.

Figure 3.1: The image used as a input for the system.

The geometry for both scene is modelled using blender 3d graphics suite [Fou]. A view of the geometry can be seen in figure 3.2

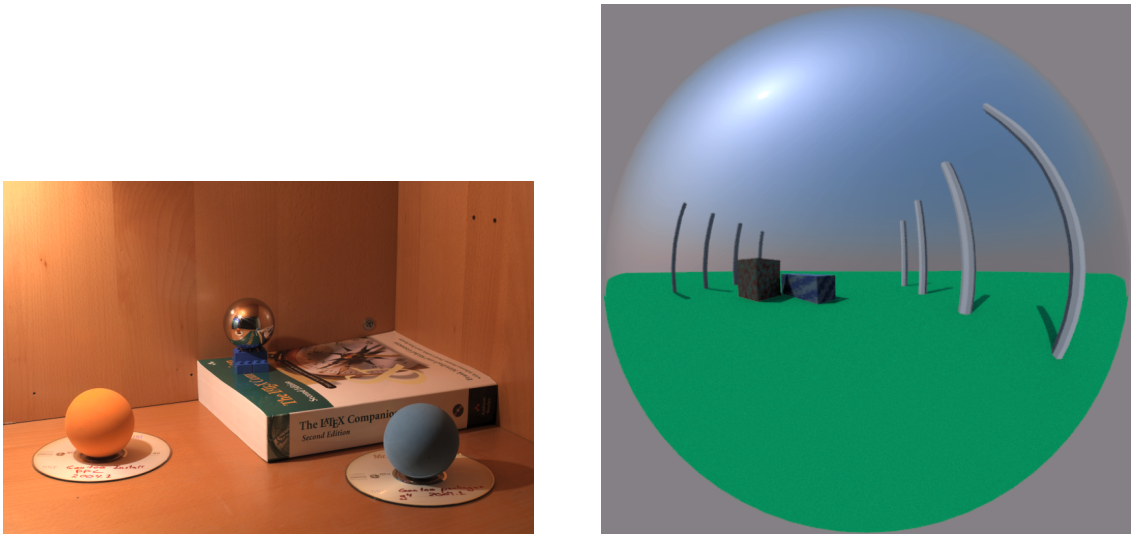


(a) The bookcase real scene.

(b) The outdoor virtual scene.

Figure 3.2: Geometry of the two scenes used for experiments.

The experiment is to try to use a raytracer to simulate the initial illumination, but using different strategies for illuminating the scene. First by using a light-probe (figure 3.3 on the facing page). Then by using an estimation of the original illumination source. And last by a combination of the lightprobe capture, and an estimation of the principal illumination. The raytracer is used since it directly supports all these lighting methods.



(a) A lightprobe sphere in real scene.

(b) A lightprobe rendering in the virtual scene.

Figure 3.3: Lightprobe captures in the two experiments.

3.6 Results

Rendering the synthetic scene with purely white surfaces results in figures 3.5 on the next page, 3.6 on the following page, and 3.7 on page 27. The renderings shows a blue color-tint because of the sky, and sharp shadows from the single predominant light-source.

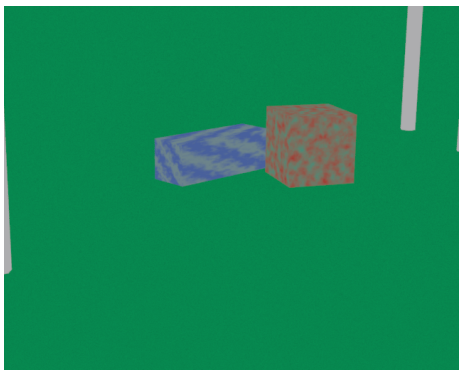
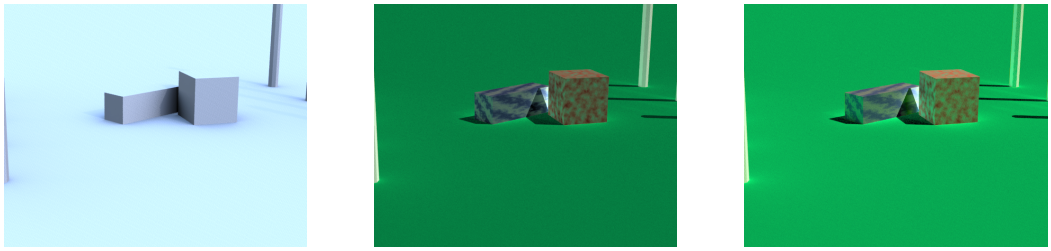


Figure 3.4: The textures of the synthetic sceen mapped onto the geometry.

Synthetic scene — Generally. Generally all the rendering methods does not capture the color-bleeding from the dominant green floor to the scene objects. Color-bleeding is a global illumiantion effect where the light reflected of a surface is so bright that it colors the nearby surfaces. This is seen by a general gren tint in the textures, which should have been removed if the green interreflected light from the ground could effect the other objects. The color-bleeding

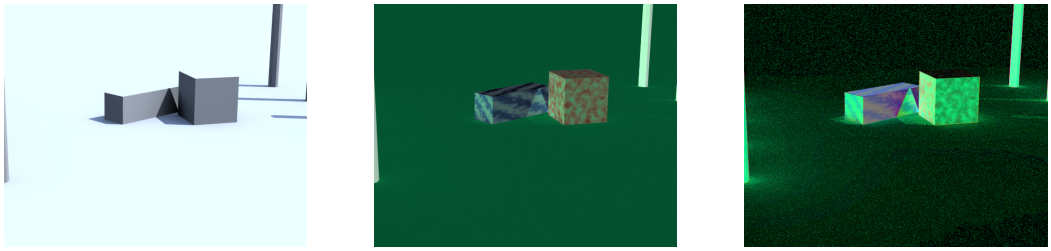


(a) Initial illumination estimate

(b) Unlit textures

(c) Difference to texture reference

Figure 3.5: Initial analysis through probe estimation.



(a) Initial illumination estimate

(b) Unlit textures

(c) Difference to texture reference

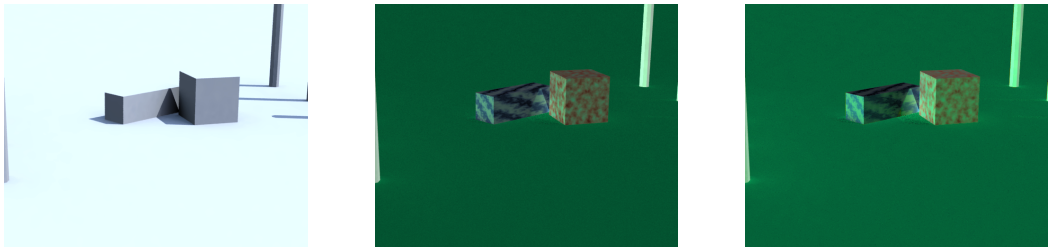
Figure 3.6: Initial analysis through original illumination.

problem is most predominant when using only the original lighting, sinc the light-probe does capture a small part of the green area. Because the entire scene is synthetic materials were known, and a comparison of the recovered texture and the texture originallly used is calculated. The used texture is shown in figure 3.4 on the preceding page

Synthetic scene — Light probe. Rendering with the captured lightprobe shows decreased quality of the shadows in the image (figure 3.5(a)). This is because sampling of a lightprobe does not represent the lightsources very precisely. Since the ligt-source is less defined, so will the shadows be. A second problem here is that the light-probe generally is only valid for a certain location in the scene, and as such does not give an accurate result for the entire scene.

Synthetic scene — Original light estimation. In the synthetic scene the original light is already known, and just used directly instead of an estimation. Using this technique the shape of the shadows are captured quite precisely, but the fact that reflectivity is unaccounted for in the global illumiantion solution means that regions in shadow from the lightsources especially is missing the green color-bleeding from the ground.

Synthetic scene — Combination probe and principal illumination. Attempting to combine the knowledge of a probe with a principal lighting, this shows a little less of the color-bleeding problem while still maintaining the fine shadow boundaries. The error is spread



(a) Initial illumination estimate

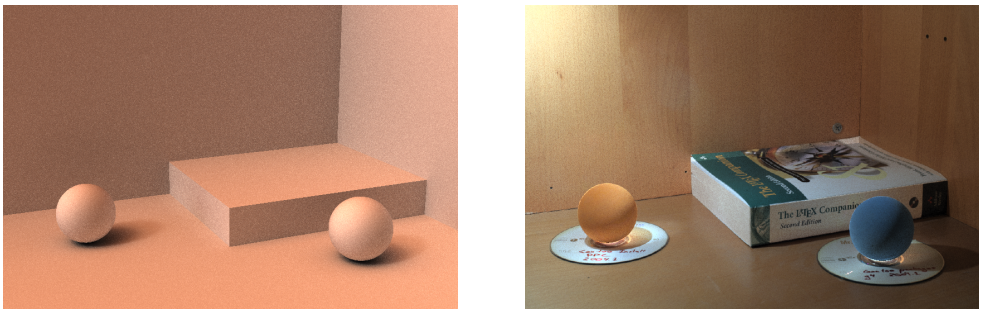
(b) Unlit textures

(c) Difference to texture reference

Figure 3.7: Initial analysis through combination of probe estimation and original illumination.

more generally over the image that with the other two methods, and shows an averaging of the effects of the approximations. Also the errors are generally not so obvious in this image, and as such is the perceptually nicest method (figure 3.7(b)).

The real scene rendered with pure white diffuse reflectors is shown in figures 3.8, 3.9 on the next page, and 3.10 on the following page.



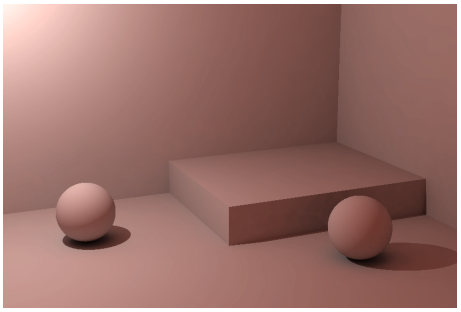
(a) Initial illumination estimate.

(b) Unlit textures.

Figure 3.8: Initial analysis through probe estimation.

Real scene — Light probe. With the real scene the light-probe render shows the same shadow artifacts as the synthetic scene (figure 3.8(a)). In fact the light in the real scene is closer than it was in the virtual scene, so the shadows are more misshaped. Also looking at the area near the light-source in the upper left corner the area is brighter in the original photograph because of its proximity to the light-source (figure 3.8(b)). This is not captured because when using a light-probe the distance to the light-source is not accounted for as the light-probe does not change over space.

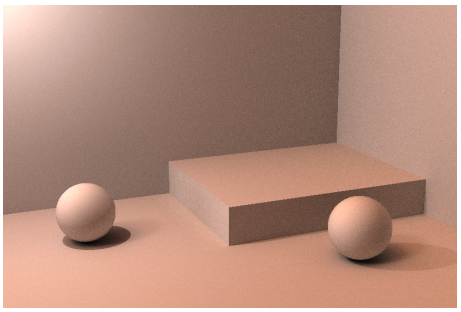
Real scene — Primary light-source estimation. By estimating the position of the light-source using the placement of shadows as a reference the shadows can be neutralized far more effectively (figure 3.9(b) on the following page). This rendering however has problems with matching of the environment color. The color of the regions with shadow has a slight tint. This happens for two reasons, first it is hard for the user to adjust the background value correctly since it is a slow trial and error process. Secondly the original probe (figure 3.3(a)) shows



(a) Initial illumination estimate.

(b) Unlit textures.

Figure 3.9: Initial analysis through original illumination.



(a) Initial illumination estimate.

(b) Unlit textures.

Figure 3.10: Initial analysis through combination of probe estimation and original illumination.

parts of the environment being much darker, and of a different color. This is not captured in the simple one-color background used here.

Real scene — Combination probe and principal illumination. Using the probe does help significantly with the wrong colors in the shadows, but does also introduce another issue. As seen in the probe rendering the lightsource appears to be at a wrong location because of the poor handling of local lights using light-probes. This also affects this rendering, and the fake shadows are clearly visible.

3.7 Summary

The aspect of color-bleeding must be handled for scenes with large areas of a single color, like the ground in the virtual scene. This might be handled by an iterative approach where the estimated textures are used to generate an approximate color for each surface, and the initial render is repeated. Because the surfaces rendered are no longer pure white it also becomes important to remove the approximated surface color from the new texture estimate.

The approach which produces the best results with minimal user intervention seems to be the one based on using the principal light-sources to help a probe estimate. This method

produces shadows which more accurate than just a light-probe, and uses a good estimate for environment radiance. However this system would require som method for removing the principal light-source from the probe to avoid dual-shadows.

Another technique that would help the renderings is importance sampling. In the experiments the light-source was only a tiny part of the entire probe, but ,becuase it is so much brighter than the rest, has a large impact upon the final result. By driving the sampling of the light-probe to sample the region with the light-source more than the rest will give better results. There exists methods which can be used to improve the sampling of an light-probe for example [CJAMJ05] and [ARBJ03].

A reverse way to look at the importance sampling problem is to make sure that the illumination is sampled more accurately in parts of the image where shadows are expected to be. The next chapter details how this can be achieved by adapting a sampling-mesh which serves as a reference for where higher fidelity is needed in the shadows.

Chapter 4

Mesh subdivision

This chapter is about methods for subdividing the mesh of the scene for use as a sampling-grid which will help improve the render of the initial illumination distribution. First methods to adaptively subdivide a mesh in regions of image change are discussed. Then the possibility of using a mesh-relaxation technique to improve the performance of the adaptive subdivision scheme is explored.

As per section 2.5 the mesh initially recoved from the input data must be adapted to reflect the shadow placements. This is because with a finite mesh based method the resolution of the mesh will greatly impact the calculation of a radiance distribution. And for methods which does not rely on the mesh it can still serve as a guide for where to kconcentrate sampling efforts.

While calculating the initial illumination distribution there is the advantage of having a solution readilly at hand. The image captured already has the information from the initial illumination included. This means it also carries information about shadows and where shadow edges are. Therefore an image based method can be used to refine the mesh from the original photograph.

4.1 Mesh datastructure

The datatructure used for the mesh must handle the subdivision and relaxation operations. Subdivision operations splits lines in the mesh one by one to attain a mesh that follows the illumination distribution. Relaxation operations adjust vertices to improve mesh structure. So the two major operations that must be supported are:

Edge split: A new vertex is created on the edge that is to be split. Each triangle containing the edge must also be split into two triangles. Figure 4.1 shows an example of an edge split.

Vertex move: A vertex can be displaced. To avoid getting reversed triangles constraints are placed on the vertex not crossing the opposite edge in a triangle where it is a corner.

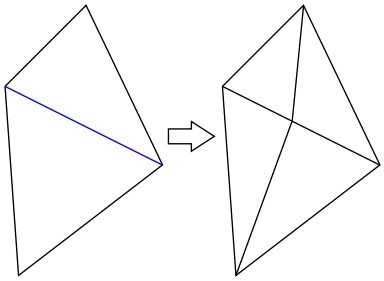


Figure 4.1: Example of an edge being split.

The minimum connectivity in in a mesh dictates that each triangle in the mesh must have knowledge of the three vertices making its corners. A mesh with this connectivity will support rendering the mesh, however by increasing the connectivity the mesh the operations described above can become more time effective.

To support the vertex move each vertex should have knowledge of triangles in which it is a corner. Then when making sure constraints are kept it is just a search of each connected triangle, to see whether the move crosses any opposing edges.

By having a structure for edges it becomes possible to directely relate vertex-pairs which are connected. This becomes important when splitting the edges, where each edge must be examined to determine if it should be split. By creating a explicit edge this can be done without having to search through all triangles. Edges should be connected to the vertices at each end.

Each edge has information about triangles using it. This is used when splitting the edge, because this split must also split the triangles. Instead of having to search each triangle using the vertices at each end this becomes a simple lookup.

The above extra connectivity results in a datastructure defined by these collections:

- Vertex (v): $Index \hookrightarrow (x, y, z, \{triangles\})$
- Triangle (t): $Index \hookrightarrow (v_1, v_2, v_3)$
- Edge (e): $Index \hookrightarrow (v_1, v_2, t_{left}, t_{right})$

4.2 Image based adaptive surface refinement

To increase the resolution of the lighting mesh where the lighting is changing, the mesh is adaptively subdivided by examining the original photograph. Each edge in the mesh is examined to determine if it crosses a shadow boundary. If the edge crosses a boundary, then it is divided into two edges, and so are the two triangles that share the edge. Figure 4.2, 4.3, and 4.4 shows steps through this algorithm. Green lines signify lines that are sufficiently subdivided, while blue lines still needs to be subdivided. In this figure the mesh is a simple planar camera aligned mesh. This must also be the case for the subdivision system while sampling, but the result should still be a 3D mesh, the handling of these two representations is described in section 4.2.4.

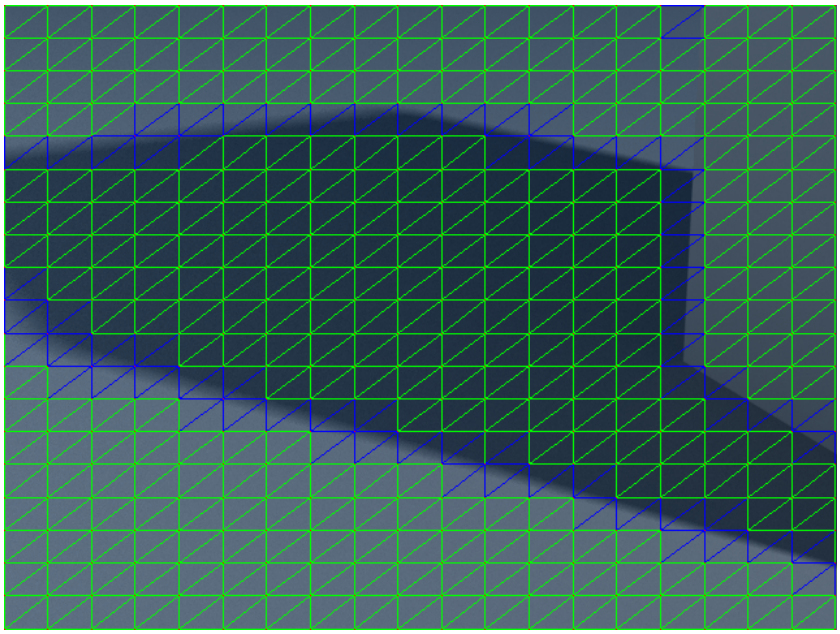


Figure 4.2: Initial mesh before any subdivision.

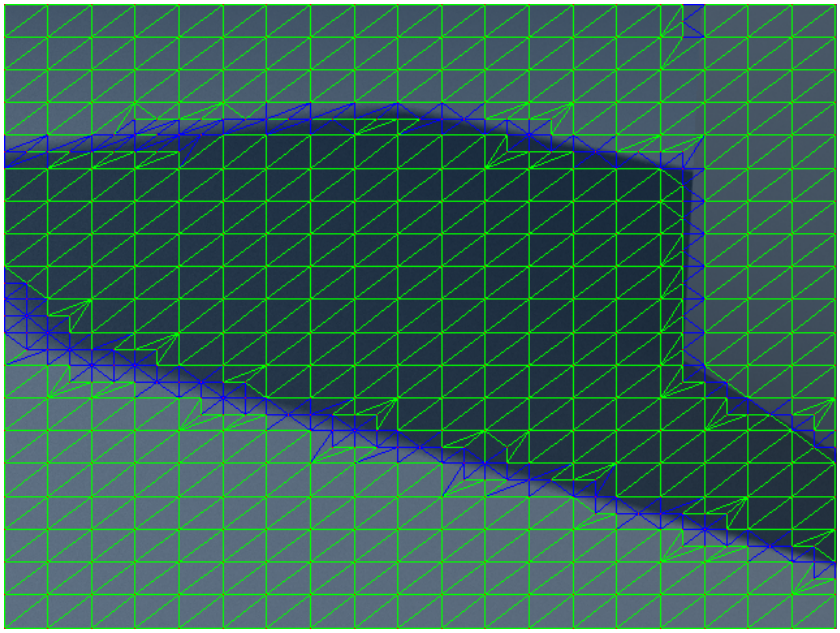


Figure 4.3: Subdivision 1 step.

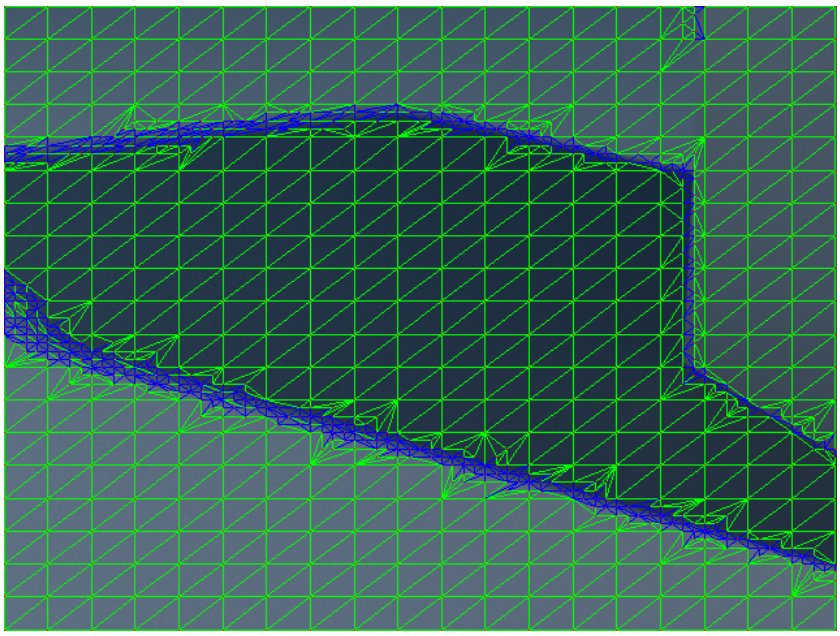


Figure 4.4: Subdivision 2 steps.

4.2.1 Determining edge boundary cross

When determining if an edge crosses a boundary, it is important to reference the interpolation of colors across a triangle from its vertices. Since this interpolation should match up with what is shown in the picture at the triangles location. Figure 4.5 shows how colors are interpolated across a triangle with two different vertex colors.

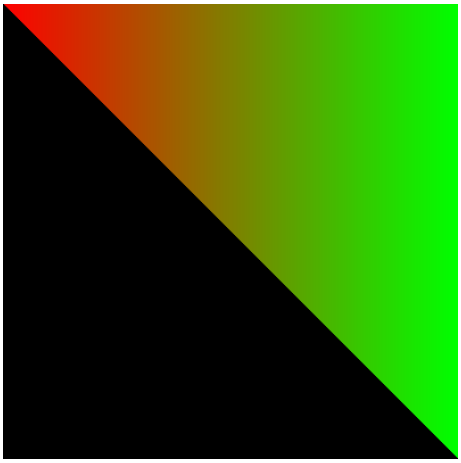


Figure 4.5: Interpolation across a two color triangle.

This interpolation across a triangle becomes important since this is what the final renderer is able to simulate. This means that even though the gradients of lighting in a real image may not follow this interpolation it must be simulated using this in the virtual scene, and therefor the geometry must allow for this interpretation. However a examination of a simple shadow

does indeed reveal this property. Appendix A shows measurements performed on a Shadow boundary revealing the linear interpolation to be observable.

Possible strategies for determining shadow cross:

Analyze line: By sampling each pixel under the line, an estimate of the actual changes in the photograph over the line can be measured. This will capture any change that crosses the boundary, but will require an examination of the entire line.

Compare points: Just comparing pixels at the endpoints of the line will capture mesh edges where one point is in shadow, and one point is not. However this method may fail to capture sharp corners in the shadow correctly (see figure 4.6).

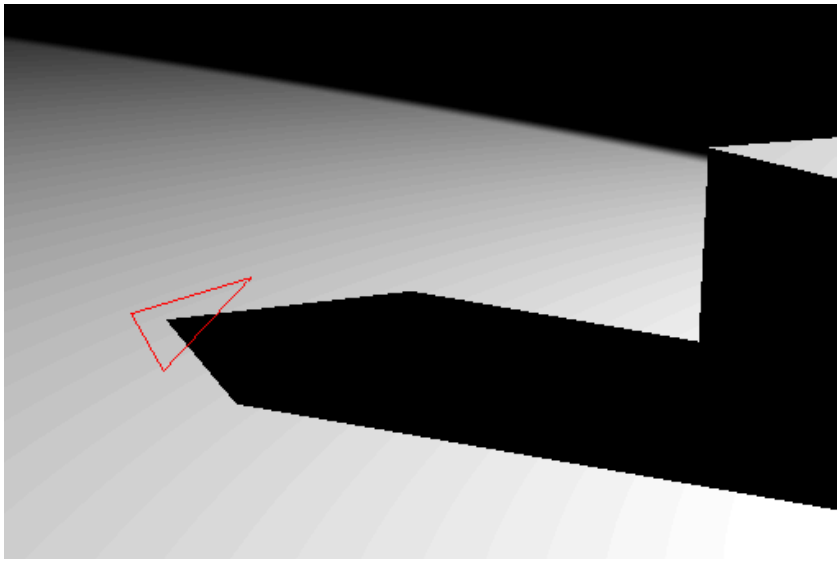


Figure 4.6: A shadow *stabbing* an edge is not captured by point comparison..

4.2.2 Analyze line

Lines are analyzed by integrating over the line in the image. Integration over a line involves finding all pixels which lie on this line, and is therefor the same problem as drawing a line. A fast method for rendering lines to a pixel device is the *Bresenham line-drawing algorithm* [Wika].This method can be adapted to integrate over a line by substituting the pixel write-out for a pixel read.

4.2.3 Definition of a changing line

To distinguish lines that cross a shadow boundary from lines that do not, there must be some measure for how close the line is to the linear interpolation of section 4.2.1 on the preceding page.

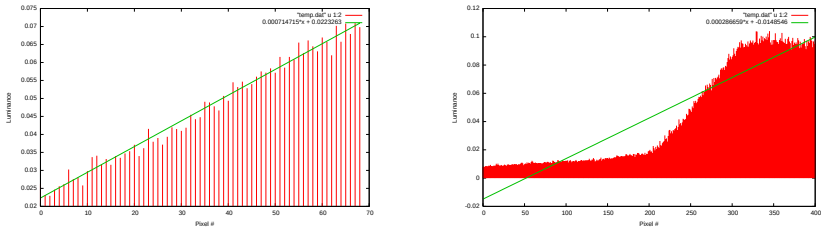
A measure for intensity change over a line is the *standard deviation* of the samples. Standard

deviation comes from statistics, and is calculated as eqn. (4.1). [Wikd]

$$\sigma^2 = \frac{N\sum_{i=1}^{N} x_i^2 - \left(\sum_{i=1}^{N} x_i\right)^2}{N^2} \tag{4.1}$$

Where each x_i is a sample on the integrated line.

Standard deviation is however not affected by order, and requires some form of preprocessing to correctly determine the linear gradients in the interpolation. Another option is to assume the points form a line when plotted in a line-distance/intensity graph. Figure 4.7(a) shows how a satisfactory line looks in this context. Looking at figure 4.7(b) it is not possible to fit a linear line through these points. So with this method the decision value is how much these points deviate from the linear gradient. Appendix A on page 57 shows further data for lines measured on the same shadow.



(a) Linear gradient line.

(b) Line crossing shadow boundary.

Figure 4.7: Graphs of intensities on lines in a photograph the green line is a least squares fit of a linear line.

A method for finding the linear line through the points is *Least squares fitting*. This method tries to minimize the squared error between the fitted line, and the actual points. The square error for a linear line is calculated as in equation 4.2. (x_j, y_j) are samples, and a and b are parameters of the line. The error is the vertical error, or, for intensity lines, the intensity error. [Kre99, pp. 914–915]

$$q = \sum_{j=1}^{n} (y_j - a - bx_j)^2 \tag{4.2}$$

a and b can be found by solving the linear system in equation 4.3.

$$an + b\sum x_j = \sum y_j \tag{4.3}$$

$$a\sum x_j + b\sum x_j^2 = \sum x_j y_j \tag{4.4}$$

When using this linear gradient fit method, a possibility is using the same data used for fitting the line to determine the optimum place to split the edge. Finding the pixel which exhibits the greatest error compared to the fitted line will yield an estimated optimum position to split. In figure 4.7(b) this would mean splitting twice, once at the lower break, and once at the top. Care must be taken however to ensure that the line is not split at the endpoints as a result of the line-fitting. ie. looking at the figure there is quite a difference between the first sample and the fitted line, but a split here is not desirable.

4.2.4 3D mesh and planar subdivision

As stated the image-based subdivision is performed in planar screen-oriented space. But since the subdivided mesh should still be a 3D mesh, the transformation is performed for each edge-test. The transformation to the planar space means that when splitting an edge the new vertex must be created from the original coordinates to be usable in the original mesh.

By this the algorithm for testing a line for subdivision becomes:

1. Transform endpoints of edge into screen-space.
2. Use line read to determine fitness of current line.
3. If line is determined to cross shadow boundary:
   (a) Calculate position of new vertex between edge end-points in untransformed space.
   (b) Split edge.
   (c) Split neighboring triangles.
   (d) Add new edges to list of edges to be examined for subdivision.

4.3 Clipping

Because only the mesh within the image is needed for the subdivision the mesh is first clipped to within the screen area. This is feasible because only a single image is used, and the aim is to only reproduce the original view with a new illumination. Thus this mesh can also be used when rendering a new illumination.

4.3.1 Triangles

Since the entire system works from triangles it is reasonable to only work with clipping of such primitives. In the rendering pipeline clipping is performed per triangle individually. This is a good approach when the clipped triangles are just needed for rendering to the screen. However since this clipping in the remeshing part of the system is followed by other mesh operations, it becomes important that the mesh retains its connectivity as described in section 4.1. A way to obtain this property is by splitting edges instead of clipping triangles. When the edge is split traingles outside the clipping region can be discarded.

By clipping each boundary individually the possible scenarios that require splitting is two, see figure 4.8 on the following page. As the figure shows each of these scenarios can be handled by splitting both lines crossing the boundary one by one. Therefore clipping the entire mesh can be accomplished by splitting each edge one by one.

4.4 Mesh relaxation

Mesh relaxation can be applied to the vertices of the mesh while performing the subdivision. Thereby a better fit may be obtained, and possibly some subdivision avoided. Relaxation works by moving each vertex so that it will better fit to the shadow boundaries in the original photograph. The following describes a method for relaxing a mesh, which use a technique called VPR.

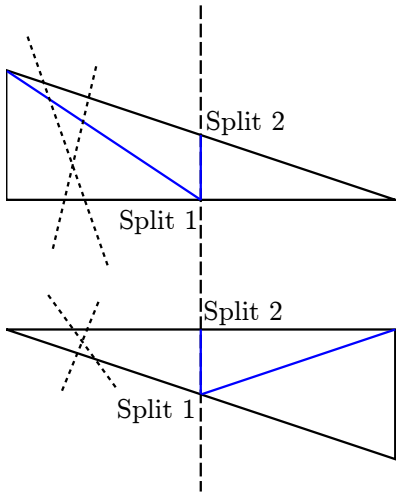


Figure 4.8: Clipping scenarios with a single boundary and a triangle.

4.4.1 Uphill Maximum Gradient Path

VPR is used to find local maxima in the changing in an image. The dual of VPR, downhill maximum gradient path [MVSM95], is often used in image segmentation, since it connects each pixel with its least changing surrounding area. VPR however allows a point distribution to be transformed into another distribution wich better captures changes in the image. VPR works with a gradient image. A gradient image coresponds to the derivative of the original image, and captures edges in the image. Figure 4.9 shows an example of an image and its gradient magnitude image.

Figure 4.9: Original photo (left), and resulting gradient image (right).

When performing VPR the gradient image is virtually considered as a landscape, where regions of great change are mountains, and regions of little change are valleys. The dual of VPR (downhill MGP) will now let a droplet of water fall from each desired point, and the path of this droplet is the downhill maximum gradient path. Computationally the path of the waterdroplet is simulated by allways selecting the next pixel which has the least gradient intensity. Now for VPR the gravity is virtually inverted, so that the droplet will find the locally highest peak instead of the locally lowest valley. Another way to simulate this is to use the inverted gradient image. Figure 4.10 on the facing page shows the image from figure 4.9 as

an inverse landscape, where image changes are valleys.

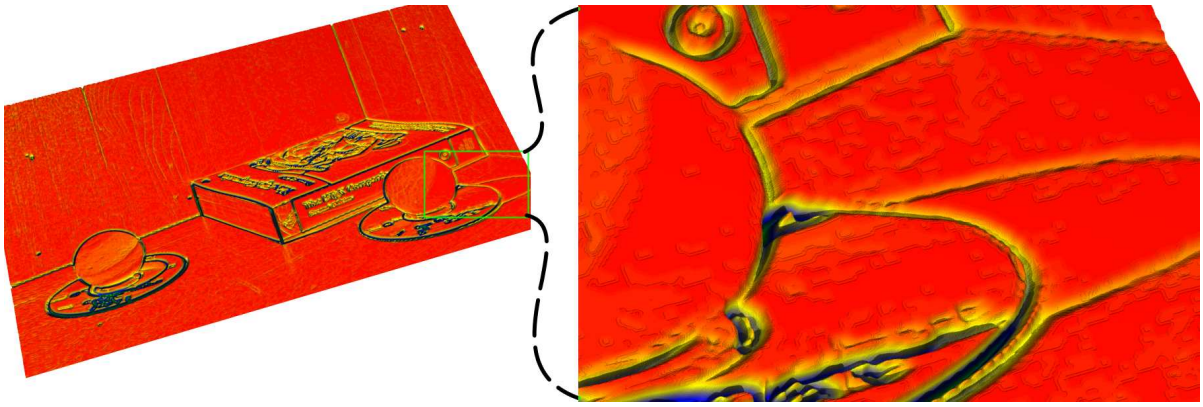


Figure 4.10: Example landscape for vertex mesh relaxation, right is magnified portion.

Figure 4.12 on page 41 shows a simulation of waterdroplets falling in a landscape produced by sweeping a sinuscurve in two dimensions. A number of random waterdroplets have been started, and allowed to flow down the slopes of the landscape. The algorithm for producing a flow-path from a starting pixel is outlined in the algorithm in figure 4.11 on the next page. At each step this algorithm finds the pixel in the surrounding area to the current pixel which has the smallest intensity, and chooses this as the next pixel in the path.

4.4.2 Using VPR for vertex mesh relaxation

By utilizing the water flow-paths from VPR it is possible to apply a relaxation to vertices in a mesh. By allowing each vertex of the mesh be relaxed in the direction of the water flow-path the mesh will dynamically adapt to the contours of the image.

Looking at the method used to create the gradient image, a possible optimization can be implemented since the system is used to move points, and not to find paths. Normally the gradient image is produced by calculating the horizontal and vertical derivatives, and then calculating the magnitude of this vector. Since the idea in the mesh relaxation is to displace points this might just as well be performed in the direction of the derivative, than by normal VPR.

When a point is moved some care must be taken in preserving the orientation of the triangles. That is if a point crosses the opposing edge of any triangle it belongs to, that triangle is flipped, an event that disrupts the mesh. Therefor when moving a vertex it is important to check that it does not cross any of the edges in the adjoining triangles. A method for testing this property is to examine for each edge whether the move will cross the edge. Either by calculating to which side of the edge the point will finally be, or by determining if the vector corresponding to the move is intersecting the edge. Figure 4.13 sumarize the two methods. Testing the intersection of the move and edge is an instance of segment to segment intersection [SE03, pp. 241–245]. While testing the before, and after points of the move for edge cross can be performed by testing which side of the edge the points are on, which is the same test used in polygon inclusion tests [SE03, pp. 695–697].

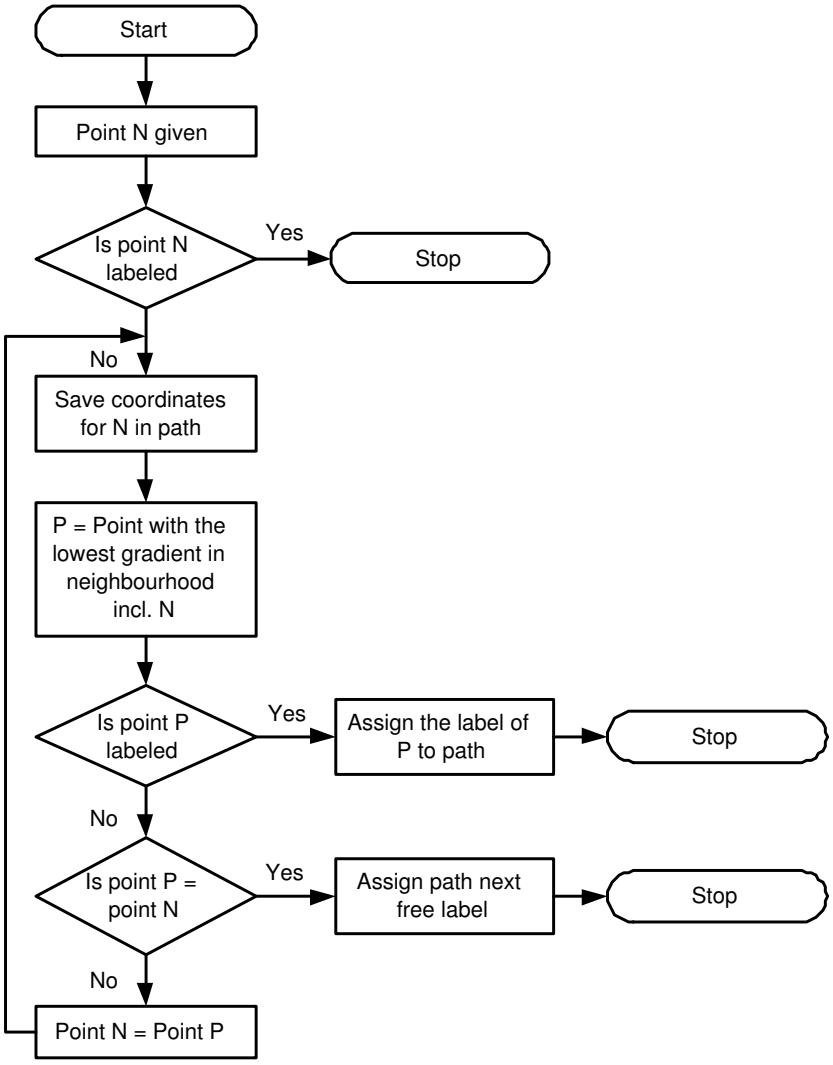


Figure 4.11: Flowchart showing the algorithm for Downhill MGP.

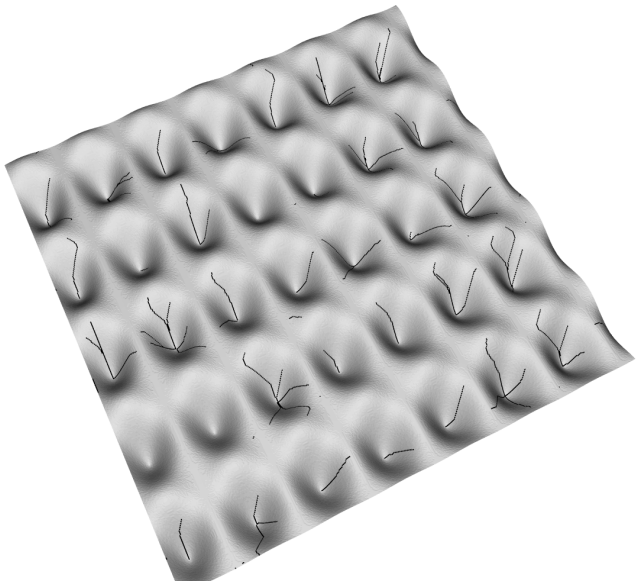


Figure 4.12: Waterdroplet simulation, black lines are droplet paths.

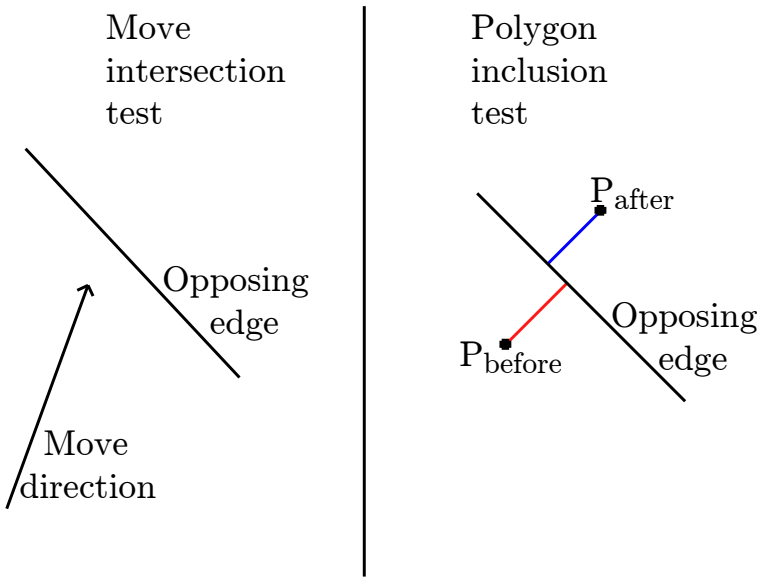


Figure 4.13: Methods to test the validity of a relaxation move..

Chapter 5

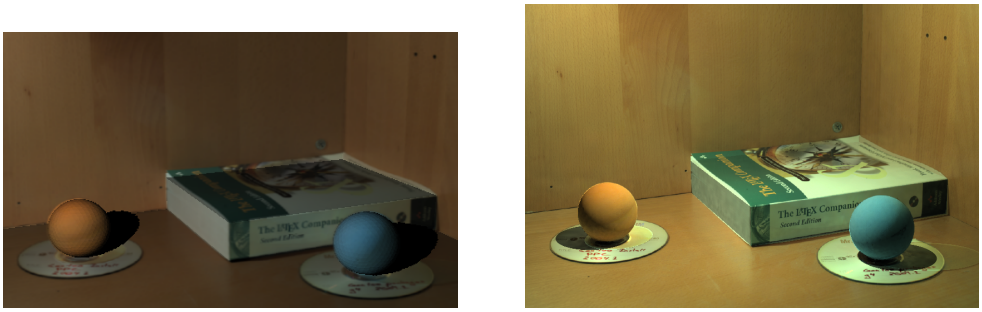
Interactive render of new illumination

This chapter details rendering of a new novel illumination. The system must perform this in real-time, so that it can be used in interactive illumination design.

A relighting system requires a phase for calculating the new illumination. To allow the user to interactively play with a new lighting scheme this step uses the interactive pipeline render model (see section 2.1.2 on page 10).

The idea is to render the new illumination to the frame-buffer while using the unlit image as a texture. A feature of the rendering pipeline is used to calculate texture coordinates in screen aligned space automatically [SWND04, pp. 422–432]. A shadow map is used to render shadows onto the image [AMH02, pp. 269–275].

Figure 5.1(a) shows a new illumination rendered with the interactive application. The real scene from section 3.5 on page 23 is used as the basis. For reference figure 5.1(b) shows the same scene using another new illumination, but this time rendered with a ray tracer.



(a) Real-time application.

(b) Raytraced.

Figure 5.1: Renders of the real scene using new illuminations.

Chapter 6

User interface

This chapter looks at the user interface for the interactive relighting part of the system.

6.1 The users

The application is ment for use when designing illumination of a scene. A typical user is the editor of a movie who wants to change the ligthing of the movie and investigate which illumination design yields the best result. The user can also be a photographer who wants to investigate where his lights should be for the perfect photo. In general the user is a visual oriented person with some tecnical expertice and as such the user interface is designed with this in mind.

The input of the application is a picture of a scene and a representation of the geometry in this scene.

6.2 The main form

The main form has two areas. The left area is for information about the added ligths. In the right area the scene is shown with the new lights added. The design of the form can be seen on figure 6.1. and the dialog for adding new light-sources in figure 6.2.

6.2.1 The left area:

At the top is a representation of the geometry of the scene. On this the added lighs are represended as dots. The dots have different colors according to their status.

red: The currently selected light source.

blue: A shown light source that is not active.

yellow: A deactivated light source.

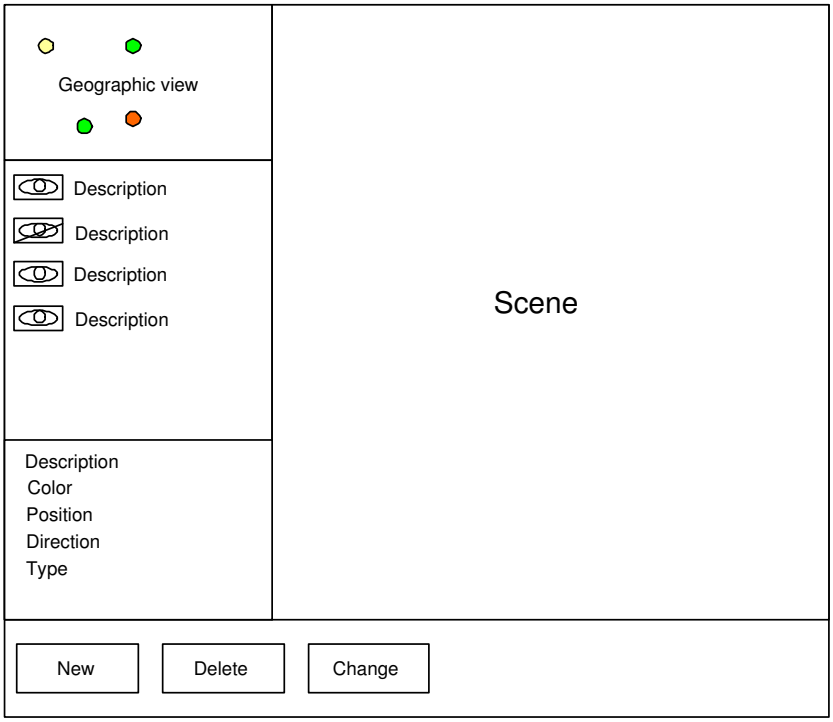


Figure 6.1: Main form.

Below the geometry wiew is a list of all the lights added represented by a short description. In front of each list item is a small checkbox with an eye icon. This is used to deactivate the light source but still keep it for later use.

Under the list is information about the selected light. This is the direction, color and position of the light source.

6.2.2 The right area:

The scene is shown in the right area. The effects of the inserted light sources are shown here.

There are 3 buttons at the bottom of the main form "New", "Change" and "Delete".

6.3 functionality

This section is about design of the user interface. It describes the functionality of the finished application. However the user interface has not been implemented, but the functionality is described as if it was. A finished application will have the following main functions:

Add a new light source

When adding a light source a new dialog box will appear. This allows the user to state a description, color, position and direction of the added light source. The description is what is shown in the light source list. Color is given in hex format. Position is given as 3D coordinates,

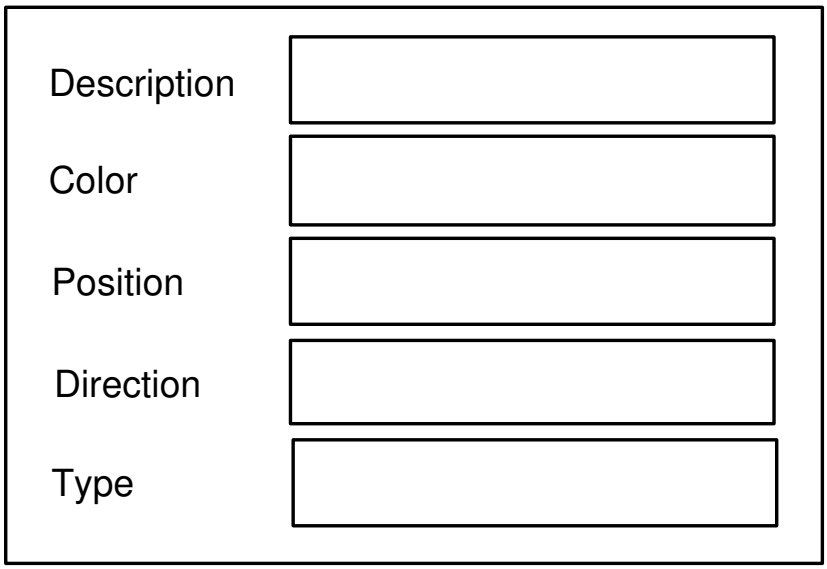


Figure 6.2: Dialog for adding new light-source.

and so is direction indication a vector. The type of light can also be selected. There are the following types available: Spotlight, point-light, area-light.

Selecting a light source

The user has 2 ways to select a light source. Clicking on it in the geometric view or selection it on the list. When a light source is selected it appears red in the geometric view. A selected light source can be moved, changed or deleted.

Moving a light source

Moving a light source can be done in two ways. By the arrow keys or by changing its position by clicking the change button. The up and down key moves the light up or down in the geometric view. The left and right keys move the light source to the sides. Home and End moves the light source forwards or backwards in the geometric view. Pressing SHIFT while moving the light alow it to be moved a longer distance at a time.

changing a light source

The user can change the settings of a light source by clicking the "Change" button. the settings are description, color, position, direction and type. This will bring out the same dialog that is shown when adding a light source.

Removing a light source

A light source can be removed both temporary and permanently. Permanently by pressing the "Delete" button or the DEL key and temporary by clicking the box with the eye icon on the light source list. When the icon is clicked the light source will no longer appear on the scene and the light source will appear in yellow in the geometric view. If the user want to show temporarely removed light source again, he must reclick the icon on the list.

Chapter 7

Conclusion

The thesis describes details about a system for relighting a photograph. The system consists of three phases: analyzing initial illumination, render new illumination, and combine results. The two last phases are combined in an interactive application for light experimentation.

Analyzing initial illumination Analysis of the initial illumination is performed by simulation. Analysis by simulation is a technique in which as much of the context of the original capture as possible is used to simulate a new result. Then the difference between the capture and this new result is caused by the unknown parameters.

Three different approaches for capturing the lighting of the scene was tested on virtual and real user provided data. The results show that the most accurate removal of the illumination is possible by estimating the position and types of light-sources affecting the scene. However using light-probes gives a better matching of ambient illumination, and ultimately means an image where the color appears most correct.

A method for improving the solution is proposed and analyzed. This method uses image-based techniques to create a mesh that captures the composition of shadows in the original scene. Both by subdivision and relaxation.

Render new illumination An application for interactively calculating and displaying a new illumination was coded. In this application the user can move a light-source to experiment with different illuminations.

Bibliography

[AMH02] Thomas Akenine-Möller and Eric Haines. *Real-time rendering.* A K Peters, 2 edition, 2002. 10, 43

[ARBJ03] Sameer Agarwal, Ravi Ramamoorthi, Serge Belongie, and Henrik Wann Jensen. Structured importance sampling of environment maps. *ACM Trans. Graph.*, 22(3):605–612, 2003. 29

[Ash] Ian Ashdown. Photometry and radiometry. webpage: http://www.ledalite.com/download/white/photomet.doc. Valid as of June 4th, 2005. 11

[BG01] Samuel Boivin and André Gagalowicz. Image-based rendering of diffuse, specular and glossy surfaces from a single image. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 107–116, New York, NY, USA, 2001. ACM Press. 15

[BG02] Samuel Boivin and André Gagalowicz. Inverse rendering from a single image. In *IS&T's First Europ. Conf. on Color in Graphics, Images and Vision*, pages 268–277, Poitiers, France, April 2002. 15

[CJAMJ05] Petrik Clarberg, Wojciech Jarosz, Tomas Akenine-Möller, and Henrik Wann Jensen. Wavelet importance sampling: Efficiently evaluating products of complex functions. In *Proceedings of ACM SIGGRAPH 2005.* ACM Press, 2005. 29

[DBB03] Philip Dutré, Philippe Bekaert, and Kavita Bala. *Advanced global illumination.* A K Peters Ltd., 2003. 23

[Deb98] Paul Debevec. Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 189–198, New York, NY, USA, 1998. ACM Press. 5

[DM97] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 369–378, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. 3

[DS97] George Drettakis and François X. Sillion. Interactive update of global illumination using a line-space hierarchy. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 57–64, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. 16

[DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 11–20, New York, NY, USA, 1996. ACM Press. 3, 4

[Eli] Hugo Elias. Radiosity. webpage: http://freespace.virgin.net/hugo.elias/radiosity/radiosity.htm. Valid as of June 13th, 2005. 22, 23

[Fou] The Blender Foundation. blender3d.org :: Home. webpage: http://www.blender3d.com/. Valid as of June 11th, 2005. 24

[GHH01] Simon Gibson, Toby Howard, and Roger Hubbold. Flexible image-based photometric reconstruction using virtual light sources. 20(3):296–305, September 2001. 3

[GW02] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice-Hall, Inc., 2 edition, 2002. 18

[Hec90] Paul S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 145–154, New York, NY, USA, 1990. ACM Press. 12

[HSA91] Pat Hanrahan, David Salzman, and Larry Aupperle. A rapid hierarchical radiosity algorithm. *SIGGRAPH Comput. Graph.*, 25(4):197–206, 1991. 7, 16, 18

[JL04] Katrien Jacobs and Céline Loscos. Classification of illumination methods for mixed reality. In *State of the Art Reports*, pages 95–118. Eurographics, 2004. 15

[Kaj86] James T. Kajiya. The rendering equation. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150. ACM Press, 1986. i, 11

[Kre99] Erwin Kreyszig. *Advanced engineering mathematics*. John Wiley & Sons, Inc., 8 edition, 1999. 36

[LDR00] Céline Loscos, George Drettakis, and Luc Robert. Interactive virtual relighting of real scenes. *IEEE Transactions on Visualization and Computer Graphics*, 6(4):289–305, 2000. 3, 16, 18, 22

[MDA02] Vincent Masselus, Philip Dutré, and Frederik Anrys. The free-form light stage. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 247–256, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association. 5, 17

[MK94] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE Transactions on Information Systems*, E77-D(12), December 1994. 2

[ML04] Claus .B. Madsen and Rune Laursen. Image relighting: Getting the sun to set in an image taken at noon. In *13th Danish Conference on Pattern Recognition and Image Analysis*, pages 13–20, Copenhagen, Denmark, August 2004. 16, 22

[MVSM95] Frederik Maes, Dirk Vandermeulen, Paul Suetens, and Guy Marchal. Computer-aided interactive object delineation using an intelligent paintbrush technique. In *CVRMed '95: Proceedings of the First International Conference on Computer Vision, Virtual Reality and Robotics in Medicine*, pages 77–83, London, UK, 1995. Springer-Verlag. 38

[Nvi] Nvidia. High dynamic range rendering on the geforce 6800. webpage: http://developer.nvidia.com/object/hdr_on_6800.html. Valid as of June 13th, 2005. 23

[OCDD01] Byong Mok Oh, Max Chen, Julie Dorsey, and Frédo Durand. Image-based modeling and photo editing. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 433–442, New York, NY, USA, 2001. ACM Press. 2, 17

[OHHM02] Marc Olano, John C. Hart, Wolfgang Heidrich, and Michael McCool. *Real-Time Shading*. A K Peters, 2002. 13

[PH04] Matt Pharr and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kufmann, 2004. 23

[SE03] Phillip J. Schneider and David H. Eberly. *Geometric tools for computer graphics*. The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling. Morgan Kaufmann, 2003. 39

[SWND04] Dave Shreiner, Mason Woo, Jackie Neider, and Tom Davis. *OpenGL Programming Guide*. Addison Wesley, 4 edition, 2004. 43

[War92] Gregory J. Ward. Measuring and modeling anisotropic reflection. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 265–272, New York, NY, USA, 1992. ACM Press. 15, 21

[Wei] Eric W. Weisstein. Voronoi diagram — from mathworld. Webpage: http://mathworld.wolfram.com/VoronoiDiagram.html. Valid as of June 10th, 2005. 17

[Wika] Wikipedia. Bresenham's line algorithm. webpage: http://en.wikipedia.org/wiki/Bresenham's_line_algorithm. Valid as of June 12th, 2005. 35

[Wikb] Wikipedia. Compositing. webpage: http://en.wikipedia.org/wiki/Compositing. Valid as of June 4th, 2005. 4

[Wikc] Wikipedia. Laser range scanner. webpage: http://en.wikipedia.org/wiki/Laser_-range_scanner. Valid as of June 4th, 2005. 3

[Wikd] Wikipedia. Standard deviation. webpage: http://en.wikipedia.org/wiki/Standard_deviation. Valid as of June 13th, 2005. 36

[Wil] Keith Wills. Dodge and burn project. webpage: http://www.scphoto.com/html/dodge_burn.html. Valid as of June 4th, 2005. 1

[YDMH99] Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins. 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, 1999. ACM Press/Addison-Wesley Publishing Co. 3, 15, 21

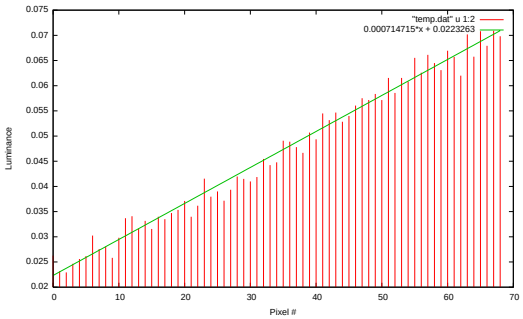
Appendices

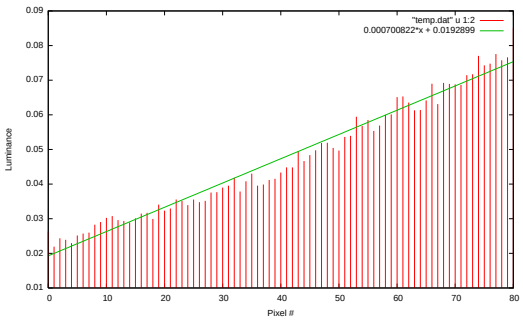
Appendix A

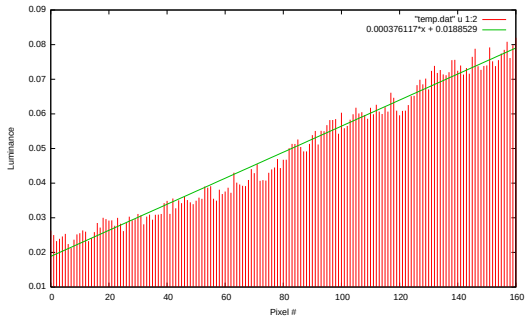
Data fitting for lines in soft shadows

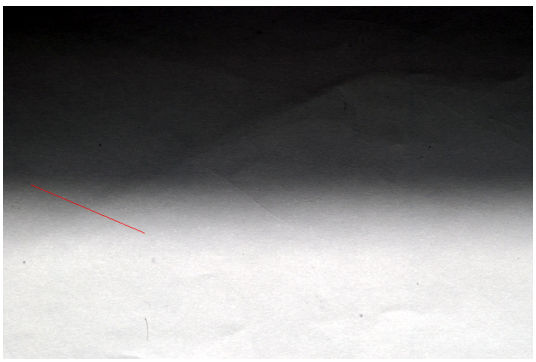
These are data fit result obtained by taking pictures of a shadow. Each left image shows intensities over the line, and a least sqaures fit to a linear function. The right image shows from which part of the shadow the line is sampled. The first 17 images are of lines within the penumbra of the shadow. The last 17 are of lines passing from the umbra through the penumbra to unshadowed.

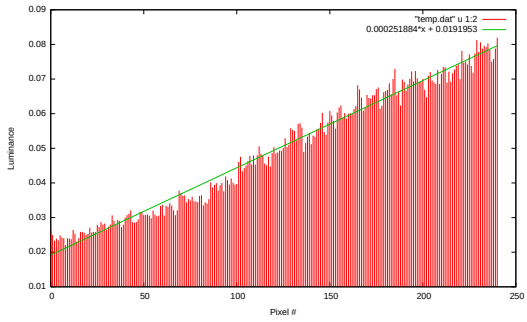
A.1 Penumbra

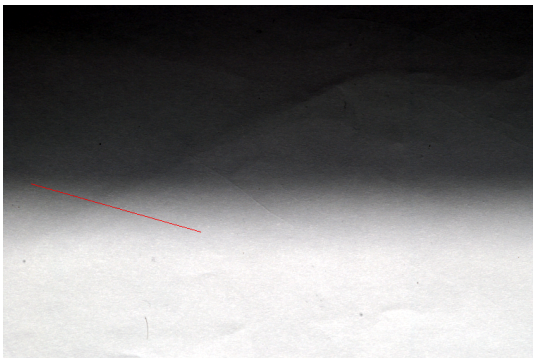


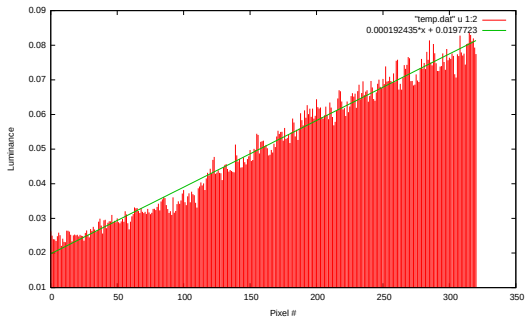


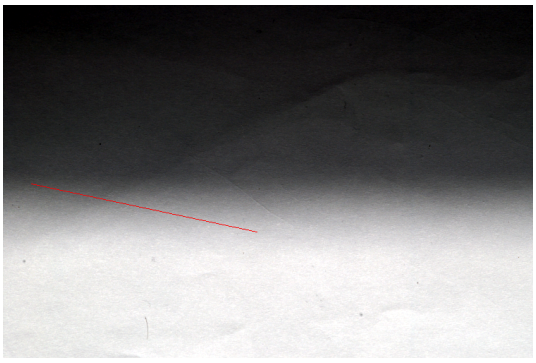

"temp.dat" u 1:2
0.000376117*x + 0.0188529
Luminance
Pixel #

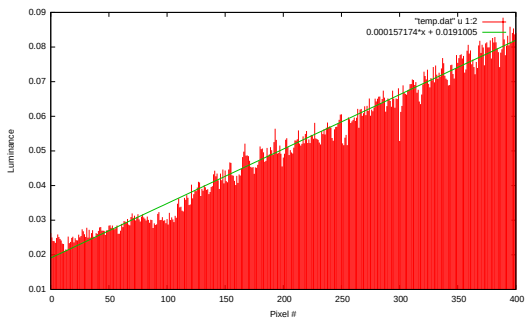


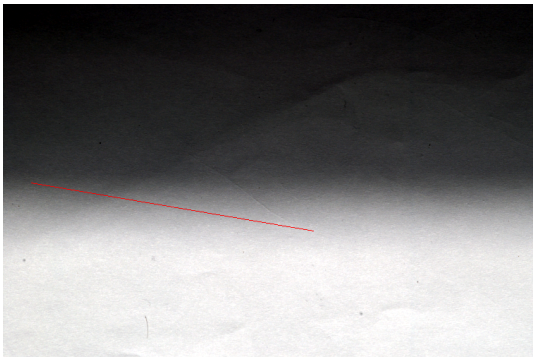

"temp.dat" u 1:2
0.000251884*x + 0.0191953
Luminance
Pixel #

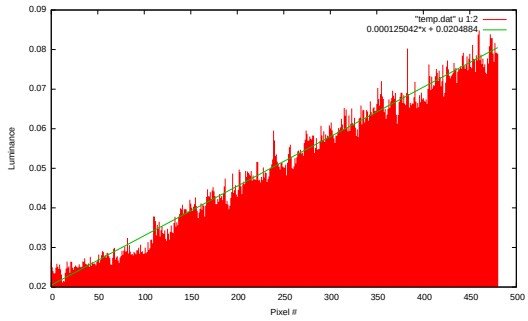


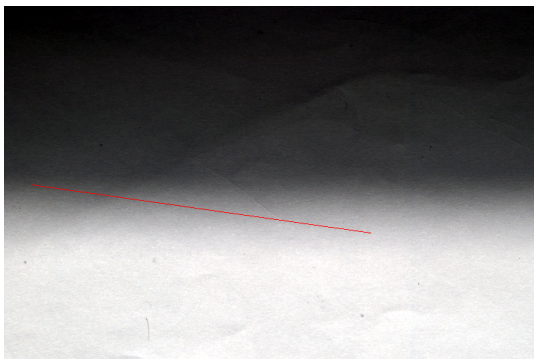

"temp.dat" u 1:2
0.000192435*x + 0.0197723
Luminance
Pixel #

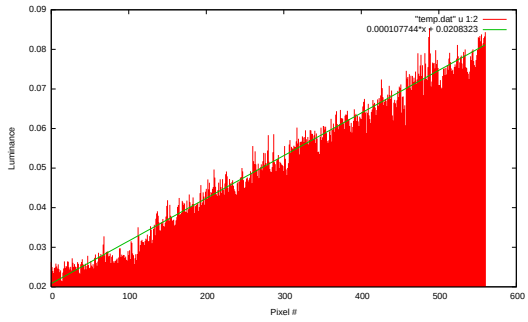


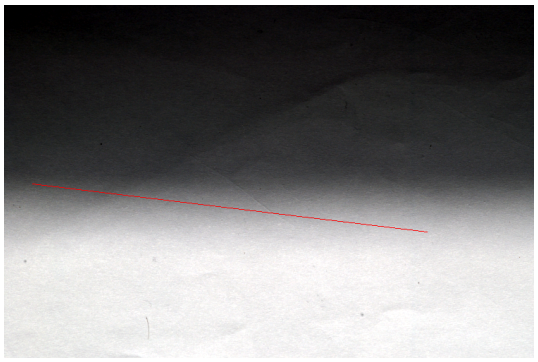

"temp.dat" u 1:2
0.000157174*x + 0.0191005
Luminance
Pixel #

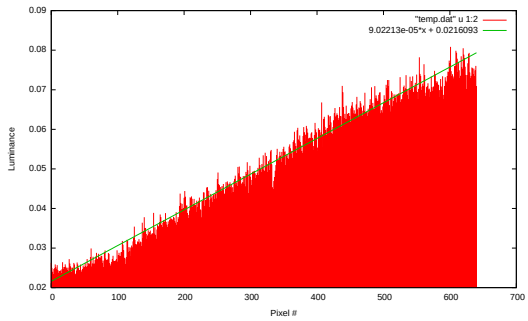


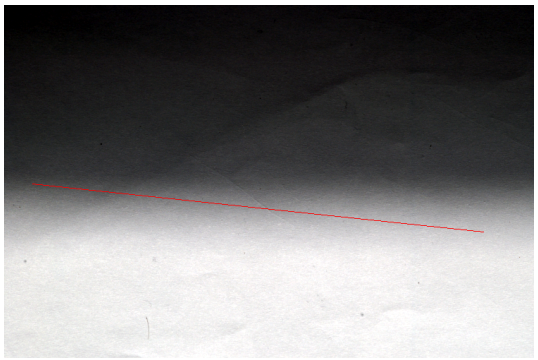




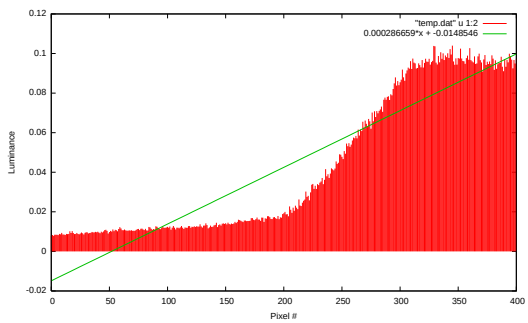


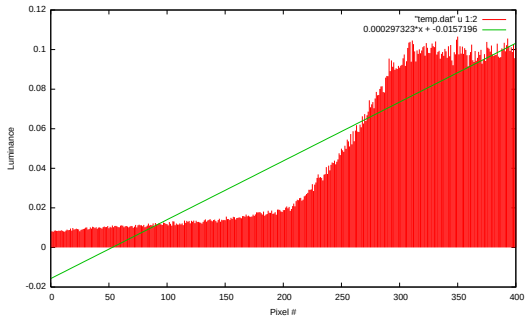


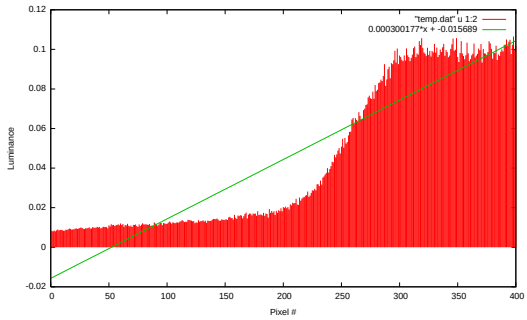


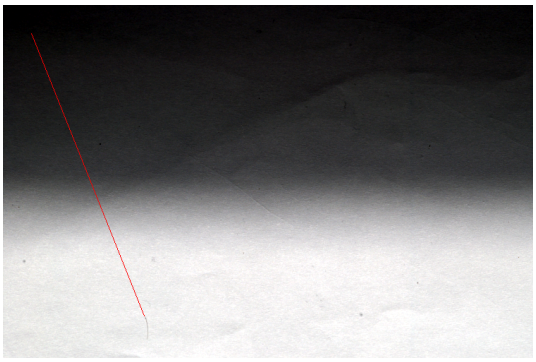


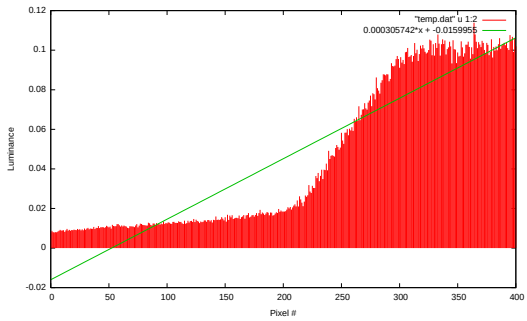
A.2 Umbra to unshadowed

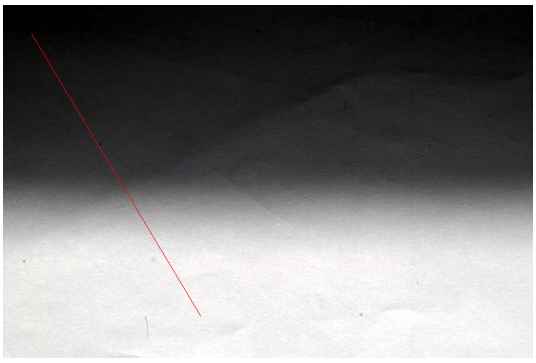


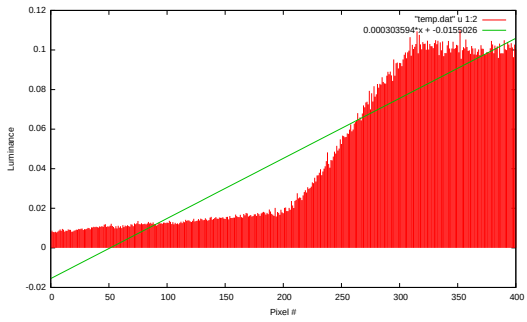

"temp.dat" u 1:2
0.000297323*x + -0.0157196
Luminance
Pixel #

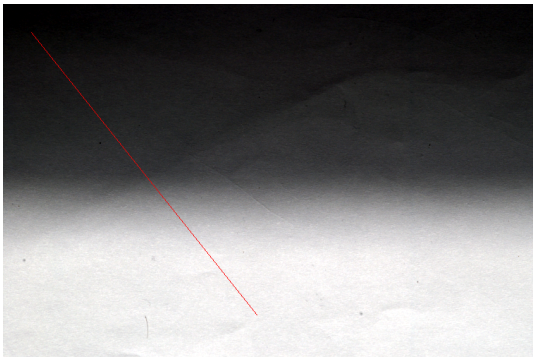

"temp.dat" u 1:2
0.000300177*x + -0.015689
Luminance
Pixel #

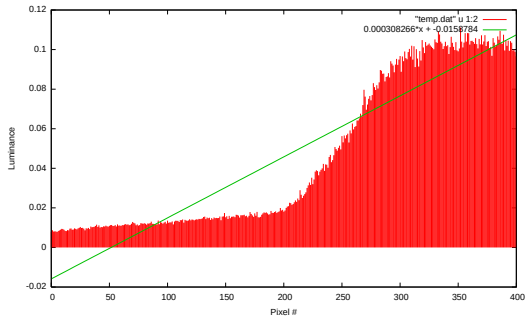


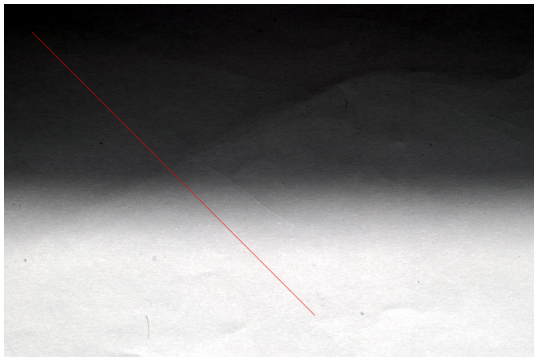

"temp.dat" u 1:2
0.000305742*x + -0.0159955
Luminance
Pixel #

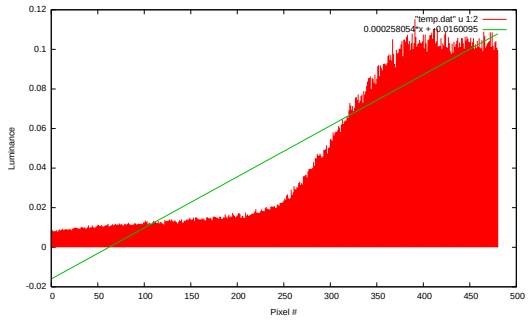


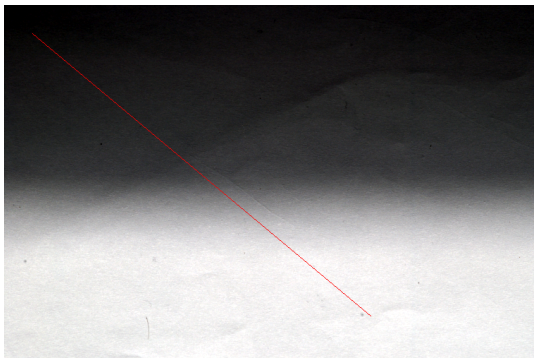

"temp.dat" u 1:2
0.000303594*x + -0.0155026
Luminance
Pixel #

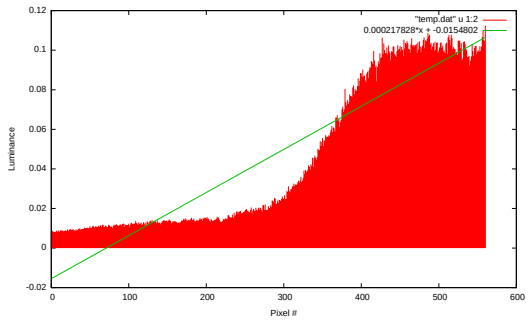


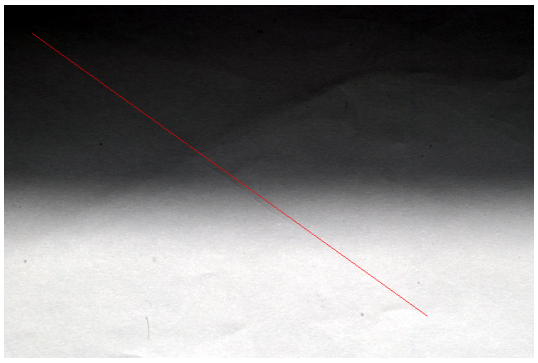

"temp.dat" u 1:2
0.000308266*x + -0.0158784
Luminance
Pixel #

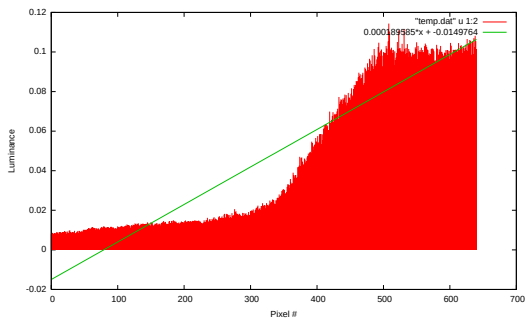



"temp.dat" u 1:2
0.000258054*x + -0.0160095
Luminance
Pixel #




"temp.dat" u 1:2
0.000217828*x + -0.0154802
Luminance
Pixel #




"temp.dat" u 1:2
0.000189585*x + -0.0149764
Luminance
Pixel #