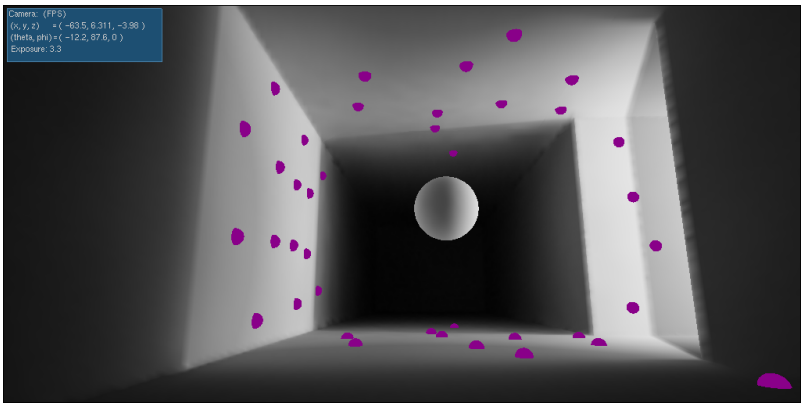
Fusing Environment Maps for IBL in Large Scenes

DEPARTMENT OF MEDIA TECHNOLOGY AND ENGINEERING SCIENCE

COMPUTER VISION AND GRAPHICS



Group 1021

Jesper Højgaard Pedersen

August 6th, 2008

AALBORG UNIVERSITY

AALBORG UNIVERSITET

Laboratory for Computer Vision and Media Technology
Niels Jernes Vej 14
Phone +45 99 40 87 89
http://www.cvmt.aau.dk

Title: Fusing Environment Maps for IBL in Large Scenes

Theme: Specialisation in Computer Vision and Graphics

Project period:
Master's Thesis, Spring 2008

Project group:
Group 1021

Group members:
Jesper Højgaard Pedersen

Supervisor:
Claus B. Madsen

Number of Prints: 4

Pages: 57

Finished: August 6th, 2008

Abstract:

This report documents the development of a method for augmenting a large scene with a virtual object so that the shading and shadowing of the virtual object is consistent with the spatially varying illumination in the scene.
This is done through a preliminary analysis that reviews previous work, which is used as a basis for proposing a new method that takes spatially varying illumination into account. It is found that varying illumination can be captured using multiple HDR light probes. The light probes are then back-projected onto a geometric proxy that is a rough model of the scene geometry. To shade the virtual object, the the radiance distribution at the position of the object is approximated by rendering an environment map, which then can be used to shade and shadow the object in a scene-consistent manner.

Preface

This thesis documents the work performed by Jesper Højgaard Pedersen for the degree of M.Sc.CE during the 9th and 10th semester at the specialization of Computer Vision and Graphics at Aalborg University, Denmark. The research for this thesis was mainly carried out at the Computer Graphics Lab at the University of California, San Diego (UCSD). The study abroad lasted seven months from November 11th 2007 to June 15th 2008. The thesis was then written on the basis of the work from June 17th 2008 to August 6th 2008 at Aalborg University.

The stay at the Computer Graphics Lab at UCSD was an amazing experience and I would like to thank both Claus B. Madsen and the people at the lab, especially Henrik Wann Jensen for making it possible.

For complete understanding of the report a technical and scientific level corresponding to that of 9th and 10th semester students at the Department of Media Technology and Engineering Science is assumed.
Literature references are written in brackets and listed alphabetically in the bibliography. References to appendices are listed alphabetically as: Appendix A, B, etc.

Readers are encouraged to explore the enclosed CD. It is found at the back page of the report and contains this thesis, the source code for the developed applications together with various data that has been used in the process.

Aalborg University,
6th August 2008

Jesper Højgaard Pedersen

Contents

1 Introduction **7**

2 Preliminary Analysis **9**

2.1 Introduction 9

2.2 Previous Work 10

2.2.1 Environment Mapping 10

2.2.2 Image-Based Lighting 11

2.2.3 Image-Based Rendering 13

2.2.4 Global Illumination in Synthetic Scenes 13

2.3 Discussion 15

2.4 Proposed Method 16

3 Problem Definition **19**

3.1 Delimitation of Study 20

4 System Preview **21**

4.1 Preprocessing 21

4.2 Run-time 23

5 Problem Analysis **27**

5.1 Modeling Scene Geometry 27

5.1.1 Projecting Light Probes Onto Geometry 28

5.1.2 Parallax Distortion 30

5.1.3 Sampling Strategy 30

5.2 Fusing Light Probes 30

5.3 Shading From EM 35

5.3.1 Spatial Radiance Approximation 35

5.3.2 Frequential Radiance Representation 37

5.3.3 Comparison of Representations 39

6 Implementation **41**

6.1 Summed Area Tables 41

6.2 GPU Utilization 42

7 Results and Evaluation **45**

8 Conclusion **51**

8.1 Future Work 52

Bibliography **52**

1 Introduction

Sometimes, the world is not enough...

...and thus, in many respects it could be desirable to seamlessly integrate additional items into a scene. One way of doing so would be to overlay real-world imagery with computer graphics. This is what the field of Augmented Reality (AR) focuses on. More specifically, Azuma defined in [Azu97] an AR system based on three criterions. It must

- Combine real and virtual
- Be interactive in real-time
- Be registered in 3-D.

Such systems have a wide field of applications, and one can think of several scenarios, where it would be of great value to realistically augment existing, real-world scenery with computer generated content.

Car-manufacturer BMW is actively researching how AR can be utilised for an enhanced driving experience [TSK+05]. Here, one of the usages of AR is to create a next-generation heads-up-display (HUD), that directs the attention of the driver to dangerous situations around the car, see Figure 1.1.

Figure 1.1: *AR used in cars to improve security by visualising areas that require the drivers attention. (Adopted from [TSK+05].)*

Another example is within the fields of interior design. Here, the ability to see how a new piece of furniture e.g. a bed as shown in Figure 1.2 would fit into a given

scene without explicitly having to place the actual piece of furniture there would be a significant improvement to the work-flow.

Figure 1.2: *An example of how AR can be applied in the context of interior design. (Rendering by Pavel Diplodok, used with permission)*

In that sense, it would be interesting to be able to easily "sample" a large scene with complex illumination and followingly have the possibility to navigate virtually around the scene. On top of that, to be able to add more objects to the sampled scene and interactively see how such objects look in the scene in a true-to-life manner would be useful. Thus, the ability to enhance or augment large real-world scenes with virtual objects poses as an interesting problem to solve.

A crucial aspect of such a system is to have the virtual objects appear as integral parts of the scene. That is, the appearance must be consistent with the other objects in the scene. If an object for instance is placed in front of a bright light or window, the shadows of the object need to be cast in a direction away from the light and possibly onto other objects in the scene.

Grounded in this scenario, an initial problem for this thesis may be formulated as:

How is it possible to realistically augment a large, real-world scene with a virtual object?

2
Preliminary Analysis

This chapter serves as a preliminary analysis of possible solutions for the initial problem previously stated in the Introduction. The chapter elaborates on how to augment a real-world scene with virtual objects in a scene-consistent manner in the sense that the objects are correctly shaded, cast and receive shadows, and may be occluded by other objects in the scene. In order to do so, first a theoretical basis is established through a review of related previous works. Then, it is discussed how the previous work may be improved and synthesized into a new method, which solves the initial problem.

2.1 Introduction

One way of approaching the problem could be to thoroughly inspect the real-world scene and separate it into geometry, surfaces, and lights. Then, using digital content creation (DCC) tools such as Autodesk's Maya or Lightwave from Newtek, one could try to remodel the geometry and place the synthetic object in the scene. With sufficient knowledge of all surfaces and scene illumination, the lighting in the complete scene can now be simulated to render how the scene would look from specific viewpoint. Given correct input data and an precise model for light transport, this solution yields the desired result: A view of the scene with the virtual object included.
This is basically the subject of Global Illumination (GI), which was formalised in The Rendering Equation by Kajiya in [Kaj86]. Generally, the equation defines radiance leaving a point on a surface in a given direction as the product of the incident radiance and the surface's bidirectional reflectance distribution function (BRDF). For an introduction to radiometry in general, see e.g. [Dut03] or [Jen01]. However, in praxis this is not a viable solution. Even though the rendering equation is being approximated in real-time, it still requires a great deal of work to model large scenes down to the last detail.

A radically different approach could then be to take a lot of pictures of the scene. By taking a picture from nearly all points in space in all directions, the scene could later

be virtually restored and navigation through it is done by simply picking the correct image. This equivilates sampling a subspace of The Plenoptic Function[AB91]. The Plenoptic function $P(\phi, \theta, x, y, z, \lambda, t)$ is an idealized 7-D function that parameterizes the light given a point in space, direction, time, wavelength. In other words, the pleoptic function stores all possible picture at all times for all locations. Clearly, this is also a bit too much, and one could settle for something less extensive.

Letting the rendering equation and the plenoptic function span a possible solution space, the following section evaluates the relevant work that has previously been carried out within the subject matter.

2.2 Previous Work

In this section, prior research of relevance to the problem is presented and briefly evaluated. The presentation of notable previous works places this project in a scientific context and helps clarify where novel ideas are introduced.

2.2.1 Environment Mapping

A classical method for rendering highly specular objects is through the use of texture mapping[BN76]. This was later extended by Miller et al. in [MH84], who used photographs as environment maps (EMs) to create very realistic compositions of real-world scenery with synthetic objects.

In [ML03] Meyer and Loscos presented a method for computing the reflection of static environment on highly specular dynamic objects. Their particular application of the technique was rendering vehicles in a city. A naïve approach to simulate the reflections of buildings on a moving car would be to generate an EM for each frame and project the EM onto the car geometry. However, such an approach is very fill-rate intensive as the scene needs to rendered multiple times from different views to generate the EM. To alleviate this, Meyer et al. introduce Reference Environment Maps (REMs). A REM is a cube-map, that has been pre-computed at some specific position in the scene, see Figure 2.1.

To calculate the reflection on the surface they cast a ray in the direction of the reflection vector and compute the point I where the ray intersects a building. As the building are represented in a height-map, this can be done very quickly. The vectors $\overline{C_A I}$ and $\overline{C_B I}$ are used as look up vectors in the REMs. Finally, the REMs are weighted based on their distance to the point I_p which is the projection of I onto $\overline{C_A C_B}$.

While this method speeds up the computation of dynamic EMs, it also possesses some limitations. Due to the precomputation of the REMs the method need to assume a static environment and because of the simplified raycasting scheme, the

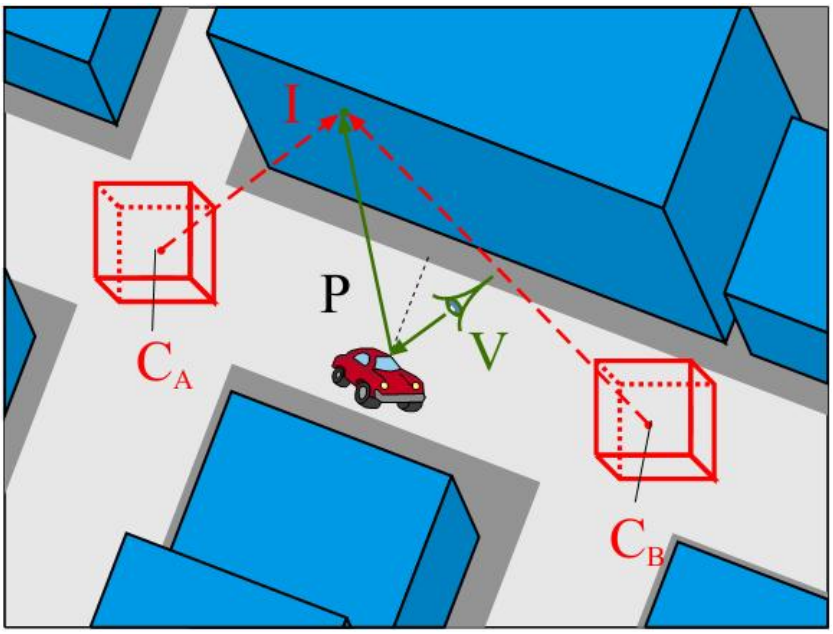


Figure 2.1: *Reference Environment Maps (C_A and C_B) are prerendered at representative locations in the scene. The local reflection at point P may then be calculated based on the intersection point I and the maps C_A, C_B. From [ML03].*

geometry of the environment must be representable as a height field. Further, the authors only consider fully reflective surfaces. Another restriction they do not address is that the raycasting does not handle concave, self-reflecting objects or other objects that may appear in the scene.

Wang et al. also suggest a method for interpolating EMs in real-time to generating a dynamic EM[WWL+04].

2.2.2 Image-Based Lighting

Debevec presented in [Deb98] a novel method for compositing computer generated imagery with real-world scenes. The key insight was, that when placing a synthetic object in the a scene, only a limited part of the scene is photometrically affected by the object. Thus, for the majority of synthetic objects (not counting objects with highly emissive surfaces) the lighting in most of the scene remains the same after the synthetic object has been added. This effectively allows for a partition of the scene into three components: the distant part, which remains unaffected by the object, a local part of the scene that changes due to the addition of the object, and thirdly, the synthetic object. We only need to consider the incident illumination from the distant scene at the position of the virtual object. Thus a light-based model of the distant scene will suffice. The incident radiance at the point of interest is captured using a light probe. In practice, this can be obtained using a zoom lens and photographing a

highly reflective sphere at different exposure levels to achieve a high dynamic range (HDR) environment map (a.k.a. radiance map). The local scene component may often be modeled by a simple geometry and Debevec proposed to manually fit a simple BRDF until sufficiently similar appearance was found. Such a brute-force approach inherently limits the complexity of applicable BRDFs, but the results from assuming trivial Lambertian surfaces are nonetheless convincing.
Having the distant scene as a radiance map and the simple local scene, a global illumination solution including the virtual object is straight-forward to calculate in a raytracer. This may however be slow if done brute-force since the light contribution from each pixel in the radiance map must be calculated. Not long after Debevec published his findings ATI were able to do a real-time implementation [ATI] of the method using preconvolved irradiance maps.

Even though the presented method, commonly known as Image-Based Lighting (IBL), can be used for highly realistic renderings, it has some limitations. Real-world surfaces may exhibit both anisotropic and spatially varying[1] properties. By using a single light probe, the surface radiances in the scene are only captured at one point and in one specific direction. The incident radiance measurements from the light probe are only valid in the vicinity of where it was captured. From this follows that not all surface properties are not captured and reproduced correctly.
Various improvements to this limitation have been proposed; recently a stereo probe setup was presented in [CCC08]. While this notably improves the representation of spatially varying radiance distributions, the method is still only applicable in smaller scenes where two light probes are sufficient to captures the illumination. This restriction applies similarly to other solutions such as the Lumigraph[GGSC96] and light field rendering[LH96; MRP98; WAA+00].

IBL is may also be used for real-time shading of objects without diffuse preconvolution. In such cases, the radiance distribution from the light probe image is approximated at run-time using different approaches. In [ML07a] Madsen et al. presented a method for augmenting a live video stream with a virtual object. The object was shaded in real-time based on a light probe image that had been acquired in the scene at the position of the object. The light probe image was the approximated by a configuration of directional light sources, which had their position and colour based on the Median Cut algorithm[Deb05].
Another method for placing a number of directional light sources was developed in [BSKS05]. Here, they decompose the light probe image into a number of cells, so that each cells has approximately the same radiance and the size of the cells is inversely proportional to the radiance. The decomposition is done by placing random samples and then use Lloyds relaxation to even out the sample distribution resulting in a Voronoi mesh.
A third option for sampling a light probe image in real-time is to use Importance

[1] For the sake of completion, models of temporally varying BRDFs also exist, see e.g. [GTR+06].

Sampling[HSK+05]. Here, it is shown how importance sampling can be used for sampling a video stream. By sampling the stream using a specific sequence, each of the samples can be indexed consistently over many frames, whereby their position and colour can be low-pass filtered to obtain temporal stability.

Common to the above-mentioned systems is that they only use one light probe to capture the illumination in the scene which requires the object to stay in the vicinity of where the light probe was captured.

2.2.3 Image-Based Rendering

The idea of storing the radiance distribution in a scene as a set of images and from that set create novel view points is known as image-based rendering (IBR). Using this technique it is possible to capture anisotropic BRDFs in the form of view-dependent texture mapping (VDTM). Debevec first published such a method in [DTM96], which was later refined in [DYB98]. The key idea is to capture images of a given scene from multiple view points. Then, to create a novel view point, the three images from the original view points closest to the new view point are located. The three images are projected onto a simplified geometry model of the scene and blended with barycentric weights determined by how close they are to the new view point. One advantage of this method is, that it requires only a coarse geometric representation of the scene. It is less attractive, that the cameras need to be accurately calibrated, and a high number of view points are needed to capture specularities adequately and leads to high memory requirements. This was sought alleviated in [MA07] by fitting the VDTMs to a parameterised model to avoid explicitly storing all texture data. However, their solution does not scale well to large scenes with many different surfaces.
In a similar way, [EDM+08] recently presented Floating Textures. Is this work, they propose a new method for IBR with less strict requirements to camera calibration. Similarly to VDTM, from a novel view point, images from multiple view-points are back-projected onto a proxy geometry. However contrary to applying a simple linear blending scheme, they use an optical flow algorithm [HS80] and novel view points are rendered at interactive rates.

2.2.4 Global Illumination in Synthetic Scenes

Rendering a virtual object into a large scene requires knowledge of the scene radiance distribution in the vicinity of the virtual object. Calculating the scene radiance distribution is also an integral part of certain methods for simulating global illumination (GI) in synthetic scenes. Thus, it may be of interest to have a look at such methods.

A key component of GI is to account for indirect lighting. This is opposed to direct

lighting, where only light arriving from distinct light sources is considered. The indirect lighting is light that that arrives at a surface point from all other surfaces. Another way of putting it to say that light is diffusely reflected between surfaces, and it is this diffuse interreflection that need be calculated.
A method for this was presented by Ward et al. in [WRC88]. It is computationally infeasible to calculate the irradiance at all points on all surfaces in a scene. However, irradiance varies slowly over a surface so it can b argued that it is sufficient to perform the irradiance calculation at certain locations, so-called cache-points. Then, when rendering the image the irradiance at a surface point is obtained by linear interpolation of the nearest irradiance caches. This reduces the computational cost at the expense of image quality.
The irradiance caching method was further improved by Ward and Heckbert in [WH92]. Similarly to irradiance caching, the irradiance at a surface point is calculated by sampling the incoming radiance and integrating over the hemisphere. During the hemispherical sampling additional information such as the distance, brightness, and direction of all visible surfaces can be obtained as well. By using this extra information the gradient of the irradiance function can be calculated. Knowing the gradient, a higher-order interpolation scheme may be employed, effectively increasing the image quality with little computational overhead.

Instead of calculating the irradiance on a surfaces, Greger et al. suggest storing and calculating the illumination using an Irradiance Volume[GSHG98]. A visualisation of an Irradiance Volume is shown in Figure 2.2. In this representation, the scene radiance field is sampled in a tree-dimensional grid where an irradiance distribution function is computed. For each cell in the grid, the incident radiance on a differential surface is sampled and the irradiance is calculated. This is done in all directions, resulting in a 5-D function, three indicating the position and two representing the direction of the surface normal. The resulting irradiance distribution function is a continuous, low-frequency, spherical function.

To shade an object, the irradiance volume is queried on a per-vertex basis. For each vertex, the irradiance is calculated from trilinear interpolation of the eight nearest cells, Figure 2.2B shows how this is done in 2-dimension. It is therefore assumed that the irradiance field varies smoothly wrt. position and direction. As the irradiance distribution function is a low-frequency, spherical function Ramamoorthi showed in [RH01] that it can be compactly represented in a spherical harmonics (SH) basis. Moreover, it is faster to project the irradiance function into an SH basis and shade an object using the SH coefficients rather than performing a hemispherical integration of the incident radiance. This is discussed in further detail in Section 5.3.2.

In [MPM02] and [NPG03] achieve diffuse interreflection at interactive rates by representing an irradiance volume as a 3-D grid of cube maps. Each cube map is projected into a SH and the resulting coefficients can be interpolated to calculate a SH representation of the irradiance for a given vertex. A new implementation of

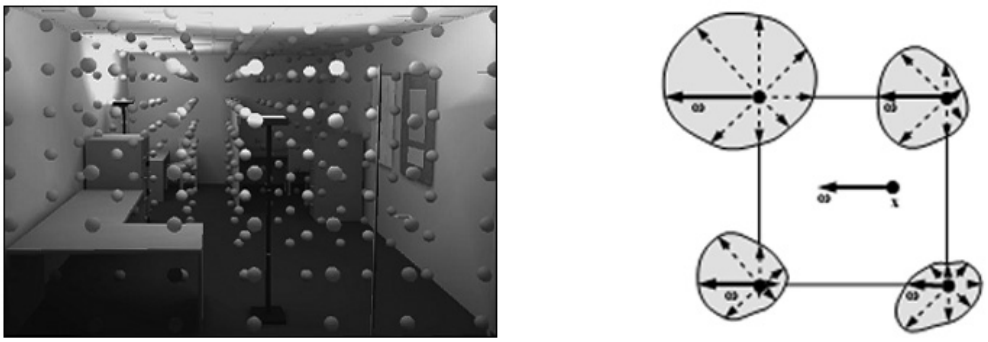


Figure 2.2: *Figure A shows a scene rendered using an Irradiance Volume with each cell represented with a sphere. Figure B illustrates how to calculate the irradiance for a vertex by interpolating between the nearest cells. From [GSHG98].*

this method was presented in a demo from ATI[Per07]. Using Shader Model 4 hardware and DirectX 10.1, one can write a geometry shader that makes it possible to render to multiple cube maps in one pass. This significantly increases the framerate compared to previous implementations that require multiple passes per cube map update.

2.3 Discussion

The initial problem was to realistically place a virtual object in a large, real-world scene. For the object to appear realistic and as an integral part of the scene, it must be shaded and cast shadows consistently with the scene illumination. The illumination in a real-world scene is rarely constant and will often vary depending on which various local light sources that may be present. So, to realistically shade a virtual object moving around in a large scene, it is necessary to take the varying illumination into account.

In synthetic scenes where the lighting is simulated using a GI tool, one way of representing the illumination is with an irradiance volume. As the irradiance volume is a discrete representation of the scene's irradiance field, it therefore stores how the illumination varies across the scene. However, by only storing the irradiance, information about the lighting is discarded and consequently, objects with glossy surfaces cannot be rendered using this method alone.
Perhaps more importantly, the irradiance volume is only suitable for scenes with low-frequency illumination. To properly capture high frequency illumination, i.e. where the illumination changes abruptly from one point to another, a prohibitively large number of irradiance samples will be needed. For scenes with high-frequency

illumination, a representation of the radiance field therefore need to be considered. Moreover, to be able to render objects with arbitrary surfaces the scene radiance distribution need to be known near the object.
A way of obtaining the incoming radiance on the object surface is to generate an environment map centered around the object. This is similar to IBL methods and allows for very realistic, real-time shading. Typically, however, only one light probe is used, which leads to limited movement of the object within the scene.

To capture spatially varying illumination in a real-world scene, one method could be to acquire multiple light probes from around the scene. Then, by "fusing" the light probes the radiance distribution at the location of the object can be determined. In order to do so, it is necessary to determine which light probes are visible from the object. A possible solution could be to model a low-detail representation of the scene geometry and back-project the light probes onto that proxy geometry. Thereby, visibility calculations similar to those performed in the shadow mapping algorithm[Wil78] can be applied.
Additionally, the object can interact with the scene in terms of casting shadows onto the scene geometry. As the surfaces in the scene are captured from different viewpoints when using multiple light probes, more complex, non-lambertian surfaces are accounted for as well.

The method requires the scene to be reasonably approximated by the proxy geometry and back-projection of the light probes. That makes the method well suited for simulating lighting in rooms, halls, and similar, confined spaces. Outdoor landscape scenes may not benefit notably from using this method as the scene is illumination is from a distant light source (e.g. the sun or moon) and thus spatially invariant. In scenes containing both far and near lighting, e.g. if distant scenery can been seen through a window in a room, it may be necessary to handle parallax errors.The proposed method is only applicable to static scenes. The virtual object moves freely around but the scene geometry and the light probes are static. Havran has looked into using video cameras as source, but a method is still needed for approximating the scene geometry adequately in real-time.

From this discussion, a novel method is proposed to render a virtual object into a large scene with spatially varying illumination. The method, that may be seen as a hybrid between image-based and model-based techniques, is briefly described next.

2.4 Proposed Method

The proposed method can be divided in two key parts:

1. Offline: Generation or acquisition of scene radiance

2. Run-time: Rendering the virtual objects together with the scene

1. Offline

1. Acquire the illumination in the scene
This may be done using a reflective ball on a tripod as light probe. To account for the poor sampling of radiance coming directly behind the ball, two or more pictures are taken and stitched together. Alternatively, a camera with a fish-eye lensmay be used and possibly lead to better results.
The captured light probes must be HDR images.

2. Model a simple geometrical proxy of the scene
Calibrate the light probes to match the geometry when projected back onto the geometry to get the position in the virtual scene that corresponds to the position of the light probes in the real scene.

3. Generate a depth map for each light probe
A virtual camera is then placed at the positions in the scene that corresponds to where the light probes were captured and the depth is calculated for each texel in the EM. Since the depth is calculated for each corresponding texel the depth maps are omni-directional as well.

2. Run-time

1. Shade the virtual object
Generate a dynamic, cubic EM by placing the camera in the center of the object and render the scene for each side in the cube map. When generating the EM determine which light probes are visible for a given direction and fuse those according to how they contribute to the incident radiance.

2. Render the scene
Back-project the light probes onto the proxy geometry and render scene together with the object.

The remaining parts of the thesis is about the problems that this method entails and how they can be solved. The first step is to define the exact goals, which is done in the following chapter.

3
Problem Definition

In this chapter the objectives for the rest of the thesis are defined. This is done based on what was found in the previous chapter.

From the Preliminary Analysis it was found that one way to augment large scenes with virtual objects could be to sample the scene illumination using multiple light probes and back-project the probes onto a proxy geometry of the scene. Then a local EM could be generated from fusing the light probes and using the depth information available.

For this method to work there are three key areas that need to be further explored which the rest of this thesis will focus on. The three focus-areas are

1. Geometric Scene Model
2. Back-projection and fusion of light probes
3. Shading and shadows from a dynamically generated EM

Developing valid solutions for these three sub-tasks result in a solution that can be used for augmenting a real scene with a virtual object.

For the augmentation of an real environment it is crucial that the appearance of the virtual object is realistic. This means that the developed method should be able to handle various aspects of shading and shadowing. For instance, the object may become occluded by other parts of the scene and only be partially visible. A complete method should also be able to handle both shadows that may be cast from the scene onto the object and shadows the object would cast onto the scene. Summed up into one sentence:

The virtual object should at all times appear and interact consistently with the scene in real-time.

Even within the scope bounded by the above-mentioned areas focus the task of augmenting a scene is far from trivial. It is therefore necessary to leave out certain aspects of a such task.

3.1 Delimitation of Study

To better focus on the afore mentioned areas the following limitations will need to be introduced:

- Shadows are only considered to a limited extent as this is a vast area of research in its own. The developed method should however be easy to extend to use well-known methods for shadowing.
- Calibration of real-world light probes to match the scene geometry is not considered as this can be done independent of the developed method.
- Similarly, the task of creating a suitable geometric model of the scene is not fully explored.
- Only time-invariant scenes are handled.
- Only use synthetic scenes are used. By rendering the scenes and creating the light probes using a global illumination renderer the quality of the method can be evaluated against a Ground Truth solution. Also, the limited time-frame of this project did not allow for testing the method with real-world scenes.
- For the sake of simplicity, the method focuses only on augmenting the scene with a single object. It should however be easily extendable to handle many objects.

A preview of the developed method is given in the following chapter. This will hopefully help to give the reader a more concrete idea of the concepts that are introduced in the later chapters.

4 System Preview

In this chapter an overview of the developed solution is presented. The intention is that by showing the methods in the context they are used the upcoming purpose and discussion of them will be clearer. To put it in another way, this chapter focuses on *what* has been done. *Why* and *how* is then to be discussed in the next chapters. Moreover, the preview should help to clarify and concretize the goal of the thesis as defined previously in the Problem Definition.

This chapter presents a method for augmenting a large scene with a virtual objects. The method takes the varying illumination in a scene into account and realistically shades the object in a scene-consistent manner. The method is divided into two parts. The first part is a preprocessing step that prepares the required scene information. The second part is then the run-time process that augments the scene based on the given information.

The developed application can be found on the accompanying CD in the back of this thesis.

4.1 Preprocessing

Depending on whether the application is used with a real or a synthetic scene, the preprocessing differs a bit. In the following, it is assumed it is a real-world scene that is sought to be augmented.

1. First, the illumination is the scene is acquired. Obtaining an omni-directional image from a point in the scene is done using light probes. Traditionally, light probe images are created by taking one or more high dynamic range (HDR) photographs of a reflective ball placed on a tripod and stitching photographs together. Another option is to use a wide-angle lens. Figure 4.1 shows an example of such a light probe reprojected to Latitude-Longitude (LL) format.

For complex scenes, one should acquire enough probes such that the radiance from all major surfaces is captured by at least one light probe. The light probes are then converted to cube maps for later use.

Figure 4.1: *Example of a light probe in Latitude-Longitude format. The light probe is created by stitching two photos taken with a fish-eye lens. The light probe is captured in HDR so the pixel values have here been mapped to a low-dynamic range to better visualise them. (Courtesy of Claus B. Madsen)*

2. The next step is to create a rough geometric model of the scene–sometimes referred to as a scene-proxy. Figure 4.2 shows two examples of proxy-geometries.

3. The last step in the preprocessing is to append depth information to the light probes. There are multiple ways to generate the depth information, e.g. by loading the scene into a raytracer or utilizing programmable shaders on a GPU. Then, determine the positions in the virtual scene that corresponds to where the light probes were captured in the real scene. For each of the positions the scene is rendered in six directions to generate a cube map (which must have the same dimensions as the light probe cube maps). Now, when rendering the scene the depth buffer is stored instead the usual pixel intensities, so each texel contains the distance from the light probe to the geometry[1]. Figure 4.3 shows an example of a depth buffer with the content mapped to grayscale values and converted to LL form.

If the method is used for purely synthetic scenes Step 1 can be omitted and the light probes can instead be created using a global illumination renderer together with the depth maps in Step 3.

With the initial preparations out of the way it is now time to make use of the collected information. In the following section a visual run-through of the main algorithm in the method is given. This part works the same regardless of whether the input scene is synthetic of real.

[1]Make sure the depth buffer stores the distances linearly and not $1/z$ as is commonly used in hardware implementations.

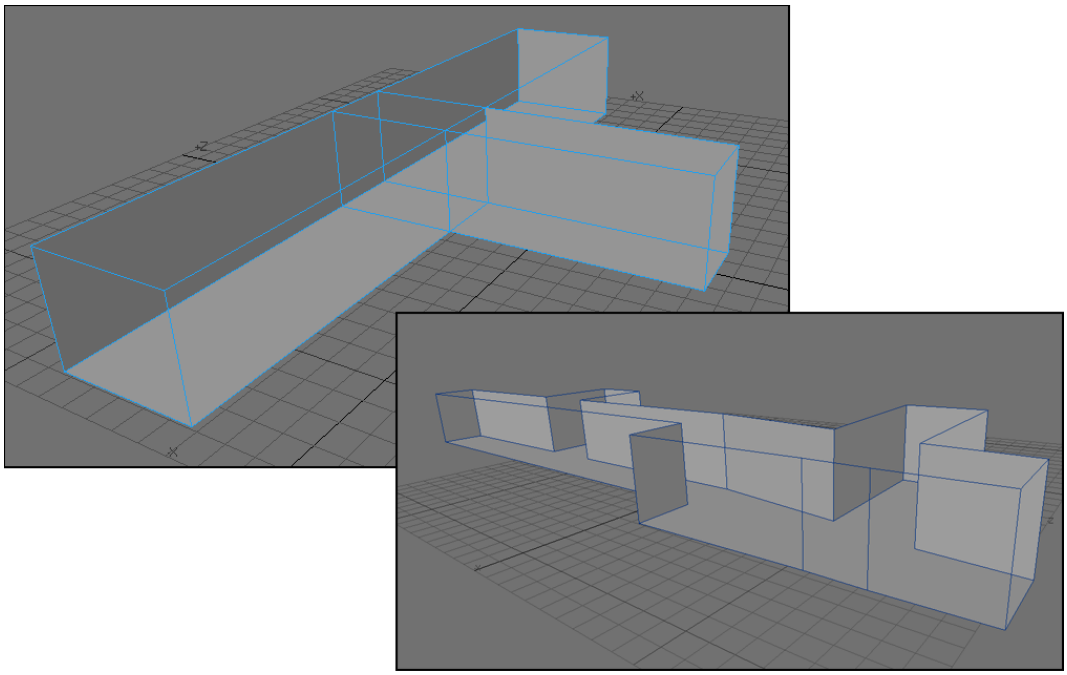


Figure 4.2: *Two examples of a scene-proxy. The top-most model is a synthetic test scene, while the one in the bottom corner is a model of a hall in the Computer Science and Engineering Building at the University of California, San Diego.*

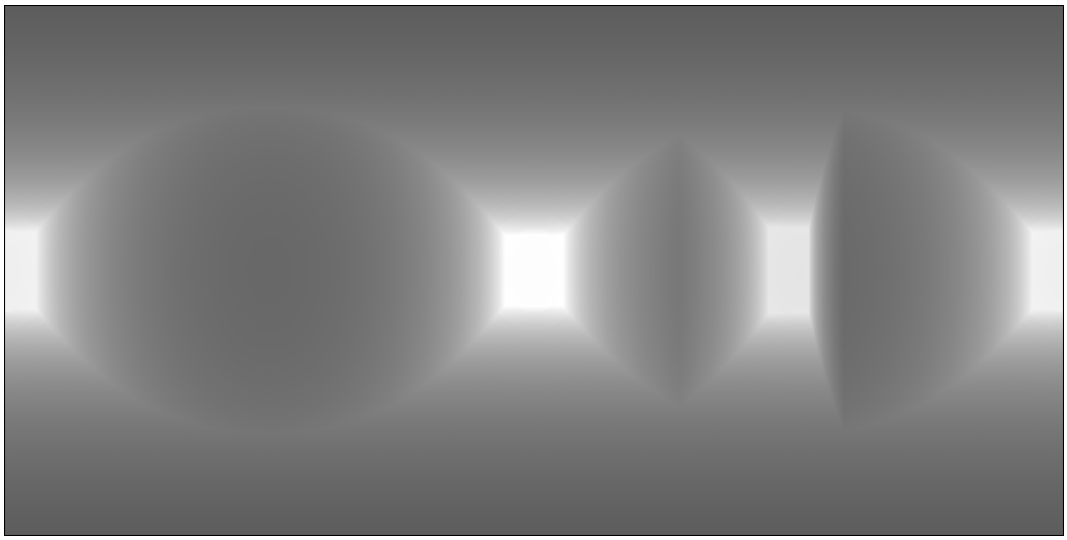


Figure 4.3: *The test scene rendered as a depth map. The distance is mapped to grayscale where brighter texels are furthest away.*

4.2 Run-time

The run-time part can be divided into two sub-tasks; shading the object and rendering the scene geometry.

To shade the object a cube map is created and dynamically updated for each frame. This is done by placing the camera in the center of the object and rendering the directions of each face in the cube. For each render pass the incoming radiance is calculated for each fragment. This is done by rasterizing the scene geometry with a custom shader that fuses the light probes in the scene. First, the Multiple Light Probes (MLP) shader determines the 3-D position of the fragment. By calculating the distance from the fragment to each light probe center and comparing it with the depth information at the according light probe texel it can be determined which light probes are visible. This is illustrated in Figure 4.4

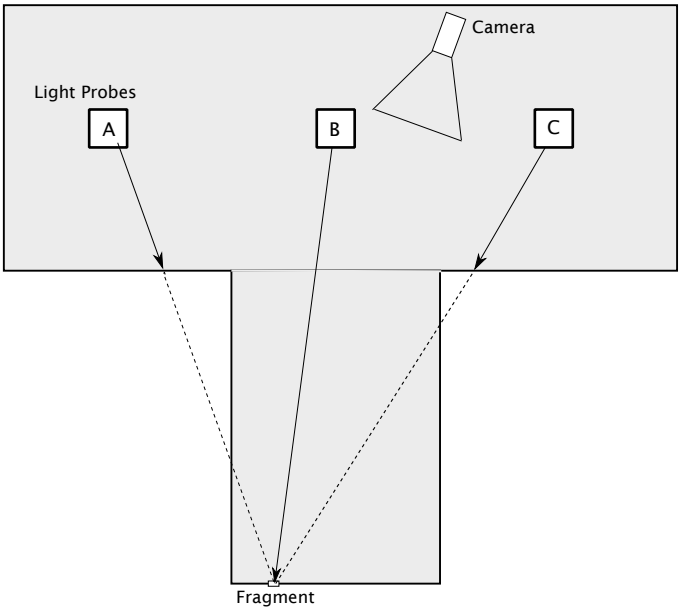


Figure 4.4: *The distance from a fragment to the light probes is calculated and compared to distances stored in each light probe. If the distance from fragment to light probe is smaller than the stored value the light probe is not visible and is omitted when calculating the exitant radiance from the fragment.*

The radiances from the visible light probes are blended according to how close their direction matches the direction to the camera. This is done for all texels for the 6 faces of the cube map and each texel stores the incident radiance and the distance to the scene-geometry. This cube map can now be used to shade the object.

To simulate highly reflective surfaces the cube map can be mapped directly onto the object. However to render more diffuse surfaces the incoming radiance is approximated using a number of directional light sources using the Median Cut (MC) algorithm. The MC algorithm needs the input cube map to be converted to LL map so this is performed on the GPU in a shader, see Figure 4.5.

Using the Median Cut algorithm, the LL map is recursively divided into regions of

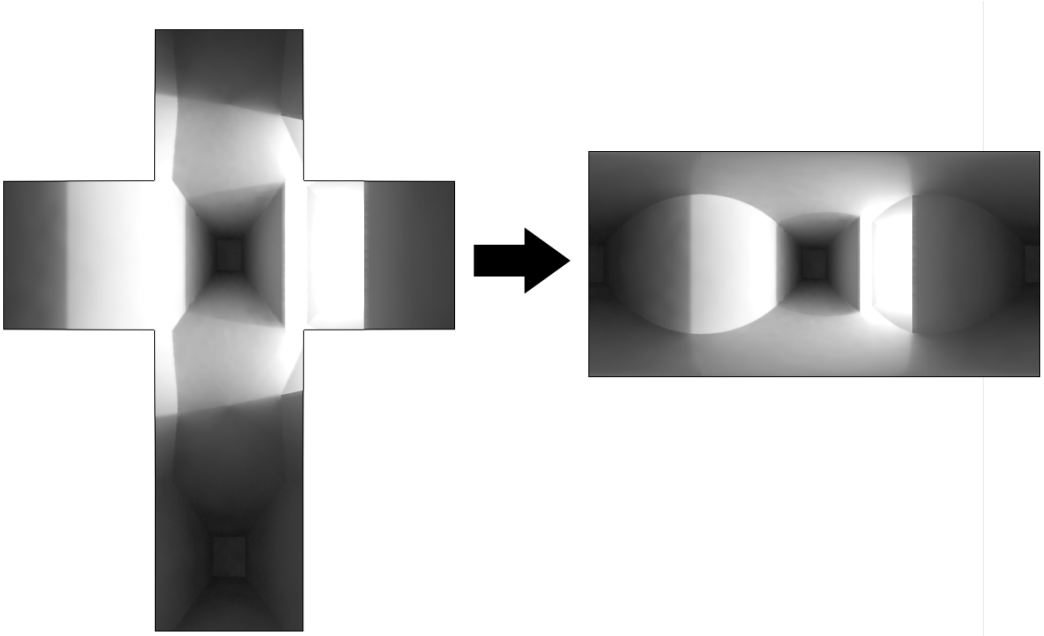


Figure 4.5: *The incoming radiance at the position of the object along with the distance to the scene geometry is sampled and stored in a cube map. The cube map is then converted to LL map on the GPU.*

circa same intensity, where each region is represented as a directional light source. The result of applying MC to an LL map can be seen in Figure 4.6 where each purple square is the position of a light source.

In the last step, the scene geometry is rasterized using the MLP shader similarly to when the dynamic cube map was generated. The object is shaded using the newly found positions of the directional light sources. Since the LL map also holds the distance to the geometry the light source can be localised in the scene using spherical coordinates. Depending on the number of light sources, both lambertian and glossy BRDFs can be rendered.

Since all calculations are performed in HDR, the final render pass is to tone map the frame buffer before it is displayed on the screen. A screen shot from the application is shown in Figure 4.7.

The run-time part of the method here in Section 4.2 runs unoptimized in real-time on a GeForce 6800GT.

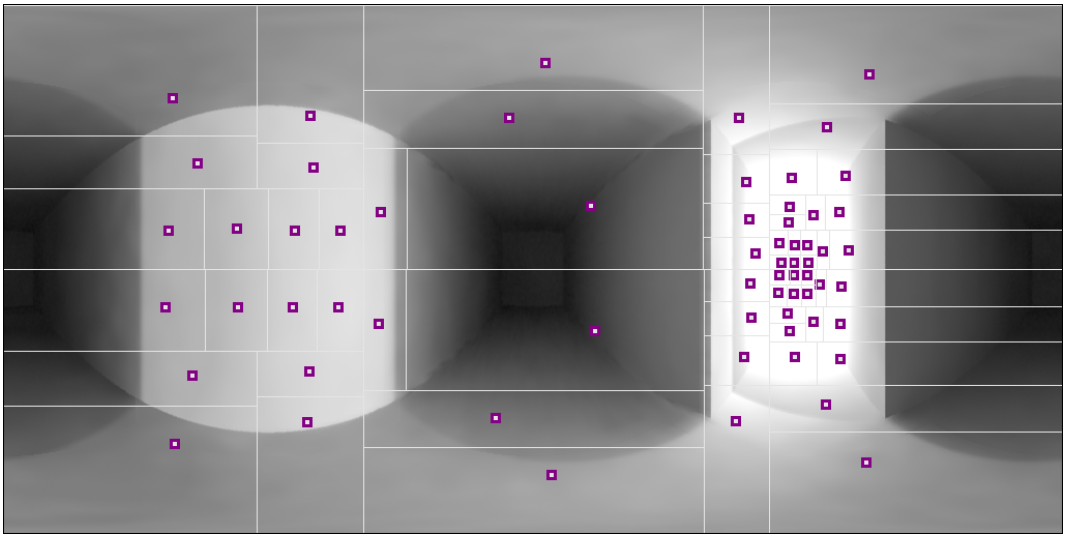


Figure 4.6: *Result of applying the Median Cut algorithm to the LL map. The purple squares indicate the positions of where the directional light should be placed to approximate the incident radiance.*

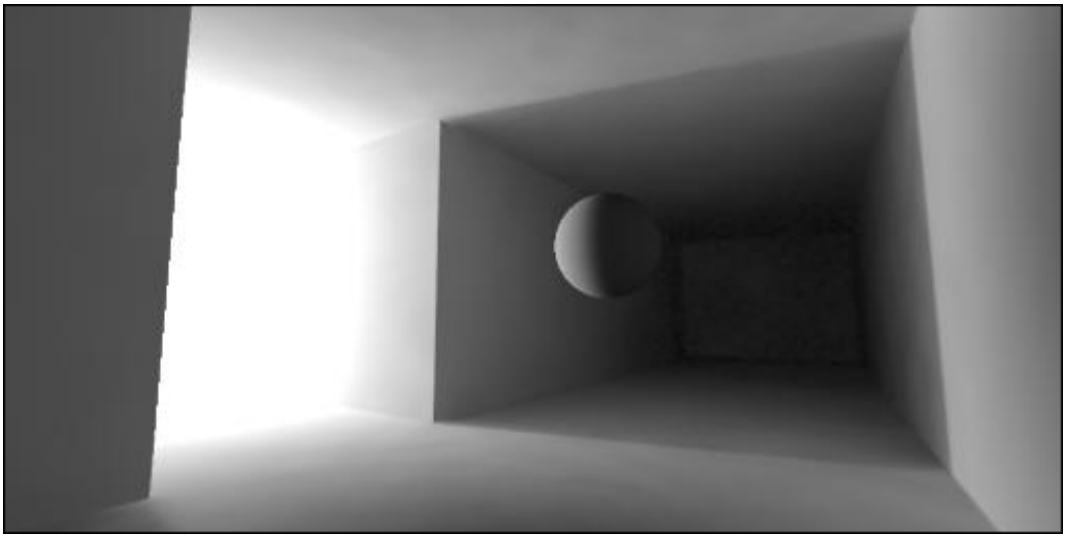


Figure 4.7: *A scene, albeit synthetic, has been augmented with an object.*

5
Problem Analysis

This chapter seeks to develop a method that may fulfill the goals stated previously in the Problem Definition. This is done based on the more general method that was presented in Section 2.4. First, the representation of the scene is discussed and then a method for handling the scene illumination is developed. Finally, two approaches for shading the virtual object are explored to find out which meets the goals stated in the Problem Definition better. The fundamental difference between the methods is that one is based on a spatial representation of the scene illumination whereas the second method works in the frequency domain. Both methods are then discussed with respect to the goals, and one will be selected for further implementation.

When augmenting a scene with a virtual object there are two criterions that must be met to make the object appear as an integral part of the scene. The two criterions are

- Geometric coherence
- Radiometrical coherence.

Modeling an explicit, although approximate, scene geometry can help achieve both requirements. The scene geometry can be used for determining spatial interaction with the virtual object, and the geometry further serves a radiometric purpose as a part of the method for shading the object and when casting shadows.

5.1 Modeling Scene Geometry

Accounting for geometrical coherence is the task of determining the spatial interaction between the object and the scene. This serves as an important clue to the position of the object in relation to other non-virtual objects and essentially let the virtual object “become a part of the scene”.

Reconstruction of the scene can be done either manually or more-or-less automatically, where the latter also is known as Image-based Modeling (IBM). Depending

on the complexity of the scene and how precise the reconstructed geometry should be, IBM requires that the scene is photographed from many view points. Since this is also the case when capturing the light probes, it could be considered done in the same set. Such an approach is however beyond the scope of this project, so it was chosen to manually create a low-detail approximation of the geometry. Geometries are created by roughly measuring out the dimensions of the scene and building a model of it in a CAD program based on these measurements. An example of two models was shown in Figure 4.2 on page 23. This is acceptable since highly detailed models are not needed for AR.

Having the virtual object and the scene geometry in the same space, it is now possible to resolve occlusions. This is done in real-time in hardware by rasterizing the geometries to the same depth buffer and enabling depth-testing on the GPU. Now, when the augmented scene is rendered the z-buffer algorithm is applied and non-visible fragments are discarded. On older hardware with 16-bit depth buffers a phenomenon known as "z-fighting" could occur due to lack of precision, but on modern GPUs with 24- or 32-bit buffers this is less of a problem.

With the geometry in place, the next step is to handle the information in the light probes. This is done in the following sections.

5.1.1 Projecting Light Probes Onto Geometry

With a geometric model of the scene in place the information from the light probes now need to be mapped onto that geometry. This texture mapping can be done in various ways. The typical way of doing it is via uv-mapping where the geometry is transformed to a 2-D coordinate system. The texture is mapped onto the geometry according to the position of the faces in the uv-coordinate system, see Figure 5.1A. The generation of texture coordinates can also be done analytically using for instance cylindrical, spherical, or planar coordinates, Figure 5.1B. Finally, the texture can be projected onto the geometry from an arbitrary view point, illustrated in Figure 5.1C.

That means, one way of mapping the light probes onto the scene geometry could be to first project them into a reference texture space which could then be mapped onto the geometry using standard techniques. It is however not necessary to generate explicit texture coordinates. Knowing the position and rotation of a light probe in the scene, the radiance data can be projected back onto the geometry using some canonical reference frame. This is shown in Figure 5.2.

The affine transformation from real world to virtual is

$$E = [R \,|\, t]\,, \tag{5.1}$$

where R is a 3x3 rotation matrix and t is the translation. Since the geometry is only

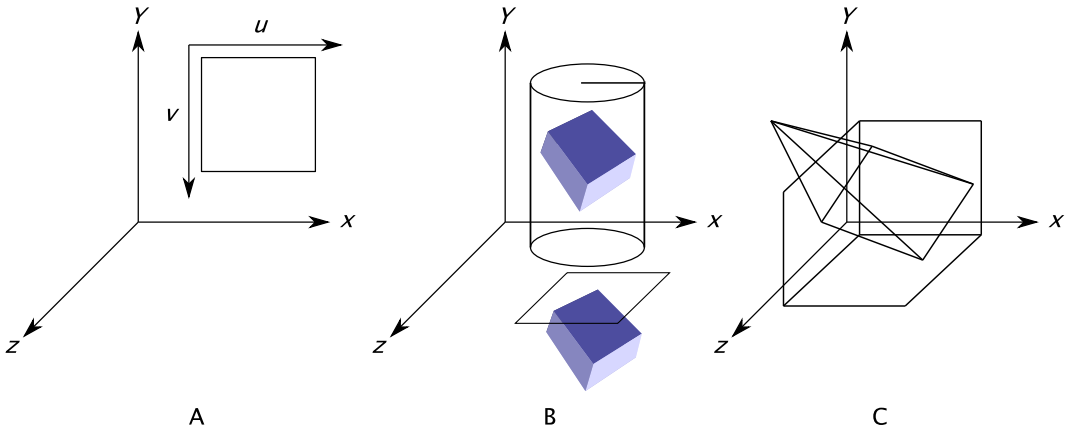


Figure 5.1: *Different types of mapping texture to geometry. In A is shown uv-mapping, B exemplifies analytical mappings such as cylindrical and planar projection, and in C is shown projection from an arbitrary view point.*

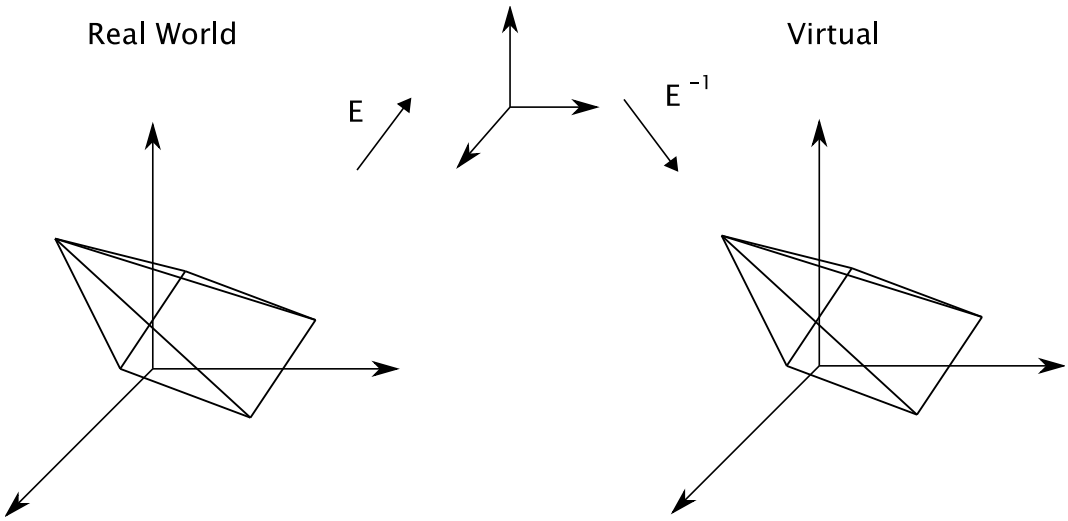


Figure 5.2: *The position and rotation of a light probe can be restored in the virtual scene using a reference frame.*

a rough model, E^{-1} can be approximated manually beginning with "a qualified guess" and interactively calibrating until the light probes are in place. If not all surfaces has been captured by a light probe it may be necessary to employ some hole filling algorithm, see e.g. [DTM96].

Now, when rendering the scene from the virtual camera, the back-projection onto the geometry is done by multiplying all vertices with the inverse view matrix and the position of the vertices are linearly interpolated per fragment. Then, for each fragment a vector is created from the fragment position to the position of the light probe, and this vector can be used as a lookup in the light probe cube map.

5.1.2 Parallax Distortion

In scenes with large depth variation, parallax errors may occur. A typical scenario would be if the scene to be augmented is a room with windows. One way of handling this could then be to model the windows as hole in the scene geometry and have it surrounded with an additional "skybox" (infinitely) distant away. For this skybox a separate light probe is used, perhaps captured from the roof of the building.
As the scene is rendered the depth of the geometry is checked as usual and it is registered if fragments are placed at this infinite distance. In such case, the separate sky light probe is used instead.

5.1.3 Sampling Strategy

In the Irradiance Volume, Greger uses an adaptive 3-D grid structure for storing the irradiance distributions. In praxis, capturing light probes in such a fixed structure may not be manageable for other than small grid-sizes. It is also not strictly necessary as the method presented in this thesis bears no restrictions towards a specific sampling pattern. Some general guidelines to the sampling, based on frequency considerations, is that more samples are needed for glossy surfaces and high-frequency illumination, that is, where the illumination changes abruptly, for instance near windows in bright day light.

The following section discusses the second goal in the Problem Definition, Chapter 3. The goal is to investigate a way of recreating the scene illumination in order to generate a dynamic EM that later can be used for shading the virtual object.

5.2 Fusing Light Probes

An important part of the scene augmentation process is to shade the virtual object. To do so, a method for estimating the incident radiance at the position of the object must be found. This is done by dynamically generating an EM from fusing both the information stored in the captured light probes and the knowledge of the scene geometry. In order to establish such a method, it may be useful to consider exactly what the data in the light probes represents.

Consider the radiance distribution function

$$L(x,\omega) = \int_{\Omega_n} L'(x,\omega')f(x,\omega,\omega')(\omega' \cdot n)d\omega', \tag{5.2}$$

where L is the exitant radiance from the surface point x in the direction ω and L'

is the incoming radiance from direction ω'. The direction vectors are unit-length. The term f is the Bidirectional Reflectance Distribution Function (BRDF), Ω_n is the hemisphere defined by the surface normal n, and $d\omega'$ is the differential solid angle in direction ω'. The terms in Equation (5.2) is visualised in Figure 5.3.

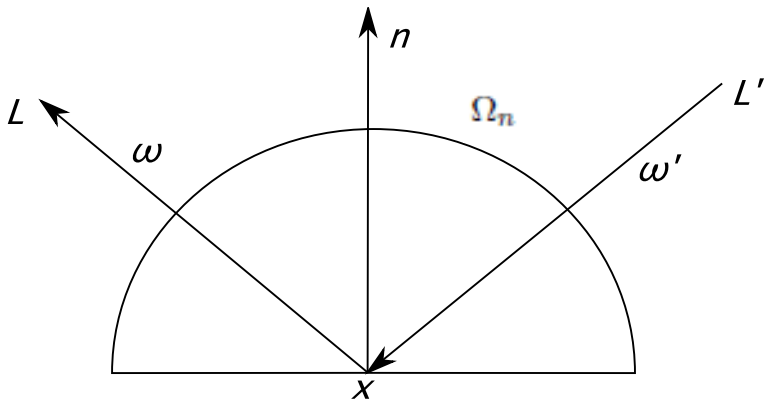


Figure 5.3: *Visualisation in 2-D of the terms in Equation 5.2 where L is the exitant radiance and L' is the incident radiance. The direction vectors ω and ω' are unit-length, Ω_n is the hemisphere defined on a surface point x with the normal n.*

When capturing light probes, it is essentially the solution to this spherical integration that is discretely sampled. This is illustrated in Figure 5.4

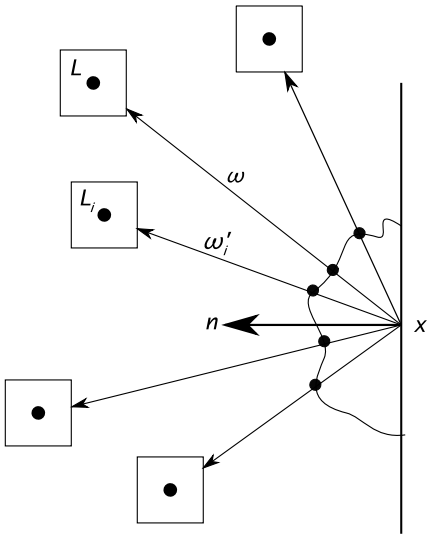


Figure 5.4: *The pixel value stored in a light probe $L_i(\omega_i')$ is a sampling of the radiance distribution at point x. To generate a dynamic EM at some position in the scene the value of $L(x, \omega)$ must be determined.*

Looked at it in this way, generating the dynamic EM at a position in the scene becomes the task of finding $L(x, \omega)$ for all pixels in the EM. Since it is unfeasible

to evaluate Equation (5.2) for each pixel, the incident radiance is estimated based on the light probes in the scene.

An obvious solution to this could be to simply use the value stored in a light probe L_i which represents the direction ω_i' most similar to the direction ω from point x

$$L(x, \omega) = \underset{i}{\operatorname{argmax}}\ \omega \cdot \omega_i', \tag{5.3}$$

where i indexes the set of light probes.

This approach has however a significant limitation. It only works for simple, convex scene geometries. If the geometry has concavities then not all light probes may be visible from all surface points and the radiance estimate will obviously be wrong if a non-visible light probe is selected. Therefore, this needs to be taken into consideration.

Since the scene is explicitly modeled, this can be used to include a visibility term in Equation (5.3). The idea is similar to the depth comparison in the shadow mapping algorithm[Wil78] and is illustrated in Figure 5.5.

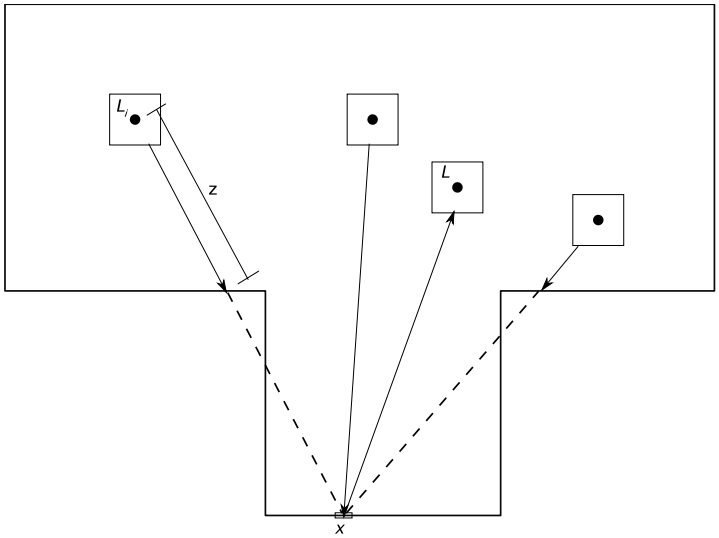


Figure 5.5: *A light probe is not valid if the distance from a surface point x to the position of the light probe is bigger than the Z value stored in the light probe for the same direction.*

The visibility term V is a binary function that can be written as

$$V(x, x_i) = \begin{cases} 0, & \text{if dist}(x, x_i) < L_{i,Z}, \\ 1, & \text{if dist}(x, x_i) > L_{i,Z} \end{cases} \tag{5.4}$$

where x is the surface point, x_i the position of the light probe L_i, and Z is the distance from the light probe to the scene geometry in that direction, also known as

depth values.

In praxis, the depth values can be stored in the alpha channel of the light probe image. Since the scene in assumed to be static, generating the depth values need only to be done once as a preprocess step. The light probes are stored as cube maps. For each face of the cube map the geometry is rasterized where the depth values are interpolated between the vertices. An example of a vertex and fragment shader pair for doing so can be seen in Listings 5.1 and 5.2 respectively.

Listing 5.1: *Vertex shader in GLSL for generating depth values and appending them to the light probe data*

```
varying vec3 fragPos;

void main()
{
       vec4 eyeVertex = gl_ModelViewMatrix * gl_Vertex; // view space
       fragPos = eyeVertex.xyz;

       gl_TexCoord[0] = gl_MultiTexCoord0;
       gl_Position = ftransform();
}
```

Listing 5.2: *Fragment shader in GLSL for generating depth values and appending them to the light probe data*

```
uniform samplerCube probe;
varying vec3 fragPos;                    // Frag pos interpolated

void main()
{
       vec3 dir = -fragPos;        // Vector from origo to fragpos
       vec4 colour = textureCube( probe, dir ); // Sample the probe
       colour.a = fragPos.z;        // Change alpha to store depth

       gl_FragData[0] = colour;
}
```

After pruning non-visible light probes from the set of all light probes, the method for generating the dynamic EM can be refined. Unless the scene illumination has been excessively sampled, selecting the best matching light probe may still be erroneous. Therefore, more advanced schemes for fusing multiple light probes are now considered.

The fusing of light probes is a matter of approximating the radiance distribution and evaluating how much is sent in direction of the dynamic EM. One way of taking multiple light probes into account, is to calculate a weighted average of the intensities of N light probes where the weights depend on how closely the direction

ω' from x to the i'th light probes matches the direction ω from x to the dynamic EM:

$$L(x, \omega) = \frac{1}{N} \sum_{i}^{N} L_i(\omega \cdot \omega_i'). \tag{5.5}$$

Instead of always requiring a constant number of light probes, N, the scheme can be made more adaptive by summing over all visible light probes, but excluding those with a similarity measure $\omega \cdot \omega'$ below a user-specified threshold.

Depending on the sampling scheme other fusing methods could be of interest too. In Figure 5.6 is shown different cases that may arise when the dynamic EM moves around in the scene.

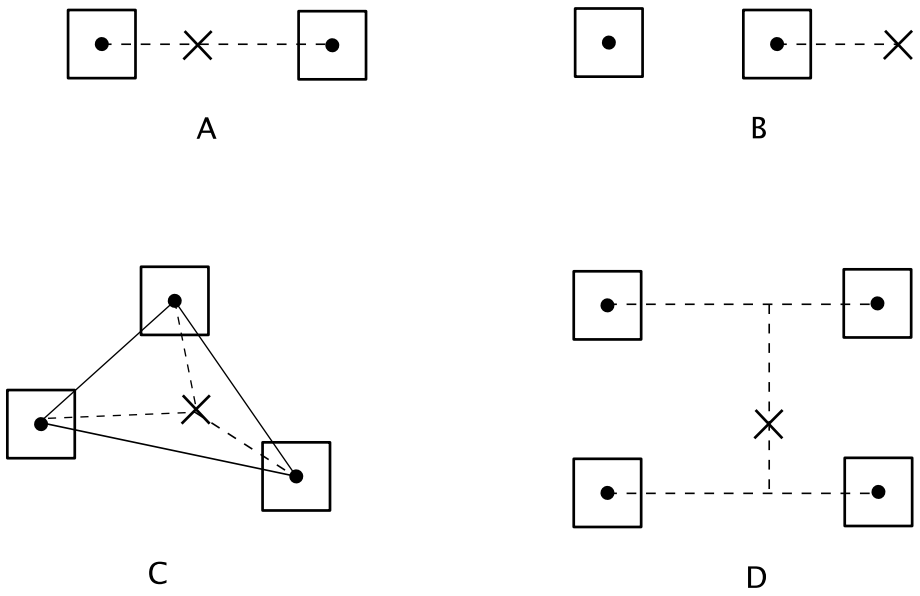


Figure 5.6: *Different cases arise when the object moves around relative to the light probes and alternative blending schemes for the dynamic EM may be considered. Figure A illustrates probes sequentially and a linear blend is adequate. Figure B shows the special case where the object is outside the probe sequence. In such case, no weighting should be applied. Figures C and D illustrate how either barycentric weighting or bilinear interpolation could be used depending on the sampling strategy.*

Having a method to dynamically generate an EM that stores the incident radiance at a given position in the scene, it is now possible to shade the object. This is discussed in the following section.

5.3 Shading From EM

The generated dynamic EM stores a discrete representation of the radiance distribution at that point in the scene. This can be thought of as each pixel in the EM is a small area light source, so shading an object now becomes at matter of calculating

$$L(x,\omega) = \sum_{i=1}^{N} L_i' f(x,\omega,\omega_i')(\omega_i' \cdot n)\Delta\omega_i', \tag{5.6}$$

where N is the number of pixels in the EM, L_i' is the radiance stored in the i'th pixel, ω_i' is the direction to the pixel, and $\Delta\omega_i'$ is the solid angle subtended by the pixel, which varies over the sphere depending on the used EM representation. To take this into account a weight map must be generated and used to modulate the pixel values, and the weights are normalized so

$$\sum_{i=1}^{N} \Delta\omega_i' \approx 4\pi. \tag{5.7}$$

Except for very small EMs, it is not feasible in real-time to sum over all pixels and therefore the EM need to be approximated in some way. This can be done either in the spatial domain or the EM can be transformed into frequency domain and approximated there. A method for each of the domains is now considered and the most suitable is chosen.

5.3.1 Spatial Radiance Approximation

When approximating the EM in the spatial domain, the task is to find a distribution of directional light sources that each represents an area of the EM. To do so, the Median Cut algorithm as presented by Debevec in [Deb05] is chosen because it is fast and conceptually straight-forward.

Median Cut

The Median Cut algorithm divides the EM into 2^K regions of circa same intensity where each region is represented by a directional light source. Since the regions are sought to have equal intensities, areas of high intensity in the EM are sampled more densely. The light source is placed in the centroid of the region and has the total summed radiance for that region.

The algorithm is as follows:

1. Add the entire light probe image to the region list as a single region.

2. For each region in the list, subdivide along the longest dimension such that its light energy is divided evenly.
3. If the number of iterations is less than K, return to Step 2.
4. Place a light source at the centroid of each region, and set the light source radiance to the sum of the pixel values within the region.

The energy of a pixel is calculated as

$$Y = 0.2125R + 0.7154G + 0.0721B. \tag{5.8}$$

The input EM is in LL format and therefore a weight map must be generated that corrects the oversampling near the poles and scale the pixel values so that a higher resolution EM does not equate stronger incident radiance.

The LL format maps a sphere into 2D as shown in Figure 5.7.

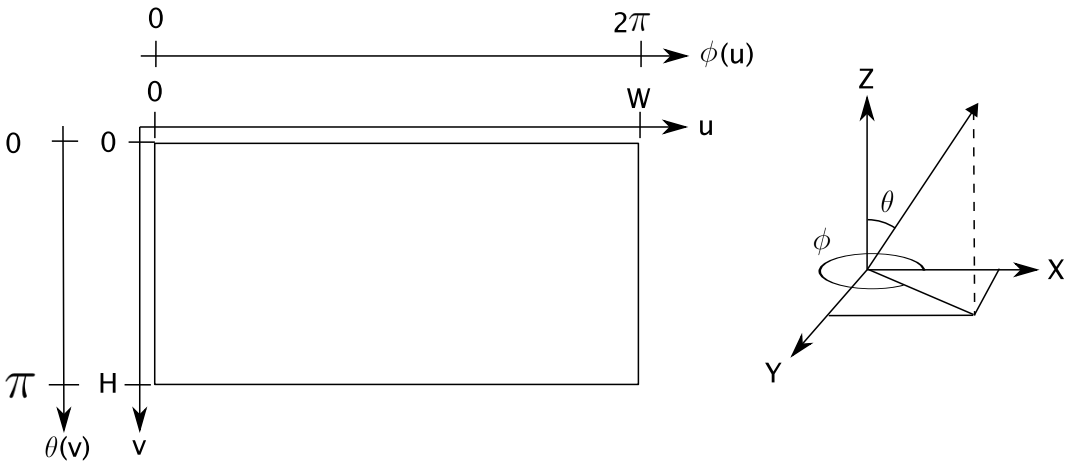


Figure 5.7: *Mapping from spherical coordinates to LL map.*

It can then be calculated how many steradians each pixel subtends with

$$\phi(u) = u\Delta_\phi,\ \Delta_\phi = \frac{2\pi}{W} \tag{5.9}$$

$$\theta(v) = v\Delta_\theta,\ \Delta_\theta = \frac{\pi}{H}. \tag{5.10}$$

To compensate for oversampling near the poles a sine-term must be added. Thus, the weight $W(u, v)$ for a given pixel, corresponding to the solid angle it subtends, is

$$W(u, v) = \phi(u)\theta(v)\sin(\theta(v)). \tag{5.11}$$

That this is correct can be verified by considering the summation

$$\sum_{u}^{W}\sum_{v}^{H}\sin(\theta(v))\Delta_\theta\Delta_\phi \approx 4\pi, \tag{5.12}$$

where the left-hand side approaches π as the size of the EM (WxH) increases.

An example of the resulting distribution of light sources was shown in Figure 4.6 on page 26.

Now, shading the object is done by summing over all light sources within the hemisphere while taking the BRDF and the direction of the light sources into account

$$\tilde{L}(x,\omega) = \sum_{i}^{N} L_i f(x,\omega,\omega_i')(\omega_i' \cdot n). \tag{5.13}$$

For typical Blinn-Phong shading

$$f(x,\omega,\omega_i') = \frac{k_d(\omega_i' \cdot n) + k_s(\omega \cdot \omega_{refl})^m}{(\omega_i' \cdot n)}. \tag{5.14}$$

Shadow Handling

The uv positions of the directional light sources on the LL map corresponds to a direction in spherical coordinates. When the EM is generated, the distances to the geometry is also calculated and thereby each light source can be localized in the scene and some shadowing method, e.g. shadow mapping or newer variations hereof, can be employed. Since it may be unfeasible to cast shadows from all lights, a good heuristic would be only to cast shadows from the M lights with the smallest area. This is reasoned with that these lights produce the strongest, most noticeable shadows.

Casting shadows onto the scene geometry has the effect that it changes the illumination from which the EM was generated in the first place. The consequences of this will therefore need to be further investigated in another project.

5.3.2 Frequential Radiance Representation

Another way of shading the object based on an EM, is to transform the EM into the frequency domain and perform the approximation and shading from there.

In [RH01] it was shown how this could be done by projecting the EM into a Spherical Harmonics (SH) basis. The basis function is the real associated Legendre poly-

nomials parameterized using spherical coordinates. How to calculate the basis functions is explained in [Gre03]; Figure 5.8 shows a visualisation of the five first bands.

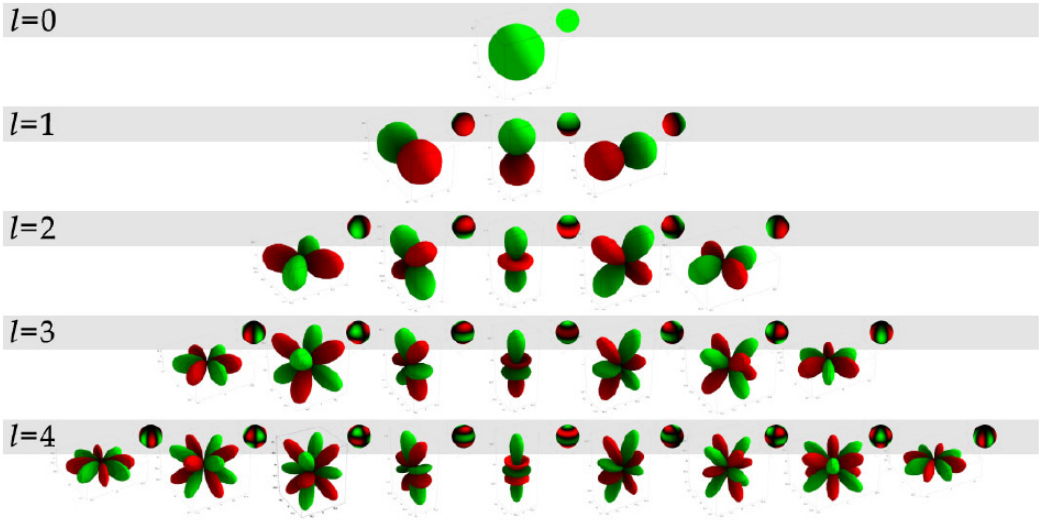


Figure 5.8: *Visualisation of the first five bands of the spherical hamonics basis. From [Gre03].*

A spherical signal is projected into frequency domain using the SH transform by integrating over the sphere S for each SH coefficient

$$L_l^m = \int_S L'(\omega')Y_l^m(\omega')d\omega', \tag{5.15}$$

where L_l^m is the m'th coefficient for the l'th band, $l \geq 0, -l \geq m \geq l$, and Y_l^m is matching SH basis function evaluated in the direction ω'.
So, the result of this projection is a vector of SH coefficients, L_l^m for each colour channel, where each coefficient is a measure of "how much of this basis function" is present in the signal. For a discrete EM (5.15) can be written as

$$L_l^m = \sum_i^N L_i' Y_l^m(\omega_i')\Delta\omega_i', \tag{5.16}$$

where N is the number of pixels in the EM, L_i is the intensity of the i'th pixel, ω_i' is the direction to the pixel, and $\Delta\omega_i'$ is the projected solid angle like in Equation 5.6 on page 35.

To reconstruct the signal the inverse SH transform is applied

$$\tilde{L}(\omega) = \sum_{l=0}^{n-1} \sum_{m=-l}^{l} L_l^m Y_l^M(\omega), \tag{5.17}$$

and the more coefficients,n, the more precise is the reconstruction. For low-frequency signals such as an irradiance distribution Ramamoorthi showed that 9 coefficients are adequate, however due to the low-frequency nature of the basis functions a high-frequency signal is not well represented in the SH basis.

Having the frequency representation of the signal, convolutions become multiplications. So, to perform a diffuse convolution, the signal can now be multiplied with the with the SH coefficients from transforming the consine-term. Sloan presented a method called Precomputed Radiance Transfer (PRT) in [PJJ02] to further extended the shading to include global illumination effects such as diffuse interreflection on the object.
Briefly explained, a transfer function can be calculated for each vertex in the object in an offline process. The transfer function stores how incident radiance is reflected at that vertex including visibility and diffuse interreflections on the object. The transfer functions are transformed to SH basis, and when shading the object calculating, the exitant radiance is simply a matter of multiplying the incident radiance coefficients with the transfer function. Later, the method was further extended to account for more complex transfers e.g. local deformations of the object[SLS05].

In the following section, shading using representation in either spatial or frequential in evaluated with respect to applicability in this project.

5.3.3 Comparison of Representations

Since its introduction shading in the frequency domain has become increasingly popular and basic SH and PRT functions are included in the Direct3D API[Mic]. The methods are good for low-frequency lighting and in cases where the transfer functions can be precalculated, realistic shading including subsurface scattering can be simulated.
However, since these methods are typically used in synthetic scenes, where the synthetic light can be "baked" into an SH based lighting for soft, diffuse shading. The same light can the also be used for high-frequency direct lighting wiht following hard shadows.

In this application, all illumination information is stored in light probe images, so to have hard shadows (which then can be softened if needed), the lighting needs to be localised. Even though Sloan et al. presented a method for dynamic shadows in dynamic scenes with non-localised illumination, it is still limited to low-frequency lighting[SGNS07].

Based on this, it is chosen to approximate the EM using Median Cut and directional light sources[1].

[1] At this years SIGGRAPH, Kautz et al. presents a method for all-frequency shadows

In the following chapter are some of the details in implementing the proposed method described.

from an EM[ADM+08], it was however discovered too late to be included in this project.

6 Implementation

This chapter discusses specific issues that was taken into consideration when implementing the methods proposed in the previous chapter.

6.1 Summed Area Tables

In the Median Cut algorithm, calculating the intensities of the regions require summing over the region many times. To accelerate this task, Debevec suggests using Summed Area Tables (SAT)[Deb05]. This makes it possible to calculate the sum of an arbitrarily sized region in constant time[Cro84]. A SAT is generated using

$$s_{mn} = \sum_{i=1}^{m} \sum_{j=1}^{n} t_{ij}, \tag{6.1}$$

where s_{mn} is an element is the SAT that contains the value of all elements t above and to the left of the correspond element in the EM. This is exemplified in Figure 6.1.

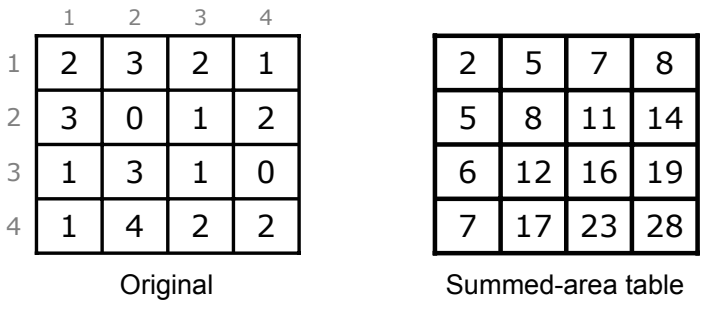


Figure 6.1: *Example of a Summed Area Table along with the original table. Adapted from [HS05].*

Now, the values of the corners of the region are looked up and the sum is calculated

as

$$\text{Sum} = LR - UR - LL + UL, \tag{6.2}$$

as illustrated in Figure 6.2.

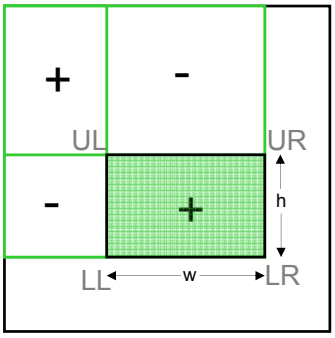


Figure 6.2: *Summation using a SAT. Adapted from [HS05].*

6.2 GPU Utilization

Normally, generating an EM by rendering the scene from 6 views is considered unfeasible due to the increased fillrate requirements. However, because of the limited complexity of the scene, this is not an issue for the developed application. The EM is stored as a cube map in six 32 bit render-textures. The cube map then need to be converted to an LL map to be used with the Median Cut. The conversion is also performed on the GPU to take advantage of parallelisation and cube map look up are performed in hardware. To do so, the viewport is set to the size of the LL map and a screen-sized quad is rasterized with the shader shown in Listing 6.1.

Listing 6.1: *Shader code in GLSL for projecting a cube map into an longitude-latitude map.*

```
#define PI     3.14159265

uniform samplerCube texCube;

void main()
{
       float theta = 2*PI * gl_TexCoord[0].x;
       float phi  = gl_TexCoord[0].y * PI;

       vec3 dir;
       dir.x = sin(phi)*cos(theta);
       dir.y = -cos(phi);
```

```
        dir.z = -sin(phi)*sin(theta);

        gl_FragData[0] = textureCube( texCube, dir );
}
```

The LL map is read into system memory, the Median Cut is performed on it, and the resulting light positions and intensities are uploaded to the graphics card again to be used for shading the object. Since only 8 light sources are supported when using fixed-function pipeline, Listing 6.2 shows how this can be extended using a custom shader.

Listing 6.2: *Proof-of-concept diffuse shading using more than 8 directional light sources.*

```
#define NUM_LIGHTS          64

uniform vec3 lightPos[ NUM_LIGHTS ];      // In World Coordinates
uniform vec3 lightCol[ NUM_LIGHTS ];

varying vec3 diffuse;

void main()
{
        vec3 ecVertex = vec3( gl_ModelViewMatrix * gl_Vertex );
        vec3 normal = normalize( gl_NormalMatrix * gl_Normal );

        diffuse = vec3(0.);

        for( int i = 0; i < NUM_LIGHTS; i++ )
        {
                vec3 ecLightPos = vec3( gl_ModelViewMatrix *
                    vec4(lightPos[i], 1.) );
                vec3 ecLightDir = normalize( ecLightPos );

                float intensity = max( 0., dot(normal, ecLightDir) );

                diffuse += lightCol[i] * intensity;
        }

        gl_Position = ftransform();
}
```

By implementing the lighting calculations in shaders, this further allows for more advanced BRDFs than the standard Blinn-Phong model used in OpenGL. The back-projection and fusing of light probe images were also performed on the GPU using custom shaders written in GLSL.

To test the proposed methods in the Problem Analysis, an application was developed.

7

Results and Evaluation

In this chapter, the resulting application is evaluated and the developed methods are discussed.

In its current state, the application supports loading a scene geometry together with an object to be used for augmentation. It also loads two HDR light probes from which the illumination in the scene is based. The application works as described in Chapter 4. A screenshot from the application can be seen in Figure 7.1 where the approximating directional light sources have been visualised and localised in the scene.

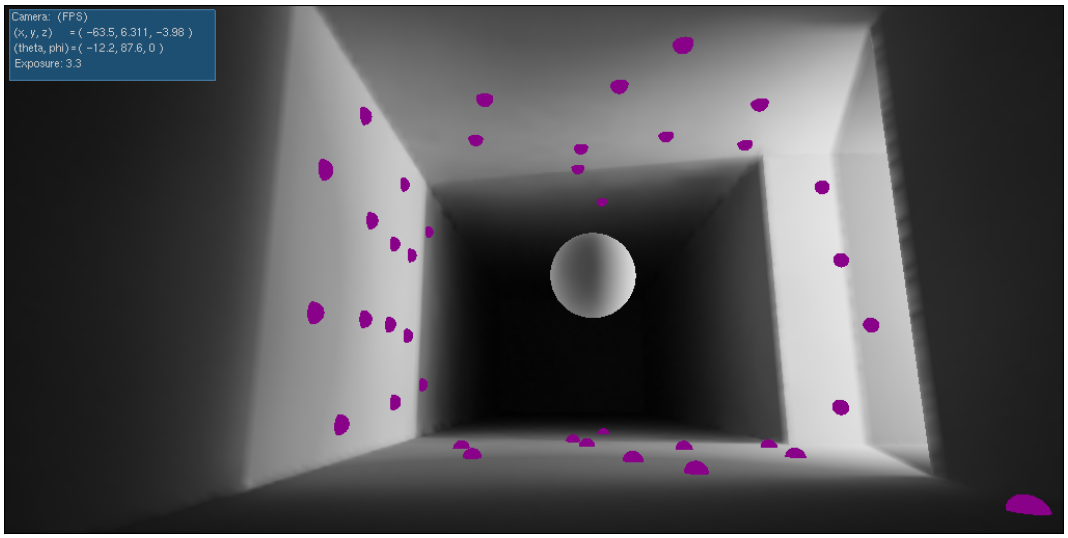


Figure 7.1: *Screenshot from the developed application. The sphere in the center of the image is lit from 64 directional light sources, based on an EM that has been dynamically generated at the position of the sphere. The light sources are visualised by the smaller, purple spheres. The user can freely navigate the camera as well as the sphere around in the scene.*

When updating the dynamic EM, even small a change in the position leads to a new configuration of light sources. For diffuse shading of the object, this is not critical,

but when the light sources are used for shadowing (esp. if only few, hard shadows are supported) the shadows could be subject to much flickerering. An immidiate solution would be to treat the position and intensity of each light source as two signals and apply a low-pass filter to each. This in itself is not a trivial problem and considerations similar to those in [HSK+05] must be made.

Late in the process it was found, that some calculations were erroneous, and had to be corrected. This set back the development process and unfortunately implied that little time was left for implementing more advanced features.

Specifically, there were problems with getting the excitant radiance correct when using directional lights. To alleviate this, a test-application was developed which takes a light probe image as input and generates the corresponding irradiance map.

The first test was to input a unit-radiance map. From the definition of irradiance, the resulting values should then approximate π as the number of (equally distributed) light sources increases. This was verified from the results listed in Table 7.1.

Table 7.1: *Irradiance approximation of unit-radiance map. As the number of light sources, 16 and 64, increases the average value of the irradiance map should go towards the Ground Truth (GT).*

	Avg.	RMS	Error (%)
GT	3.142	–	–
16	3.046	0.241	3.02
64	3.108	0.050	1.05

The next tests were then to approximate a known, preconvolved diffuse map. The well-known Uffizi and Grace Cathedral maps were acquired from [Lab] where the corresponding diffuse convolution maps also were found to be used for Ground Truth. A pixel in a diffuse map stores the excitant radiance for that direction whereas the approximations using directional light sources calculate the irradiance (and thereby the radiosity for a non-absorbing surface) for that direction. Assuming a Lambertian surface, this means that the values must be scaled with

$$\frac{1}{\int_\Omega (n \cdot \omega) d\omega} = \frac{1}{\pi} \tag{7.1}$$

to account for the radiance reflected (uniformly) in other directions on the hemisphere.

The results from approximating these can be seen in Figures 7.2 and 7.3. Except for the 16 lights approximation of the Uffizi map that has a bit too much contrast, the maps are visually indistinguishable. Quantitatively, the error rates in Table 7.2 and Table 7.3 are a bit higher than those reported in [ML07b]. This may be explained by the numerical imprecision when using a GPU.

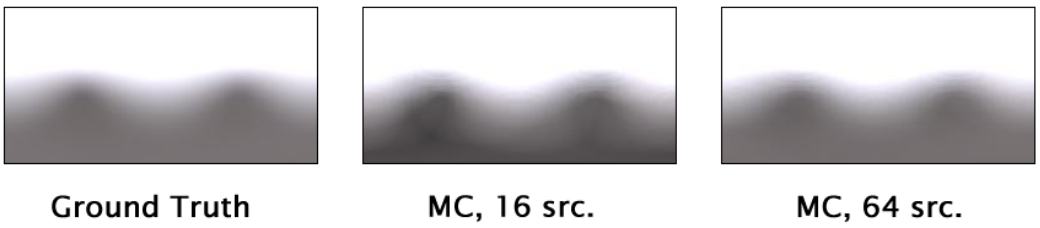


Figure 7.2: *Approximation of diffuse convolution using Median Cut with 16 and 64 directional light sources, respectively (Uffizi).*

Figure 7.3: *Approximation of diffuse convolution using Median Cut with 16 and 64 directional light sources, respectively (Grace).*

Table 7.2: *Diffuse convolution approximation of Uffizi (512x256) on GF6800GT*

Uffizi	Avg. Error (%)	RMS
MC 16	19.1	0.197
MC 64	6.0	0.034

Table 7.3: *Diffuse convolution approximation of Grace (512x256) on GF6800GT*

Grace	Avg. Error (%)	RMS
MC 16	17.3	0.082
MC 64	10.7	0.083

During these tests, it became clear that modern GPUs are designed for speed on behalf of precision. Table 7.4 lists the result of calculating the irradiance on two different GPUs using the exact same code and input data. The differences are notable even though 32-bit pr. channel floating-point buffers were used for the calculations.

Table 7.4: *Numerical Accuracy. MC16, 256x128, Grace*

	R_{avg}	G_{avg}	B_{avg}
ATI X300 Mobility	1.157	0.750	0.537
NVIDIA GForce 6800GT	1.090	0.720	0.514

To evaluate the quality of the output, a simple test scene was constructed. The scene geometry has the shape of a 'T' where one very bright light source is placed down the corridor, resulting in a sharp change in the illumination from dark to bright. A rendering of the scene is shown in Figure 7.4.

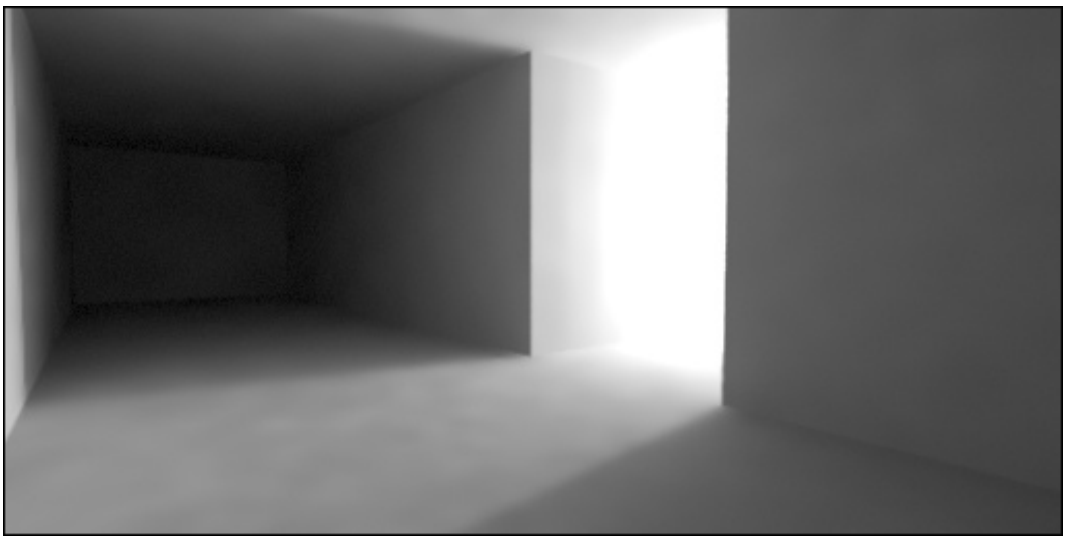


Figure 7.4: *Ground Truth of the test scene rendered using the global illumination engine in Lightwave 9.*

Now, to test that the way of back-projecting the light probes was correct, a synthetic light probe was created by placing a camera in the middle of the hall and rendering the scene as a cube map using global illumination. This light probe was loaded into the application and the result of this can be seen in Figure 7.5.

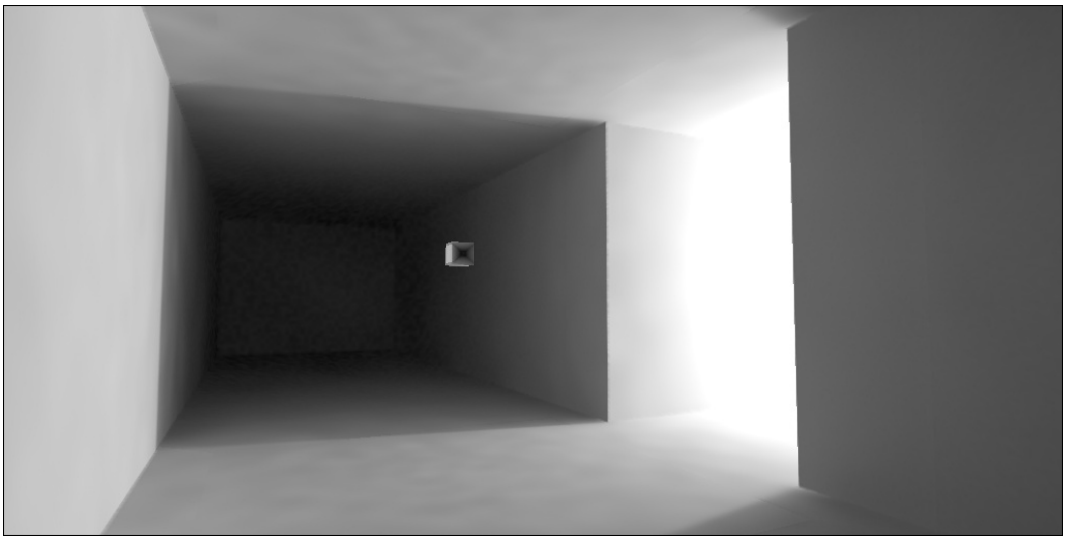


Figure 7.5: *Back-projection of a single light probe.*

To test the fusing method, the so-called Multiple Light Probes (MLP) shader, two test light probes were created; one with a blue-checkered pattern and one with a red-checkered pattern. The test scene was set up as illustrated in Figure 7.6.

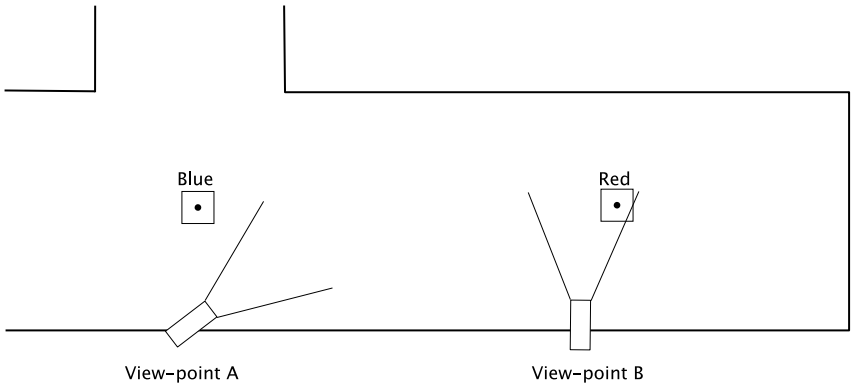


Figure 7.6: *Fuse test setup.*

The fusing is done as described back in Equation (5.5), however checking for the visibility of a light probe had to be omitted due to time constraints. The result of this is displayed in Figure 7.7, which shows how the shading of the scene geometry correctly changes depending on the view-point. The circular blending that can be seen in View-point A is due to the cosine weighting. Also, the high frequencies in the patters are preserved. This fusing scheme seems promising, however tests should of course also be made with images from real-world light probes.

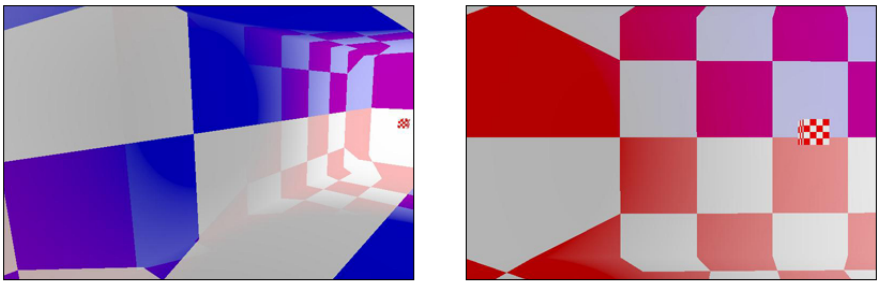


Figure 7.7: *Fusing two light probes. The weighting of each light probes depends on the view-point in relation to the light probe.*

As a final test, a second synthetic light probe was created similar to the one used to test the back-projection, only it was positioned the same place as the red-checkered test probe. The result of using these two light probes can be seen in Figure 7.8. There is a noticeable visual artifact down towards the light when compared to the raytraced image back in Figure 7.4. The attenuation is to be expected as the light probe, that is not visible from this position is included in calculating the radiance.

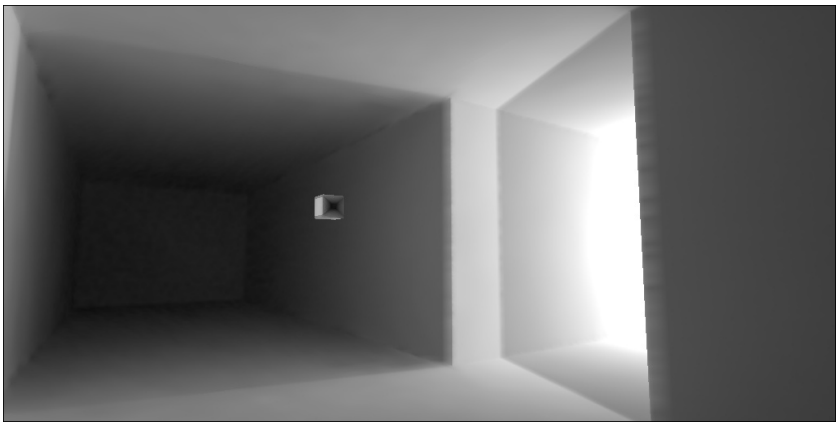


Figure 7.8: *Two light probes back-projected and fused.*

In the following chapter, the conclusion of this report are made.

8 Conclusion

The overall goal of this report was to develop a method for augmenting a real-world scene with a virtual object. The shading of the virtual object should be consistent with spatially varying illumination present in large scenes. To do so, a method based on fusing multiple light probes was proposed. The method depended on three underlying areas that needed to be considered.

To allow the virtual object to interact with the scene, a rough, geometric representation of the scene is modeled. This geometry is used to resolve occlusions and also gives the ability to cast and receive shadows. This worked well, however, shadow casting could not be implemented in due time.

Another function of the scene geometry is that the light probes can be back-projected onto it. This was successfully implemented and showed good results for the test scenes. More tests are needed with real-world probes to see whether noticeable distortions arise where the geometry does not entirely match the scene.
After the probes are projected back, they are fused together and used for view-dependent shading of the scene geometry. A method for fusing the probes was developed grounded in radiometric theory. The preliminary test results are promising, but since the method is primarily aimed at fusing real-world light probes, such a test set is needed. Potential problems from the fusing include blurriness and ghosting artifacts.

The last focus-area was to develop and test a method for shading a virtual object. This was done by dynamically generating an EM at the position of the object to capture the incident radiance. This EM should then be approximated by directional light sources with which the object could be shaded.
Calculating the irradiance from a unit-radiance map gave satisfying results with an error of 3% to 1% for 16 and 64 light sources, respectively. When approximating real-world light probes, the errors were higher (ranging from 6% to 19%), however qualitatively the method is considered to be applicable to the given task.

Based on the above, the main objective –to have a virtual object appear and interact consistently with the scene– is only partially fulfilled. Developing and implementing a novel, solid AR method is an involving task with many underlying complex-

ities that need to be addressed, and therefore more work need to be carried out to give a conclusive answer about the validity of the proposed method.

8.1 Future Work

In the immediate future, more work need to be done with respect to finishing the implementation of the proposed method. Once this is completed, the method stands as an interesting framework upon which more refined techniques can be built. It could be interesting to experiment with more advanced fusing schemes, perhaps incorporating non-linear blendings similar to the Floating Textures. Another important aspect is to stabilize the light sources over time. It is not know at the time of writing if this is posible when using Median Cut or if importance sampling should be applied similar to the work of Havran.

Bibliography

[AB91] E.H. Adelson and J. Bergen. The plenoptic function and the elements of early vision. *Computational Models of Visual Processing*, pages 3–20, 1991.

[ADM+08] T. Annen, Z. Dong, T. Mertens, P. Bekaert, H.-P. Seidel, and J. Kautz. Real-time, all-frequency shadows in dynamic scenes. In *ACM Transactions on Graphics (Proceedings SIGGRAPH 2008)*, August 2008.

[ATI] ATI. Rendering with natural light. `http://ati.amd.com/developer/demos/R9700.html`.

[Azu97] Ronald T. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, August 1997.

[BN76] J. F. Blinn and M. E. Newell. Texture and reflection in computer generated images. In *Communications of the ACM, No. 10*, volume 19, 1976.

[BSKS05] A. Barsi, L. Szirmay-Kalos, and L. Szécsi. Image-based illumination on the gpu. *MG&V*, 14(2):159–169, 2005.

[CCC08] Massimiliano Corsini, Marco Callieri, and Paolo Cignoni. Stereo light probe. *EUROGRAPHICS*, 27, 2008.

[Cro84] Franklin C. Crow. Summed-area tables for texture mapping. *SIGGRAPH Comput. Graph.*, 18(3):207–212, 1984.

[Deb98] Paul E. Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH '98*, 1998.

[Deb05] Paul Debevec. A median cut algorithm for light probe sampling, 2005.

[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, 1996.

[Dut03] Philip Dutre. *Advanced Global Illumination*. AK Peters, 2003.

[DYB98] Paul Debevec, Yizhou Yu, and George Boshokov. Efficient view-dependent image-based rendering with projective texture-mapping. Technical Report CSD-98-1003, University of California at Berkeley, 20, 1998.

[EDM+08] Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson de Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. Floating Textures. *Computer Graphics Forum (Proc. Eurographics EG'08)*, 27(2), 4 2008.

[GGSC96] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph, 1996.

[Gre03] Robin Green. Spherical harmonic lighting the gritty details. `http://www.research.scea.com/gdc2003/spherical-harmonic-lighting.html`, January 2003.

[GSHG98] Gene Greger, Peter Shirley, Philip M. Hubbard, and Donald P. Greenberg. The irradiance volume. *IEEE Computer Graphics and Applications*, 18(2):32–43, 1998.

[GTR+06] J. Gu, C. Tu, R. Ramamoorthi, P. Belhumeur, W. Matusik, and S. K. Nayar. Time-varying surface appearance: Acquisition, modeling, and rendering. In *ACM SIGGRAPH*, 2006.

[HS80] Berthod K.P. Horn and Brian G. Schunck. Determining optical flow. In *AI Memos 1959–2004*, 1980.

[HS05] Justin Hensley and Thorsten Scheuermann. Dynamic glossy reflections using summed-area tables. In *ShaderX 4*, pages 187–200. Charles River Media, 2005.

[HSK+05] Vlastimil Havran, Miloslaw Smyk, Gfzegorz Krawczyk, Karol Myszkowski, and Hans-Peter Seidel. Importance Sampling for Video Environment Maps. In Kavita Bala and Philip Dutré, editors, *Eurographics Symposium on Rendering 2005*, pages 31–42,311, Konstanz, Germany, 2005. ACM SIGGRAPH.

[Jen01] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping*. AK Peters, 2001.

[Kaj86] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, 1986.

[Lab] ICT Graphics Lab. High-resolution light probe image gallery. `http://gl.ict.usc.edu/Data/HighResProbes/`. Retrieved 2 Aug. 2008.

[LH96] M. Levoy and P. Hanrahan. Light field rendering, 1996.

[MA07] Bjarne Kondrup Mortensen and Jens Rosenkjær Andersen. Modelling view-dependent surface point radiance with parameter maps. Master's thesis, Aalborg University, Denmark, 2007.

[MH84] Gene S. Miller and C. Robert Hoffman. Illumination and reflection maps: Simulated objects in simulated and real environments. In *Course Notes for Advanced Computer Graphics Animation*. SIGGRAPH, 1984.

[Mic] Microsoft. Precomputer radiance transfer (direct3d 9). `http://msdn.microsoft.com/en-us/library/bb147287(VS.85).aspx`. Retrieved 5 Aug. 2008.

[ML03] Alexandre Meyer and Céline Loscos. Real-time reflection on moving vehicles in urban environments. In *VRST '03: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 32–40, 2003.

[ML07a] C. B. Madsen and R. Laursen. A scalable gpubased approach to shading and shadowing for photorealistic real-time augmented reality. In *International Conference on Graphics Theory and Applications*, 2007.

[ML07b] Claus B. Madsen and Rune E. Laursen. Performance comparison of techniques for approximating image-based lighting by directional light sources. In *SCIA*, pages 888–897, 2007.

[MPM02] Rafał Mantiuk, Sumanta Pattanaik, and Karol Myszkowski. Cubemap data structure for interactive global illumination computation in

dynamic diffuse environments. In *Conference Proceedings of International Conference on Computer Vision and Graphics*, pages 530–538, 2002.

[MRP98] G. Miller, S. Rubin, and D. Ponceleon. Lazy decompression of surface light fields for precomputed global illumination. In *9th Eurographics Workshop on Rendering*, 1998.

[NPG03] Mangesh Nijasure, Sumanta Pattanaik, and Vineet Goel. Interactive global illumination in dynamic environments using commodity graphics hardware. In *Proceedings of Pacific Graphics*, pages 450–454, 2003.

[Per07] Emil Persson. Global illumination, november 2007.

[PJJ02] Sloan P.P., Kautz J., and Snyder J. Pre-computed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *ACM Transactionson Graphics 22*, 2002.

[RH01] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 497–500, 2001.

[SGNS07] Peter-Pike Sloan, Naga K. Govindaraju, Derek Nowrouzezahrai, and John Snyder. Image-based proxy accumulation for real-time soft global illumination. In *PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, pages 97–105, Washington, DC, USA, 2007. IEEE Computer Society.

[SLS05] Peter-Pike Sloan, Ben Luna, and John Snyder. Local, deformable precomputed radiance transfer. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 1216–1224, New York, NY, USA, 2005. ACM.

[TSK+05] Marcus Toennis, Christian Sandor, Gudrun Klinker, Christian Lange, and Heiner Bubb. Experimental evaluation of an augmented reality visualization for directing a car drivers attention. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR 05)*, 2005.

[WAA+00] Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle. Surface light fields for 3d photography. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 287–296, 2000.

[WH92] Gregory J. Ward and Paul Heckbert. Irradiance Gradients. In *Third Eurographics Workshop on Rendering*, pages 85–98, 1992.

[Wil78] Lance Williams. Casting curved shadows on curved surfaces. *SIGGRAPH Comput. Graph.*, 12(3):270–274, 1978.

[WRC88] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 85–92. ACM, 1988.

[WWL+04] Wenle Wang, Lifeng Wang, Stephen Lin, Jianmin Wang, and Baining Guo. Real-time environment map interpolation. In *ICIG '04: Proceedings of the Third International Conference on Image and Graphics*, pages 382–389. IEEE Computer Society, 2004.