A Practical Guide to Global Illumination using Photon Mapping

Siggraph 2002 Course 43

Tuesday, July 23, 2002

Organizer

Henrik Wann Jensen
Stanford University

Lecturers

Per H. Christensen
Pixar

Henrik Wann Jensen
Stanford University

Toshiaki Kato
Square USA / Rhythm & Hues

Frank Suykens
K.U.Leuven Belgium

Abstract

This course serves as a practical guide to photon maps. Any reader who can implement a ray tracer should be able to add an efficient implementation of photon maps to his or her ray tracer after attending this course and reading the course notes.

There are many reasons to augment a ray tracer with photon maps. Photon maps makes it possible to efficiently compute global illumination including caustics, diffuse color bleeding, and participating media. Photon maps can be used in scenes containing many complex objects of general type (i.e. the method is not restricted to tessellated models). The method is capable of handling advanced material descriptions based on a mixture of specular, diffuse, and non-diffuse components. Furthermore, the method is easy to implement and experiment with.

This course is structured as a half day course. We will therefore assume that the participants have knowledge of global illumination algorithms (in particular ray tracing), material models, and radiometric terms such as radiance and flux. We will discuss in detail photon tracing, the photon map data structure, the photon map radiance estimate, and rendering techniques based on photon maps. We will emphasize the techniques for efficient computation throughout the presentation. Finally, we will present several examples of scenes rendered with photon maps and explain the important aspects that we considered when rendering each scene.

Lecturers

Per H. Christensen

Pixar
per.christensen@acm.org

Dr. Per Christensen is a senior developer at Pixar's Seattle office. He is working on new features for Pixar's PhotoRealistic RenderMan renderer. Prior to joining Pixar, he was a senior software engineer at Square USA in Honolulu, developing and implementing efficient global illumination methods in the massively parallel renderer Kilauea. He has also co-implemented the first commercial implementation of the photon map method. Dr. Christensen received his Ph.D. from the University of Washington for research in hierarchical techniques for glossy global illumination.

Henrik Wann Jensen

Stanford University
henrik@graphics.stanford.edu
http://graphics.stanford.edu/˜henrik

Dr. Henrik Wann Jensen is a Research Associate at Stanford University where he is working in the computer graphics group on realistic image synthesis, global illumination, and new appearance models. His contributions to computer graphics include the photon mapping algorithm for global illumination, and the first BSSRDF for simulating subsurface scattering in translucent materials. He is the author of "Realistic Image Synthesis using Photon Mapping", AK Peters 2001. Prior to coming to Stanford in 1999, he was a postdoctoral researcher at MIT, and a research scientist in industry where he added photon maps to a commercial renderer. He received his M.Sc. and Ph.D. in Computer Science from the Technical University of Denmark for developing the photon mapping method.

Toshiaki Kato

Square USA Honolulu Studio (now Rhythm & Hues)
toshi@squareusa.com

For eighteen years Toshi Kato has devoted his time to the development of high-end production-level renderers. He was the lead of the project “Kilauea,” a parallel ray tracer developed at Square USA’s R&D division. The development of ”Kilauea” involved various fields including system architecture design, implementation of a message passing layer, and a parallel shading and ray tracing engine. From 1993 to 1998 he developed an in-house scanline-based renderer at Rhythm & Hues in LA and participated in the production of movies such as Mouse Hunt (1998), Kazaam (1996) and Babe (1995). He also developed a special stereoscopic renderer for IMAX’s Omnimax Solido dome.

Frank Suykens

K.U.Leuven
Frank.Suykens@cs.kuleuven.ac.be
http://www.cs.kuleuven.ac.be/˜franks/

Frank Suykens is a Ph.d. candidate at the university K.U.Leuven in Belgium. His research focusses on photorealistic rendering. The main topics concern efficient Monte Carlo algorithms for bidirectional and multipass methods. Recent work involves making weighted combinations of multipass methods and extensions of the photon map method using viewpoint information to control error and density of photon maps locally in the scene. All research is implemented into RenderPark, a freely available rendering package developed by the graphics research group (K.U.Leuven) that is used for research and educational purposes.

Course Overview

5 minutes: Introduction and Welcome

Henrik Wann Jensen

Overview of the course and motivation for attending it.

15 minutes: Overview of Global Illlumination

Henrik Wann Jensen

A brief overview of state-of-the-art in global illumination including a description of finite element radiosity and Monte Carlo ray tracing methods. The advantages and disadvantages of each method will be discussed, and an overview of the photon mapping algorithm will be given.

40 minutes: Photon Tracing: Building the Photon Maps

Henrik Wann Jensen

This part of the course will cover efficient techniques for:

- Emitting photons from the light sources in the scene
- The use of projection maps
- Simulating scattering and absorption of photons using Russian Roulette
- Storing photons in the photon map
- Preparing the photon map for rendering

Also the use of several photon maps for the simulation of caustics, soft indirect illumination and participating media will be described.

45 minutes: Rendering using Photon Mapping

Henrik Wann Jensen

A description of how the photon maps are used to render complex scenes with global illumination. We will describe how the rendering equation is split into several components that each can be rendered using specialized techniques based on the photon maps, and we will cover methods for rendering caustics, indirect illumination, participating media and subsurface scattering.

We will describe how to apply useful algorithms such as irradiance caching and Russian roulette to significantly reduce the rendering time. We will also present some useful and practical ”tricks” that dramatically improve the speed of the photon map.

There will be several examples examples of different scenes rendered using photon maps with a description of how the photon maps were used, and some insight on the issues that were important to ensure good quality and fast results.

15 minutes: Break

35 minutes: Visual Importance

Frank Suykens

A description of visual importance and how it can be used to build a better and more compact photon map. This includes methods to control the local density of photons to get an appropriate accuracy throughout the scene, while reducing memory requirements, and a brief discussion about the use of importance sampling on light sources in order to guide photons to the important locations.

30 minutes: Faster Photon Mapping

Per H. Christensen

A number of optimizations for making the photon map faster will be presented. This includes precomputing information at the cost of slightly higher memory consumption, and also methods for making the search for photons more effective. New details on frame-coherent use of random numbers in the photon tracing pass to reduce flickering in animations.

30 minutes: Photon Mapping at SquareUSA (the Kilauea Renderer)

Toshi Kato

Applying global illumination to complex scenes used in the actual production work gives rise to various issues related to image quality and rendering speed. Techniques and improvements researched and developed to overcome these issues at Square USA for the parallel global illumination renderer ”Kilauea” are explained, using sample images rendered by Kilauea.

5 minutes: Final Remarks and Questions

All

Contents

Chapter 1

Introduction

This course material describes in detail the practical aspects of the photon map algorithm. The text is based on previously published papers, technical reports and dissertations (in particular [Jensen96c]). It also reflects the experience obtained with the implementation of the photon map as it was developed at the Technical University of Denmark. After reading this course material, it should be relatively straightforward to add an efficient implementation of the photon map algorithm to any ray tracer.

1.1 Motivation

The photon mapping method is an extension of ray tracing. In 1989, Andrew Glassner wrote about ray tracing [Glassner89]:

> "Today ray tracing is one of the most popular and powerful techniques in the image synthesis repertoire: it is simple, elegant, and easily implemented. [However] there are some aspects of the real world that ray tracing doesn't handle very well (or at all!) as of this writing. Perhaps the most important omissions are diffuse inter-reflections (e.g. the 'bleeding' of colored light from a dull red file cabinet onto a white carpet, giving the carpet a pink tint), and caustics (focused light, like the shimmering waves at the bottom of a swimming pool)."

At the time of the development of the photon map algorithm in 1993, these problems were still not addressed efficiently by any ray tracing algorithm. The photon map method offers a solution to both problems. Diffuse interreflections and caustics are both indirect illumination of diffuse surfaces; with the photon map method, such illumination is estimated using precomputed photon maps. Extending ray tracing with photon maps yields a method capable of efficiently simulating all types of direct and indirect illumination. Furthermore, the photon map method can handle participating media and it is fairly simple to parallelize [Jensen00].

1.2 What is photon mapping?

The photon map algorithm was developed in 1993–1994 and the first papers on the method were published in 1995. It is a versatile algorithm capable of simulating global illumination including caustics, diffuse interreflections, and participating media in complex scenes. It provides the same flexibility as general Monte Carlo ray tracing methods using only a fraction of the computation time.

The global illumination algorithm based on photon maps is a two-pass method. The first pass builds the photon map by emitting photons from the light sources into the scene and storing them in a *photon map* when they hit non-specular objects. The second pass, the rendering pass, uses statistical techniques on the photon map to extract information about incoming flux and reflected radiance at any point in the scene. The photon map is decoupled from the geometric representation of the scene. This is a key feature of the algorithm, making it capable of simulating global illumination in complex scenes containing millions of triangles, instanced geometry, and complex procedurally defined objects.

Compared with finite element radiosity, photon maps have the advantage that no meshing is required. The radiosity algorithm is faster for simple diffuse scenes but as the complexity of the scene increases, photon maps tend to scale better. Also the photon map method handles non-diffuse surfaces and caustics.

Monte Carlo ray tracing methods such as path tracing, bidirectional path tracing, and Metropolis can simulate all global illumination effects in complex scenes with very little memory overhead. The main benefit of the photon map compared with these methods is efficiency, and the price paid is the extra memory used to store the photons. For most scenes the photon map algorithm is significantly faster, and the result looks better since the error in the photon map method is of low frequency which is less noticeable than the high frequency noise of general Monte Carlo methods.

Another big advantage of photon maps (from a commercial point of view) is that there is no patent on the method; anyone can add photon maps to their renderer.

As a result several commercial systems use photon maps for rendering caustics and global illumination.

1.3 Overview of the course material

The first part describes the basic photon mapping algorithm. Section 2.1 describes emission, tracing, and storing of photons. Section 2.2 describes how to organize the photons in a balanced kd-tree for improved performance in the rendering step. The radiance estimate based on photons is outlined in section 2.3. This section also contains information on how to filter the estimate to obtain better quality and it contains a description of how to locate photons efficiently given the balanced kd-tree. The rendering pass is presented in section 2.4 with information on how to split the rendering equation and use the photon map to efficiently compute different parts of the equation. Section 2.5 we give a number of examples of scenes rendered with the photon map algorithm.

The second part provides some information about recent research on visual importance. How can we send the photons to the parts of the model that we are concerned about?

The last part provides the details for a number of practical tricks that can make photon mapping significantly faster.

1.4 More information

For more information about photon mapping, all the practical details, the theory and the insight for understanding the technique see:

Henrik Wann Jensen
Realistic Image Synthesis using Photon Mapping
AK Peters, 2001

This book also contains additional information about participating media and subsurface scattering. Finally, it contains an implementation with source code in C++ of the photon map data structure.

1.5 Acknowledgements

The inspiration behind this course is Alan Chalmers. Without his suggestion (over a beer) this course might never have materialized. Thanks, Alan! In addition, thanks to Niels Jørgen Christensen, Martin Blais, Byong Mok Oh, Gernot Schaufler and Maryann Simmons for helpful comments on the notes. Finally, we would like to thank intellectual property counsel Karen Hersey at Massachusetts Institute of Technology.

Chapter 2

A Practical Guide to Global Illumination using Photon Mapping

2.1 Photon tracing

The purpose of the photon tracing pass is to compute indirect illumination on diffuse surfaces. This is done by emitting photons from the light sources, tracing them through the scene, and storing them at diffuse surfaces.

2.1.1 Photon emission

This section describes how photons are emitted from a single light source and from multiple light sources, and describes the use of projection maps which can increase the emission efficiency considerably.

Emission from a single light source

The photons emitted from a light source should have a distribution corresponding to the distribution of emissive power of the light source. This ensures that the emitted photons carry the same flux — ie. we do not waste computational resources on photons with low power.

Photons from a diffuse point light source are emitted in uniformly distributed random directions from the point. Photons from a directional light are all emitted in the same direction, but from origins outside the scene. Photons from a diffuse square light source are emitted from random positions on the square, with directions limited to a hemisphere. The emission directions are chosen from a cosine distribution: there is zero probability of a photon being emitted in the direction parallel to the plane of the square, and highest probability of emission is in the direction perpendicular to the square.

In general, the light source can have any shape and emission characteristics — the intensity of the emitted light varies with both origin and direction. For example, a (matte) light bulb has a nontrivial shape and the intensity of the light emitted from it varies with both position and direction. The photon emission should follow this variation, so in general, the probability of emission varies depending on the position on the surface of the light source and the direction.

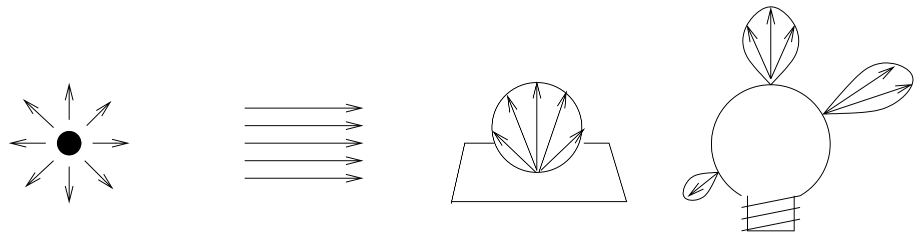


Figure 2.1: Emission from light sources: (a) point light, (b) directional light, (c) square light, (d) general light.

Figure 2.1 shows the emission from these different types of light sources.

The power ("wattage") of the light source has to be distributed among the photons emitted from it. If the power of the light is P_{light} and the number of emitted photons is n_e, the power of each emitted photon is

$$P_{photon} = \frac{P_{light}}{n_e} \, . \tag{2.1}$$

Pseudocode for a simple example of photon emission from a diffuse point light source is given in Figure 2.2.

To further reduce variation in the computed indirect illumination (during rendering), it is desirable that the photons are emitted as evenly as possible. This can for example be done with stratification [Rubinstein81] or by using low-discrepancy quasi-random sampling [Keller96].

```
emit_photons_from_diffuse_point_light() {
  $n_e$ = 0                number of emitted photons
  while (not enough photons) {
    do {                   use simple rejection sampling to find diffuse photon direction
      $x$ = random number between -1 and 1
      $y$ = random number between -1 and 1
      $z$ = random number between -1 and 1
    } while ( $x^2 + y^2 + z^2 > 1$ )

    $\vec{d}$ = $<\ x,\ y,\ z\ >$
    $\vec{p}$ = light source position
    trace photon from $\vec{p}$ in direction $\vec{d}$
    $n_e$ = $n_e$ + 1
  }
  scale power of stored photons with $1/n_e$
}
```

Figure 2.2: Pseudocode for emission of photons from a diffuse point light

Multiple lights

If the scene contains multiple light sources, photons should be emitted from each light source. More photons should be emitted from brighter lights than from dim lights, to make the power of all emitted photons approximately even. (The information in the photon map is best utilized if the power of the stored photons is approximately even). One might worry that scenes with many light sources would require many more photons to be emitted than scenes with a single light source. Fortunately, it is not so. In a scene with many light sources, each light contributes less to the overall illumination, and typically fewer photons can be emitted from each light. If, however, only a few light sources are important one might use an importance sampling map [Peter98] to concentrate the photons in the areas that are of interest to the observer. The tricky part about using an importance map is that we do not want to generate photons with energy levels that are too different since this will require a larger number of photons in the radiance estimate (see section 2.3) to ensure good quality.

Projection maps

In scenes with sparse geometry, many emitted photons will not hit any objects. Emitting these photons is a waste of time. To optimize the emission, *projection map*s can be used [Jensen93, Jensen95a]. A projection map is simply a map of the geometry as seen from the light source. This map consists of many little cells. A cell is "on" if there is geometry in that direction, and "off" if not. For example, a projection map is a spherical projection of the scene for a point light, and it is a planar projection of the scene for a directional light. To simplify the projection it is convenient to project the bounding sphere around each object or around a cluster of objects [Jensen95a]. This also significantly speeds up the computation of the projection map since we do not have to examine every geometric element in the scene. The most important aspect about the projection map is that it gives a conservative estimate of the directions in which it is necessary to emit photons from the light source. Had the estimate not been conservative (e.g. we could have sampled the scene with a few photons first), we could risk missing important effects, such as caustics.

The emission of photons using a projection map is very simple. One can either loop over the cells that contain objects and emit a random photon into the directions represented by the cell. This method can, however, lead to slightly biased results since the photon map can be "full" before all the cells have been visited. An alternative approach is to generate random directions and check if the cell corresponding to that direction has any objects (if not a new random direction should be tried). This approach usually works well, but it can be costly in sparse scenes. For sparse scenes it is better to generate photons randomly for the cells which have objects. A simple approach is to pick a random cell with objects and then pick a random direction for the emitted photon for that cell [Jensen93]. In all circumstances it is necessary to scale the energy of the stored photons based on the number of active cells in the projection map and the number of photons emitted [Jensen93]. This leads to a slight modification of formula 2.1:

$$P_{photon} = \frac{P_{light}}{n_e} \frac{\text{cells with objects}}{\text{total number of cells}} \ . \tag{2.2}$$

Another important optimization for the projection map is to identify objects with specular properties (i.e. objects that can generate caustics) [Jensen93]. As it will be described later, caustics are generated separately, and since specular objects often are distributed sparsely it is very beneficial to use the projection map for caustics.

2.1.2 Photon tracing

Once a photon has been emitted, it is traced through the scene using photon tracing (also known as “light ray tracing”, “backward ray tracing”, “forward ray tracing”, and “backward path tracing”). Photon tracing works in exactly the same way as ray tracing except for the fact that photons propagate flux whereas rays gather radiance. This is an important distinction since the interaction of a photon with a material can be different than the interaction of a ray. A notable example is refraction where radiance is changed based on the relative index of refraction[Hall88] — this does not happen to photons.

When a photon hits an object, it can either be reflected, transmitted, or absorbed. Whether it is reflected, transmitted, or absorbed is decided probabilistically based on the material parameters of the surface. The technique used to decide the type of interaction is known as Russian roulette [Arvo90] — basically we roll a dice and decide whether the photon should survive and be allowed to perform another photon tracing step.

Examples of photon paths are shown in Figure 2.3.

Reflection, transmission, or absorption?

For a simple example, we first consider a monochromatic simulation. For a reflective surface with a diffuse reflection coefficient d and specular reflection coefficient s (with $d + s \leq 1$) we use a uniformly distributed random variable $\xi \in [0, 1]$ (computed with for example `drand48()`) and make the following decision:

$$\begin{array}{lcl} \xi \in [0, d] & \longrightarrow & \text{diffuse reflection} \\ \xi \in]d, s+d] & \longrightarrow & \text{specular reflection} \\ \xi \in]s+d, 1] & \longrightarrow & \text{absorption} \end{array}$$

In this simple example, the use of Russian roulette means that we do not have to modify the power of the reflected photon — the correctness is ensured by averaging several photon interactions over time. Consider for example a surface that reflects 50% of the incoming light. With Russian roulette only half of the incoming photons will be reflected, but with full energy. For example, if you shoot 1000 photons at the surface, you can either reflect 1000 photons with half the energy or 500 photons with full energy. It can be seen that Russian roulette is a powerful technique for reducing the computational requirements for photon tracing.

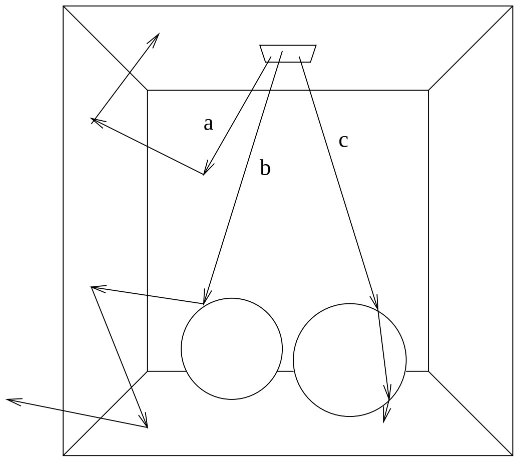


Figure 2.3: Photon paths in a scene (a "Cornell box" with a chrome sphere on left and a glass sphere on right): (a) two diffuse reflections followed by absorption, (b) a specular reflection followed by two diffuse reflections, (c) two specular transmissions followed by absorption.

With more color bands (for example RGB colors), the decision gets slightly more complicated. Consider again a surface with some diffuse reflection and some specular reflection, but this time with different reflection coefficients in the three color bands. The probabilities for specular and diffuse reflection can be based on the total energy reflected by each type of reflection or on the maximum energy reflected in any color band. If we base the decision on maximum energy, we can for example compute the probability P_d for diffuse reflection as

$$P_d = \frac{\max(d_r P_r, d_g P_g, d_b P_b)}{\max(P_r, P_g, P_b)}$$

where (d_r, d_g, d_b) are the diffuse reflection coefficients in the red, green, and blue color bands, and (P_r, P_g, P_b) are the powers of the incident photon in the same three color bands.

Similarly, the probability P_s for specular reflection is

$$P_s = \frac{\max(s_r P_r, s_g P_g, s_b P_b)}{\max(P_r, P_g, P_b)}$$

where (s_r, s_g, s_b) are the specular reflection coefficients.

The probability of absorbtion is $P_a = 1 - P_d - P_s$. With these probabilities, the decision of which type of reflection or absorption should be chosen takes the following form:

$$\begin{array}{lll} \xi \in [0, P_d] & \longrightarrow & \text{diffuse reflection} \\ \xi \in]P_d, P_s + P_d] & \longrightarrow & \text{specular reflection} \\ \xi \in]P_s + P_d, 1] & \longrightarrow & \text{absorption} \end{array}$$

The power of the reflected photon needs to be adjusted to account for the probability of survival. If, for example, specular reflection was chosen in the example above, the power P_{refl} of the reflected photon is:

$$\begin{array}{l} P_{refl,r} = P_{inc,r}\, s_r / P_s \\ P_{refl,g} = P_{inc,g}\, s_g / P_s \\ P_{refl,b} = P_{inc,b}\, s_b / P_s \end{array}$$

where P_{inc} is the power of the incident photon.

The computed probabilities again ensure us that we do not waste time emitting photons with very low power.

It is simple to extend the selection scheme to also handle transmission, to handle more than three color bands, and to handle other reflection types (for example glossy and directional diffuse).

Why Russian roulette?

Why do we go through this effort to decide what to do with a photon? Why not just trace new photons in the diffuse and specular directions and scale the photon energy accordingly? There are two main reasons why the use of Russian roulette is a very good idea. Firstly, we prefer photons with similar power in the photon map. This makes the radiance estimate much better using only a few photons. Secondly, if we generate, say, two photons per surface interaction then we will have 2^8 photons after 8 interactions. This means 256 photons after 8 interactions compared to 1 photon coming directly from the light source! Clearly this is not good. We want at least as many photons that have only 1–2 bounces as photons that have made 5–8 bounces. The use of Russian roulette is therefore very important in photon tracing.

There is one caveat with Russian roulette. It increases variance on the solution. Instead of using the exact values for reflection and transmission to scale the photon energy we now rely on a sampling of these values that will converge to the correct result as enough photons are used.

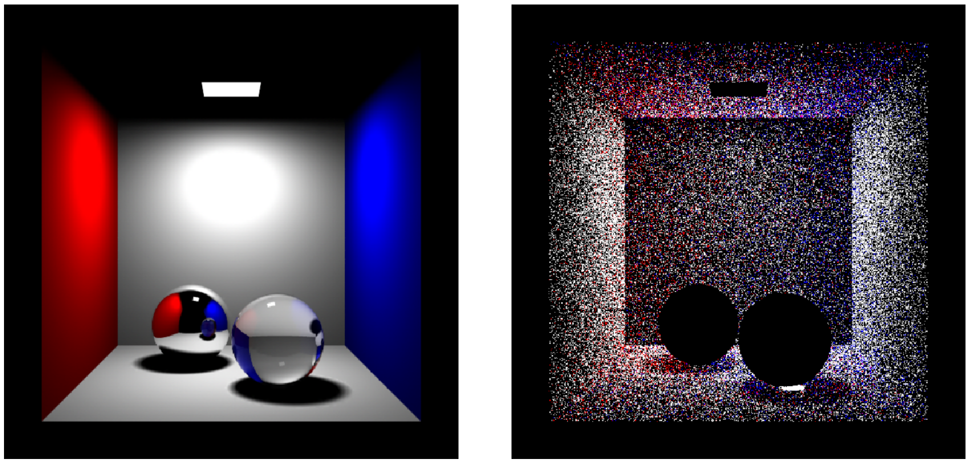

(a) (b)

Figure 2.4: "Cornell box" with glass and chrome spheres: (a) ray traced image (direct illumination and specular reflection and transmission), (b) the photons in the corresponding photon map.

Details on photon tracing and Russian roulette can be found in [Shirley90, Pattanaik93, Glassner95].

2.1.3 Photon storing

This section describes which photon-surface interactions are stored in the photon map. It also describes in more detail the photon map data structure.

Which photon-surface interactions are stored?

Photons are only stored where they hit diffuse surfaces (or, more precisely, non-specular surfaces). The reason is that storing photons on specular surfaces does not give any useful information: the probability of having a matching incoming photon from the specular direction is zero, so if we want to render accurate specular reflections the best way is to trace a ray in the mirror direction using standard ray tracing. For all other photon-surface interactions, data is stored in a global data structure, the *photon map*. Note that each emitted photon can be stored several times along its path. Also, information about a photon is stored at the surface where it is absorbed if that surface is diffuse.

For each photon-surface interaction, the position, incoming photon power, and incident direction are stored. (For practical reasons, there is also space reserved for a flag with each set of photon data. The flag is used during sorting and look-up in the photon map. More on this in the following.)

As an example, consider again the simple scene from Figure 2.3, a “Cornell box” with two spheres. Figure 2.4(a) shows a traditional ray traced image (direct illumination and specular reflection and transmission) of this scene. Figure 2.4(b) shows the photons in the photon map generated for this scene. The high concentration of photons under the glass sphere is caused by focusing of the photons by the glass sphere.

Data structure

Expressed in `C` the following structure is used for each photon [Jensen96b]:

```
struct photon {
  float x,y,z;      // position
  char p[4];        // power packed as 4 chars
  char phi, theta;  // compressed incident direction
  short flag;       // flag used in kdtree
}
```

The power of the photon is represented compactly as 4 bytes using Ward’s packed rgb-format [Ward91]. If memory is not of concern one can use 3 floats to store the power in the red, green, and blue color band (or, in general, one float per color band if a spectral simulation is performed).

The incident direction is a mapping of the spherical coordinates of the photon direction to 65536 possible directions. They are computed as:

```
phi = 255 * (atan2(dy,dx)+PI) / (2*PI)
theta = 255 * acos(dx) / PI
```

where `atan2` is from the standard C library. The direction is used to compute the contribution for non-Lambertian surfaces [Jensen96a], and for Lambertian surfaces it is used to check if a photon arrived at the front of the surface. Since the photon direction is used often during rendering it pays to have a lookup table that maps the theta, phi direction to three floats directly instead of using the formula for spherical coordinates which involves the use of the costly `cos()` and `sin()` functions.

A minor note is that the flag in the structure is a short. Only 2 bits of this flag are used (this is for the splitting plane axis in the kd-tree), and it would be possible to use just one byte for the flag. However for alignment reasons it is preferable to have a 20 byte photon rather than a 19 byte photon — on some architectures it is even a necessity since the float-value in subsequent photons must be aligned on a 4 byte address.

We might be able to compress the information more by using the fact that we know the cube in which the photon is located. The position is, however, used very often when the photons are processed and by using standard float we avoid the overhead involved in extracting the true position from a specialized format.

During the photon tracing pass the photon map is arranged as a flat array of photons. For efficiency reasons this array is re-organized into a balanced kdtree before rendering as explained in section 2.2.

2.1.4 Extension to participating media

Up to this point, all photon interactions have been assumed to happen at object surfaces; all volumes were implicitly assumed to not affect the photons. However, it is simple to extend the photon map method to handle *participating media*, i.e. volumes that participate in the light transport. In scenes with participating media, the photons are stored within the media in a seperate *volume photon map* [Jensen98].

Photon emission, tracing, and storage

Photons can be emitted from volumes as well as from surfaces and points. For example, the light from a candle flame can be simulated by emitting photons from a flame-shaped volume.

When a photon travels through a participating medium, it has a certain probability of being scattered or absorbed in the medium. The probability depends on the density of the medium and on the distance the photon travels through the medium: the denser the medium, the shorter the average distance before a photon interaction happens. Photons are stored at the positions where a scattering event happens. The exception is photons that come directly from the light source since direct illumination is evaluated using ray tracing. This separation was introduced in [Jensen98] and it allows us to compute the in-scattered radiance in a medium simply by a lookup in the photon map.

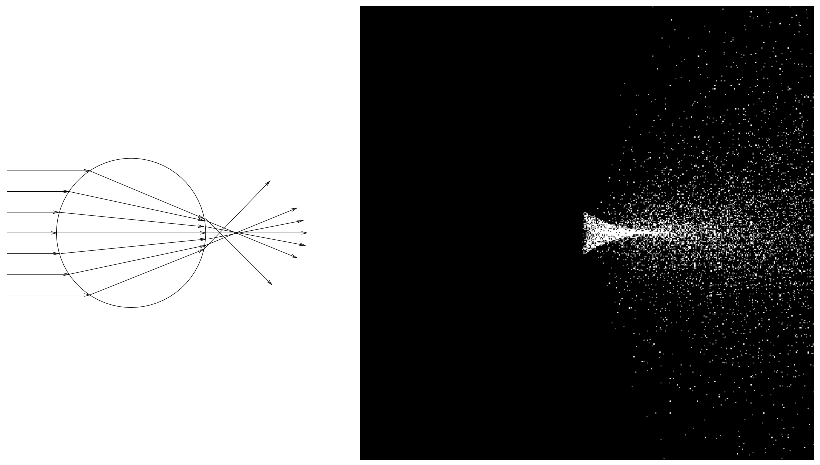


Figure 2.5: Sphere in fog: (a) schematic diagram of light paths, (b) the caustic photons in the photon map.

As an example, consider a glass sphere in fog illuminated by directional light. Figure 2.5(a) shows a schematic diagram of the photon paths as photons are being focused by refraction in the glass sphere. Figure 2.5(b) shows the caustic photons stored in the photon map.

Multiple scattering, anisotropic scattering, and non-homogeneous media

The simple example above only shows the photon interaction in the fog after refraction by the glass sphere, and the photon paths are terminated at the first scattering event. General multiple scattering is simulated simply by letting the photons scatter everywhere and continuously after the first interaction. The path can be terminated using Russian roulette.

The fog in the example has uniform density, but it is not difficult to handle media with nonuniform density (aka. nonhomogeneous media), since we use ray marching to integrate the properties of the medium. A simple ray marcher works by dividing the medium into little steps [Ebert94]. The accumulated density (integrated extinction coefficient) is updated at each step, and based on a precomputed probability it is determined whether the photon should be absorbed, scattered, or whether another step is necessary.

For more complicated examples of scattering in participating media, including anisotropic and nonhomogeneous media and complex geometry, see [Jensen98].

2.1.5 Three photon maps

For efficiency reasons, it pays off to divide the stored photons into three photon maps:

Caustic photon map: contains photons that have been through at least one specular reflection before hitting a diffuse surface: LS^+D.

Global photon map: an approximate representation of the global illumination solution for the scene for all diffuse surfaces: $L\{S|D|V\}^*D$

Volume photon map: indirect illumination of a participating medium: $L\{S|D|V\}^+V$.

Here, we used the grammar from [Heckbert90] to describe the photon paths: L means emission from the light source, S is specular reflection or transmission, D is diffuse (ie. non-specular) reflection or transmission, and V is volume scattering. The notation $\{x|y|z\}$ means "one of x, y, or z", x^+ means one or several repeats of x, and x^* means zero or several repeats of x.

The reason for keeping three separate photon maps will become clear in section 2.4. A separate photon tracing pass is performed for the caustic photon map since it should be of high quality and therefore often needs more photons than the global photon map and the volume photon map.

The construction of the photon maps is most easily achieved by using two separate photon tracing steps in order to build the caustics photon map and the global photon map (including the volume photon map). This is illustrated in Figure 2.6 for a simple test scene with a glass sphere and 2 diffuse walls. Figure 2.6(a) shows the construction of the caustics photon map with a dense distribution of photons, and Figure 2.6(b) shows the construction of the global photon map with a more coarse distribution of photons.

2.2 Preparing the photon map for rendering

Photons are only generated during the photon tracing pass — in the rendering pass the photon map is a static data structure that is used to compute estimates of the

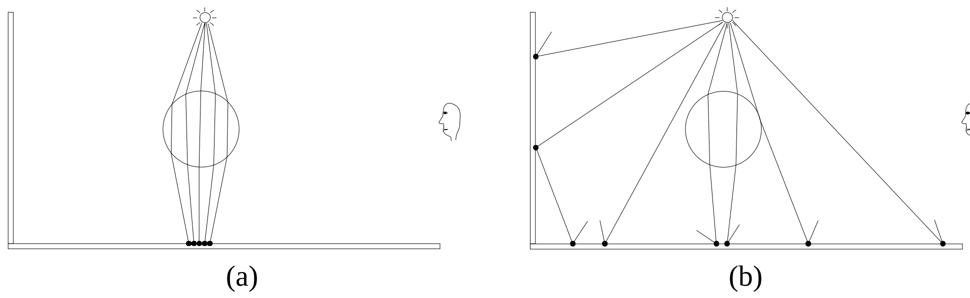


Figure 2.6: Building (a) the caustics photon map and (b) the global photon map.

incoming flux and the reflected radiance at many points in the scene. To do this it is necessary to locate the nearest photons in the photon map. This is an operation that is done extremely often, and it is therefore a good idea to optimize the representation of the photon map before the rendering pass such that finding the nearest photons is as fast as possible.

First, we need to select a good data structure for representing the photon map. The data structure should be compact and at the same time allow for fast nearest neighbor searching. It should also be able to handle highly non-uniform distributions — this is very often the case in the caustics photon map. A natural candidate that handles these requirements is *a balanced kd-tree* [Bentley75]. Examples of using a balanced versus an unbalanced kd-tree can be found in [Jensen96a].

2.2.1 The balanced kd-tree

The time it takes to locate one photon in a balanced kd-tree has a worst time performance of $O(\log N)$, where N is the number of photons in the tree. Since the photon map is created by tracing photons randomly through a model one might think that a dynamically built kd-tree would be quite well balanced already. However, the fact that the generation of the photons at the light source is based on the projection map combined with the fact that models often contain highly directional reflectance models easily results in a skewed tree. Since the tree is created only once and used many times during rendering it is quite natural to consider balancing the tree. Another argument that is perhaps even more important is the fact that a balanced kd-tree can be represented using a heap-like data-structure [Sedgewick92] which means that explicitly storing the pointers to the sub-trees at each node is no longer

```
kdtree *balance( points ) {
  Find the cube surrounding the points
  Select dimension $dim$ in which the cube is largest
  Find median of the points in $dim$
  $s1$ = all points below median
  $s2$ = all points above median
  node = median
  node.left = balance( $s1$ )
  node.right = balance( $s2$ )
  return node
}
```

Figure 2.7: Pseudocode for balancing the photon map

necessary. (Array element 1 is the tree root, and element i has element $2i$ as left child and element $2i + 1$ as right child.) This can lead to considerable savings in memory when a large number of photons is used.

2.2.2 Balancing

Balancing a kd-tree is similar to balancing a binary tree. The main difference is the choice at each node of a splitting dimension. When a splitting dimension of a set is selected, the median of the points in that dimension is chosen as the root node of the tree representing the set and the left and right subtrees are constructed from the two sets separated by the median point. The choice of a splitting dimension is based on the distribution of points within the set. One might use either the variance or the maximum distance between the points as a criterion. We prefer a choice based upon maximum distance since it can be computed very efficiently (even though a choice based upon variance might be slightly better). The splitting dimension is thus chosen as the one which has the largest maximum distance between the points.

Figure 2.7 contains a pseudocode outline for the balancing algorithm [Jensen96c].

To speed up the balancing process it is convenient to use an array of pointers to the photons. In this way only pointers needs to be shuffled during the median search. An efficient median search algorithm can be found in most textbooks on algorithms — see for example [Sedgewick92] or [Cormen89].

The complexity of the balancing algorithm is $O(N \log N)$ where N is the number of photons in the photon map. In practice, this step only takes a few seconds even for several million photons.

2.3 The radiance estimate

A fundamental component of the photon map method is the ability to compute radiance estimates at any non-specular surface point in any given direction.

2.3.1 Radiance estimate at a surface

The photon map can be seen as a representation of the incoming flux; to compute radiance we need to integrate this information. This can be done using the expression for reflected radiance:

$$L_r(x,\vec{\omega}) = \int_{\Omega_x} f_r(x,\vec{\omega}',\vec{\omega}) L_i(x,\vec{\omega}') |\vec{n}_x \cdot \vec{\omega}'| \, d\omega_i' \,, \tag{2.3}$$

where L_r is the reflected radiance at x in direction $\vec{\omega}$. Ω_x is the (hemi)sphere of incoming directions, f_r is the BRDF (bidirectional reflectance distribution function) [Nicodemus77] at x and L_i is the incoming radiance. To evaluate this integral we need information about the incoming radiance. Since the photon map provides information about the incoming flux we need to rewrite this term. This can be done using the relationship between radiance and flux:

$$L_i(x,\vec{\omega}') = \frac{d^2\Phi_i(x,\vec{\omega}')}{\cos\theta_i \, d\omega_i' \, dA_i} \,, \tag{2.4}$$

and we can rewrite the integral as

$$\begin{aligned} L_r(x,\vec{\omega}) &= \int_{\Omega_x} f_r(x,\vec{\omega}',\vec{\omega}) \frac{d^2\Phi_i(x,\vec{\omega}')}{\cos\theta_i \, d\omega_i' \, dA_i} |\vec{n}_x \cdot \vec{\omega}'| \, d\omega_i' \\ &= \int_{\Omega_x} f_r(x,\vec{\omega}',\vec{\omega}) \frac{d^2\Phi_i(x,\vec{\omega}')}{dA_i} \,. \end{aligned} \tag{2.5}$$

The incoming flux Φ_i is approximated using the photon map by locating the n photons that has the shortest distance to x. Each photon p has the power $\Delta\Phi_p(\vec{\omega}_p)$ and by assuming that the photons intersects the surface at x we obtain

$$L_r(x,\vec{\omega}) \approx \sum_{p=1}^{n} f_r(x,\vec{\omega}_p,\vec{\omega}) \frac{\Delta\Phi_p(x,\vec{\omega}_p)}{\Delta A} \,. \tag{2.6}$$

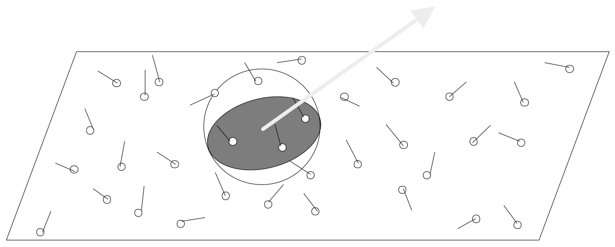


Figure 2.8: Radiance is estimated using the nearest photons in the photon map.

The procedure can be imagined as expanding a sphere around x until it contains n photons (see Figure 2.8) and then using these n photons to estimate the radiance.

Equation 2.6 still contains ΔA which is related to the density of the photons around x. By assuming that the surface is locally flat around x we can compute this area by projecting the sphere onto the surface and use the area of the resulting circle. This is indicated by the hatched area in Figure 2.8 and equals:

$$\Delta A = \pi r^2 \, , \tag{2.7}$$

where r is the radius of the sphere – ie. the largest distance between x and each of the photons.

This results in the following equation for computing reflected radiance at a surface using the photon map:

$$L_r(x, \vec{\omega}) \approx \frac{1}{\pi r^2} \sum_{p=1}^{N} f_r(x, \vec{\omega}_p, \vec{\omega}) \Delta\Phi_p(x, \vec{\omega}_p) \, . \tag{2.8}$$

This estimate is based on many assumptions and the accuracy depends on the number of photons used in the photon map and in the formula. Since a sphere is used to locate the photons one might easily include wrong photons in the estimate in particular in corners and at sharp edges of objects. Edges and corners also causes the area estimate to be wrong. The size of those regions in which these errors occur depends largely on the number of photons in the photon map and in the estimate. As more photons are used in the estimate and in the photon map, formula 2.8 becomes more accurate. If we ignore the error due to limited accuracy of the representation of the position, direction and flux, then we can go to the limit and

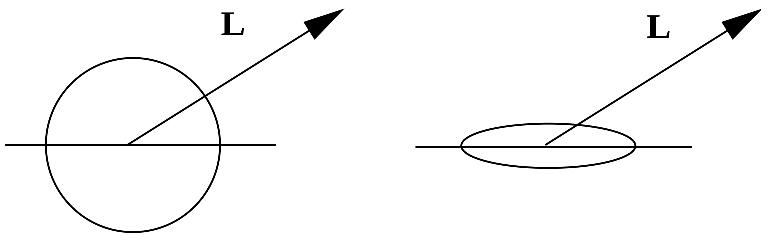


Figure 2.9: Using a sphere (left) and using a disc (right) to locate the photons.

increase the number of photons to infinity. This gives the following interesting result where N is the number of photons in the photon map:

$$\lim_{N\to\infty} \frac{1}{\pi r^2} \sum_{p=1}^{\lfloor N^\alpha \rfloor} f_r(x, \vec{\omega}_p, \vec{\omega}) \Delta\Phi_p(x, \vec{\omega}_p) = L_r(x, \vec{\omega}) \text{ for } \alpha \in]0, 1[\,. \tag{2.9}$$

This formulation applies to all points x located on a locally flat part of a surface for which the BRDF, does not contain the Dirac delta function (this excludes perfect specular reflection). The principle in equation 2.9 is that not only will an infinite amount of photons be used to represent the flux within the model but an infinite amount of photons will also be used to estimate radiance and the photons in the estimate will be located within an infinitesimal sphere. The different degrees of infinity are controlled by the term N^α where $\alpha \in]0, 1[$. This ensures that the number of photons in the estimate will be infinitely fewer than the number of photons in the photon map.

Equation 2.9 means that we can obtain arbitrarily good radiance estimates by just using enough photons! In finite element based approaches it is more complicated to obtain arbitrary accuracy since the error depends on the resolution of the mesh, the resolution of the directional representation of radiance and the accuracy of the light simulation.

Figure 2.8 shows how locating the nearest photons is similar to expanding a sphere around x and using the photons within this sphere. It is possible to use other volumes than the sphere in this process. One might use a cube instead, a cylinder or perhaps a disc. This could be useful to either obtain an algorithm that is faster at locating the nearest photons or perhaps more accurate in the selection of photons. If a different volume is used then ΔA in equation 2.6 should be replaced by the area of the intersection between the volume and the tangent plane touching the surface at x. The sphere has the obvious advantage that the projected area and the distance computations are very simple and thus efficiently computed. A more accurate volume can be obtained by modifying the sphere into a disc (ellipsoid) by compressing the sphere in the direction of the surface normal at x (shown in Figure 2.9) [Jensen96c]. The advantage of using a disc would be that fewer "false photons" are used in the estimate at edges and in corners. This modification works pretty well at the edges in a room, for instance, since it prevents photons on the walls to leak down to the floor. One issue that still occurs, however, is that the area estimate might be wrong or photons may leak into areas where they do not belong. This problem is handled primarily by the use of filtering.

2.3.2 Filtering

If the number of photons in the photon map is too low, the radiance estimates becomes blurry at the edges. This artifact can be pleasing when the photon map is used to estimate indirect illumination for a distribution ray tracer (see section 2.4 and Figure 2.15) but it is unwanted in situations where the radiance estimate represents caustics. Caustics often have sharp edges and it would be nice to preserve these edges without requiring too many photons.

To reduce the amount of blur at edges, the radiance estimate is filtered. The idea behind filtering is to increase the weight of photons that are close to the point of interest, x. Since we use a sphere to locate the photons it would be natural to assume that the filters should be three-dimensional. However, photons are stored at surfaces which are two-dimensional. The area estimate is also based on the assumption that photons are located on a surface. We therefore need a 2d-filter (similar to image filters) which is normalized over the region defined by the photons.

The idea of filtering caustics is not new. Collins [Collins94] has examined several filters in combination with illumination maps. The filters we have examined are two radially symmetric filters: the cone filter and the Gaussian filter [Jensen96c], and the specialized differential filter introduced in [Jensen95a]. For examples of more advanced filters see Myszkowski et al. [Myszkowski97].

The cone filter

The cone-filter [Jensen96c] assigns a weight, w_{pc}, to each photon based on the distance, d_p, between x and the photon p. This weight is:

$$w_{pc} = 1 - \frac{d_p}{k\,r}\,, \tag{2.10}$$

where $k \geq 1$ is a filter constant characterizing the filter and r is the maximum distance. The normalization of the filter based on a 2d-distribution of the photons is $1 - \frac{2}{3k}$ and the filtered radiance estimate becomes:

$$L_r(x, \vec{\omega}) \approx \frac{\sum_{p=1}^{N} f_r(x, \vec{\omega}_p, \vec{\omega}) \Delta\Phi_p(x, \vec{\omega}_p) w_{pc}}{(1 - \frac{2}{3k})\pi r^2}\,. \tag{2.11}$$

The Gaussian filter

The Gaussian filter [Jensen96c] has previously been reported to give good results when filtering caustics in illumination maps [Collins94]. It is easy to use the Gaussian filter with the photon map since we do not need to warp the filter to some surface function. Instead we use the assumption about the locally flat surfaces and we can use a simple image based Gaussian filter [Pavicic90] and the weight w_{pg} of each photon becomes

$$w_{pg} = \alpha \left[1 - \frac{1 - e^{-\beta \frac{d_p^2}{2r^2}}}{1 - e^{-\beta}} \right]\,, \tag{2.12}$$

where d_p is the distance between the photon p and x and $\alpha = 0.918$ and $\beta = 1.953$ (see [Pavicic90] for details). This filter is normalized and the only change to equation 2.8 is that each photon contribution is multiplied by w_{pg}:

$$L_r(x, \vec{\omega}) \approx \sum_{p=1}^{N} f_r(x, \vec{\omega}_p, \vec{\omega}) \Delta\Phi_p(x, \vec{\omega}_p) w_{pg}\,. \tag{2.13}$$

Differential checking

In [Jensen95a] it was suggested to use a filter based on differential checking. The idea is to detect regions near edges in the estimation process and use less photons in these regions. In this way we might get some noise in the estimate but that is often preferable to blurry edges.

The radiance estimate is modified based on the following observation: when adding photons to the estimate, near an edge the changes of the estimate will be monotonic. That is, if we are just outside a caustic and we begin to add photons to the estimate (by increasing the size of the sphere centered at x that contains the photons), then it can be observed that the value of the estimate is increasing as we add more photons; and vice versa when we are inside the caustic. Based on this observation, differential checking can be added to the estimate — we stop adding photons and use the estimate available if we observe that the estimate is either constantly increasing or decreasing as more photons are added.

2.3.3 The radiance estimate in a participating medium

For the radiance estimate presented so far we have assumed that the photons are located on a surface. For photons in a participating medium the formula changes to [Jensen98]:

$$\begin{aligned} L_i(x,\vec{\omega}) &= \int_\Omega f(x,\vec{\omega}',\vec{\omega})\, L(x,\vec{\omega}')\, d\omega' \\ &= \int_\Omega f(x,\vec{\omega}',\vec{\omega})\, \frac{d^2\Phi(x,\vec{\omega}')}{\sigma_s(x)\, d\omega'\, dV}\, d\omega' \\ &= \frac{1}{\sigma_s(x)} \int_\Omega f(x,\vec{\omega}',\vec{\omega})\, \frac{d^2\Phi(x,\vec{\omega}')}{dV} \\ &\approx \frac{1}{\sigma_s(x)} \sum_{p=1}^{n} f(x,\vec{\omega}_p',\vec{\omega})\, \frac{\Delta\Phi_p(x,\vec{\omega}_p')}{\frac{4}{3}\pi r^3}\,, \end{aligned} \tag{2.14}$$

where L_i is the in-scattered radiance, and the volume $dV = \frac{4}{3}\pi r^3$ is the volume of the sphere containing the photons. $\sigma_s(x)$ is the scattering coefficient at x and f is the phase-function.

2.3.4 Locating the nearest photons

Efficiently locating the nearest photons is critical for good performance of the photon map algorithm. In scenes with caustics, multiple diffuse reflections, and/or participating media there can be a large number of photon map queries.

Fortunately the simplicity of the kd-tree permits us to implement a simple but quite efficient search algorithm. This search algorithm is a straight forward extension of standard search algorithms for binary trees [Cormen89, Sedgewick92, Horowitz93]. It is also related to range searching where kd-trees are commonly used as they have optimal storage and good performance [Preparata85]. The nearest neighbors query for kd-trees has been described extensively in several publications by Bentley et al. including [Bentley75, Bentley79a, Bentley79b, Bentley80]. More recent publications include [Preparata85, Sedgewick92]. Some of these papers go beyond our description of a nearest neighbors query by adding modifications and extensions to the kd-tree to further reduce the cost of searching. We do not implement these extensions because we want to maintain the low storage overhead of the kd-tree as this is an important aspect of the photon map.

Locating the nearest neighbors in a kd-tree is similar to range searching [Preparata85] in the sense that we want to locate photons within a given volume. For the photon map it makes sense to restrict the size of the initial search range since the contribution from a fixed number of photons becomes small for large regions. This simple observation is particularly important for caustics since they often are concentrated in a small region. A search algorithm that does not limit the search range will be slow in such situations since a large part of the kd-tree will be visited for regions with a sparse number of photons.

A generic nearest neighbors search algorithm begins at the root of the kd-tree, and adds photons to a list if they are within a certain distance. For the n nearest neighbors the list is sorted such that the photon that is furthest away can be deleted if the list contains n photons and a new closer photon is found. Instead of naive sorting of the full list it is better to use a max-heap [Preparata85, Sedgewick92, Horowitz93]. A max-heap (also known as a priority queue) is a very efficient way of keeping track of the element that is furthest away from the point of interest. When the max-heap is full, we can use the distance d to the root element (ie. the photon that is furthest away) to adjust the range of the query. Thus we skip parts of the kd-tree that are further away than d.

Another simple observation is that we can use squared distances — we do not need the real distance. This removes the need of a square root calculation per distance check.

The pseudo-code for the search algorithm is given in Figure 2.10. A simple implementation of this routine is available with source code at [MegaPov00].

For this search algorithm it is necessary to provide an initial maximum search radius. A well-chosen radius allows for good pruning of the search reducing the number of photons tested. A maximum radius that is too low will on the other hand

```
given the photon map, a position $x$ and a max search distance $d^2$
this recursive function returns a heap $h$ with the nearest photons.
Call with locate_photons(1) to initiate search at the root of the kd-tree

locate_photons( $p$ ) {
  if ( $2p+1$ < number of photons ) {
    examine child nodes
            Compute distance to plane (just a subtract)
    $\delta$ = signed distance to splitting plane of node $n$
    if ($\delta < 0$) {
            We are left of the plane - search left subtree first
      locate_photons( $2p$ )
      if ( $\delta^2 < d^2$ )
        locate_photons( $2p+1$ )      check right subtree
    } else {
            We are right of the plane - search right subtree first
      locate_photons( $2p+1$ )
      if ( $\delta^2 < d^2$ )
        locate_photons( $2p$ )      check left subtree
    }
  }
          Compute true squared distance to photon
  $\delta^2$ = squared distance from photon $p$ to $x$
  if ( $\delta^2 < d^2$ ) {            Check if the photon is close enough?
    insert photon into max heap $h$
            Adjust maximum distance to prune the search
    $d^2$ = squared distance to photon in root node of $h$
  }
}
```

Figure 2.10: Pseudocode for locating the nearest photons in the photon map

introduce noise in the photon map estimates. The radius can be chosen based on an error metric or the size of the scene. The error metric could for example take the average energy of the stored photons into account and compute a maximum radius from that assuming some allowed error in the radiance estimate.

A few extra optimizations can be added to this routine, for example a delayed construction of the max heap to the time when the number of photons needed has been found. This is particularly useful when the requested number of photons is large.

Nathan Kopp has implemented a slightly different optimization in an extended

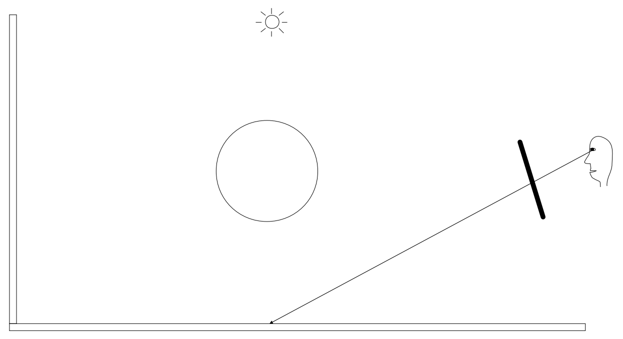


Figure 2.11: Tracing a ray through a pixel.

version of the Persistence Of Vision Ray Tracer (POV) called `MegaPov` (available at [MegaPov00]). In his implementation the initial maximum search radius is set to a very low value. If this value turns out to be too low, another search is performed with a higher maximum radius. He reports good timings and results from this technique [Kopp99].

Another change to the search routine is to use the disc check as described earlier. This is useful to avoid incorrect color bleeding and particularly helpful if the gathering step is not used and the photons are visualized directly.

2.4 Rendering

Given the photon map and the ability to compute a radiance estimate from it, we can proceed with the rendering pass. The photon map is view independent, and therefore a single photon map constructured for an environment can be utilized to render the scene from any desired view. There are several different ways in which the photon map can be visualized. A very fast visualization technique has been presented by Myszkowski et al. [Myszkowski97, Volevich99] where photons are used to compute radiosity values at the vertices of a mesh.

In this note we will focus on the full global illumination approach as presented in [Jensen96b]. Initially we will ignore the presence of participating media; at the end of the note we have added some notes for this case.

The final image is rendered using distribution ray tracing in which the pixel radiance is computed by averaging a number of sample estimates. Each sample consists of tracing a ray from the eye through a pixel into the scene (see Figure 2.11). The radiance returned by each ray equals the outgoing radiance in the direction of the ray leaving the point of intersection at the first surface intersected by the ray. The outgoing radiance, L_o, is the sum of the emitted, L_e, and the reflected radiance

$$L_o(x,\vec{\omega}) = L_e(x,\vec{\omega}) + L_r(x,\vec{\omega})\,, \tag{2.15}$$

where the reflected radiance, L_r, is computed by integrating the contribution from the incoming radiance, L_i,

$$L_r(x,\vec{\omega}) = \int\limits_{\Omega_x} f_r(x,\vec{\omega}',\vec{\omega}) L_i(x,\vec{\omega}') \cos\theta_i \, d\omega_i'\,, \tag{2.16}$$

where f_r is the bidirectional reflectance distribution function (BRDF), and Ω_x is the set of incoming directions around x.

L_r can be computed using Monte Carlo integration techniques like path tracing and distribution ray tracing. These methods are very costly in terms of rendering time and a more efficient approach can be obtained by using the photon map in combination with our knowledge of the BRDF and the incoming radiance.

The BRDF is separated into a sum of two components: A specular/glossy, $f_{r,s}$, and a diffuse, $f_{r,d}$

$$f_r(x,\vec{\omega}',\vec{\omega}) = f_{r,s}(x,\vec{\omega}',\vec{\omega}) + f_{r,d}(x,\vec{\omega}',\vec{\omega})\,. \tag{2.17}$$

The incoming radiance is classified using three components:

- $L_{i,l}(x,\vec{\omega}')$ is direct illumination by light coming from the light sources.
- $L_{i,c}(x,\vec{\omega}')$ is caustics — indirect illumination from the light sources via specular reflection or transmission.
- $L_{i,d}(x,\vec{\omega}')$ is indirect illumination from the light sources which has been reflected diffusely at least once.

The incoming radiance is the sum of these three components:

$$L_i(x,\vec{\omega}') = L_{i,l}(x,\vec{\omega}') + L_{i,c}(x,\vec{\omega}') + L_{i,d}(x,\vec{\omega}')\,. \tag{2.18}$$

By using the classifications of the BRDF and the incoming radiance we can split the expression for reflected radiance into a sum of four integrals:

$$\begin{aligned} L_r(x,\vec{\omega}) &= \int_{\Omega_x} f_r(x,\vec{\omega}',\vec{\omega}) L_i(x,\vec{\omega}') \cos\theta_i \, d\omega_i' \\ &= \int_{\Omega_x} f_r(x,\vec{\omega}',\vec{\omega}) L_{i,l}(x,\vec{\omega}') \cos\theta_i \, d\omega_i' + \\ & \quad \int_{\Omega_x} f_{r,s}(x,\vec{\omega}',\vec{\omega}) (L_{i,c}(x,\vec{\omega}') + L_{i,d}(x,\vec{\omega}')) \cos\theta_i \, d\omega_i' + \\ & \quad \int_{\Omega_x} f_{r,d}(x,\vec{\omega}',\vec{\omega}) L_{i,c}(x,\vec{\omega}') \cos\theta_i \, d\omega_i' + \\ & \quad \int_{\Omega_x} f_{r,d}(x,\vec{\omega}',\vec{\omega}) L_{i,d}(x,\vec{\omega}') \cos\theta_i \, d\omega_i' \, . \end{aligned} \tag{2.19}$$

This is the equation used whenever we need to compute the reflected radiance from a surface. In the following sections we discuss the evaluation of each of the integrals in the equation in more detail. We distinguish between two different situations: an accurate and an approximate.

The accurate computation is used if the surface is seen directly by the eye or perhaps via a few specular reflections. It is also used if the distance between the ray origin and the intersection point is below a small threshold value — to eliminate potential inaccurate color bleeding effects in corners. The approximate evaluation is used if the ray intersecting the surface has been reflected diffusely since it left the eye or if the ray contributes only little to the pixel radiance.

2.4.1 Direct illumination

Direct illumination is given by the term

$$\int_{\Omega_x} f_r(x,\vec{\omega}',\vec{\omega}) L_{i,l}(x,\vec{\omega}') \cos\theta_i \, d\omega_i' \, ,$$

and it represents the contribution to the reflected radiance due to direct illumination. This term is often the most important part of the reflected radiance and it has to be computed accurately since it determines light effects to which the eye is highly sensitive such as shadow edges.

Computing the contribution from the light sources is quite simple in ray tracing based methods. At the point of interest shadow rays are sent towards the light sources to test for possible occlusion by objects. This is illustrated in Figure 2.12.

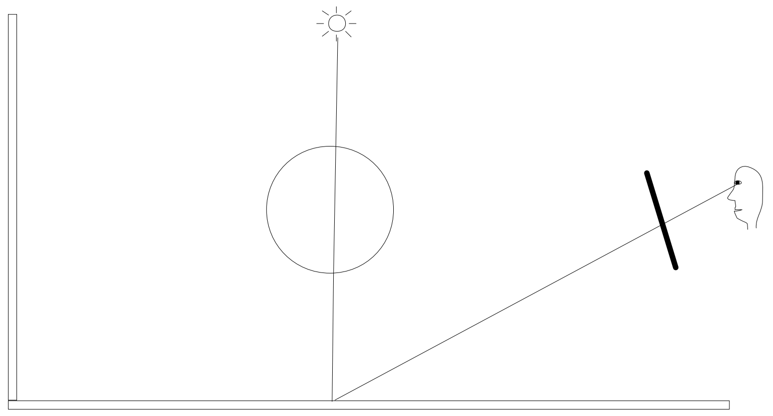


Figure 2.12: Accurate evaluation of the direct illumination.

If a shadow ray does not hit an object the contribution from the light source is included in the integral otherwise it is neglected. For large area light sources several shadow rays are used to properly integrate the contribution and correctly render penumbra regions. This strategy can however be very costly since a large number of shadow rays is needed to properly integrate the direct illumination.

Using a derivative of the photon map method we can compute shadows more efficiently using shadow photons [Jensen95c]. This approach can lead to considerable speedups in scenes with large penumbra-regions that are normally very costly to render using standard ray tracing. The approach is stochastic though, so it might miss shadows from small objects in case these aren't intersected by any photons. This is a problem with all techniques that use stochastic evaluation of visibility.

The approximate evaluation is simply the radiance estimate obtained from the global photon map (no shadow rays or light source evaluations are used). This is seen in Figure 2.15 where the global photon map is used in the evaluation of the incoming light for the secondary diffuse reflection.

2.4.2 Specular and glossy reflection

Specular and glossy reflection is computed by evaluation of the term

$$\int_{\Omega_x} f_{r,s}(x, \vec{\omega}', \vec{\omega})(L_{i,c}(x, \vec{\omega}') + L_{i,d}(x, \vec{\omega}')) \cos\theta_i \, d\omega_i' \, .$$

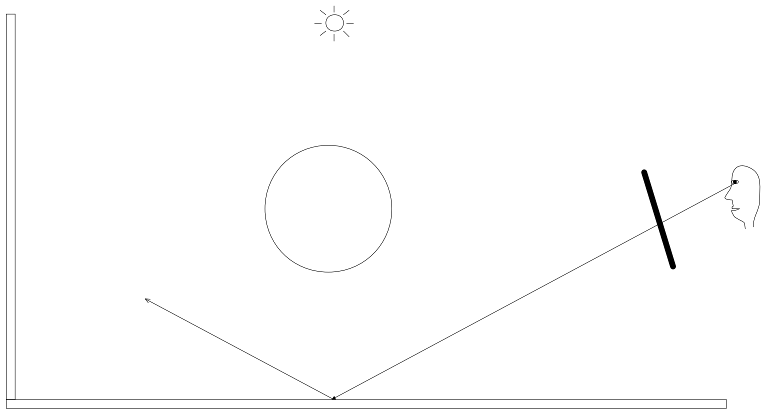


Figure 2.13: Rendering specular and glossy reflections.

The photon map is not used in the evaluation of this integral since it is strongly dominated by $f_{r,s}$ which has a narrow peak around the mirror direction. Using the photon map to optimize the integral would require a huge number of photons in order to make a useful classification of the different directions within the narrow peak of $f_{r,s}$. To save memory this strategy is not used and the integral is evaluated using standard Monte Carlo ray tracing optimized with importance sampling based on $f_{r,s}$. This is still quite efficient for glossy surfaces and the integral can in most situations be computed using only a small number of sample rays.

This is illustrated in Figure 2.13.

2.4.3 Caustics

Caustics are represented by the integral

$$\int_{\Omega_x} f_{r,d}(x, \vec{\omega}', \vec{\omega}) L_{i,c}(x, \vec{\omega}') \cos\theta_i \, d\omega_i' \, .$$

The evaluation of this term is dependent on whether an accurate or an approximate computation is required. In the accurate computation, the term is solved by using a radiance estimate from the caustics photon map. The number of photons in the caustics photon map is high and we can expect good quality of the estimate. Caustics are never computed using Monte Carlo ray tracing since this is a very inefficient method when it comes to rendering caustics. The approximate evaluation

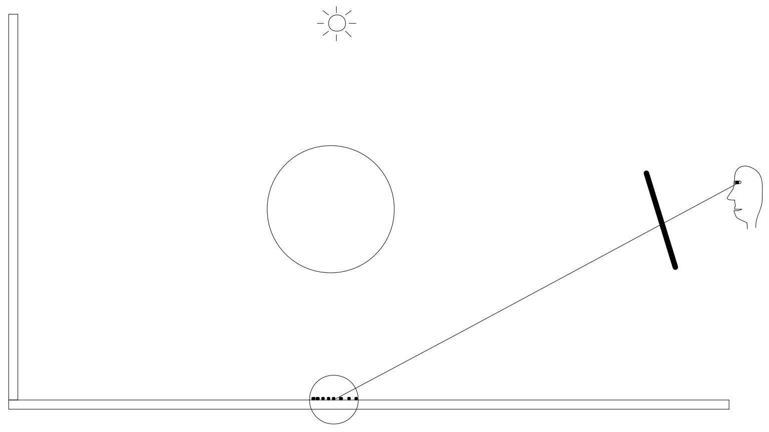


Figure 2.14: Rendering caustics.

of the integral is included in the radiance estimate from the global photon map.

This is illustrated in Figure 2.14.

2.4.4 Multiple diffuse reflections

The last term in equation 2.19 is

$$\int_{\Omega_x} f_{r,d}(x, \vec{\omega}', \vec{\omega}) L_{i,d}(x, \vec{\omega}') \cos\theta_i \, d\omega_i' \, .$$

This term represents incoming light that has been reflected diffusely at least once since it left the light source. The light is then reflected diffusely by the surface (using $f_{r,d}$). Consequently the resulting illumination is very “soft”.

The approximate evaluation of this integral is a part of the radiance estimate based on the global photon map.

The accurate evaluation of the integral is calculated using Monte Carlo ray tracing optimized using the BRDF with an estimate of the flux as described in [Jensen95b]. An important optimization at Lambertian surfaces is the use of Ward’s irradiance gradient caching scheme [Ward88, Ward92]. This means that we only compute indirect illumination on Lambertian surfaces if we cannot interpolate with sufficient accuracy from previously computed values. The advantage of using the photon map compared to just using the irradiance gradient caching method is that we avoid having to trace multiple bounces of indirect illumination and we can use the information in the photon map to concentrate our samples into the important directions.

This is illustrated in Figure 2.15.

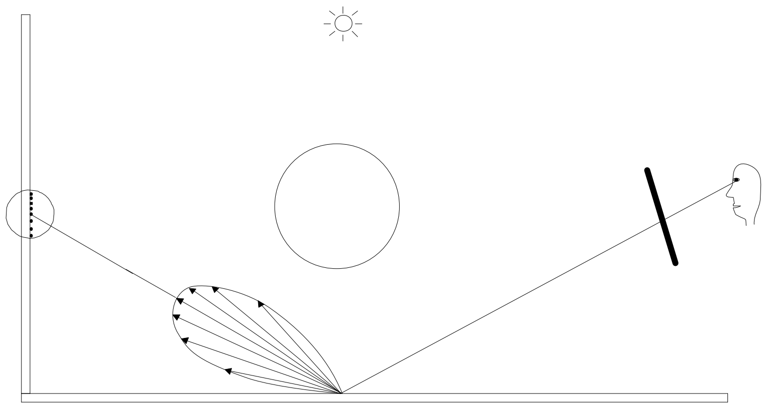


Figure 2.15: Computing indirect diffuse illumination with importance sampling.

2.4.5 Participating media

In the presence of participating media we can still use the framework as presented so far. The main difference is that we need to take the media into account as we trace rays through the scene. This can be done quite efficiently using ray marching and the volume radiance estimate as described in [Jensen98].

2.4.6 Why distribution ray tracing?

The rendering method presented here is a combination of many algorithms. In order to render accurate images without using too many photons a distribution ray tracer is used to compute illumination seen directly by the eye. One might consider visualizing the global photon map directly, and this would indeed be a full global illumination solution (it would be similar to the density estimation approach presented in [Shirley95]). The problem with this approach is that an accurate solution requires a large number of photons. Significantly fewer photons are necessary when a distribution ray tracer is used to evaluate the first diffuse reflection. If a blurry solution is not a problem (for example for previewing) then a direct visualization of the photon map can be used. For more accurate results it is often necessary to use more than 1000 photons in the radiance estimate (see the results section for some examples).

2.5 Examples

In this section we present some examples of scenes rendered using photon maps. Please see the photon map web-page at `http://www.gk.dtu.dk/photonmap` for the latest results. Also refer to the papers included in these notes for more examples.

All the images have been rendered using the `Dali` rendering program. `Dali` is an extremely flexible renderer that supports ray tracing with global illumination and participating media. The global illumination simulation code based on photon maps is a module in `Dali` that is loaded at runtime. All material and geometry code is also represented via modules that are loaded at runtime. `Dali` is multi-threaded and all images have been rendered on a dual PentiumII-400 PC running Linux. The width of each image is 1024 pixels and 4 samples per pixel have been used.

2.5.1 The Cornell box

Most global illumination papers feature a simulation of the Cornell box, and so does this note. Since we are not limited to radiosity our version of the Cornell box is slightly different. It has a mirror sphere and a glass sphere instead of the two cubes featured in the original Cornell box (the original Cornell box can be found at `http://www.graphics.cornell.edu/online/box/`). Classic radiosity methods have difficulties handling curved specular objects, but ray tracing methods (including the photon map method) have no problems with these.

Ray tracing

The image in Figure 2.16 shows the *ray traced* version of the Cornell box. Notice the sharp shadows and the black ceiling of the box due to lack of area lights and global illumination. Rendering time was 3.5 seconds.

Ray tracing with soft shadows

In Figure 2.17 soft shadows have been added. It has been reported that some people associate soft shadows with global illumination, but in the Cornell box example it is still obvious that something is missing. The ceiling is still black. Rendering time was 21 seconds.

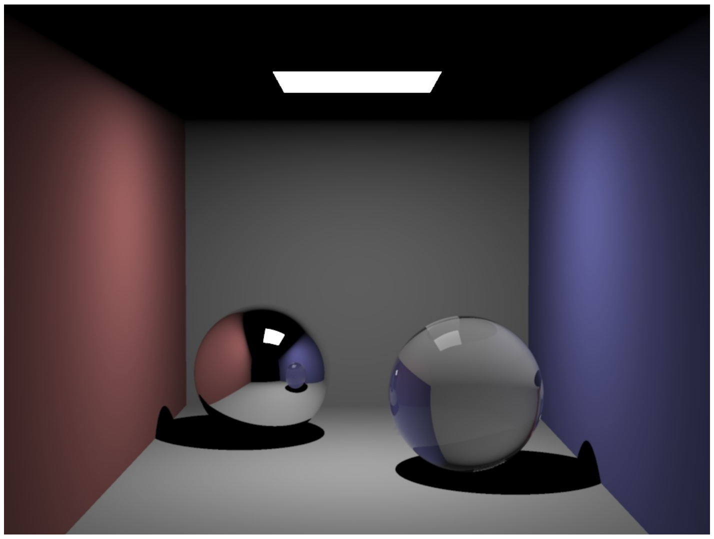


Figure 2.16: Ray traced Cornell box with sharp shadows.

Adding caustics

The image in Figure 2.18 includes the caustics photon map. Notice the bright spot below the glass sphere and on the right wall (due to light reflected of the mirror sphere and transmitted through the glass sphere). Also notice the faint illumination of the ceiling. The caustics photon map has 50000 photons and the estimate uses up to 60 photons. Photon tracing took 2 seconds. Rendering time was 34 seconds. We did not use any filtering of the caustics photons. A maximum search distance of 0.15 was used for the caustics photon map (the depth of the Cornell box is 5 units). Using a search distance of 0.5 increased the rendering time to 42 seconds. For an unlimited initial search radius the rendering time was 43 seconds. The computed images looked very similar. The faint illumination of the ceiling is a caustic (created by the bright caustic on the floor) — it becomes a little softer with the increased search radius. For a search radius of 0.01 the caustics became more noisy, and the rendering time was 25 seconds. For other scenes where the caustics are more localized the influence of the maximum search radius on the rendering time can be more dramatic than for the Cornell box.

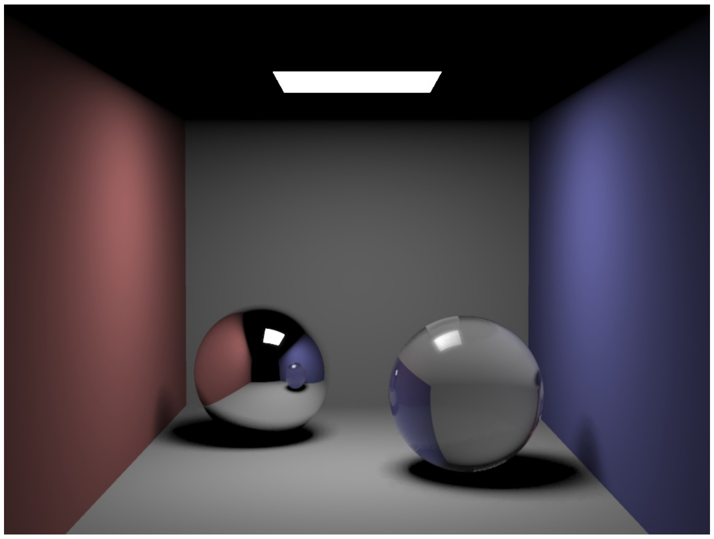


Figure 2.17: Ray traced Cornell box with soft shadows.

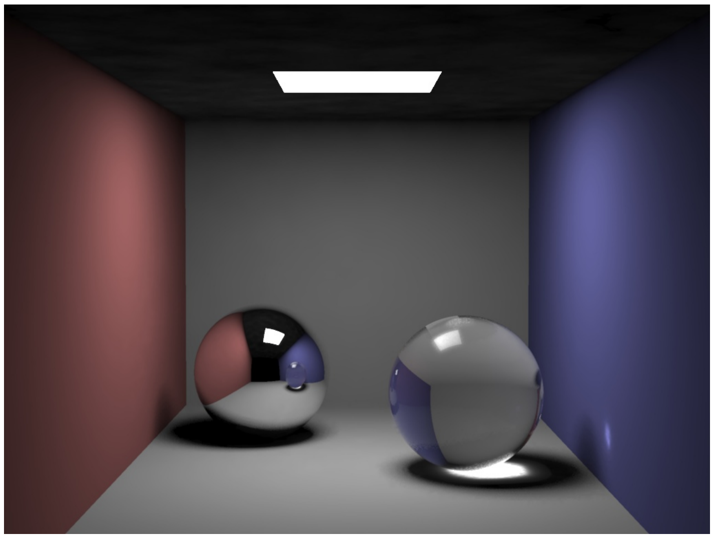


Figure 2.18: Cornell box with caustics.

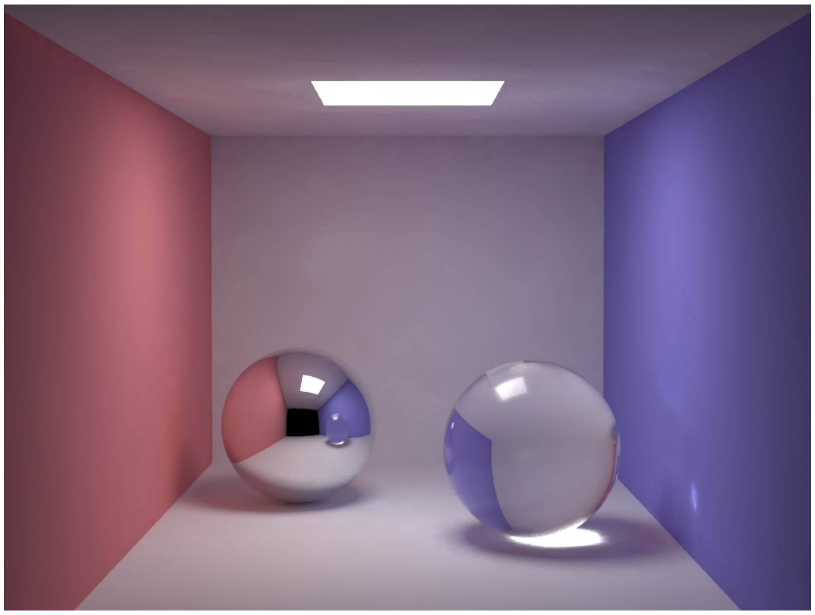


Figure 2.19: Cornell box with global illumination.

Global illumination

In Figure 2.19 global illumination has been computed. The image is much brighter and the ceiling is illuminated. 200000 photons were used in the global photon map and 100 photons in the estimate. The caustic photon map parameters are the same. Photon tracing took 4 seconds. Rendering time was 66 seconds.

The radiance estimate from the global photon map

Finally in Figure 2.20 we have visualized the radiance estimates from the global photon map directly. We have shown images with 100 and 500 photons in the estimate. Notice how the illumination becomes softer and more pleasing with more photons, but also more blurry and with more false color bleeding at the edges. The edge problem can be solved partially by using an ellipsoid or disc to locate the photons (see section 2.3) — with 500 photons in the estimate and the ellipsoid search activated we get the image in Figure 2.21 These images took 30–35 seconds to render. Notice how the quality of the direct visualization gives a reasonable

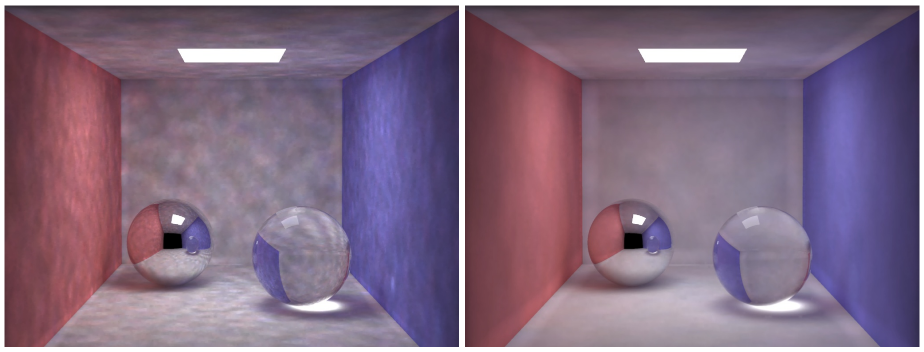


Figure 2.20: Global photon map radiance estimates visualized directly using 100 photons (left) and 500 photons (right) in the radiance estimate.

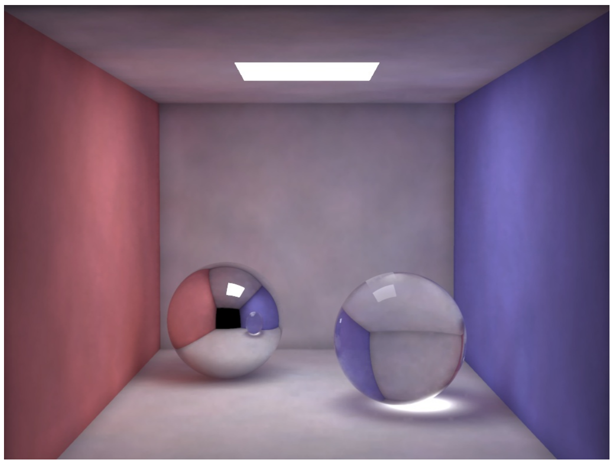


Figure 2.21: Global photon map radiance estimates visualized directly using 500 photons and a disc to locate the photons. Notice the reduced false color bleeding at the edges.

estimate of the overall illumination in the scene. This is the information we benefit from in the full rendering step since we do not have to sample the incoming light recursively.

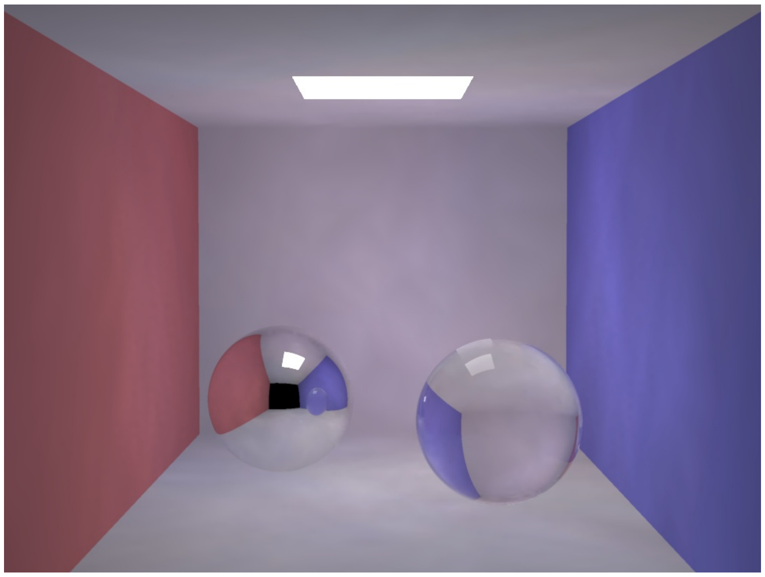


Figure 2.22: Fast visualization of the radiance estimate based on 50 photons and a global photon map with just 200 photons. Rendering time was 4 seconds.

Fast global illumination estimate

For fast visualization of global illumination one can use very few photons in the global photon map. In Figure 2.22 we have visualized the radiance estimate from a global photon map with just 200 photons! We used up to 50 photons in the radiance estimate. The illumination is very blurry and as a consequence the shadows and the caustics are missing, but the overall illumination is approximately correct, and this visualization is representative of the final rendering as shown in Figure 2.19. Photon tracing took 0.03 seconds and the rendering time for the image was 4 seconds. This is almost as fast as the simple ray tracing version, and the main reason is that we only used ray tracing to compute the first intersection and the mirror reflections and transmissions. The global photon map was used to estimate both indirect and direct light.

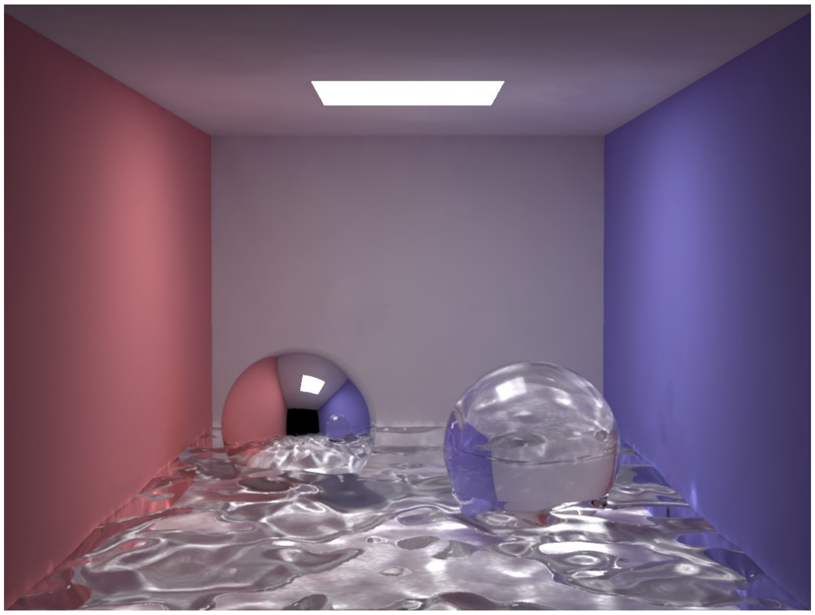


Figure 2.23: Cornell box with water.

2.5.2 Cornell box with water

In the Cornell box in Figure 2.23 we have inserted a displacement-mapped water surface. To render this scene we used 500000 photons in both the caustics and the global photon map, and up to 100 photons in the radiance estimate. We used a higher number of caustic photons due to the water surface which causes the entire floor to be illuminated by the photons in the caustics photon map. Also the number of photons in the global photon map have been increased to account for the more complex indirect illumination in the scene. The water surface is made of 20000 triangles. The rendering time for the image was 11 minutes.

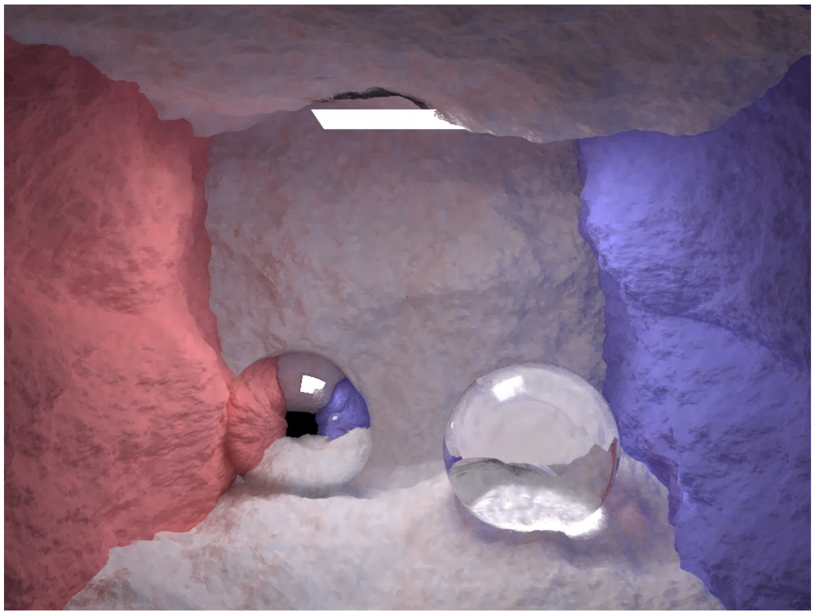


Figure 2.24: Fractal Cornell box.

2.5.3 Fractal Cornell box

An example of a more complex scene is shown in Figure 2.24. The walls have been replaced with displacement mapped surfaces (generated using a fractal midpoint subdivision algorithm) and the model contains a little more than 1.6 million elements. Notice that each wall segment is an instanced copy of the same fractal surface. With photon maps it is easy to take advantage of instancing and the geometry does not have to be explicitly represented. We used 200000 photons in the global photon map and 50000 in the caustics photon map. This is the same number of photons as in the simple Cornell box and our reasoning for choosing the same values are that the complexity of the illumination is more or less the same as in the simple Cornell box. We want to capture the color bleeding from the colored walls and the indirect illumination of the ceiling. All in all we used the same parameters for the photon map as in the simple Cornell box. We only changed the parameters for the acceleration structure to handle the larger amount of geometry. The rendering time for the scene was 14 minutes.

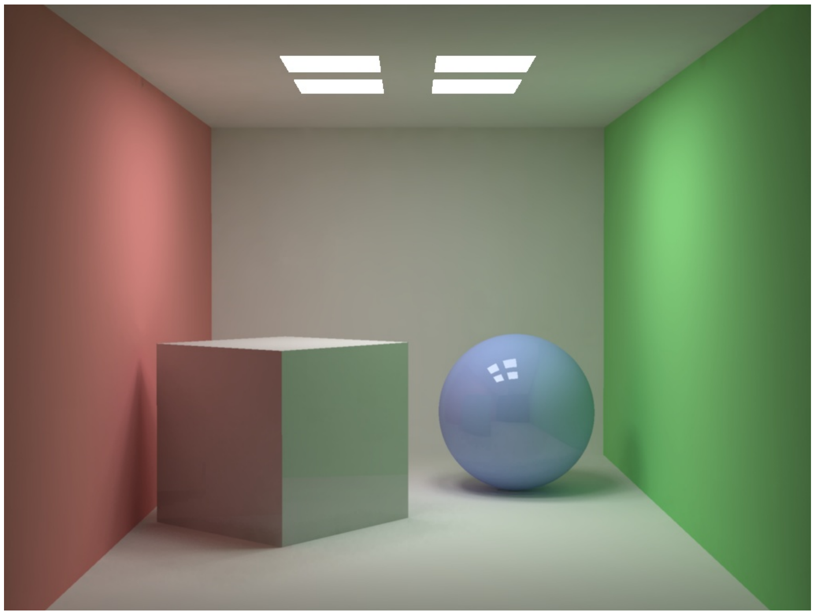


Figure 2.25: Cornell box variation with 4 light sources.

2.5.4 Cornell box with multiple lights

A simple example of a scene with multiple light sources is the variation of the Cornell box scene shown in Figure 2.25. We generated 100000 photons from each light source and the resulting global photon map has 400000 photons. Other than that the rendering parameters were the same as for the other Cornell box with 1 light source. The rendering time for this scene was 90 seconds.

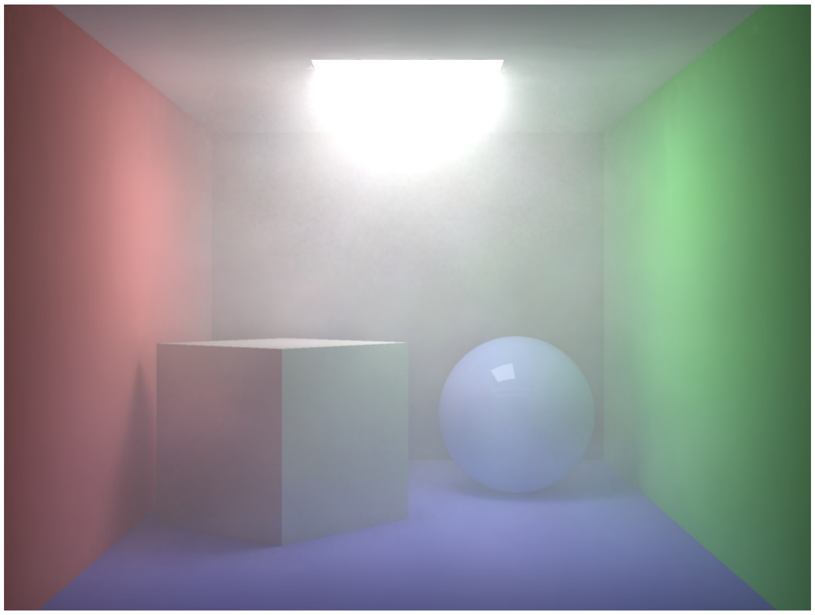


Figure 2.26: Cornell box with a participating medium.

2.5.5 Cornell box with smoke

The Cornell box scene shown in Figure 2.26 is an example of a scene with a uniform participating medium. To simulate this scene we used 100000 photons in the global photon map and 150000 photons in the volume photon map. A simple non-adaptive ray marcher has been implemented so the step size had to be set to a low value which is extra costly. The rendering time for the scene was 44 minutes.

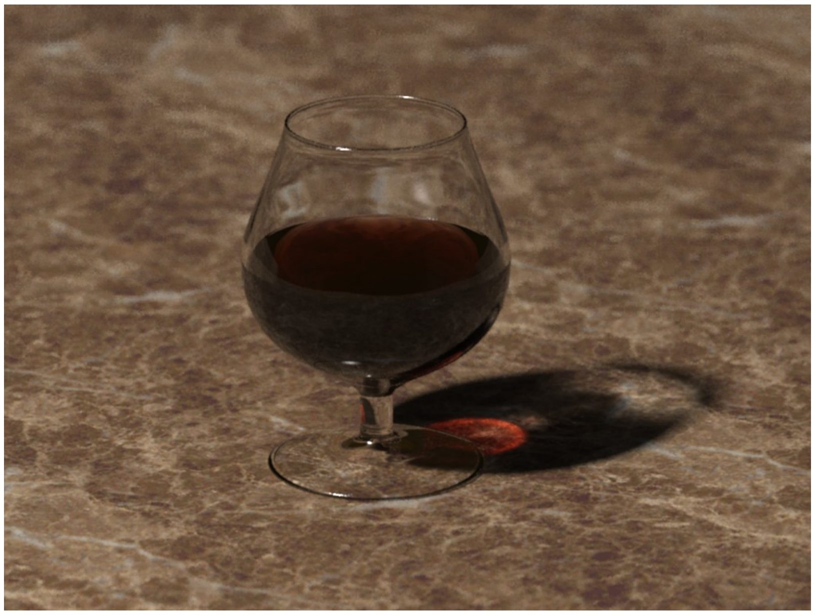


Figure 2.27: A cognac glass with a caustic.

2.5.6 Cognac glass

Figure 2.27 shows an example of a caustic from a cognac glass. The glass is an object of revolution approximated with 12000 triangles. To generate the caustic we used 200000 photons. The radiance estimates for the caustic were computed using up to 40 photons. The rendering time for the image was 8 minutes and 10 seconds — part of this rendering time is due to the ray traced depth of field simulation.

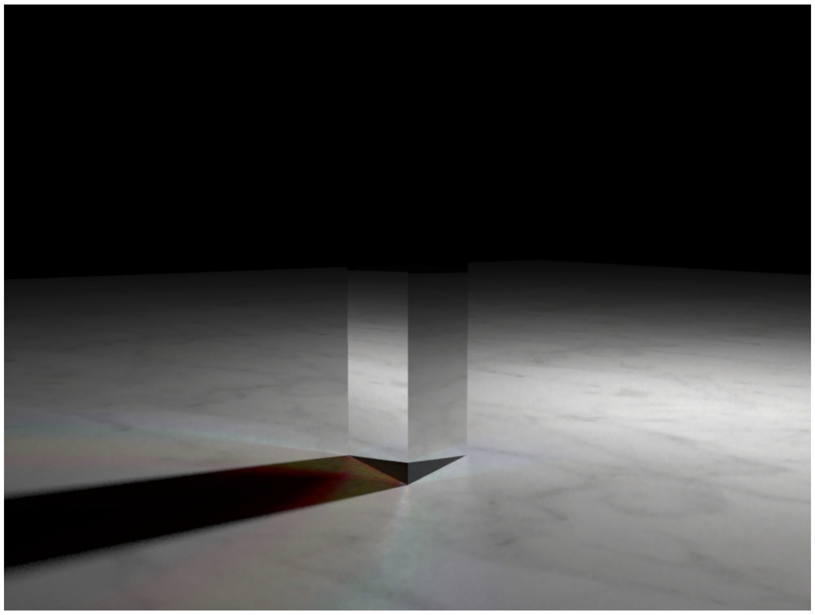


Figure 2.28: Caustics through a prism with dispersion.

2.5.7 Prism with dispersion

The classic example of dispersion with glass prism is shown in Figure 2.28. Even though only three separate wavelengths have been sampled, the color variations in the caustics are smooth. An accurate color representation would require more wavelength samples; such an extension to the photon map is easy to implement. 500000 photons were used in the caustics and 80 photons were used in the radiance estimate. The rendering time for the image was 32 seconds.

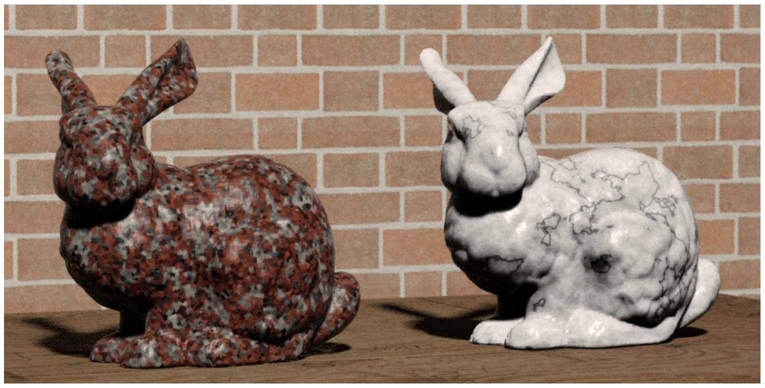


Figure 2.29: Granite bunny next to a marble bunny — both models are rendered using subsurface scattering. The photon map is used to compute multiple scattering inside the stone material.

2.5.8 Subsurface scattering

A recent addition to the photon map is the simulation of subsurface scattering [Jensen99, Dorsey99]. For subsurface scattering we use the photon map to compute the effect of multiple scattering within a given material. This is often very costly to compute and therefore mostly omitted from approaches dealing with subsurface scattering. This is unfortunate since multiple scattering leads to very nice and subtle color bleeding effects inside the material which improves the quality of the rendering.

Figure 2.29 shows a granite bunny next to a marble bunny. Both of these stone models are rendered using subsurface scattering with 100000 photons used to simulate multiple scattering. The rendering time for the picture was 21 minutes. This bunny is the original Stanford bunny and the scene contains 140000 triangles, and it is rendered with full global illumination and depth of field.

Figure 2.30 shows a bust of Diana the Huntress made of translucent marble. For this scene the light source was behind the bust to emphasize the effect of subsurface scattering. Notice the translucency of the hair and the nose region. This image was rendered in 21 minutes using 200000 photons.

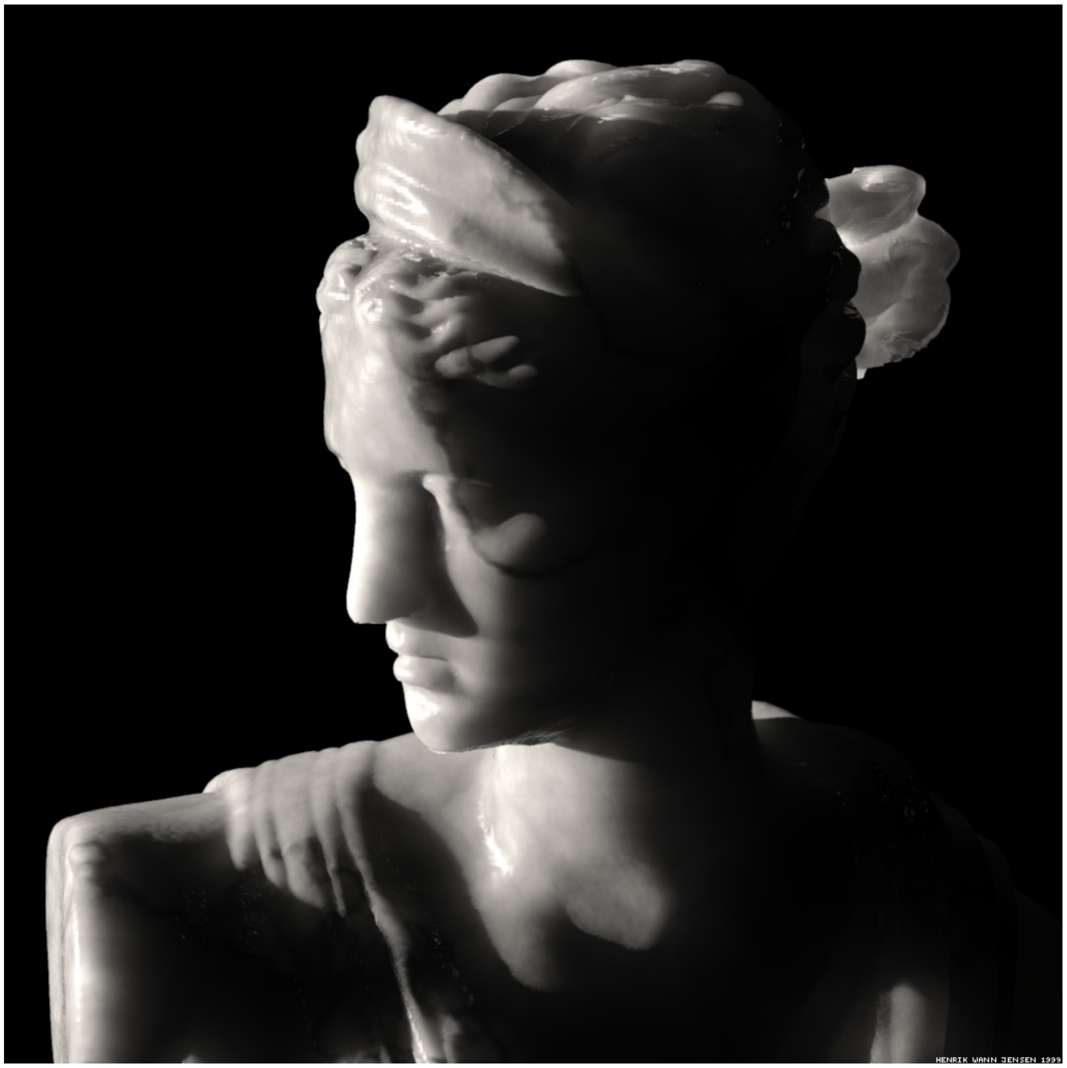


Figure 2.30: Translucent marble bust illuminated from behind

2.6 Where to get programs with photon maps

Photon maps are already available on the Internet for downloading. We have collected the following links as of the writing of these notes.

RenderPark (a photorealistic rendering tool) has photon maps (as well as many

other global illumination algorithms). See
`http://www.cs.kuleuven.ac.be/cwis/research/graphics/RENDERPARK/` for more information.

Nathan Kopp has made a photon map extension to `MegaPov` [MegaPov00] (an extended version of POV ray). Free source code and executable can be found at: `www.nathan.kopp.com/patched.htm`

Blue Moon Rendering Tools (a free Renderman compliant renderer) has photon maps. See `www.bmrt.org` for more information.

The following commercial products supports photon maps for rendering caustics and/or indirect illumination: Lightflow, LightWave, Luminaire, Maya and Twister. In addition the Inspirer rendering system uses a fast photon mapping approach — a mesh based view independent solution is computed based on the local density of photons.

Several other people have implemented photon mapping: a few research packages provides photon maps (these packages might not be publicly available), and some production houses use photon mapping (an example is Kilauea, the in-house renderer used for film production at Square USA).

Chapter 3

Visual Importance

Contact: Frank Suykens
Computer Graphics Research Group
Department of Computer Science
K.U.Leuven
Belgium

E-mail: Frank.Suykens@cs.kuleuven.ac.be

3.1 Introduction

The photon map construction, as described in the previous chapters, stores photons in all parts of the scene that receive light. Since the photon powers in the map are approximately equal, differences in the received illumination on a surface correspond to a change in the density of the photon map. The higher the density —the number of photons per unit area—, the brighter the incoming illumination.

The position of the viewer or camera, used to render an image of the scene, is not taken into account during the photon map construction. For large scenes, many photons may be stored in unimportant regions of the scene, wasting a lot of precious memory.

The *visual importance* determines which parts of the scene are important to the viewer. If a rendering method stores an approximate radiance solution in the scene, the visual importance tells us where this solution should be the most accurate. A peek forward to figure 3.7 (page 84) shows the visual importance for a camera looking towards a single desk in an office (figure 3.10 shows the particular view).

A bright color corresponds to a high visual importance. Clearly, the overall visual importance is low, except near the desk in sight, which is where we want to store most of the photons.

In this chapter we will describe methods to incorporate the visual importance into the construction of photon maps. The visual importance will be used to control the density of the photon maps. Three important issues arise when developing such an importance driven method:

- The first problem is the computation of the visual importance itself: If we want to use importance during photon map construction, it must be known for every position in the scene. As we will see, this can be done by constructing an importance map.

- Second, a relationship between the visual importance and the *required density* of the photon map must be established. This required density criterion can be found through an analysis of the reconstruction error in the photon map with respect to the error in a pixel.

- Finally, the construction of the photon map should adapts its density according to the required density criterion.

In this text we will propose solutions to all three problems and incorporate them in the following three-pass algorithm:

```
1 Compute an importance map by tracing importons
    from the viewer
2 while(photons to trace/store)
    trace photon
    if (density of photon map at this photon position
        is too low)
      store photon
    else
      distribute photon power over nearest neighbours
3 Render image as with the standard photon map method
```

This approach is mainly based on a paper published in the Eurographics Workshop on Rendering 2000 [Suykens00]. Some extensions, improvements and implementation tricks that we have added since then, will also be discussed.

The results of the importance driven method show that less photons need to be stored, compared to a standard photon map construction, while equally accurate results are obtained.

Organization of this chapter

The rest of this chapter is organized as follows: Section 3.2 explains the concept of visual importance and gives an overview of related research in which importance was used. In section 3.3 a required density criterion based on importance will be developed. Section 3.4 shows how to actually compute an importance map, which is similar to a photon map but containing importance information. In section 3.5 caustic and global photon maps are constructed taking into account the required density criterion. Section 3.6 has a few notes on rendering using the importance driven photon maps. Results are shown in 3.7 and conclusions and open issues are discussed in 3.8.

3.2 What is Visual Importance

3.2.1 Intuitive definition

Light sources propagate light (or radiance) through the scene. This light is reflected by surfaces depending on the BRDF and the geometry. All the light that reaches the viewer through a certain pixel determines the intensity of this pixel.

While this is the way images are recorded with real life cameras and our own eyes, numerical simulation of light transport can be reversed by starting from the viewer and propagate *importance* until it reaches the light sources. The transport equations for importance are the same as, or at least very similar to, the transport of radiance, depending on the exact definition of importance. Christensen [Christensen95] notes that importance can be visualized by replacing the camera by a light source that emits light through the screen. The resulting brightness of the surfaces indicate their importance. They indicate how much 'real light' emitted from a surface would contribute to the image.

3.2.2 A mathematical definition

The radiance (or rendering) equation that describes the transport of light in a scene is given by

$$L(x \rightarrow \vec{\omega}) = L_e(x \rightarrow \vec{\omega}) + \int_{\Omega_x} L_i(x \leftarrow \vec{\omega}') f_r(x, \vec{\omega}, \vec{\omega}') cos(\theta') d\omega'$$

with L_e the self-emitted light which is non-zero when x is located on a light source.

A similar transport equation exists for visual importance (See e.g. Pattanaik [Pattanaik95] for more information). Let $W_e(x \rightarrow \vec{\omega})$ be the self-emitted potential[1], which is set to 1 for x equal to the camera position[2] and for all directions through the screen (or through the pixel if we're examining individual pixels) and zero otherwise. The transport equation for exitant potential is:

$$W(x \rightarrow \vec{\omega}) = W_e(x \rightarrow \vec{\omega}) + \int_{\Omega_x} W_i(x \leftarrow \vec{\omega}') f_r(x, \vec{\omega}, \vec{\omega}') cos(\theta') d\omega'$$

However, for computing flux through a pixel or the screen, the *incident* potential is needed, which is given by:

$$W_i(x' \leftarrow \vec{\omega}') = W(x \rightarrow \vec{\omega}')$$

with x' the nearest point on a surface seen from x in direction $\vec{\omega}'$.

If the potential solution is known, the light flux that reaches the camera through the screen (or pixel) can now be expressed in terms of radiance and potential by integrating over the light sources and all possible directions:

$$\Phi_{\text{scr}} = \int_A \int_\Omega W_i(x \leftarrow \vec{\omega}) L_e(x \rightarrow \vec{\omega}) cos\theta d\omega dx$$

Alternatively, the flux could be computed using direct potential (only non-zero for points visible by the camera) and a radiance solution:

$$\Phi_{\text{scr}} = \int_A \int_\Omega W_i^{\text{direct}}(x \leftarrow \vec{\omega}) L(x \rightarrow \vec{\omega}) cos\theta d\omega dx$$

[1]The name importance has been used for several different quantities. We adopt the naming of visual importance quantities as proposed by Pattanaik. *Potential* is the 'visual importance equivalent' of radiance and *importance* corresponds to irradiance (or radiosity). Importance is potential integrated over the hemisphere.

[2]We consider a pinhole camera. For a finite aperture, the potential is 1 for each point x on the aperture area in any viewing direction

Note that if L is approximated using a photon map, this equation corresponds to a direct visualization of the map. In fact, an infinite number of combinations to compute flux from (partial) radiance and potential solutions is possible.

These potential equations will be used further on to derive a criterion for the required density of both the caustic and the global photon map.

3.2.3 Previous work on importance

Importance and adjoint equations were first used in neutron transport simulation (see for example [Kalos86]).

It was introduced into graphics by Smits et. al. [Smits92] who used importance to drive the refinement in a hierarchical radiosity algorithm. Christensen [Christensen95] extended this principle to a non-diffuse finite element method.

Pattanaik and Mudur [Pattanaik95] applied importance (or potential) to particle tracing radiosity. By estimating importance on light sources, more particles could be directed towards relevant regions in the scene. This idea was further explored by Dutré et. al. [Dutre95] who also used the estimated importance to drive the sampling of directions on surfaces during particle tracing.

In '98 Peter and Pietrek [Peter98] presented an importance driven three pass algorithm for photon map construction. In a first pass 'importons' were shot from the camera into the scene to construct an importance map. This map was used to gather importance on light sources. In a second pass photons were emitted into the scene based on the light source importance. When a photon is reflected the importance map is used to guide the photons towards important regions. The third pass consisted of importance driven path tracing. The photon map was only used for direction sampling but not for illumination reconstruction.

While photons are effectively concentrated in relevant parts of the scene, the method results in a photon map that has a mixture of high and low powered photons. This results in an increased variance when one would reconstruct radiance by locating the nearest photons. The radius of influence of a few high powered photons is clearly visible in the reconstruction, because they increase the estimate by a large amount.

Therefore, the number of photons used in the reconstruction must be very high in order to reduce variance. This prevents straight application of the technique to the direct use of a global and caustic map.

A few techniques have been presented that try to control the density of the photon map while keeping the photon powers homogeneous throughout the map. We will only briefly mention them here as they will be discussed in greater detail later on:

- In [Suykens00] we presented a method for density control that redistributes photon powers locally when the density of the map is already high enough. The required density was determined by a special importance map constructed in a first pass. A slightly improved variation of this method will be explained in detail in this chapter.

- Keller and Wald [Keller00] use a discrete acceptance probability to decide if a certain photon is stored or not. The probability is based on the importance, also computed in a first pass.

- Per Christensen uses a similar method to control the density, but combines it with a variation of Peter and Pietrek's method, so that photons are also sent out in important directions from the light sources. Special care is taken to ensure a homogeneous photon map. This method is outlined in another chapter of the course notes.

3.3 Error analysis and required density

In this section, an error analysis will relate the error in a pixel with the reconstruction error of a photon map (3.3.1). This is were importance comes into play. This relation will be used to derive a required density criterion that gives the target density of the photon map for any point in the scene (3.3.2).

3.3.1 Error analysis

To analyze the error in a pixel, we will use the following notation:

- Φ_{pix} : The flux through one pixel in the image

- $\Delta\Phi_{\text{pix}}$: The flux error for the pixel

- $\Gamma(x) = \int_\Omega W_i(x \leftarrow \vec{\omega}) cos\theta \; d\omega$: The (incident) importance in x.

- $\Psi = \int_A \Gamma(x) dA$: The total importance for a certain area. This quantity is equivalent to (light) flux.

The pixel flux, computed in the final ray tracing pass consists of several separate parts: direct illumination, for which the light sources are sampled directly, caustics, using the caustic map, and indirect illumination from the global map.

We are interested in the error in a pixel due to reconstruction errors in both the caustic and the global map. The error analysis is given for the caustic map, but it is similar for the global map.

The caustic map is used for directly visible surfaces and also after one or more specular reflections or refractions. Let the radiance due to caustics be L_c; now the contribution to the pixel flux is defined by all (ES^*L_c) paths, where E is the eye. By defining *caustic potential* W_c as the potential due to these (ES^*) paths, the contribution to the pixel flux is given by

$$\Phi_{\mathrm{pix},c} = \int_A \int_\Omega W_c(x \leftarrow \vec{\omega}) L_c(x \rightarrow \vec{\omega}) cos\theta \; d\omega \; dA$$

with A the total surface area in the scene[3].

However the caustic radiance reconstruction from the photon map is only approximate. The reconstruction error ΔL_c causes an error in the pixel flux $\Delta\Phi_{\mathrm{pix},c}$:

$$\Phi_{\mathrm{pix},c} + \Delta\Phi_{\mathrm{pix},c} = \int_A \int_\Omega W_c(x \leftarrow \vec{\omega})(L_c(x \rightarrow \vec{\omega}) + \Delta L_c(x \rightarrow \vec{\omega})) cos\theta \; d\omega \; dA$$

$$\Delta\Phi_{\mathrm{pix},c} = \int_A \int_\Omega W_c(x \leftarrow \vec{\omega}) \Delta L_c(x \rightarrow \vec{\omega}) cos\theta \; d\omega \; dA$$

In this error analysis we will assume surfaces to be diffuse (L_c independent of direction), but for glossy materials a directional error estimate might be better. The error can be rewritten as follows:

$$\Delta\Phi_{\mathrm{pix},c} = \int_A (\int_\Omega W_c(x \leftarrow \vec{\omega}) cos\theta \; d\omega \;) \Delta L_c(x) \; dA$$

The integral over Ω corresponds to the importance Γ_c in x:

$$\Delta\Phi_{\mathrm{pix},c} = \int_A \Gamma_c(x) \Delta L_c(x) \; dA \tag{3.1}$$

This equation states the total pixel error in terms of the importance and the reconstruction error. Since we are interested in a heuristic for the reconstruction error

[3]Note that this specific integral is not actually computed in the rendering pass, as no explicit representation of W_c is constructed. Instead eye paths are traced and L_c is evaluated for these paths.

itself, we still have to remove the integral. The portion of the pixel error, due to one particular position x is given by the integrand of equation 3.1:

$$\Delta\Phi_{\text{pix},c}(x) = \Gamma_c(x)\Delta L_c(x)$$

Bounding this error by a maximum $\Delta\Phi^{\text{max}}_{\text{pix},c}(x)$ gives a bound for the reconstruction error:

$$\Delta L_c(x) \leq \Delta\Phi^{\text{max}}_{\text{pix},c}(x) \;/\; \Gamma_c(x)$$

If each position x is allowed to contribute an equal amount to the pixel error, the reconstruction error can is bounded by

$$\Delta L_c(x) \leq \frac{\Delta\Phi^{\text{max}}_{\text{pix},c}}{A_{\text{tot}}} \;/\; \Gamma_c(x)$$

with $\Delta\Phi^{\text{max}}_{\text{pix},c} = \int_{A_{\text{tot}}} \Delta\Phi^{\text{max}}_{\text{pix},c}(x)\; dA$, and A_{tot} the total area in the scene.

From this analysis we learn that the ratio between the pixel flux error and the reconstruction error is given by the importance. If we can compute Γ_c throughout the scene and if we can derive a useful relationship between the reconstruction error and the photon map density, then we can use that to tune the storage of photons in the caustic map.

A similar error analysis can be performed for the global map. The difference lies in how the reconstructed radiance L_g from the global map is used in the final pass, namely after a diffuse or moderately glossy bounce. The contributing paths during rendering are $(E(S|D_{tc})^*D(S^*)L_g)$, with D_{tc} meaning a diffuse or glossy bounce that reaches a surface under the distance threshold (too close-by). This results in a different importance Γ_g over the surfaces and requires a separate global importance map.

3.3.2 Required Density

Given an estimated bound on the reconstruction error, we still need to relate it to the density of the photon map. This is quite a difficult problem. The reconstruction is in fact a form of nearest neighbor density estimation, but a detailed error analysis is difficult and has only been investigated for specific cases [Silverman86].

In [Suykens00] we simply assumed the error to be inversely proportional to the density D of the photon map. For a given error ΔL, the required density of a photon map must at least be:

$$D_r \geq \frac{C}{\Delta L}$$

The constant C was determined by hand, but was relatively independent of the scene.

Currently we use a more intuitive approach to determine the error/density proportionality:

- The importance on a position x_{scr} on the screen is $\Gamma_{\text{scr}}(x_{\text{scr}}) = 1$. This follows from the definition of W_e and the fact that we use a pinhole camera (only the direction from the camera position to x_{scr} counts).

- A target pixel density is chosen as: M_r/A_{pix}. This density corresponds to a unit importance. The user can choose the number of target photons M_r per pixel area.

- Given the importance on a position x in the scene, the required density is set to:
$$D_r(x) = \Gamma(x) \times M_r/A_{\text{pix}} \tag{3.2}$$

Note that:

- This procedure still assumes an inverse proportionality between error and density. In fact, it just defines the factors C and $\Delta\Phi^{\text{max}}_{\text{pix},c}$ in the error analysis.

- For the global map, we used values for M_r around 1 or 2, meaning that we want 1 or 2 photons per pixel in the photon map.

- The caustic photon map is visualized directly and needs to be more accurate. We used $M_r = 25$ in our examples.

- The accuracy of the reconstruction is also dependent on the number of nearest photons used. One might make M_r dependent on this number also, but it is still an open problem how this should be done.

Two things remain to be worked out before we can make use of the required density criterion (eq. 3.2):

- The importance Γ must be known for every position in the scene. This can be done by initially computing importance maps (see 3.4).

- During photon map construction, the required density should be taken into account when storing (and possibly when shooting) photons (see 3.5).

3.4 Importance maps

In this section we will show how an importance map can be constructed and how Γ can be estimated from this map.

3.4.1 Building importance maps

Tracing importons

The construction of an importance map is very similar to the construction of a normal photon map, but *importons* —instead of photons— are emitted from the camera into the scene.

An importon has the same properties as a photon: a position, a direction and the importance, which is equivalent to the power of a photon. An emitted importon is created by sampling a uniform point on a pixel or on the screen. The starting position of the importon is the camera position itself; the direction is given by the normalized vector from the camera to the sampled image point.

The importance of a single importon is derived from the total emitted importance and the number of emitted importons. The total emitted importance for one pixel is $\Psi_{\text{pix}} = A_{\text{pix}}$. For the whole screen this is $\Psi_{\text{scr}} = A_{\text{scr}}$.

If N importons are shot into the scene, the importance of a single importon i is given by $\Psi_i = \Psi/N$. Scattering of importons (and the corresponding power adjustment) is exactly the same as the scattering of photons. Note that due to scattering an emitted importon may result in several stored importons, just as with photons.

Importons only need to be stored on glossy or diffuse surfaces, since photons will also be stored on these surfaces only. Two importance maps are needed: one for the caustic map and one for the global map. Depending on the history of the importon, it is added to the caustic or global importance map. The importance paths must mimic exactly the paths traced in the final pass:

- If no diffuse or glossy '$(D|G)$' bounce was made before (or if the distance between the bounce and the subsequent hit point is too small), then the global map will not be used for radiance reconstruction in the final pass. These importons must be stored in the caustic importon map. For example importons that hit surfaces directly from the eye, corresponding to direct importance, will be stored in this map.

- Once an acceptable $(D|G)$ bounce is made, the importon is stored in the global importance map, because in the final pass the same eye path would be using the global photon map. Note that further scattering of the importon should only include specular bounces, because further $(D|G)$ bounces will never be generated in the final rendering pass.

During importance map construction, the importons are stored in an array. Afterwards, this array is transformed into a balanced kd-tree for faster access during photon map construction.

Importance reconstruction

To evaluate the required density, we need to reconstruct importance from the importance map. The reconstruction of importance corresponds to the reconstruction of *irradiance* from a photon map. The M nearest importons are located and the importance is estimated as follows:

$$\Gamma(x) \approx \frac{\sum_{i=1}^{M} \Psi_i}{\pi r_M^2(x)} \tag{3.3}$$

A few implementation details:

- The importance will determine the local density of the photon map. A smoothly varying density is beneficial for the photon map reconstruction and for the redistribution that will be used for selective storage. This suggests using a large M for a smooth changing importance solution.

- The importance must be evaluated whenever a photon hits a diffuse or glossy surface. To accelerate these lookups we precompute importance at the importon locations as proposed by Christensen [Christensen99]. However the precomputation is not necessary on *all* importon locations (Christensen suggests 1/4th of the photons/importons). During photon tracing only the precomputed importance is used, and the other importons can be discarded to save memory.

3.4.2 Pixel versus screen importance

Until now we have always considered the pixel flux error. When an image is computed, we are interested in the error over the whole image. An error metric for images is needed and can be specified in different ways:

- **Maximum pixel error (∞-norm):** This error metric bounds the error on each pixel separately. This ensures that *all* pixels are accurate. This error metric, however, would require a separate importance map for each pixel. To determine the required density at a certain point in the scene, the *maximum* of the reconstructed importances should be used. This ensures that the photon map density will be high enough for the pixel that is most influenced by that point.

  Computing an importance map for each pixel, however, is totally infeasible as it would require emitting quite a number of importons *per pixel*, consuming too much time and memory. Therefore, it is common practice to compute the screen importance and use the pixel-sum error metric (see next item). Recently we have experimented with path differentials to estimate single pixel footprints over a path. This does allow for a maximum pixel importance-like quantity. More information is given in 3.4.3.

- **Total screen error (1-norm):** The total error through the screen could also be bounded. This corresponds to bounding the sum of the individual pixel errors, $\Delta\Phi_{\mathrm{pix}}$, leading to the screen error, $\Delta\Phi_{\mathrm{scr}}$. A disadvantage of this method is that individual pixels could still have a high and visible error.

  Using the total screen error, a single importance map for the whole screen at once is sufficient. This leads to a single caustic importance map and a single global importance map.

To compare the two error metrics, we have computed the pixel importance Γ_g for a few pixels for a certain scene, which is shown in figure 3.1 (b). This is compared with an equivalent importance solution for the whole screen at once, shown in figure 3.1 (a). Pixels that are projected near a corner show a high (pixel) importance on the abutting walls in (b). This is logical as the wall occupies a large part of the hemisphere around the projected pixel. For many other pixels, however, this wall is unimportant. Therefore, the screen importance underestimates the importance for such pixels (a).

In practice, most existing importance based algorithms use the screen error metric and compute a single importance solution for the whole image.

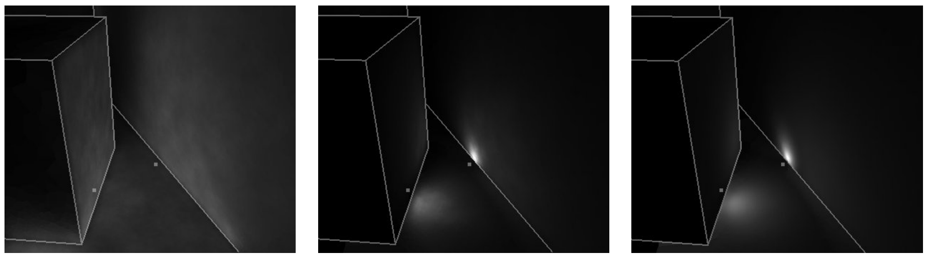


Figure 3.1: Comparison of pixel and screen importance: (a) shows the indirect importance for the complete screen. (b) shows the importance for 2 indicated pixels. The screen importance underestimates the importance on nearby walls. (c) shows a path based importance computed from path derivatives. A close match with (b) is obtained.

3.4.3 Path differentials

Since bounding each pixel to a maximum error ensures accuracy over the whole image, we have been looking for ways to compute this without requiring an importance map per pixel.

In [Suykens00] we computed a point based importance. Instead of enlarging the pixel to the full screen, it was diminished to an infinitely small area. Importance was computed for individual points on the image. While this gave quite good results for paths of length 2 (EDL_g), it was hard to generalize to longer paths that include specular bounces.

In [Suykens01] we presented path differentials. Intuitively this is a way of tracking the footprint, the region of influence of a pixel during the tracing of a path. It generalizes ray differentials, proposed by Igehy [Igehy99], so that arbitrary BRDFs can be used.

Since we believe this is a promising approach to compute importance, we will formulate the main idea and show how we applied it to construct a per pixel importance map. For more information, please consult [Suykens01, Suykens01TR].

Path sampling and path footprint

When tracing an importon, paths are constructed starting from the eye. Each vertex $\mathbf{V}$ of this path is a hit point for the importon. These paths are constructed by random sampling.

For a certain vertex $\mathbf{V}$ in the path, one can say that:

$$\mathbf{V} = g(t_1, t_2, \ldots, t_k)$$

with g the path generating function, and k the number of (unit) random numbers t_k that were used to generate the path to V.

If we apply a small perturbation ϵ_j to a variable t_j, then V moves over some distance:

$$\mathbf{V} + \delta\mathbf{V}_j = g(t_1, \ldots, t_j + \epsilon_j, \ldots, t_k)$$

This change can be approximated by a first order Taylor expansion:

$$\delta\mathbf{V}_j \approx \frac{\partial g(t_1, \ldots, t_j, \ldots, t_k)}{\partial t_j}\epsilon_j$$

The magnitude of the partial derivative determines how sensitive $\mathbf{V}$ is for changes in t_j. Simultaneous perturbation of several variables corresponds to a $\delta\mathbf{V} = \sum_j \delta\mathbf{V}_j$.

If we consider all perturbations $\epsilon_j \in [-\Delta t_j/2, \Delta t_j/2]$ then the set of perturbed vertices $\delta\mathbf{V}_j$ forms a line segment defined by a vector centered around V:

$$\Delta\mathbf{V}_j = \frac{\partial g(t_1, \ldots, t_k)}{\partial t_j}\Delta t_j$$

These are the *differential vectors*.

Given a perturbation interval Δt_j for each variable and allowing simultaneous perturbation of the variables, the set of all possible perturbed vertices $\mathbf{V} + \delta\mathbf{V}$ forms an area. We call this area *the footprint* of the path for vertex $\mathbf{V}$.

Computation of the partial derivatives and practical computation of the footprint can be found in [Suykens01].

As an example of the path differentials consider an eye path through a pixel that hits a surface x and is reflected diffusely towards x':

- Footprint in x: Two random variables determine the point in the pixel. The differential vectors determine the movement of x with respect to the pixel. The footprint corresponds to the projected area of the pixel.

- Footprint in x': The diffuse reflection introduces two new random variables that determine the outgoing direction. 4 partial derivatives give the rate of change of x' with respect to the sampled direction and image position. If x' is far away from x the footprint will be larger, as a small perturbation of the reflected direction is magnified by the large distance.

Footprint and importance

For importance estimation, we used the path differentials exactly as described in [Suykens01]. Delta's were chosen to be 1.

Since derivatives are used, the footprint can be estimated for any single path. This footprint is inversely proportional to the density of similar paths that arrive in the neighborhood of x and can be used to estimate the pixel importance in x:

$$\Gamma_{\text{pix}}(x) = 1/A_{\text{footprint}}(x)$$

The result for the few pixel importances using the footprint estimate is shown in figure 3.1 (c) . Note the close match between real pixel importance and footprint estimates.

Practical use of path differentials

To compute an importance solution based on footprints for all pixels in the image, we also store $1/A_{\text{footprint}}(x)$ with each importon. Since a single importon gives an importance estimate, we do not need to estimate the density of the importance as in equation 3.4.1. Instead we should take the maximum of pixel importances in x. This can be done by locating the nearest importons and taking the maximum of the importance. However, we found that this can cause rather abrupt changes in the required density. In practice we average several nearest pixel importances.

Whether pixel or screen importance is better than the other is still an open question. To get the best of both, we combine the pixel importance estimate using path differentials with the normal screen importance estimation. We store both footprint and screen importance information in the importon. Importance is estimated using both methods and a weighted average is taken.

The weights are chosen so that the average screen importance matches the average pixel importance:

$$\Gamma(x) = 0.5(\Gamma_{\text{scr}}(x) + \Gamma_{\text{pix}}(x)\frac{\Gamma_{\text{scr}}^{(avg)}}{\Gamma_{\text{pix}}^{(avg)}})$$

The averages are computed when precomputing importances at the importon locations.

While this is a rather arbitrary combination, it provides a fairly robust estimate for the required density. Nevertheless, using solely the path differentials or the screen importance, still gives satisfactory results.

3.5 Photon Map Construction using visual importance

After completing the first pass that builds the two importance maps, the second pass will trace photons and build the photon maps.

Several strategies have been proposed for importance driven photon map construction: Selective storage using a storage probability (3.5.1), selective storage by photon power redistribution (3.5.2) and importance driven photon shooting (3.5.3). We will focus on the redistribution.

3.5.1 Storage probability

Keller and Wald [Keller00] and Christensen both use the following technique:

When a photon hits a diffuse or glossy surface in x, it is not blindly stored but a discrete storage probability P_Γ is determined based on the (screen) importance in x. A low importance results in a low probability P_Γ and few photons will be stored in this area. If it is stored, the power of the photon is multiplied by $1/P_\Gamma$ to ensure energy conservation. Since P_Γ must be smaller than 1, importance values are scaled down by a factor (e.g. the maximum importance) and cut off at 1.

This method will be discussed in more detail by Per Christensen in another part of the course notes.

3.5.2 Density Control by redistribution

Using a storage probability, as mentioned above, does not limit the density of the photon map. A bright region that also has high importance will practically store all the arriving photons ($P_\Gamma \approx 1$), even if the density is already adequate. On the other hand for a darker but moderately important region, photons may be thrown away even though not enough photons have arrived.

It would be better to stop storing photons if the required density has been reached. This can be done by distributing the power of photons that will not be stored.

A method for selective storage and redistribution

Suppose we have traced a new photon k to a position x on a surface. An importance based required density $D_r(x)$ provides a target density for an accurate reconstruction in x.

To determine whether or not the photon is stored, we estimate the current photon map density $D_{\text{cur}}(x)$. This can be done by locating the M nearest photons and evaluating:

$$D_{\text{cur}}(x) = \frac{M}{\pi r_M^2(x)}$$

If the current density is lower than the required density, the photon is stored. No power adjustment is made for stored photons. Therefore, when a photon is not stored, its power must be somehow accounted for in order to keep the global flux in the map consistent with the flux emitted from the light sources.

This can be done by distributing the photon power over its nearest neighbors. The redistribution can be justified as follows:

Suppose we had stored the photon k, then the reconstruction of radiance using M+1 photons at the photon position x would be:

$$\tilde{L}(x, \vec{\omega}) = \frac{\sum_{i=1}^{M} f_r(x, \vec{\omega}_i', \vec{\omega})\Phi_i + f_r(x, \vec{\omega}_k', \vec{\omega})\Phi_k}{\pi r_M^2(x)}$$

Note that $r_M(x)$ without the storage of photon k, is equal to $r_{M+1}(x)$ including photon k, because the photon is located in x.

Since the photon is not stored, the power of the other photons (i) should be adjusted without changing the reconstruction in x:

$$\tilde{L}(x, \vec{\omega}) = \frac{\sum_{i=1}^{M} f_r(x, \vec{\omega}_i', \vec{\omega})(\Phi_i + \Delta\Phi_{k,i})}{\pi r_M^2(x)}$$

Different choices for $\Delta\Phi_{k,i}$ can be made depending on f_r and the distance of x to photon i:

- $f_r(x, \vec{\omega}_i', \vec{\omega})$: To get an equal reconstruction $\tilde{L}(x, \vec{\omega})$ in x, $\Delta\Phi_{k,i}$ should be zero when f_r is zero, because these photons do not contribute to $\tilde{L}$. Currently the angle between $\vec{\omega}_i'$ and the normal $\vec{n}_x$ in x is used to determine whether $\Delta\Phi_{k,i}$ should be zero (i.e. for a non-transparent material, $\Delta\Phi_{k,i} = 0$ when $cos(\vec{\omega}_i', \vec{n}_x) \leq 0$).

- Another approach could be to choose a larger delta for photons with a direction similar to the distributed photon. This might be better for non-diffuse BRDFs but at the cost of a less smoothly varying photon power.

- Distance to x: The distribution of the photon power can be seen as applying a low-pass filter (or as splatting). The dependence of $\Delta\Phi_{k,i}$ on the distance to x determines the filter kernel. We distribute the power equally over the affected photons to keep the photon powers homogeneous which, as said, is beneficial for the reconstruction.

To summarize, we choose:

$$\forall i, cos(\vec{\omega}'_i, \vec{n}_x) > 0 : \Delta\Phi_{k,i} = \Phi_k / M'$$

with M' the number of photons that have a cosine > 0.

Of course, the radiance estimate at locations different from x will give a slightly modified result. But because the current density is already sufficient—it was tested first—, this averaging can be expected not to introduce artifacts (if the required density does not change too abruptly).

Wherever D_r does change abruptly, a region of high and low density will meet. In the lower density region, photons will be distributed, possibly into the high density region. This problem is similar to the blurring of caustic edges, where a high density region also meets a very low density region. Redistribution to the nearest neighbors can introduce some visible bias into the reconstruction. Therefore it is advisable to keep the number of photons used for redistribution small (we use 20 photons) and to prevent abrupt changes in the importance and the required density. As said before, we use a large number of importons in the importance reconstruction to prevent such abrupt changes.

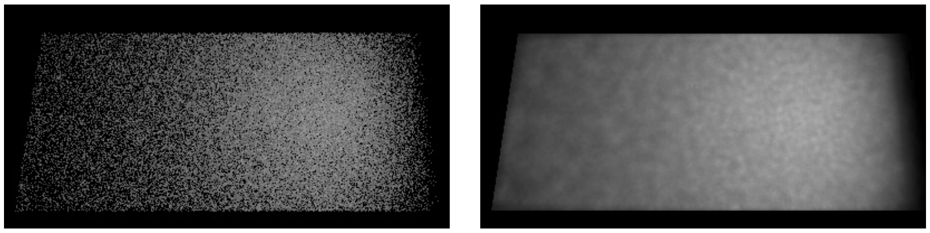
Note that the selective storage requires the estimation of the photon map density during its construction. We store the photons directly in an unbalanced kd-tree to make the lookup efficient. Before the final rendering pass this tree is balanced for even faster access.

Redistribution provides a method to control the density of photon maps based on the required density D_r. This density can be chosen based on importance, as described above, but other choices are possible too. D_r can be chosen arbitrarily, depending on the application, providing a flexible density control framework.

Redistribution results

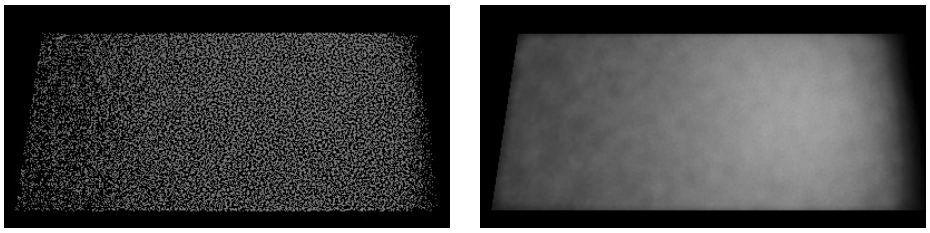
To illustrate the effect of redistribution, we constructed a simple scene containing a single grey plane lit by a light source positioned to the side of the plane. A global photon map is computed and visualized directly.

In figure 3.2 no density control is used; all photons are stored. On the right side the radiance reconstruction is shown (b), while the left side shows the photon hits in the global map (a).



(a) (b)

Figure 3.2: A simple plane lit by a light source on the side. No density control is used; all photons are stored. (a) shows the hits, (b) shows the resulting radiance in the photon map (80 nearest photons)



(a) (b)

Figure 3.3: The same plane using a constant required density ($D_r = 1000$). Photons are distributed over 20 nearest neighbors if the current photon map density at the hit point is sufficient. (a) shows the hits, (b) shows the resulting radiance in the photon map (80 nearest photons)

For figure 3.3 a constant required density was used. The radiance reconstruction is similar, while the hit density is clearly different. In the bright region photons were distributed to their neighbors. The lower density is compensated by a higher photon power. Note that the variance in the bright region is lower because the distribution of photons is a kind of low pass filter.

More results, using an importance based required density, are given in 3.7.

3.5.3 Importance driven shooting

The two previous techniques (section 3.5.1 and 3.5.2) for importance driven photon map construction do not change the way photons are emitted; only the storage is controlled.

It would be interesting to also *emit* more photons towards important regions. Straight application of Peter and Pietrek's method [Peter98] is possible, but, as said, it results in an undesirable mixture of photon powers.

Further in the course notes, Per Christensen describes a new method that allows a combination of the storage control and importance driven shooting of photons, while keeping photon power homogeneous. It would be very interesting to combine his technique with the photon redistribution.

3.6 Rendering

The final pass in photon map rendering computes the actual image. This pass does not change when using importance based photon maps.

However one could use the importance or required density to derive some heuristic for choosing how many near photons must be used in the estimate. A high importance corresponds to a high density, and more near photons could be used.

3.7 Results

Photon maps and their importance driven construction were implemented in RenderPark [RPK]. All results shown here are generated using RenderPark on a single 1GHz AMD Athlon with 256MB.

3.7.1 Caustic map

To test visual importance and density control for the caustic map, we used a room with a large glass egg in it. It is lit by two light sources.

Importance maps were computed for the view shown in figure 3.6. The caustic importance map contained 100000 importons. Importance was precomputed for each importon location using the 200 nearest importons. Tracing the importons took around 3 seconds, precomputation and kd-tree balancing took 15 seconds.

The screen importance and footprint importance are computed simultaneously and both are stored in the importon.

For the required density, M_r, the accuracy scale factor, was set to 25. This is higher than for the global maps, because the caustic map is visualized directly and a higher density is needed.

Figure 3.4 shows the required density as seen from an alternate viewpoint. Figure (a) shows the screen importance estimate and (b) the footprint estimate.

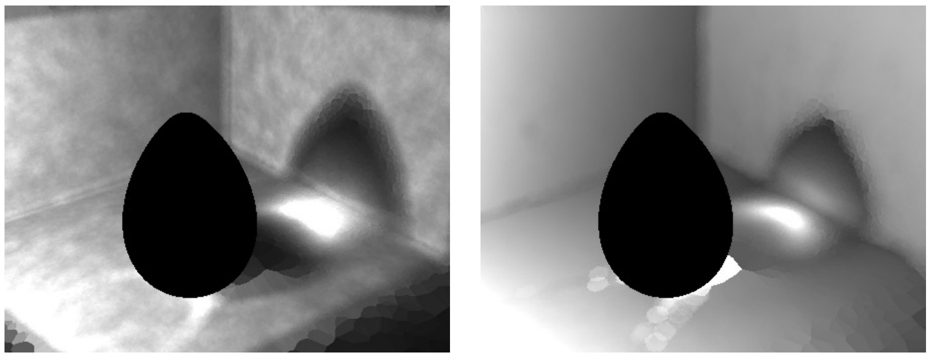

(a) (b)

Figure 3.4: Required density of the caustic map (seen from an alternate camera). Note the 'importance caustic' that indicates the region that is magnified by the egg. (a) shows screen importance, (b) shows footprint based importance.

Some interesting observations can be made:

- Importance is focused through the egg on the ground. This results in a sort of importance caustic. This is the part of the scene that is magnified most by the egg as can be seen in the final rendering in figure 3.6.
- The screen importance (a) shows a much higher variance. This is because of density variations in the importance map, that show up in the importance estimate. When using the footprint, each importon carries an importance estimate and the density is not used.
- Some importons had very small footprints that are caused by eye paths that are extremely focused by the glass egg. Although few of these paths occur, the small footprints cause 'spike circles' in the required density estimate. We removed the worst spikes by discarding importons that have a footprint 100 times smaller than the average footprint (i.e. required 100 times more density than the average). The average footprint is computed while storing the importons. Still some circles show up in the footprint estimate.

To compute the caustic photon map, we used the average of screen and footprint importance. Figure 3.5 shows the resulting density of the caustic map. Around 200000 photons are stored in the map. Without density control 400000 would have been stored at this point in time. If more photons are shot, the difference grows because more regions reach the required density, and fewer and fewer photons are

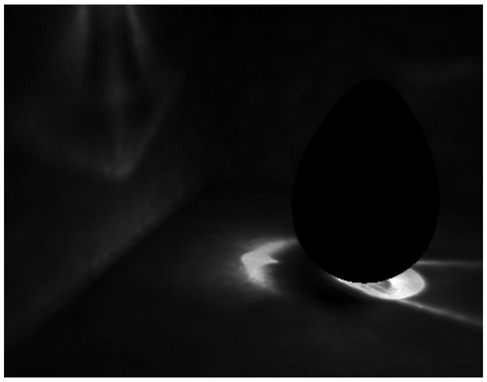


Figure 3.5: Resulting density of the caustic map. Bright parts of the caustic are relatively dark because the required density is reached and no extra photons are stored. (200000 stored photons in total)

stored in the caustic map. The gain is largest in the bright parts of the caustic, where a large fraction of the photons arrive and the required density is reached early on.

Unstored photons were distributed over 20 neighbors. These neighbors are also used to determine the current density.

Tracing and storing the photons took about 140 seconds. This is slower than tracing the 400000 photons without density control (about 100 seconds), because of a lookup in the importance map (quite fast due to precomputation) and a lookup in the current photon map to determine the current density (slower as the tree is not yet balanced). Compared to the final rendering time this difference is negligible.

Figure 3.6 shows the final rendering of the scene. Full global illumination is computed using a global photon map. The final rendering took around 80 minutes.

Note that, in the end, no photons will be stored anymore when all lit regions have reached the required density. This is an interesting advantage of the redistribution: We can just keep shooting photons until the density in difficult parts of a caustic is sufficient without worrying about the many photons that would be stored in the bright, ’easy’ parts. In practice, we end the photon tracing when just a tiny fraction of photons are added as new to the map.

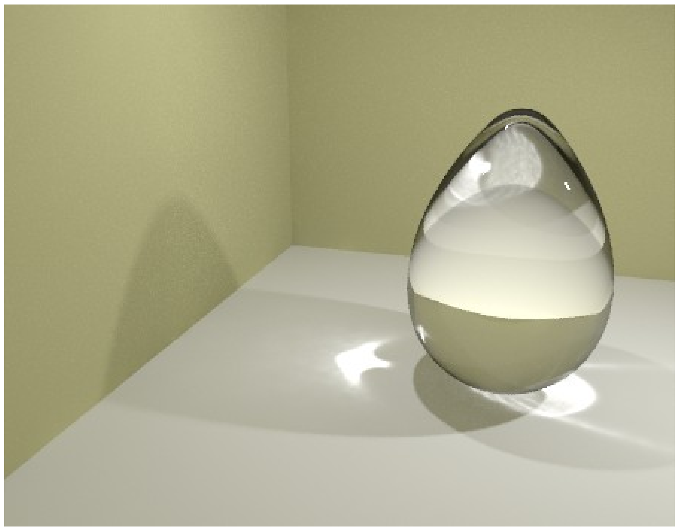


Figure 3.6: Final rendering of the egg scene.

3.7.2 Global map

To show benefits of importance for the global photon map we use an office scene with several desks and light sources. The camera is looking towards a single desk with a a glossy pad and some photo stands. The view can be seen in the final rendered figure 3.10.

Figure 3.7 shows the required density. The average of screen and footprint importance was used with an accuracy scale factor $M_r = 2$. The glossy pad causes a high required density on the photo stands and part of the wall, because the global map is visualized directly after the glossy reflection. 80000 importons were stored and importance was precomputed for the importon locations using the 200 nearest importons.

Figure 3.8 (a) shows the resulting density of the global map. 57.000 photons are stored, resulting in a good match with the required density. The 20 nearest photons were used for redistribution. Without density control, 400000 photons would have been stored (density shown in figure 3.8 (b)).

Note that some parts did not yet reach the required density. The final rendering, however, shows no artefacts. This indicates that the accuracy could be set even lower.

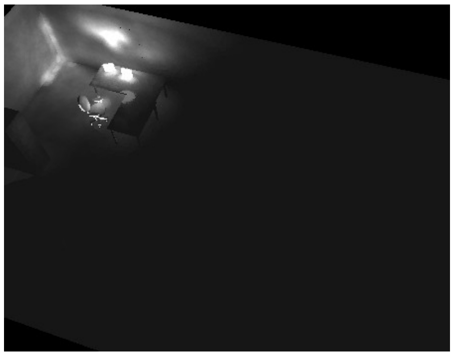


Figure 3.7: Required density of the global map. The glossy pad causes a high required density on the photo stands and part of the wall. Abutting surfaces also require a higher density.

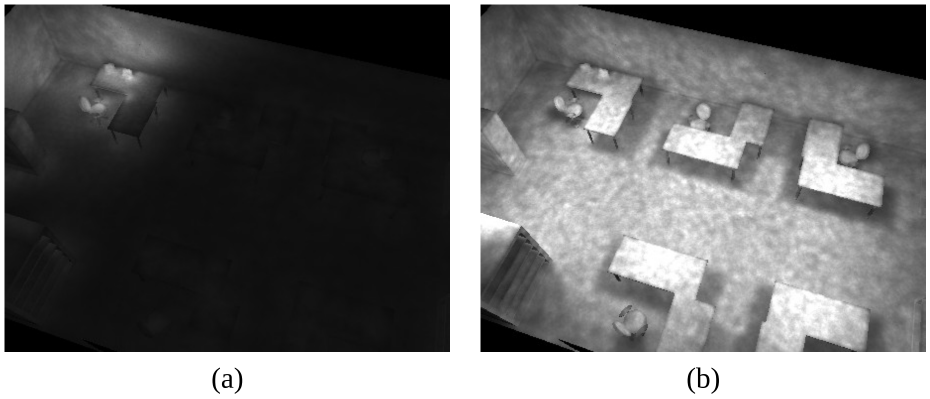


Figure 3.8: Overview of the resulting density of the global photon map. (a) uses density control, (b) does not.

Figures 3.9 (a) and (b) show a direct visualization of the global map respectively with and without density control. Diffuse irradiance was precomputed on the photon positions [Christensen99]. The 80 nearest photons were used in the radiance estimate.

While the overall illumination is similar, a courser solution can be seen in unimportant parts for the density controlled image. Note that these parts have a low variance due to the redistribution, but the bias or blurring is quite high (e.g. bad shadow boundaries under the tables).

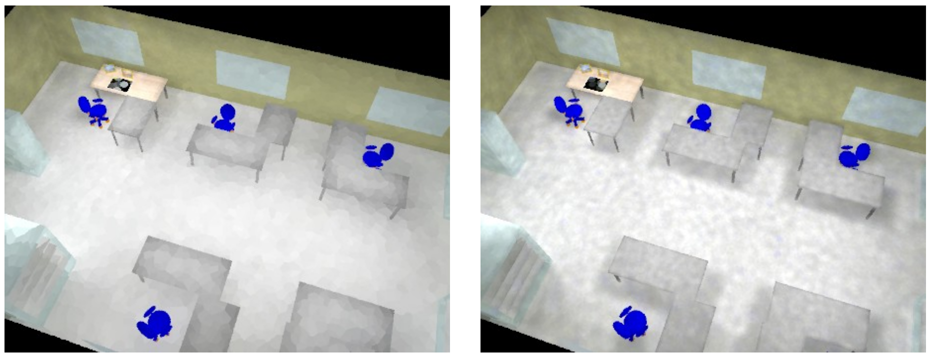

(a) (b)

Figure 3.9: Direct visualization of the global photon map. This overview shows a courser solution in unimportant regions when using density control (a, 57000 photons). Overall illumination is of course the same as without density control (b, 400000 photons).

Figure 3.10 shows the final rendering using the density controlled map. The final rendering using the 400000 photons is not shown as no visible differences could be seen. The rendering took 90 minutes, which is much more than the time needed for the importance and photon map construction.

Note that a fairly open scene was used and that even in such scenes much can be gained by using visual importance. Typical importance examples, such as a maze or a viewer standing in one room of a large building would give even better results.

3.8 Conclusions

In this chapter, we explored how visual importance can be used to build better photon maps. A new, initial pass computes importance maps that are used to control the density of the photon maps. Results show that the use of visual importance effectively reduces the number of stored photons in the photon map. The gain depends on the complexity of the scene and on what is visible through the camera, but it helps even for open, relatively simple scenes.

Many things, however, can and should be further investigated. We mention a few open issues that could be interesting for future research:

- Currently the error due to the caustic or global map reconstruction is estimated independently of any other illumination. Strong direct light for exam-

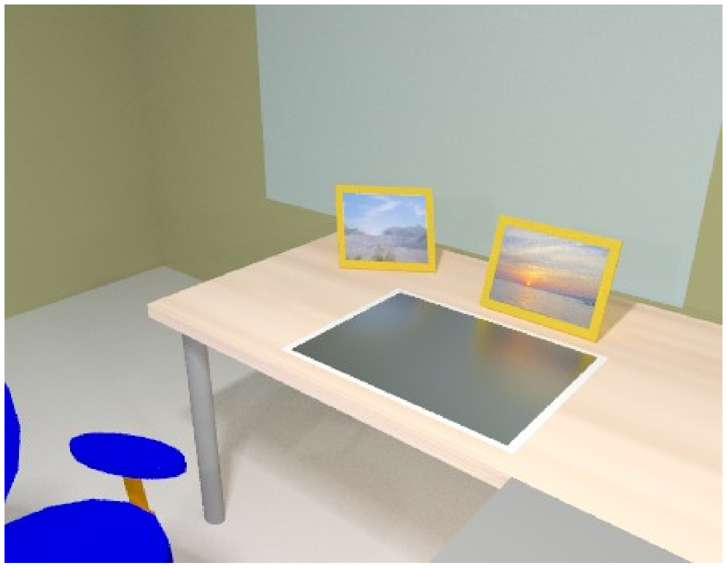


Figure 3.10: Final render image of the the office scene. The camera is looking towards one desk in the office.

ple can mask errors in the caustics or indirect illumination, so that a lower accuracy could be allowed. One could also take into account surface texture, as illumination errors are less visible on high frequency textures.

- The footprint estimate, computed using path derivatives, could be used for other purposes:
  - Compute the footprint of photons and distribute them accordingly.
  - It could be used for eye paths in the rendering pass to determine the area over which photons must be considered for illumination reconstruction. If too few photons are found the path can be extended (for the global map). This might help answering the question of how many photons to use in the reconstruction.

Many other options can be explored to make the photon map (even) more efficient. For us the ultimate goal is to derive a method that automatically constructs accurate photon maps, without the user having to tune different parameters such as the number of emitted photons and the number of nearest photons in the reconstruction. If we want to develop such a fully automatic rendering algorithm using photon maps that only requires one button ’Render’, importance is definitely necessary to incorporate error control and to help determine the parameters of the algorithm.

Acknowledgments

Thanks to my colleagues, former colleagues, Henrik and Per for interesting discussions and for proofreading the notes. Thanks to Vincent Masselus for the hand crafted VRML model of our new ’office’.

Chapter 4

Photon maps in RenderPark

RenderPark is a photo-realistic rendering tool being developed at the Computer Graphics Research Group of the Katholieke Universiteit Leuven, in Belgium. The goal is to offer a solid implementation of many existing photo-realistic rendering algorithms in order to compare them on a fair basis, evaluate benefits and shortcomings, find solutions for the latter and to develop new algorithms that are more robust and efficient than the algorithms that are available today. RenderPark will offer you several state-of-the-art rendering algorithms that are not yet present in many other rendering packages. Although RenderPark is in the first place a test-bed for rendering algorithms, it is evolving towards a full-featured physics-based global illumination rendering system.

The source code (C/C++) of RenderPark is freely available for non commercial purposes. It can be found at **www.renderpark.be**.

4.1 Overview

RenderPark can be used interactively (Motif/OpenGL) and as a batch renderer. Rendering algorithms currently implemented in RenderPark are:

- Radiosity algorithms:
  - Galerkin radiosity (hierarchical, clustering, importance driven)
  - Stochastic Jacobi radiosity (idem)
  - Various random walk radiosity methods
- Ray tracing methods (all usable as a second pass after radiosity):

  - Classical ray tracing
  - Stochastic ray tracing
  - Bidirectional path tracing

- And of course photon maps are implemented (see next section).

The main authors of RenderPark are: Philippe Bekaert, Frank Suykens, Jan Prykril and Phil Dutré. The development of RenderPark is supported by the Belgian National Science Foundation (FWO-Vl), the Flemish Institute for the Promotion of Scientific-Technological Research in Industry (IWT). RenderPark is realized within the context of various international collaboration projects as well.

4.2 Photon maps

This section contains a few specific details on the photon map implementation in RenderPark.

The construction of photon maps is implemented as a separate method. Both a global photon map and a caustic photon map are supported. The final rendering pass is built into stochastic ray tracing.

Support for importance driven construction of the photon maps is also included. In a first pass, two importance maps are build that estimate a local required density for the global and the caustic photon map. These maps are used during the construction of the photon map: no additional photons are stored in regions of sufficient density, they are distributed over several neighbors. This limits storage use. A detailed description can be found elsewhere in the course notes.

The implementation of photon maps is moderately optimized:

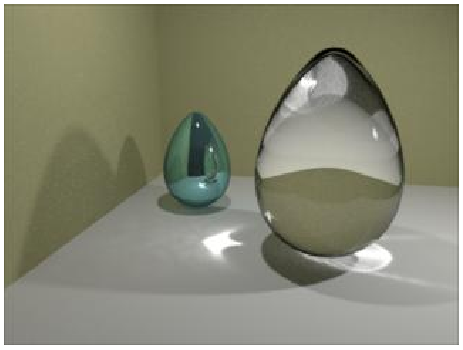
- Storage of photons can use balanced and unbalanced kd-trees (or even mixed). Lookups in balanced trees turned up to be 2 to 3 times faster.

- Irradiance and importance is precomputed for the photon/importon locations as described in [Christensen99].

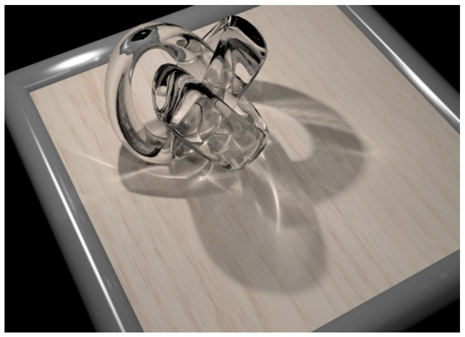
- A maximum radius estimate is used when querying the nearest photons and importons. For maps with large empty regions this can easily be an order of magnitude faster, because whole parts of the tree are skipped, that are otherwise examined completely.

- Recently we have implemented Greg Ward's irradiance caching scheme [Ward88] in the stochastic ray tracing pass. This makes rendering a lot faster, since indirect diffuse illumination is is not computed for every pixel but interpolated when possible. Speed-up in the final pass can be as high as a factor of 10 to even a 100, but sometimes the interpolation is visible in the image, especially in regions illuminated by indirect light only.

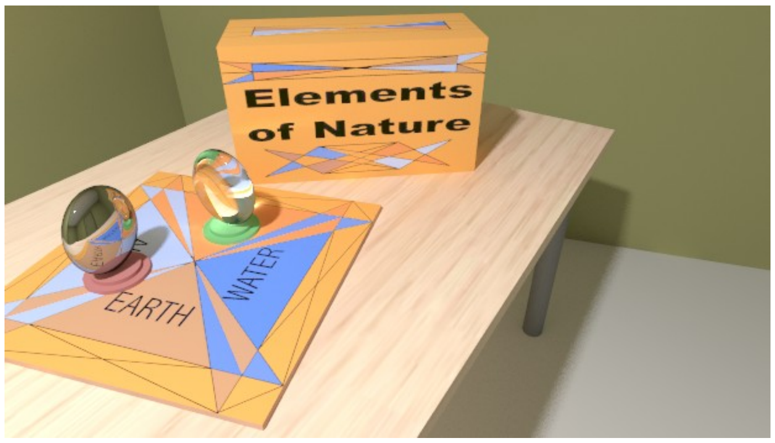
A few results rendered with photon maps in RenderPark are shown in figure 4.1. Both images compute full global illumination. Note the caustics, reflection of caustics and the indirect illumination.



- Eggs -



- Glass knot -



- Earth, wind, fire, water and photons -

Figure 4.1: Three images rendered using photon maps in RenderPark. Note the caustics and indirect illumination.

Chapter 5

Photon Mapping Tricks

Photon Mapping Tricks

Per H. Christensen

Pixar (Seattle)

Abstract
This note describes seven "tricks" for improved efficiency and accuracy of the photon map method. None of these tricks are particularly revolutionary or hard come up with, but it is my hope that this note can save some people the effort of reinventing them. The first trick reduces flicker in animations, while the following three tricks reduce computation times anywhere from 25% to factors of five-to-eight. The fifth trick investigates how to get unbiased radiance estimates. The sixth trick describes how to combine irradiance estimates from several independent photon maps (for example computed in parallel on separate computers). The last trick is somewhat more experimental, but promises a speedup close to 16 in the photon tracing phase for very complex scenes.

5.1 Frame-coherent random numbers for photon tracing

This section describes a method to reduce flickering in animations, particularly for caustics and participating media. The variance is reduced by factors of 10 or more.

5.1.1 Frame coherency

The simplest type of animation is a camera fly-through. In this case, we obviously only need to generate one photon map which can then be used in all frames of the animation. The opposite extreme is an animation where the entire illumination changes or where all objects move in all frames. In this case, we need to generate a new photon map for each frame, and no coherency can be exploited between the photon paths in each frame. However, a commonly occurring type of animation is between these extremes: only a few objects move in an otherwise static scene. In this case it pays off to keep as many photon paths as possible identical between frames, thereby reducing flickering considerably.

As an example, consider caustics from three whisky glasses — see figure 5.1. If only one of the glasses moves, only the caustic for that glass should change; we want to avoid any random fluctuations in the other two caustics.

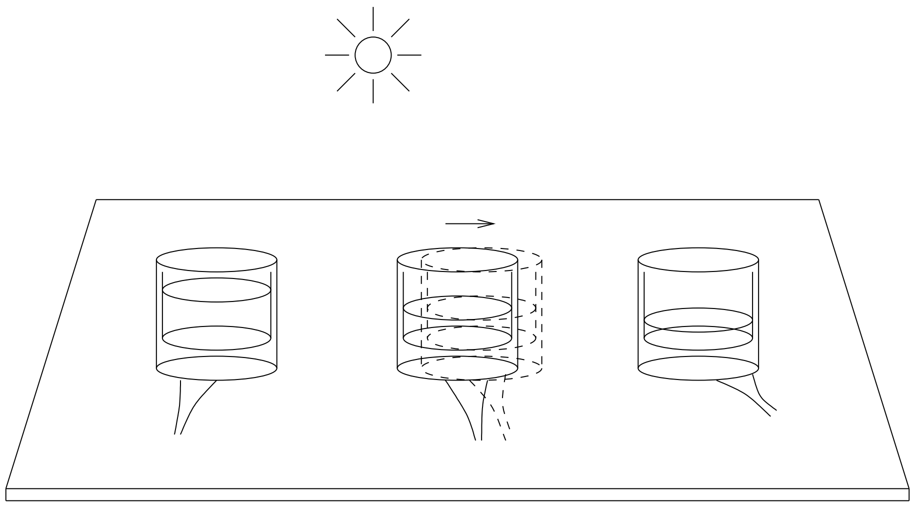


Figure 5.1: Caustics from three whisky glasses. If only the middle glass moves, only the middle caustic should change.

5.1.2 Keeping and reusing photon paths

One could keep track of all photon paths by keeping a pointer from each stored photon to the next stored photon along the same path. (Detail: for this to work, we would have to also store photons at emitters and purely specular surfaces.) In the next frame, each photon follows the same path as in the previous frame unless there is a change along the path. This works as follows. For each path step, the direction is computed as the difference between two photon positions and the photon is shot in the same direction. If the nearest "new" intersection point is identical, we keep the photon at the intersection point and continue along the path. If the intersection point is different (either because a moving object now occludes the path or because the receiving object moved), we delete the photons from the rest of that path and generate a new random path continuation.

This would probably work just fine. There is, however, a simpler way to obtain the same path coherence: we can retrace all photons while keeping the random numbers coherent from frame to frame. Then most photons will follow the same path as in the previous frame.

5.1.3 Coherent random numbers

Random numbers are used to determine photon emission origin and/or direction. Random numbers are also used when a photon hits a surface to determine whether the photon should be absorbed or scattered, and (depending on the reflection parameters) the scattering type and direction. The random numbers are typically generated by a pseudo-random number generator with good statistical properties, for example `drand48()`.

Restarting the random number generator with the same seed for each frame in an animation unfortunately does not guarantee that most photons will follow the same paths. If we just use one random number sequence for all photons, then the path length (the number of bounces) of one photon influences the random numbers for all subsequently traced photons (and therefore their paths and path lengths). In an animation, this has the unfortunate consequence that if just one photon path has a different path length (for example because one object moved slightly), then the path of all subsequent photons will be different from their paths in the previous frame.

To avoid this, we need to associate a random number sequence with each emitted photon. For example: compute a random seed as a function of the photon number, run `drand48()` a few times to get rid of coherency, and use the subsequent random number sequence for that photon. The current random number for each photon is kept with the photon while it is being traced so that we can easily compute the next random number for that photon.

As an added bonus, this use of the random numbers also makes debugging easier since local changes in geometry only give local changes in the photon paths. It also gives consistent results in parallel execution. (Multiple processors sharing a single random number generator are prone to getting different random numbers in different runs depending on the random execution order.)

5.1.4 Uses and limitations

This method greatly reduces noise in direct uses of the photon map: rendering of caustics and participating media. However, since we use final gathering to calculate soft indirect illumination on surfaces, we do not see the soft indirect photons directly. So this trick gives a smaller improvement for that type of illumination.

5.1.5 Origins and references

The technique of keeping a random number with each particle originates from the early years of Monte Carlo simulation in nuclear physics. When trying to simulate a small change in a simulation setup (for example the thickness of a radiation shield or the coolant temperature), they found that the change would often "drown" in statistical noise if they didn't associate a consistent random number sequence with each particle. Spanier and Gelbard formulated it as follows: "... it is essential to correlate the two runs positively so that, to as great an extent as is possible, only the effects of the perturbation itself are subject to statistical fluctuation." They state that quite commonly, this very simple correlation technique reduces variances in difference by factors of 10 or more. Details can be found in the following two references:

- Gerald Goertzel and Malvin H. Kalos. "Monte Carlo methods in transport problems". *Progress in Nuclear Energy*, series I, vol. 2, pp. 315–369. Pergamon Press, 1958. (Page 361)

- Jerome Spanier and Ely M. Gelbard. *Monte Carlo Principles and Neutron Transport Problems*. Addison-Wesley Publishing Co., 1969. (Sections 3.9 and 5.5)

5.2 Faster lookups, part I: automatically computed maximum search radius

This section presents a formula to automatically determine a reasonable maximum photon search radius for photon map radiance estimates.

5.2.1 Background: radiance estimates

In the photon map method, the incident irradiance and the reflected radiance are computed by a density estimate. For the reflected radiance L_r, we add the power of the n photons that hit a certain area A_n, multiply by the BRDF f_r, and divide by the size of the area:

$$L_r(x, \omega) \approx \frac{\sum_{i=1}^{n} f_r(x, \omega_i, \omega) P_i}{A_n} .$$

There are two ways to determine the photon density:

1. Fixed area: given a fixed area, find all photons within that area, add their powers, and divide by the area.

2. Fixed number of photons: find the nearest n photons, add their powers, and divide by the area they cover.

In regions with medium and high photon density, we prefer to use a fixed number of photons. This adapts nicely to varying densities, keeps the error reasonably constant, and avoids finding way too many photons in areas with very high photon density (very high irradiance). However, if we use this strategy in regions with low photon density, large parts of the kd-tree will be searched to find the nearest n photons, only to result in a very low radiance. This is a waste of computation time. The solution is to use a combination of the two photon density estimates: in regions with medium and high photon density, we use a fixed number of photons, while in regions with low photon density, we use a fixed area. Getting such a combination is actually quite simple: set an upper limit n on the number of photons to find and set an "appropriate" maximum search radius r for the kd-tree search.

5.2.2 Effects of the maximum search radius

If the radius is set too high (or infinite), large parts of the kd-tree will be searched to find the nearest n photons in sparse regions, only to result in a very low radiance.

This is a waste of computation time. If the radius is much too low, a “polka dot effect” can be seen in the images: the area of a radiance estimate either contains a photon or not, resulting in a characteristic polka dot pattern. The “appropriate” radius r for a given scene is often set manually by trial-and-error: start with a very large maximum radius and keep reducing it and rerender until artifacts become visible. To ease the use of the photon map method, it is preferable to have the maximum search radius computed automatically.

5.2.3 Maximum search radius for surfaces

The “trick” presented here is a formula to automatically compute an appropriate maximum search radius, thereby avoiding the tedious trial-and-error approach described above. The switch between the two density estimates should be done when the density is low enough that inaccuracies in the resulting radiance are not important. Let L_t be the radiance threshold for the switch and r_m the corresponding maximum search radius. Let $A_m = \pi r_m^2$ be the area corresponding to r_m, and let P_{max} be the maximum power of any of the photons stored in the kd-tree. We can bound the total power of the n nearest photons by $n\,P_{max}$ and bound f_r for a diffuse BRDF by $1/\pi$ to get:

$$L_r = \frac{\sum_{i=1}^{n} f_r(x,\omega_i,\omega)P_i}{A} \le \frac{1/\pi\, n\, P_{max}}{\pi\, r^2}\,.$$

The threshold radiance L_t is then

$$L_t = \frac{n\,P_{max}}{\pi^2 r_m^2}\,.$$

¿From this we get the maximum search radius

$$r_m = \frac{1}{\pi}\sqrt{\frac{n\,P_{max}}{L_t}}\,.$$

If we display colors between 0 and 1, we might for example choose $L_t = 0.05$. Then we get

$$r_m \approx 1.4\sqrt{n\,P_{max}}\,.$$

5.2.4 Maximum search radius for volumes

There is a corresponding formula for lookups in a volume photon map. For a volume, the radiance estimate is

$$L_r(x,\omega) = \frac{1}{\sigma(x)}\frac{\sum_{i=1}^{n} f(x,\omega_i,\omega)P_i}{\frac{4}{3}\pi\, r^3}\,.$$

— where f is the normalized phase function and σ is the volume scattering coefficient. For an isotropic (diffuse) volume the normalized phase function f is $1/(4\pi)$. The formula for the maximum search radius in a volume photon map is

$$r_m = \sqrt[3]{\frac{3\,n\,P_{max}}{16\,\pi^2\,\sigma\,L_t}}\,.$$

With, for example, a threshold radiance of $L_t = 0.05$ we get the maximum search radius

$$r_m \approx 0.72\sqrt[3]{\frac{n\,P_{max}}{\sigma}}\,.$$

5.2.5 Maximum distance to nearest photon

In the final gather stage, we can locate the nearest (single) photon and then use its precomputed radiance — see section 5.4. However, in dark regions there will be very few photons, and the nearest photon can potentially be quite far away. In that case, it can be quite incorrect to use that photon's radiance. It is more efficient (and often more correct) to disregard the nearest photon if it is far away from the lookup point and just use black for the radiance. We therefore need a reasonable cut-off distance so that if there is no photon within that distance, we can terminate the search.

There are two cases for the scenario where the nearest photon is far away from the lookup point. In one case, the nearest photon, although distant, is part of a relatively dense cluster of photons (corresponding for example to a local bright spot or the bright side of a shadow edge). In that case, that photon will have a fairly high radiance, and we would introduce a large error if we use that radiance far into the region with no photons (where the radiance should be very low). The other case is much better: the distant photon is part of a truly sparse photon distribution, and has a very low radiance value. It would be correct to use that low radiance, but we would not make a big error by using black instead.

After the nearest photon has been found, we can determine whether we can reasonably use its radiance by checking whether the lookup point is within the area of the radiance estimate of that photon. This requires us to store the radiance estimate radius r with the photon, but that is only a single additional float. (This radius would need to be stored anyways if we want to optimally combine lookups from several photon maps, as discussed in section 5.6.) If the lookup point is outside the estimate area, we have to assume that the radiance at the point is black.

This is a very valuable check after we have located the nearest photon, but for efficiency we would like to also have an a priori cut-off distance for the search.

Fortunately, we do have an a priori upper limit on the radius of the density estimate area — that's exactly what we computed in the previous two subsections! We can simply use that radius (r_m) as the cut-off distance for finding the nearest photon.

Instead of using the exact radius of the density estimate as a cut-off distance, one can of course use a fraction or a multiple of it, according to taste (and judgment of how far from the center of a radiance estimate it is reasonably valid).

5.3 Faster lookups, part II: iteration instead of recursion

This section presents an iterative algorithm for finding the nearest photons from a given point. The nearest photons are used to estimate irradiance or radiance. The iterative algorithm is up to 25% faster than a recursive version.

5.3.1 Background: recursive algorithm

A recursive algorithm to find the n photons closest to a point p is described thoroughly elsewhere in these course notes, but will be repeated here for easy reference.

The photons in the photon map are stored in a kd-tree. The tree is a left-balanced binary tree, so it is conveniently stored in an array as an implicit tree: array element 1 is the tree root, and node i has children $2i$ and $2i + 1$. There are N photons in the kd-tree array, and `Nhalf` = $\lceil N/2 \rceil$ is the array index of the last non-leaf node. `nearestDist2` is the squared maximum distance of the photons found so far, it is adjusted as more photons are found. The algorithm is initially called with $i = 1$ (the array index of the tree root node). The algorithm first descends down through the tree depth-first, at each recursive call choosing the half-space that contains point p. When a leaf is reached, its 3D distance to point p is computed and it is inserted into the heap of nearest photons if it is close enough. On the way back up from the first recursive calls, the 1D distance to the other half-space is examined. If we cannot rule out that a photon in the other half-space could be among the nearest, we make a new recursive descent, this time in that half-space. When one or both half-spaces have been searched, the photon itself is examined and possibly inserted into the heap of closest photons (if it is close enough). The recursive search algorithm is:

```
void FindNearestPhotons(point p, int i) {   // recursive version

  // Recursively examine the child nodes if node i is not a leaf
  if (i < Nhalf) {
    dist1d = signed 1D dist. from splitting plane of node i to point p
    if (dist1d < 0.0) {   // p is left of plane
      FindNearestPhotons(p, 2i)   // search left subtree first
      if (dist1d^2 < nearestDist2)
        FindNearestPhotons(p, 2i+1)   // search right subtree
    } else {   // p is right of plane
      FindNearestPhotons(p, 2i+1)   // search right subtree first
      if (dist1d^2 < nearestDist2)
        FindNearestPhotons(p, 2i)   // search left subtree
    }
  }

  // Check photon i, add it if it is among the nearest so far,
  // and update nearestDist2
  CheckAddNearest(p, i)
}
```

The inline procedure `CheckAddNearest(p, i)` checks whether the 3D distance between point p and photon i is small enough that photon i should be added to the current set of nearest photons, and if so, adds it.

5.3.2 Iterative algorithm

It is well known that any recursive algorithm can be rewritten as an iterative algorithm. Although the recursive algorithm often is most clear, the iterative version is usually faster (since recursive calls require pushing the current state on the call stack). However, the kd-tree lookup algorithm is doubly recursive, since we sometimes have to visit both children of a node. This complicates the rewriting as an iterative version, and makes the iterative version somewhat inelegant.

Again, there are N photons in the kd-tree array, and `Nhalf` = $\lceil N/2 \rceil$ is the array index of the last non-leaf node. We use two auxiliary arrays: the float array `dist1d_2[]` keeps track of (squared) 1D distances between point p and photons, while int array `chosen[]` keeps track of which child we visited first on our descend through the tree. Both arrays need to have at least as many elements as there are levels in the kd-tree, ie. $\lceil \log_2(N) \rceil$ elements. The iterative search algorithm is:

```
void FindNearestPhotons(point p) {   // iterative version

  i = 1; level = 0   // start at root node
```

```
    // Move up and down the kd-tree until return (when past the root)
    while (true) {

      // Move down through the subtrees containing p until a leaf is reached
      while (i < Nhalf) {
        dist1d = signed 1D dist. from splitting plane of node i to point p
        dist1d_2[level] = dist1d * dist1d
        i = 2i
        if (dist1d > 0.0) ++i   // choose left/right child
        chosen[level++] = i
      }

      // Check this leaf photon, add it if it is among the nearest
      // so far, and update nearestDist2
      CheckAddNearest(p, i)

      // Move up in tree until we reach a photon where we need to
      // check that photon and the other subtree
      do {
        camefrom = i
        i = i/2; --level   // go to parent
        if (i == 0) return   // we passed the root: return
      } while (dist1d_2[level] >= nearestDist2 || camefrom != chosen[level])

      // Check this non-leaf photon, add it if it is among the
      // nearest so far, and update nearestDist2
      CheckAddNearest(p, i)

      // Step into the other subtree
      i = chosen[level++]ˆ1;
    }
  }
```

Depending on the processor, optimizer, and how many photons are to be found, this iterative version can be up to 25% faster than the recursive version. The same algorithm is used to find the nearest (single) photon for reuse of its precomputed radiance value (see section 5.4).

Thanks to Piero Foscari for suggesting some improvements of this algorithm and for discussions about ways to improve it further. It might be possible to entirely rewrite this iterative algorithm to be prettier and faster than the version presented here. If you find a better version, please let me know!

5.3.3 Further reading

An interesting alternative to the kd-tree for storing the photon map was suggested in the following publication:

- Marec Vanco, Guido Brunnett, Thomas Schreiber. "A hashing strategy for efficient k-nearest neighbor computation". *Computer Graphics International*, pp. 120–128. IEEE, June 1999.

They use of a collection of hash tables to store the photons and report faster lookup times than with a kd-tree. It would of course be interesting to try their algorithm.

General discussions of algorithm optimization and kd-trees can be found in many computer science textbooks, for example:

- Thomas M. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
- Robert Sedgewick. *Algorithms in C++*. Addison-Wesley, 1992.
- Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry — Algorithms and Applications*. Springer-Verlag, 1997.

5.4 Faster lookups, part III: precompute radiance estimates

Final gathering is used for rendering soft indirect illumination. This section describes a "trick" that gives a speed-up of 5–8 for final gathering in typical scenes.

5.4.1 The basic method

This optimization is based on three observations:

1. Final gathering is the most time-consuming part of the photon map method for global illumination calculation — especially locating the nearest 50–200 photons for the radiance estimate where each final gather ray hits.
2. Many final gather rays (from different final gathers) hit nearly the same point: almost identical photon map lookups are repeated again and again.

3. Since each final gather result is the average of hundreds or thousands of radiance estimates, each estimate does not have to be very precise.

Based on these observations, the many lookups in the global photon map during final gathering can be simplified by precomputing local radiance values at the photon positions and storing those values with the photons. When a final gather ray hits a surface, we would normally do a full radiance estimate at that point — requiring finding the nearest 50–200 photons in the global photon map. Instead, we simply find the (single) nearest photon with a surface normal similar to the normal at the ray intersection point and use its precalculated radiance value. This way, the calculation of soft indirect illumination can be sped up by a factor of 5–8 in typical scenes.

The approximate radiance at a photon location is found by estimating the irradiance (by a lookup in the photon map) and multiplying that irradiance by the average diffuse reflection coefficient for the area the photon covers. The average diffuse reflection coefficient can be determined by a shader evaluation along with texture mip map lookups at an appropriate level.

Using the precomputed radiance of the nearest photon means that the radiance used for final gathering is approximated as a piecewise constant function. Formally, the photons divide the scene into a Voronoi diagram with a constant radiance within each Voronoi cell. This approximation is acceptable because the difference between the irradiance at neighboring photon positions is relatively small (since many photons are used to compute each irradiance), because each photon only covers a very small fraction of the scene, and because we only use the approximation for final gathering above a certain distance (at shorter distances we do a secondary final gathering instead).

The precalculation typically takes less than 2% of the time saved. It is not necessary to precompute the radiance at all photon locations; we have found that for typical scenes, it is sufficient to compute the radiance at 1/4 of the photon locations. It is very simple to select 1/4 of the photons since they are located in an array — simply compute the radiance for the photons in the first 1/4 of the array. In the kd-tree, this corresponds to all the photons that are neither leaves nor parents of leaves. The tree structure means that the selected 1/4 photons will automatically have a fairly good spread in the scene; we cannot end up with only the photons in one side of the scene having their radiances precomputed.

The method requires us to store the surface normal and radiance for each photon in addition to the standard position, incident direction, power, and split dimension. Using 1 byte to represent the surface normal direction and 4 bytes for the radiance (using Ward's RGBE format), the storage for each photon increases from 18 bytes to 23, an increase of less than 28%.

5.4.2 Example: interior scene

The following is an example of a scene rendered with this method. The scene consists of more than 1 million polygons. The images have 1024×768 pixels sampled with up to 16 samples per pixel. The images are computed on a Linux PC with a 733 MHz Pentium III processor and 540 MB memory. (Only a small fraction of that memory was actually used.)

Figure 5.2(a) shows the scene rendered with classic ray tracing. It shows soft shadows, specular reflections, and textures, but lacks soft indirect illumination. For example, the upper right corner is completely black since it is not directly illuminated. This image was computed in 2 minutes 34 seconds. Figure 5.2(b) shows the global photon map for this scene. Photon tracing took 8 seconds and sorting the 500,000 photons into a kd-tree took 4 seconds. Figure 5.2(c) shows the precomputed radiances at 125,000 photon positions. Even though it is a coarse approximation of the illumination, it is fully sufficient for final gathering. Computing these radiances took 15 seconds. Figure 5.2(d) is the complete image with soft shadows, specular reflection, textures, and soft indirect illumination. The final gathering took 2 minutes 45 seconds, and the total render time for this image was 5 minutes 57 seconds. For comparison, the final gather stage takes nearly 22 minutes without precomputed radiances, so the final gather time is reduced by a factor of 7.8. The 15 seconds spent on precomputing the radiances corresponds to 1.3% of the 19 minutes saved during rendering.

5.4.3 Other applications

The same method can also be used for precomputing radiances in a volume photon map. This speeds up ray marching through an isotropically reflecting participating medium quite significantly. The method is also useful for precomputing importance at the "importon" positions in an importance map.

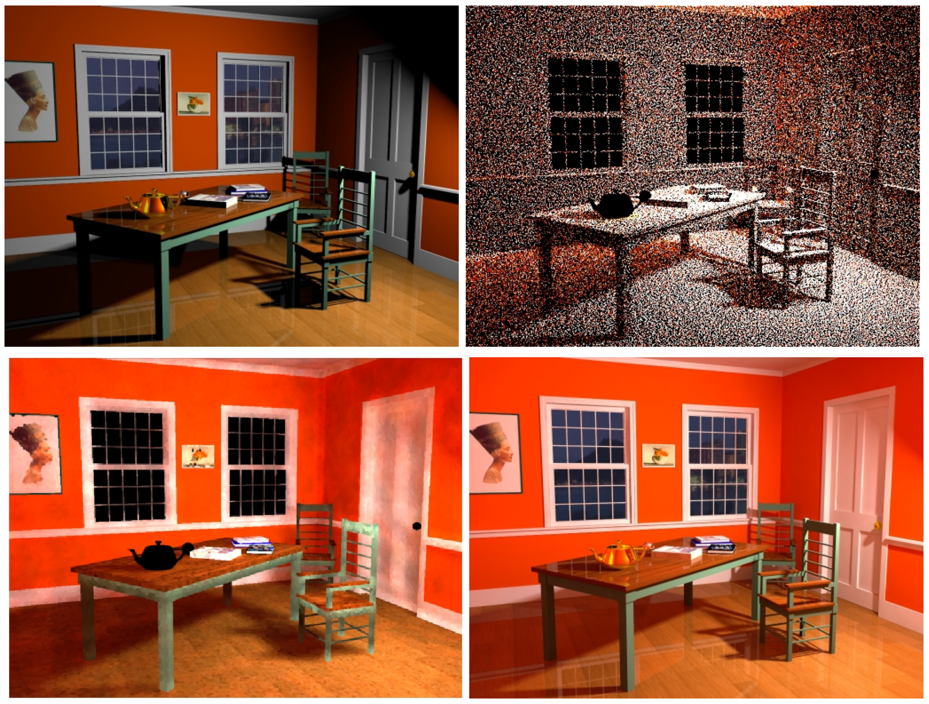


Figure 5.2: Interior scene: (a) Traditional ray tracing. (b) Photon map. (c) Precomputed radiance estimates at 1/4 of the photon positions. (d) Complete image with direct illumination, specular reflection, and soft indirect illumination.

5.4.4 Variations

There are numerous other variations and possible improvements of this method.

One could find the nearest *three* photons with appropriate normals and do a bilinear interpolation of their radiances. This would make the approximation of radiance piecewise linear instead of piecewise constant. However, we have found the resulting increase in accuracy completely unnecessary.

We can compute the radiances "on demand" instead of as a preprocess. When a final gather ray hits a point, we find the nearest photon with appropriate normal. If that photon has a radiance value stored with it, we simply use it. If the photon does not have a radiance value, we find the nearest n photons, compute the irradiance, evaluate the shader and textures, multiply the irradiance by the diffuse coefficient, store the resulting radiance with that photon, and use that radiance value.

The advantage of computing the radiance "on demand" like this is that if some photons are in parts of the scene that are never seen (neither directly nor indirectly) in the image, then their radiance will not be computed. (However, if importance is used to decide where to store photons, the number of stored unimportant photons should be relatively small. See section 5.7 for a discussion of the use of importance.)

On the other hand, computing all radiances as a preprocess has the potential advantage that coherency can be used (although I currently don't know how that should be done): the radiance computations can be ordered such that neighboring photons have their radiance computed right after each other. Then most of the found photons for the two neighboring radiance estimates will be the same. It might be possible to use this coherency. Currently, we just compute the radiance for each photon position independently, which intuitively seems wasteful. It might be that another data structure than the kd-tree is better suited for this type of coherent lookups?

Currently, we compute the radiance at all or 1/4 of the photon positions. The number 1/4 seems quite arbitrary. Why not 1/16 or 1/100? We have chosen 1/4 because it is "safe": there is hardly any visible difference between a Voronoi diagram based on all the photon positions and one based on 1/4, and the tiny differences certainly have no impact on the final gather results. With 1/16 of the photon positions, the Voronoi cells get quite large and the irradiances of neighboring cells can be quite different. At that level of approximation, one might as well have traced fewer photons in the photon tracing phase and then only precomputed radiance at 1/4 of them. Furthermore, the time to precompute radiances at 1/4 of the photons is already negligable compared to the total render time, so it is not very important to reduce it further.

When we compute radiance at a constant fraction of the photon positions, we automatically get a well distributed coverage of the scene, and we automatically get most radiance estimate calculations in the most brightly illuminated areas (areas with high photon density, i.e. high irradiance). This is pretty good. However, we can improve this by computing more radiances at places with high illumination variation (places with high photon density variation, i.e. high irradiance gradient). One can start out computing the radiances at 1/4 of the photons and then at the children of the photons that have very "skewed" local photon distributions — that is, where the photons used for the radiance estimate are very unevenly distributed within the estimation area. In other words, compute a radiance estimate at a given photon if it is one of the top 1/4 photons in the kd-tree or if its parent has a skewed photon distribution. The result is dense radiance estimates where the irradiance is high and where the irradiance variation is high. During the search for the nearest precomputed radiance (in the rendering phase), the descend down through the kd-tree stops when a photon either has no children or the children have no precomputed radiance.

Another variation is to store irradiance instead of radiance with the photons. (This is actually how this precomputation method was first used, and it might still have some merits in certain cases.) When a final gather ray hits a surface point, the precomputed irradiance estimate of the nearest photon is multiplied by the reflection coefficient precisely at the hit point. This results in more accurate radiance values than precomputed radiances since e.g. textures can be resolved much more accurately. However, computing the reflection coefficient at all final gather ray hit points can be time consuming since each computation requires the evaluation of a shader and (typically) some detailed texture map lookups (at a fine level in the mip maps — potentially trashing the texture cache). Futhermore, the ability to represent high-frequency variation in the radiance estimates is unnecessary since we are only using the radiance estimates for final gathering. In fact, having that high-frequency variation means that we need more final gather rays to get an acceptably low noise in the final gather results. (It is cheaper to sample a smooth function than a wildly varying function.)

5.4.5 Extensions

Jensen (1995) used incident photon directions to guide ray directions in the rendering phase: when a ray hits a diffuse surface, the nearest (indirect) photons are located, and their incident directions are used to bias the ray direction towards bright regions of the scene. Peter and Pietrek (1998) applied the same approach to guide photons in the photon tracing phase: when a photon hits a diffuse surface, the nearest (indirect) importons are located, and their incident directions are used to guide the photons towards important parts of the scene. In both cases, it would be beneficial to have the incident directions of nearby photons/importons stored with each photon/importon. This directional information can easily be determined during the precomputation of radiance/importance — at which time the nearest photons/importons are located anyways. The directional information can be stored and reused during rendering/photon tracing. It can for example be stored in a few bytes, with each bit corresponding to a fraction of the hemisphere: the bit is set if a significant amount of photons/importons came from that direction, so that rays/photons should have a higher probability of being scattered in that direction.

It might be possible to extend the method to handle surfaces with wide glossy (aka. directional diffuse) reflection. Each incident photon causes a drop-shaped "blobby" radiance distribution. The sum of 50–200 of these distributions will be a wider blob. To get a useful speed-up out of storing and reusing such directional radiance distributions would require a compact representation of the distribution as well as a very fast way to evaluate it in a given direction. Spherical harmonics do not seem like a promising choice, but perhaps wavelets (Christensen 1996) or multiple cosine-lobes (Lafortune 1997) are possibilities? If a suitable representation can be found, a similar representation can be used for anisotropically scattering participating media.

5.4.6 References and credits

The idea of storing irradiances with the photons was originally published in:

- Per H. Christensen. "Faster photon map global illumination". *Journal of Graphics Tools*, 4(3), pp. 1–10. ACM, 1999.

The idea of computing irradiances on demand rather than as a precomputation was first suggested to me by Toshi Kato. The pros and cons of storing radiance rather than irradiance (along with the other variations and extensions described here) were discussed at the Dagstuhl seminar "Image Synthesis and Interactive 3D" (Schloss Dagstuhl, Wadern, Germany, June 2000).

The following two references used incident photon directions to guide ray directions in the rendering phase, and incident importon directions to guide photon directions in the photon tracing phase, respectively:

- Henrik W. Jensen. "Importance driven path tracing using the photon map". *Rendering Techniques '95 (Proceedings of the 6th Eurographics Workshop on Rendering)*, pp. 326–335. Springer-Verlag, 1995.
- Ingmar Peter and Georg Pietrek. "Importance driven construction of photon maps". *Rendering Techniques '98 (Proceedings of the Ninth Eurographics Workshop on Rendering)*, pp. 269–280. Springer-Verlag, 1998.

Finally, the representation of directional distributions with wavelets and with multiple cosine-lobes are described in:

- Per H. Christensen, Eric J. Stollnitz, David H. Salesin, and Tony D. DeRose. "Global illumination of glossy environments using wavelets and importance". *Transactions on Graphics*, 15(1), pp. 37–71. ACM, January 1996.

- Eric P. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. “Non-linear approximation of reflectance functions”. *Computer Graphics (Proceedings of SIGGRAPH 97)*, pp. 117–126. ACM, August 1997.

respectively.

5.5 Unbiased radiance estimates

This section presents various ways to improve the accuracy of photon map radiance estimates.

Because of the stochastic nature of photon tracing, it is inevitable that the photon density will be too high in some places and too low in other places, resulting in noisy radiance estimates. However, despite the noise, the average of the radiance estimates should be correct. And the radiance should not be biased (ie. consistently too dark or too bright) near geometric edges.

5.5.1 Experimental setup

To evaluate and compare the various ways of estimating the radiance, we use a setup with extremely simple geometry and illumination. The test scene consists of a single diffuse square illuminated by parallel light (no indirect illumination). The correct radiance is easily calculated for reference; it is constant across the entire square.

A photon map was generated that contains 40,000 stratified photons on the square. Lookups in this photon map will be the basis of the following comparisons.

5.5.2 Disc estimate

The “classic” photon map approach to estimating the radiance at a point is to find the nearest n photons in the photon map, add their power, divide by the area of the smallest disc (centered at the point) that covers the photons, and multiply by the BRDF at that point. This is illustrated in figure 5.3 for $n = 15$.

There are (at least) three problems with this estimate: it is noisy, it is slightly too bright on average, and it is too dark near geometric edges. For a given photon map, we can reduce the noise by increasing n. (The noise can also be reduced somewhat by carefully stratifying the photon emission and scattering, as described elsewhere in these notes.)

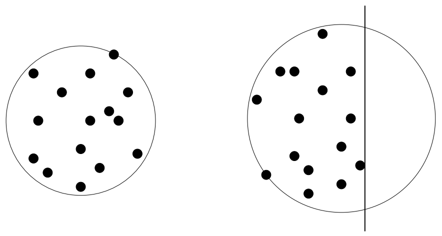


Figure 5.3: Disc estimates. (a) Far from edges. (b) Near an edge.

The slightly too bright radiance stems from the fact that the power of all n photons are added — even the (most distant) photon that is located right on the edge of the disc. So the power is actually a bit larger than what is appropriate for the area the photons cover. This problem is easy to overcome: if only half of the power of last photon is included in the sum, the average radiance will be correct. We can also get the correct average radiance by multiplying the sum of powers by $(n - 0.5)/n$ or (equivalently) by multipling the area by $n/(n - 0.5)$.

The remaining problem is the dark edges, as shown in figure 5.4. The dark edges get larger when n gets larger. So we have two conflicting goals here: we want to reduce noise by increasing n and we want to reduce the dark edges by decreasing n.

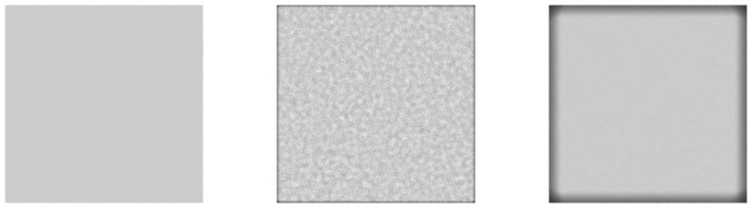


Figure 5.4: (a) Reference solution with constant intensity. (b) Disc estimate with radiance estimated from 25 photons. (c) Disc estimate with radiance estimated from 500 photons.

5.5.3 Variations of the disc estimate

The cone filter is a variation of the disc estimate. The nearest n photons within a disc are still found, but the power of each photon is attenuated depending on its distance from the lookup point: the photons furthest from the lookup point have less weight in the power summation. Cone filter radiance estimates are shown in figure 5.5. Even thought it is hard to see in the figure, the cone filter reduces the dark edges a bit. But it is impossible to eliminate the dark edges completely with the cone filter. (Also, the cone filter unfortunately gives more noise in the radiance estimates. This is probably because the randomness of the distribution of the photons inside the disc now influences the radiance estimate.) If the cone filter is used, the power of the last photon should be multiplied by 0.5 times the filter weight to get the correct average radiance.

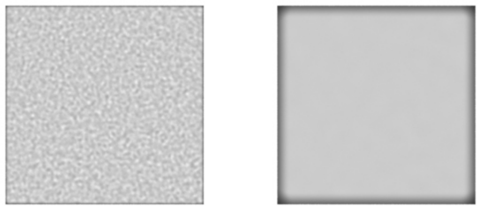


Figure 5.5: Disc estimates with cone filter ($k = 1.1$). (a) Radiance estimated from 25 photons. (b) Radiance estimated from 500 photons.

Another variation on the disc estimate is the so-called “differential checking”. Here the nearest photons are sorted according to their distance to the lookup point, and the radiance is estimated first from a few nearest photons and then from more and more photons. If there is a clear trend towards darker and darker radiance (or brighter and brighter radiance) when more photons are included, the inclusion stops. Differential checking will detect that near a geometric edge, the radiance will get darker and darker the more photons we include in the radiance estimate. Hence very few photons will be used in the radiance estimates near edges. The result is that the dark edges are avoided, but unfortunately the radiance is very noisy near the edges. Bias can be handled in the same way as for the basic disc estimate.

5.5.4 Convex hull estimate

With the convex hull estimate, the dark edges can be avoided entirely without increasing the noise in the radiance estimates. For the convex hull estimate, we still find the n nearest photons, but compute the *convex hull* of them and use the area of the convex hull in the radiance estimate. (The convex hull of a set of points S is the smallest convex polygon for which each point in S is either on the bondary of the polygon or in its interior.) Figure 5.6 shows two examples of convex hulls of photons.

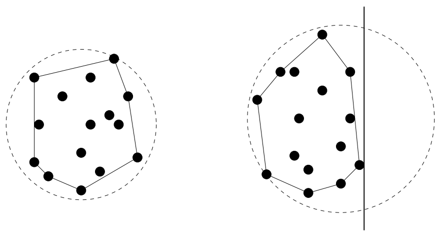


Figure 5.6: Convex hull estimates. (a) Far from edges. (b) Near an edge.

Adding the power of all the photons in the convex hull (including the photons on the boundary) and dividing by the area of the convex hull gives too bright radiance. Essentially, too much power is included for that area. Conversely, excluding all the power of the photons on the boundary gives too dark radiance. But the following heuristic seems to work very well in practice: include 20% of the power of the photons on the boundary. (This heuristic works with less than 0.5% error in the average radiance for $n \geq 15$.) As figure 5.7 shows, using the convex hull estimate eliminates the dark edges. At the same time, using the convex hull estimate does not increase the noise. One disadvantage, however, is that finding the convex hull is somewhat time-consuming. The convex hull computation will typically take up to twice as long as the disc estimate.

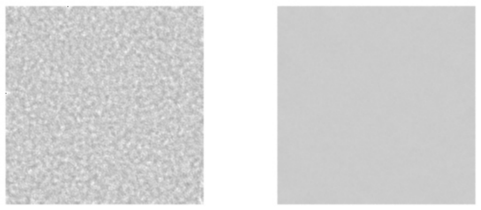


Figure 5.7: Convex hull estimates. (a) Radiance estimated from 25 photons. (b) Radiance estimated from 500 photons.

5.5.5 Origins and references

These radiance estimate methods (disc estimate, cone filter, differential checking, and convex hull) are described in sections 7.2 and 7.4 of Henrik's photon mapping book. More advanced filters are described in:

- Karol Myszkowski. "Lighting reconstruction using fast and adaptive density estimation techniques." *Rendering Techniques 97 (Proceedings of the 8th Eurographics Workshop on Rendering)*, pp. 251–262.

There are probably many more ideas for improved radiance estimates to be found in the vast literature of general density estimation.

5.6 Combining lookup results from several photon maps

This section presents a formula to combine lookup results from several photon maps.

5.6.1 Why several photon maps?

Several processors can share a photon map without conflict: storage and lookups are done in separate phases, so no locking is needed. The only reason to split a photon map is if there is not enough memory to store it on one machine. However, usually only a few million photons (of 18–24 bytes each) are needed even for complex scenes. Therefore it is no problem to store a full copy of the complete photon map on each computer to enable fully independent parallel lookups.

However, if there has to be several photon maps — either because the photon maps simply are to big or for some other reason — there are different ways the lookups can be combined.

5.6.2 Combining lookup results

We can arrange the photons such that photons located in particular parts of the scene are stored in particular photon maps. Sometimes we can then simply lookup in the appropriate photon map. This method can give problems at edges between the partial photon maps, though.

Instead, we assume that the photons are reasonably evenly distributed in the photon maps — that is, each photon is stored in a randomly chosen photon map or the photons are stored in the maps in a round robin fashion. Now, instead of finding the nearest n photons in a single photon map containing all photons, we do a lookup in M photon maps, finding the nearest n/M photons in each. The lookup in each photon map produces a power $P_m = \sum_{i=1}^{n/M} P_i$ and a radius r_m. See figure 5.8 for an illustration. Then the question is: how can we best combine the results? The first idea that comes to mind for computing the combined irradiance is to simply add the irradiances from each photon map lookup:

$$E_{total} = \sum_{m=1}^{M} \frac{P_m}{\pi r_m^2}$$

However, combining the powers and radii separately turns out to be much less sensitive to variations (probably because of the non-linearity of the formula). The powers and radii are then combined as follows:

$$E_{total} = \frac{\sum_{m=1}^{M} P_m}{\text{ave}_{m=1}^{M}(\pi r_m^2)} = \frac{M}{\pi} \frac{\sum_{m=1}^{M} P_m}{\sum_{m=1}^{M} r_m^2}$$

(This is very likely a well-known result from the density estimation field?) Note that this improved formula does not require any additional information, we're just using the available information better.

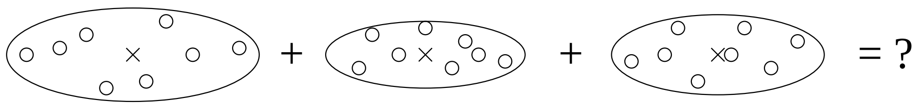


Figure 5.8: How to combine several independent density estimates for the same point?

5.7 Faster photon tracing using importance

The previous "tricks" have all been very thoroughly tested on many scenes of varying complexity. The present trick has only been tested on two scenes so far, so there is still a chance that it might fail in some cases. Anyway, here we go ...

5.7.1 Motivation

If the scene is large, if large parts of it are illuminated, and if only a small fraction of it is seen in the final image, many photons that do not contribute to the visible illumination are emitted, scattered, and stored in the first pass. The larger the illuminated parts of the scene are relative to the visible parts, the smaller the fraction of photons that are actually used.

To avoid wasting time and space, we want a photon tracing method with the following characteristics:

- Few emitted, scattered, and stored photons.
- No bias (or at least no visible bias).
- A homogeneous photon map (no mixing of photons with high and low power in the same region).

The third requirement is necessary because a few high-power photons among many low-power photons will result in extremely noisy radiance estimates: if a high-power photon is among the photons used to compute a radiance, the resulting radiance will be very high compared to the nearby radiances that didn't include that high-power photon. Visually, this means that the scene has very bright dots in the vicinity of those high-power photons, resulting in a "polka dot" pattern. It is, however, acceptable to have some regions in the photon map contain high-power photons and other regions contain low-power photons, as long as they are not mixed in the same region.

Here we present a method that satisfies all three requirements: it reduces the number of emitted, scattered, and stored photons while keeping the bias acceptable and the photon map homogeneous.

5.7.2 Previous work

Smits et al. (1992) introduced importance to computer graphics. In their seminal work, they defined importance as the adjoint of radiosity that has a source term at the view point, and used importance to reduce the number of links in a hierarchical radiosity solution. More generally, importance can be defined as the adjoint of any representation of light with the source term at the view point or at the directly visible points.

The method of Peter and Pietrek (1998) used importance to reduce the number of emitted, traced, and stored photons. Before tracing photons, they emitted importance particles ("importons") from the view point in the directions within view and traced them through the scene. The density of importance particles near a point is an estimate of the importance there. In the beginning of the photon tracing phase, they used a set of "test" photons to estimate the importance of various directions from each light source. For each light source they emitted a set of test photons and saw if any of them got traced into an important region of the scene. (The test photons were not stored in the photon map). From the results of these test photon paths, a finite element approximation of the importance in different directions from each light source was constructed. In the "real" photon tracing phase, relatively many photons were emitted in the important directions (each photon with low power), while few photons were emitted in the unimportant directions (each photon with high power). The danger in this approach is that some of the high-power photons actually might make it to one of the important regions in the scene even though none of the "test" photons did. When this happens, these high-power photons get stored in an important part of the scene, resulting in a nonhomogeneous photon map and the very bright polka dots mentioned above.

Keller and Wald (2000) used Russian roulette to determine which photons to store in regions of low importance. The photons that are stored get increased power to compensate for their low storage probability. Suykens and Willems (2000) used a variation of this where they distributed the power of non-stored photons among the nearest previously stored photons. Both methods suffer from the problem that even though the number of stored photons is dramatically reduced, the number of emitted and scattered photons is still high. In other words, the memory requirements are reduced, but the time spent on the photon tracing pass is still too high.

To summarize, none of the existing methods satisfy all three requirements listed above: reducing the number of emitted, scattered, and stored photons while keeping the bias acceptable and the photon map homogeneous.

5.7.3 Importance computation

The first step in importance-driven photon tracing is to compute the importance distribution in the scene. This can for example be done by emitting importance particles ("importons") from the eye/camera position, tracing them through the scene, and storing them in an importance map. The principles from photon tracing are also used for importon tracing, as described by Peter and Pietrek (1998). Put briefly, importons are emitted from the eye in the directions within the viewing frustum, and scattered through the scene using the material properties to determine scattering type probabilities. Russian roulette is used to terminate the paths.

The importons are stored on all diffuse surfaces they intersect on their paths.

To improve the efficiency of determining the importance at various locations during the photon tracing phase, we then compute importance at all importon positions. This is similar to the precomputation of radiance at photon positions as described in section 5.4. At each importon position, we locate the nearest importons (for example the nearest 50) and use their density to determine the importance at that importon position. These importance estimates are stored, one for each importon. The maximum computed importance is found during these computations, and used to normalize the importance such that the maximum importance in the scene is 1. These precomputed importances make it much faster to determine the importance at various locations during photon tracing.

An interesting detail is that we actually need two kinds of importance: one counting all importons (for the caustic photon map), and one counting only indirect importons, ie. importons that have bounced at least once (for the global photon map). This is because we compute soft indirect illumination with final gathering, so soft indirect photons are not visualized directly.

5.7.4 Importance-driven photon emission and storage

One of the goals of this method is to store few photons (with high power) in the regions with low importance, more photons (with intermediate powers) in the regions of medium importance, and most photons (with low power) in the regions of high importance.

In order to explain our algorithm, and see the differences from previous approaches, it is advantageous to first consider a reordering of the photon tracing method of Peter and Pietrek. A reordered version of their method would divide the emission directions from a light source into some number of strata (corresponding to the finite elements they used), and treat each stratum at a time. First emit a small number of test photons within that stratum, and see if any of them enter into an important region of the scene. The test photons are not stored. If any of these test photons reached something important, relatively many "real" photons are emitted in that set of directions — each photon with low power. If none of the test photons reach anything important, few "real" photons are emitted in that set of directions — each photon with high power. Again, the danger in this approach is that some of the (few) "real" high-power photons actually might make it to one of the important regions in the scene even though none of the test photons did. When this happens, the high-power photon gets stored among the low-power photons in an important region, resulting in a nonhomogeneous photon map and very high incorrect radiance values when the photon map is queried for radiance in the neighborhood of that photon. This shows up as the aforementioned bright polka dots in the image.

Our solution is closely related to this reordered version of Peter and Pietrek's method, but avoids the nonhomogeneous photon map. We also emit a small set of initial test photons — within a stratum of directions from each light source at a time. Each initial photon has a relatively high power. If none of the initial photons from that set of emission directions hit anything important (with importance determined by the precomputed importance of the nearest importon), we store them in the photon map. (This is the typical case.) If some of them hit an important region, we discard all of the photons emitted within that set of directions and emit a new, larger, set of photons with less power each. Some of these new low-power photons will hit unimportant regions; at those locations Russian roulette is used to determine the storage probability and stored power. The low-power photons are always stored (ie. storage probability 1) when they hit the most important regions.

A slight variation on this scheme is to never discard any photons. In the case where some of the initial high-power photons hit a region with high importance, the photons that hit the important regions have their power reduced, while Russian roulette is used to decide which of the photons should be stored in unimportant regions. It would be interesting to measure whether this variation results in more bias of the solution.

5.7.5 Discussion

As with all importance-driven methods, the savings obtained using importance can be arbitrarily high: just choose a sufficiently large test scene with a sufficiently small part visible.

We might miss a small region of high importance if none of the initial photons hit it, but this is a standard problem of Monte Carlo. The probability of this happening is reduced by splitting all photons originating from the same stratum, instead of just the photon that hits a high-importance point. It is probably possible to make an a posteriori check to see if a region of high importance has been missed completely by finding the nearest photon for each importon with high precomputed importance. If the nearest photon is closer to an importon with low importance (i.e. in a different importance Voronoi cell), there is cause for concern.

The method of Suykens and Willems, distributing the power among the nearest stored photons instead of using Russian roulette, could also be used here.

5.7.6 Future work

It should be determined experimentally how the bias and noise of this method compares to the bias and noise of the method of Peter and Pietrek. It should also be determined how the bias compares to a an alternative version of their method where the "stray" high-power photons that enter important regions are simply stored with reduced power.

The method described here could be extended to photon tracing in participating media.

5.7.7 References

The most relevant previous work on importance and photon tracing is described in the following papers:

- Brian E. Smits, James R. Arvo, and David H. Salesin. "An importance-driven radiosity algorithm". *Computer Graphics (Proceedings of SIGGRAPH 92)*, pp. 273–282. ACM, 1992.

- Henrik W. Jensen. "Importance driven path tracing using the photon map". *Rendering Techniques '95 (Proceedings of the 6th Eurographics Workshop on Rendering)*, pp. 326–335. Springer-Verlag, 1995.

- Ingmar Peter and Georg Pietrek. "Importance driven construction of photon maps". *Rendering Techniques '98 (Proceedings of the Ninth Eurographics Workshop on Rendering)*, pp. 269–280. Springer-Verlag, 1998.

- Alexander Keller and Ingo Wald. "Efficient importance sampling techniques for the photon map". *Proceedings of the Fifth Fall Workshop on Vision, Modeling, and Visualization*, pp. 271–279. IEEE, November 2000.

- Frank Suykens and Yves D. Willems. "Density control for photon maps". *Rendering Techniques 2000 (Proceedings of the Eleventh Eurographics Workshop on Rendering)*, pp. 11–22. Springer-Verlag, 2000.

5.8 Conclusion

This note has presented seven optimizations and improvements of the photon map. Simple optimizations such as the ones presented here make the photon map method significantly faster. More optimizations and improvements await discovery — we've only scratched the surface so far. So let's scratch deeper!

Acknowledgements

Thanks to my colleagues at Pixar and former colleagues at Square USA for many interesting discussions regarding global illumination and the photon map method, and for emphasizing the need for fast, flexible, and user-friendly global illumination methods in "real world" movie production work.

Chapter 6

Photon Mapping in Kilauea

Toshiaki Kato
Square USA Honolulu Studio

Hitoshi Nishimura, Tadashi Endo, Tamotsu Maruyama,
Jun Saito, Motohisa Adachi

Abstract

Kilauea is a revolutionary massively parallel global illumination renderer based on Monte Carlo ray tracing with photon mapping incorporated as the main algorithm. This course note aims to explain photon mapping in Kilauea from two aspects:

- Various tips and tricks for using photon mapping in the production work
- Speeding up photon mapping, especially final gathering

6.1 Introduction to Kilauea

The Kilauea Research Project took place in the R&D division of Square USA's Honolulu Studio from March 1998 till March 2002, to design and implement a ground-breaking high-end renderer.

6.1.1 Goals

Two ultimate goals were set when the Kilauea Research Project began. The Kilauea renderer aimed to render:

1. High quality images using global illumination;
2. Extremely complex and large scenes.

While achieving these two goals, the Kilauea renderer was required to be stable and flexible enough for the use in the production environment.

6.1.2 Concept

The global illumination computation methodology is an actively debated subject and various ideas are proposed[Goral84][Ward88][Jensen95a]. Kilauea chose a ray tracing based global illumination methodology because this offers one robust algorithm which handles diverse scene types which allows to keep the overall system architecture clean and flexible. Kilauea is fundamentally designed to compute global illumination using final gathering and photon mapping, which are based on Monte Carlo ray tracing.

However, ray tracing inherently requires large amounts of memory because the entire scene data need to be accessible. Many methods have been proposed to achieve our second goal to ray trace extremely complex scenes, such as in [Pharr97] which involves paging in only a portion of the scene from disk. Kilauea's approach for rendering large and complex data tries to be as robust as possible. An optimized algorithm for a very specific case which can handle certain extremely large scenes very efficiently, but cannot render other scenes, is undesirable. Our algorithm needed to be able to give freedom in shading implementation (such as no limitation on how the colors from different ray generations can be composed), run just like conventional ray tracers, but still be able to render extremely large scenes.

Kilauea takes advantage of parallel processing to obtain the massive computational (CPU and memory) resources required to achieve the two goals.

Cost-efficient Linux PCs were connected with low cost 100 BaseT Ethernet. Kilauea is designed to be a parallel ray tracer based on message passing, meaning that Kilauea processes running on these machines cooperate to render images.

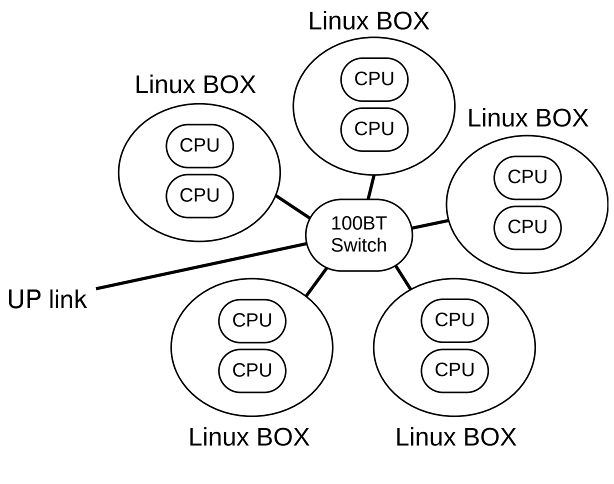


Figure 6.1: Linux cluster

One of the characteristics of Kilauea is its scalability (see section 6.1.4). If just using one machine is not fast enough, then use two machines to double the speed, three machines to triple the speed, and so on. The scalability applies to the scene size as well. If the scene does not fit in the memory of one machine, then split the scene to two machines. If two is not enough, then three, and so on. This property is especially powerful, considering the sharp price drop and performance increase of PCs.

With the concept described so far, Kilauea, the massively parallel global illumination renderer, went through research and development for four years.

For more on the Kilauea architecture, please refer to SIGGRAPH 2001 course notes #40 and the book "Practical Parallel Rendering" from A K Peters. For the latest improvements to Kilauea, please refer to SIGGRAPH 2002 course notes on "Practical Parallel Rendering".

6.1.3 Target quality

Kilauea's goals are to design and implement robust algorithms for global illumination computation and extremely large scene manipulation, and parallel processing is the approach to achieve them. The target image quality should also be defined in order to determine how much optimization is necessary to accomplish the demanded quality in a practical computation time.

Because of Kilauea's distributed processing nature, there are many implementation specific issues which distinguish Kilauea from other renderers. However, Kilauea is essentially a ray tracer and the basic quality control idea applies to Kilauea as well. The rule is simple: for high quality images, high sampling rate is necessary. This is not limited to primary ray sampling, but also applies to reflection, refraction, shadow, and final gather rays. Final gather ray tracing especially dominates the computation time in the Kilauea rendering, and the sampling rate here is significant in the final rendering speed and quality control.

We defined that the image quality obtained by 65×65 final gathers per pixel is the target quality (figure 6.2). Almost for all scenes, this $65 \times 65 = 4225$ final gathering per pixel results in an exceptional quality – perhaps an overkill in many. However, with the consideration of the quality required for making animations, Kilauea is optimized to achieve this much quality.

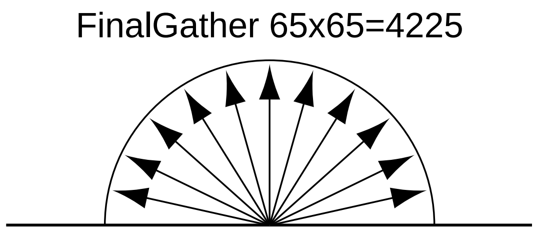


Figure 6.2: Target final gather quality

6.1.4 Parallel performance

The scalability of Kilauea is highlighted in this section, supported by timing test results from typical cases. Roughly speaking, there are two cases to be considered.

1. The scene fits in one machine
2. The scene must be distributed to several machines

For both of these cases, how Kilauea's rendering speed increases as more machines are added is discussed. Maximum of 18 nodes of dual Pentium III 1 GHz with 512 MB of memory are used in the experiment.

Case 1: scene fits in one machine

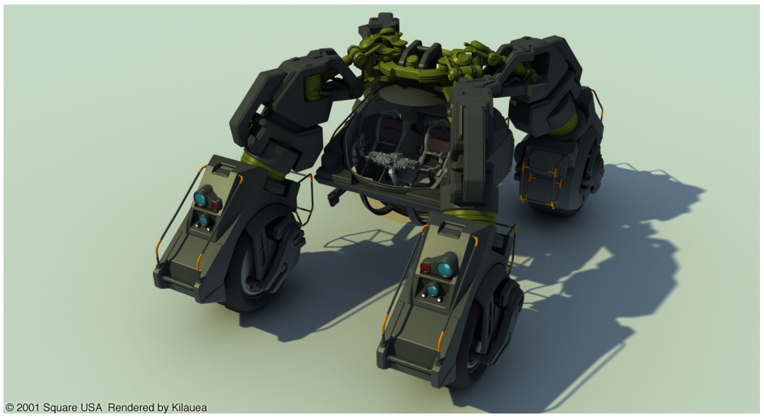


Figure 6.3: Quatro (700,233 triangles, one directional light, one sky light)

This scene is small enough to fit in the memory of one machine, and each machine involved in the rendering holds a copy of the entire scene. Two timing results are presented in the table 6.1.4 and the graph in figure 6.4. One is the time for the entire rendering, including time for Kilauea to boot and reading in the scene. Another is the time for the actual rendering, which mostly consists of ray tracing and shading.

In the actual rendering stage, Kilauea is implemented with full attention to parallel processing. Thus, the graph for timing of only the ray tracing and shading exhibits super linear scalability. This super linearity is speculated to be caused by caching at various levels. However, when all initialization stages are included, the graph gradually falls off from the optimal curve because many of the pre-rendering stages are not suitable for parallel processing, or the parallel implementation is not thoroughly considered.

A Kilauea process does not terminate after rendering one frame. Instead, it can proceed to read in the next frame, or start another rendering immediately[S2000Course40].

# of machines	All (hr:min:sec)	Ray tracing (hr:min:sec)
1	1:30:23	1:13:15
2	0:45:23	0:35:40
3	0:30:59	0:23:42
4	0:23:48	0:17:42
5	0:19:37	0:14:10
6	0:16:48	0:11:47
7	0:15:05	0:10:05
8	0:13:32	0:08:49
9	0:12:03	0:07:52
10	0:11:04	0:07:01
11	0:10:15	0:06:19
12	0:09:33	0:05:45
13	0:09:03	0:05:17
14	0:08:41	0:04:53
15	0:08:23	0:04:32
16	0:07:52	0:04:14
17	0:07:35	0:03:59
18	0:07:19	0:03:45

Table 6.1: Quatro rendering time

The user can repeatedly tune shading parameters and render again to avoid Kilauea's initialization time. For this sort of operation, only ray tracing and shading are performed, and it thus fully benefits from the super-linear scalability of Kilauea.

Case 2: scene must be distributed to several machines

For the purpose of experiment, the vehicles in the scene are fully copied, not instantiated. The scene is too large to fit in the memory of one machine, and thus is distributed to two machines. For rendering this scene, the minimum unit of the rendering is a group of two machines. In the experiment, the machines are increased by sets of two machines.

The characteristic of the timing results in table 6.1.4 and the graph in figure 6.6 are the same as in the single machine case; the actual rendering stage exhibits super-linear performance and the total rendering time curve falls gradually off from the optimal curve.

6.2 Photon map rendering techniques

Many images were rendered using Kilauea to experiment how it can be put into the actual CG production pipeline. In the course of trial and error in making these test

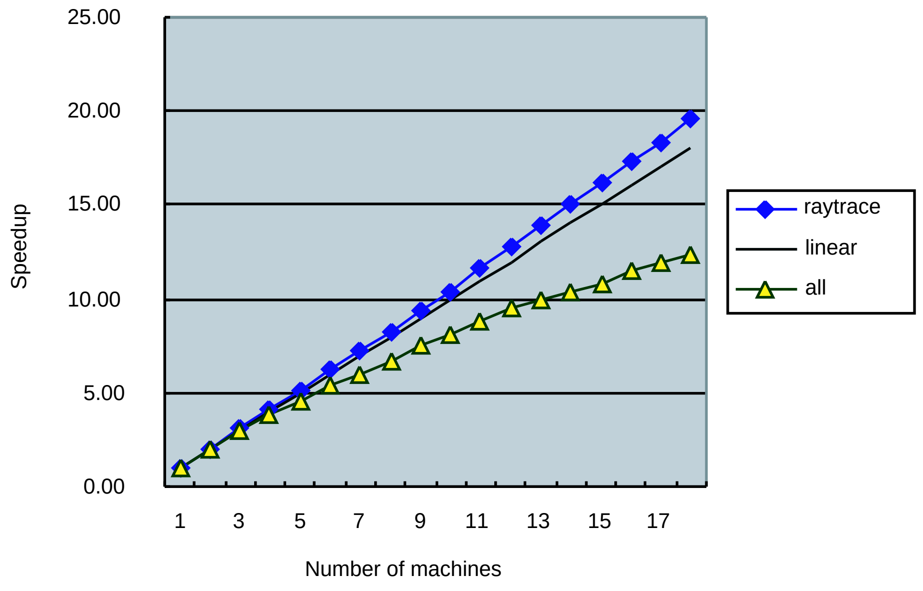


Figure 6.4: Number of machines vs. speed-up ratio

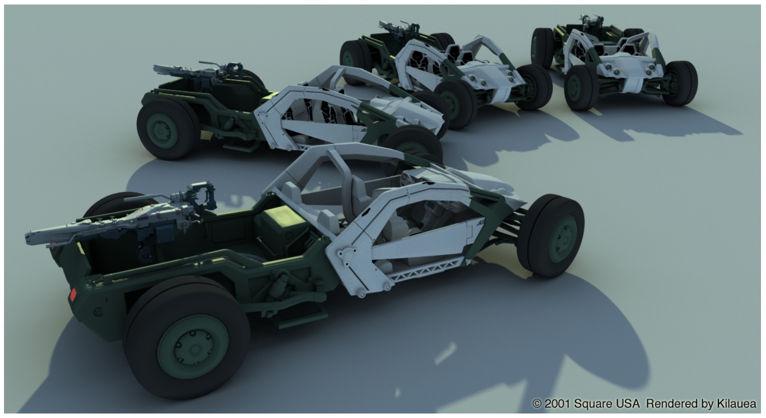


Figure 6.5: nJeep4 (2,859,636 triangles, one directional light, one sky light)

images, many tricks and techniques distinctive to photon map and global illumination rendering were developed.

6.2.1 Global photon map techniques

A typical photon map implementation has two photon maps: a global photon map and a caustic photon map. The images rendered by only referencing the global photon map (zero or negligible amount of caustic photons in the scene) is the most robust and efficient case in the Kilauea rendering. This section will cover some

# of machine groups	All (hr:min:sec)	Ray tracing (hr:min:sec)
1	1:35:45	1:23:26
2	0:47:46	0:41:19
3	0:34:08	0:27:32
4	0:25:38	0:20:37
5	0:21:00	0:16:25
6	0:17:50	0:13:29
7	0:15:30	0:11:26
8	0:13:50	0:09:55
9	0:12:38	0:08:47

Table 6.2: nJeep4 rendering time

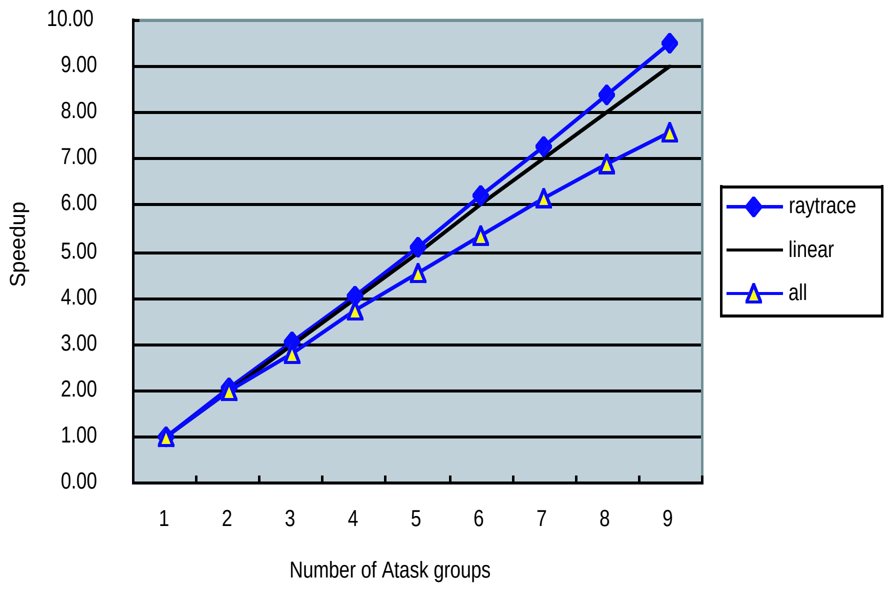


Figure 6.6: Number of machine group vs. speed-up ratio

techniques regarding the global photon map.

Sample images

The photon map method renders the following scenes in figures 6.7, 6.8, 6.9, and 6.10 with convincing quality. Expected global illumination effects such as color bleeding are observed nearly everywhere in the scene.

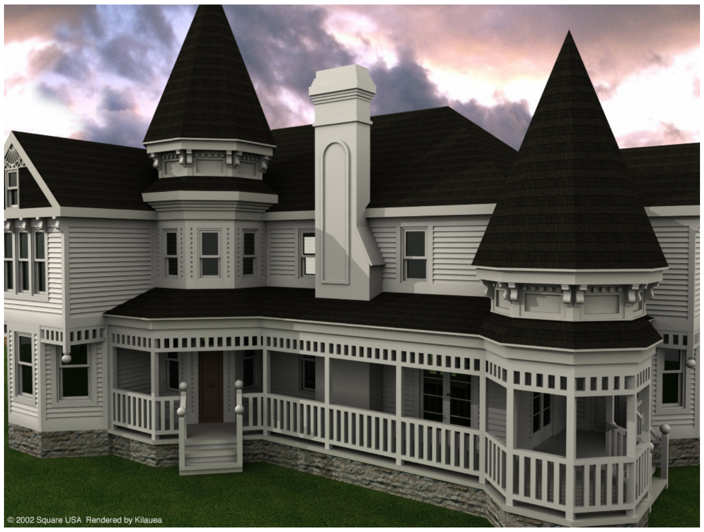


Figure 6.7: Victorian house

Figure 6.8: FA Loader

Secondary final gathering

Occasionally, problems typically seen in image 6.11 happens. This is precisely not a problem with photon mapping, but rather a problem specific to final gathering.

Final gathering computes the total power of incoming light from the surrounding surfaces to a point on the object surface. More specifically, final gather rays are shot from the point on the surface to all hemisphere directions to integrate the incoming energy to that point (figure 6.12).

There are many implementation ideas for computing final gathering. The sim-

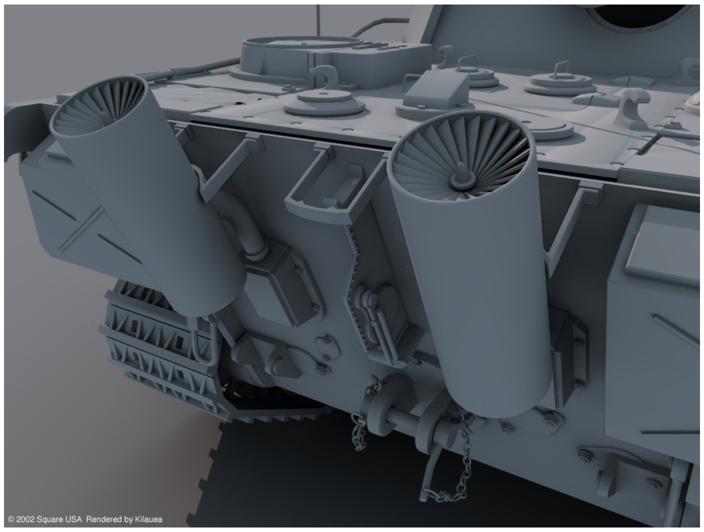


Figure 6.9: Panzer

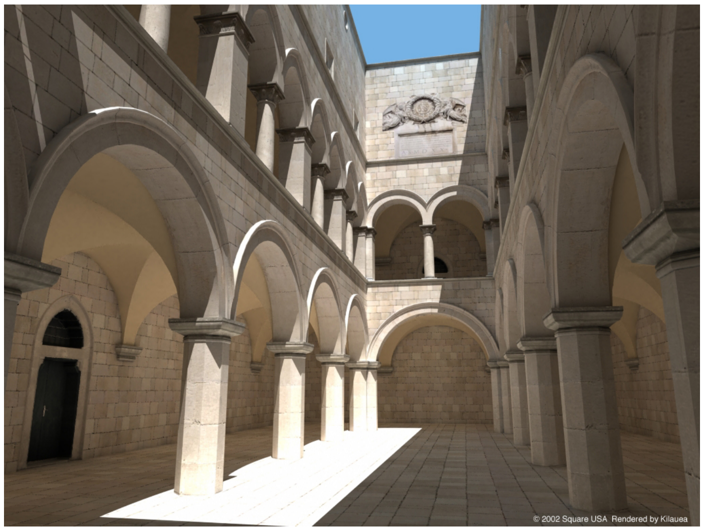


Figure 6.10: Sponza

plest idea is to simply shoot final gather rays randomly to all hemisphere directions. Another idea is to gradually increase the number of final gather rays by shooting small amount of final gather rays first and then use their results to adaptively determine where to shoot more final gather rays.

Final total power of light (irradiance from diffuse surfaces) is determined by the number of final gather rays and how they are shot. When more final gather rays are shot, the computed total power gains more accuracy.

Close examination of the artifacts on the image 6.11 reveals that high percent-

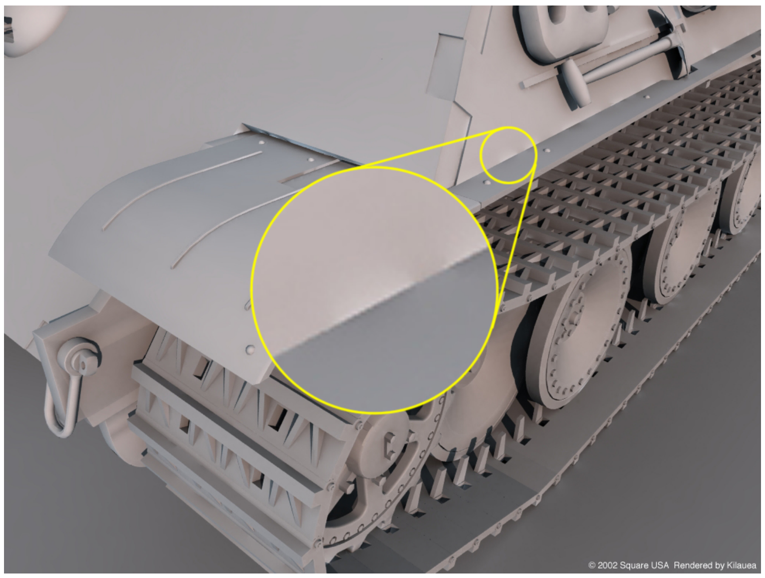


Figure 6.11: Panzer problem picture

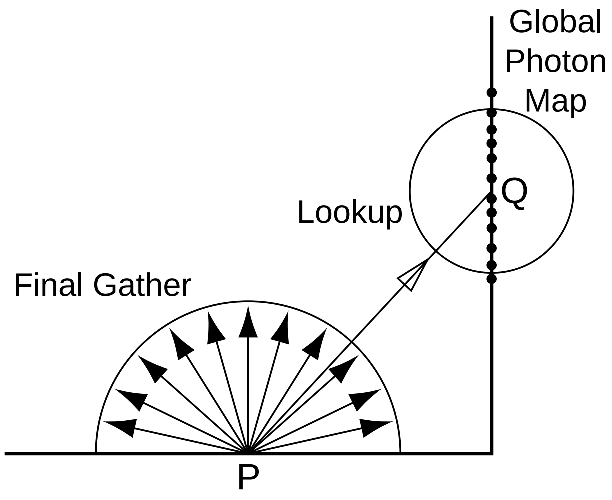


Figure 6.12: Global photon lookup

age of final gather rays shot from that problematic region hit another surface only after traveling a very short distance. When many of the final gather rays hit a very nearby surface, the accuracy of final gathering at that point does not improve by just increasing the number of final gather rays.

The problem is essentially caused by the global photon look-up executed at the point where the final gather ray hits (see section 6.2.3). The photon map is looked up to estimate the irradiance value at this point, but the problem occurs when the precision of this irradiance value is insufficient.

Photons emitted from the light sources get stored in the global photon map when they hit a diffuse surface (see section 6.2.3). Because only a finite number of photons exist, there might not be enough photons to keep the satisfactory irradiance precision in certain regions in the scene. When the accurate irradiance is requested at complex and crowded regions, as in the problematic regions in the sample image, there is often not enough information in the global photon map. The final gathering result computed mostly from inaccurate irradiance values will turn out to be inaccurate as well, leading to the degradation of the entire image quality.

One approach to overcome this problem is to increase the number of emitted photons, but this will not be a fundamental solution (there will always be regions where photons hardly reach).

There is a better solution which drastically improves the image quality. Figure 6.13 illustrates it. If the final gather ray shot from point P hits point Q on a very nearby surface, computing the irradiance by looking up global photon map is given up. Instead, additional final gathering is conducted at this point. This is called *secondary final gathering*. At each point where the final gather ray from Q hits a surface, global photon map is looked up to compute the irradiance, just like in the regular final gathering. The global photon look-up in the secondary final gathering is expected to have low precision because P and Q are close to each other, but since it is a secondary bounce, it has enough precision to improve the result at P. In a practical sense, final gather levels more than two are probably unnecessary. Indeed, Kilauea does only up to secondary final gather and is still able to remove these artifacts. Kilauea lets users define the distance to switch between regular photon look-up and secondary final gathering.

Image 6.14 is generated with the secondary final gathering turned on. As observed, the artifacts are completely gone.

Photon look-up problems

Photon look-up is a crucial operation in photon mapping, and some careful considerations are needed in this step. After final gathering, photon look-up is executed at where the final gather ray hits an object. In the direct illumination computation, the caustic photon map is looked up. The distance in the world coordinates is used to look up all photons within a certain distance from that point. In certain situations, searching photons just by distance gives undesired outcome. Some tricks to avoid this problem are listed in the following sections.

Solution A: storing incident angle

Simple photon search using distance picks up some undesired photons, as shown in 6.15. Photons stored on the back face should be ignored, but the photon look-up based only on the distance cannot exclude them.

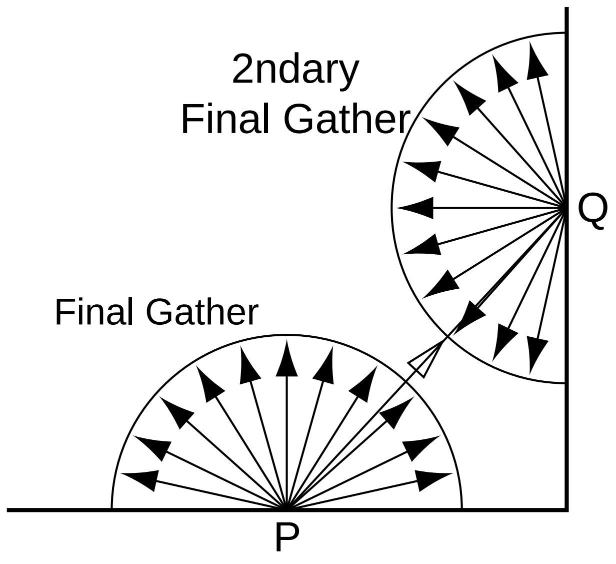


Figure 6.13: Secondary final gather

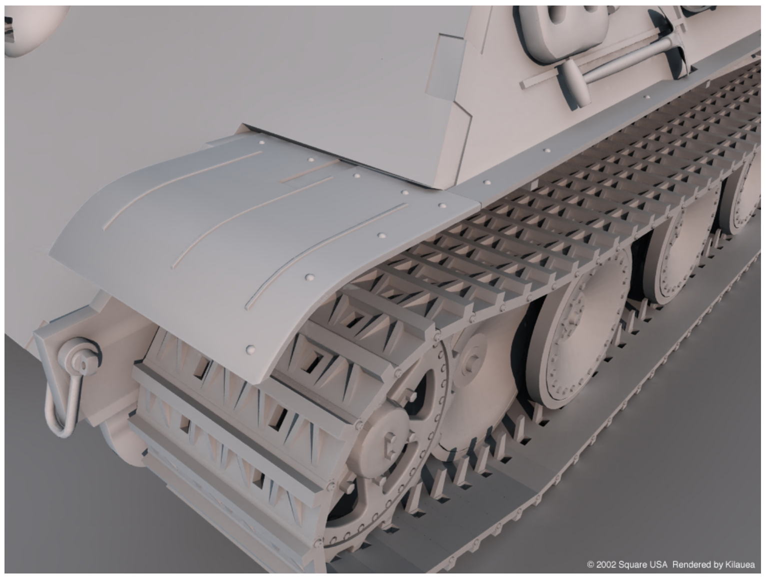


Figure 6.14: Panzer

This can be easily avoided by storing the incident direction with the photon data. For the look-up operation from point P, the normal vector at P is compared with the incident directions of the neighboring photons. This way, only the photons from the desired side of the surface can be selected. Accurate direction is not necessary for this purpose, and thus the direction can be stored with less precision than 32bit float for each of (x, y, z) to save the size of the photon map[Jensen01].

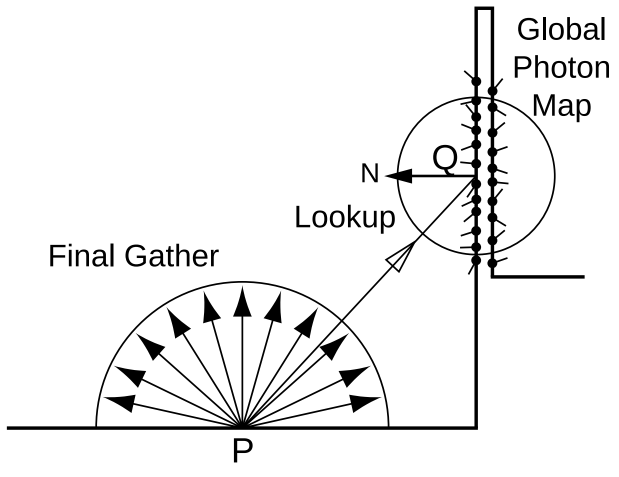


Figure 6.15: Back face photon look-up problem

Solution B: add thickness to objects

Figure 6.16 shows the case where the photon look-up solution A fails. Even if the direction of the photons are considered, undesired photons are selected.

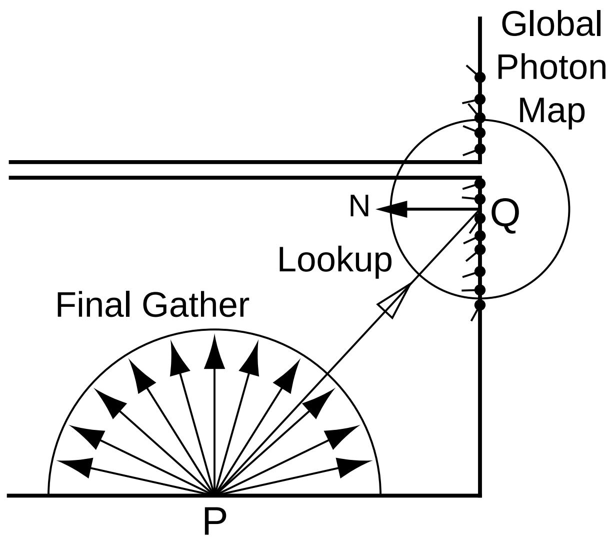


Figure 6.16: Occluded photon look-up problem

By adding more information to photons, this problem may be avoided. However, full automation of this is difficult without considerably increasing the computational cost. Essentially, this problem can be solved by modifying the scene to avoid the situation as in figure 6.16. Objects such as walls and ceilings can be modeled with some thickness to avoid photons from getting stored on the back face. This is the most intuitive and robust approach to avoid the problem.

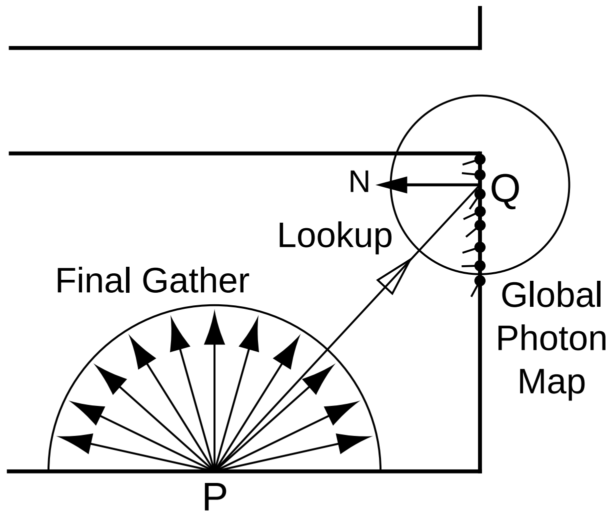


Figure 6.17: Adding thickness

Solution C: carefully plan how to emit photons

When the camera is set in a room illuminated by light from outside and the scenery outside is unimportant, there is no need to actually emit photons from outside (figure 6.18). If photons are shot in this straightforward manner, they may end up getting stored on the back side of the walls and the ceiling.

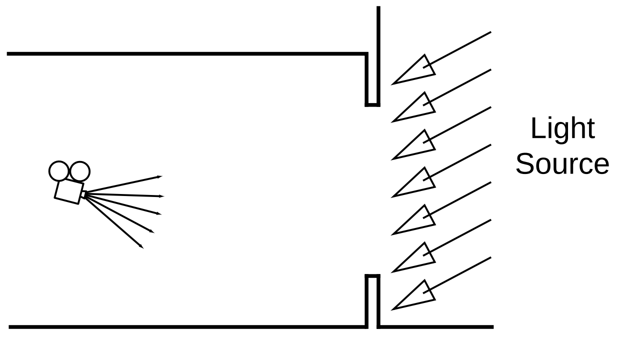


Figure 6.18: Lighting from outside

Instead, the light source should be placed at the location of the window to illuminate the room (figure 6.19). The problem is then reduced to the level of how to write such light shader, instead of the issues of photon look-up process, which is a core part of the renderer.

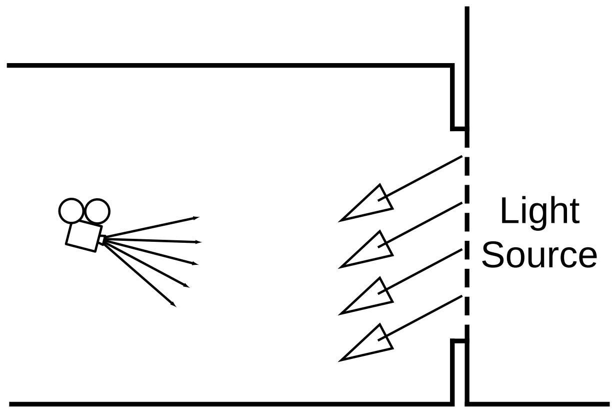


Figure 6.19: Lighting from window

Caution when creating photon map scenes

Scenes created for direct illumination renderers only need careful attention to visible portions. This in turn means that invisible portions can be drastically simplified to save memory. The basic idea for reducing the scene size applies even to the photon map rendering. However, the artist must carefully consider what will happen in the photon look-up stage when the scene is simplified. Too much simplification may result in undesirable photon look-ups.

Other than the solutions provided here, more photon look-up tricks may need to be developed to improve the usability of photon mapping in the actual production.

Sample images of photon look-up problem and solution

These images of FA Loader show how adding thickness can prevent false photon look-up.

6.2.2 Weakly focused caustic problem

One of the distinguishing effects obtained from using photon mapping is caustics. Obtaining high quality caustics rendering is often very difficult, especially when the caustic is not intense enough.

Caustics with strong focus

Figure 6.22 shows a typical caustic effect which can be rendered using a straightforward photon map implementation. This image clearly illustrates the power of the photon map algorithm.

Strong caustic effect is observed at the center of the shadow. For these types of scenes, simple caustic photon look-up gives satisfactory results because the density of the caustic photons significantly varies between the region with light focused

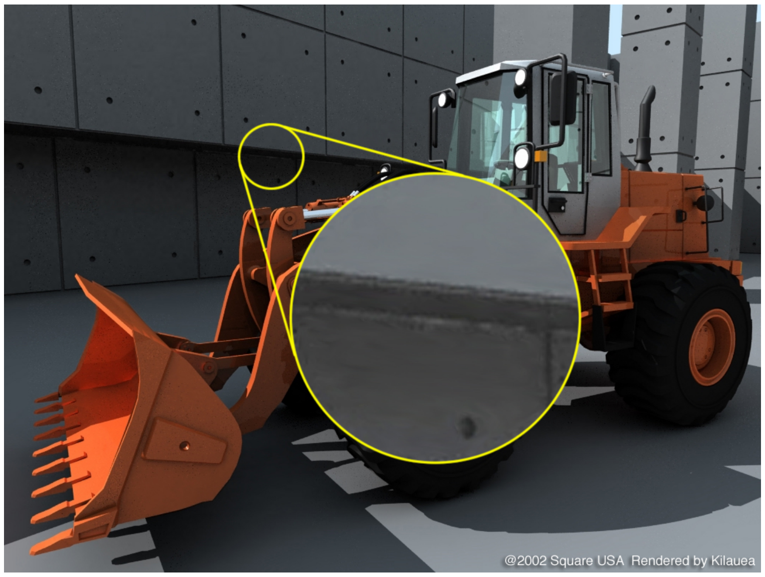


Figure 6.20: FA Loader: NG

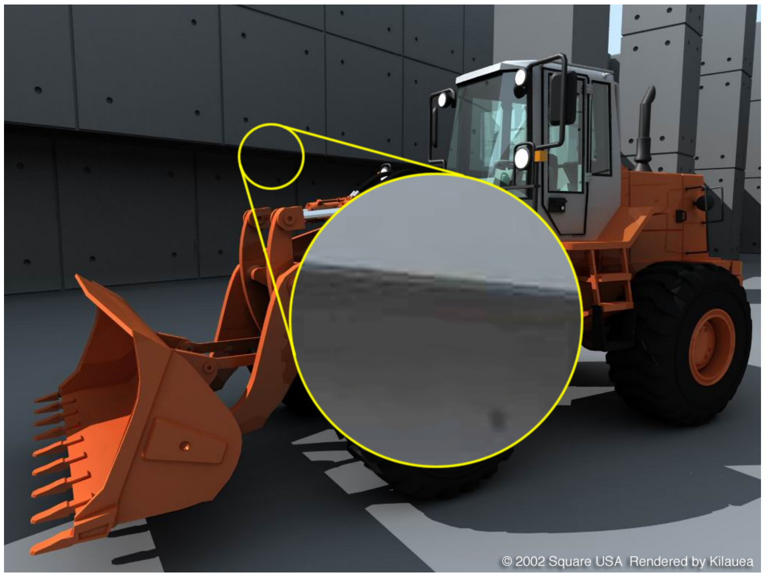


Figure 6.21: FA Loader: OK

and not focused and simple photon look-up using the distance and the number of photons is enough. As a result, the photon mapping renders convincing caustic effect with not-so-many caustic photons for these sample scenes.

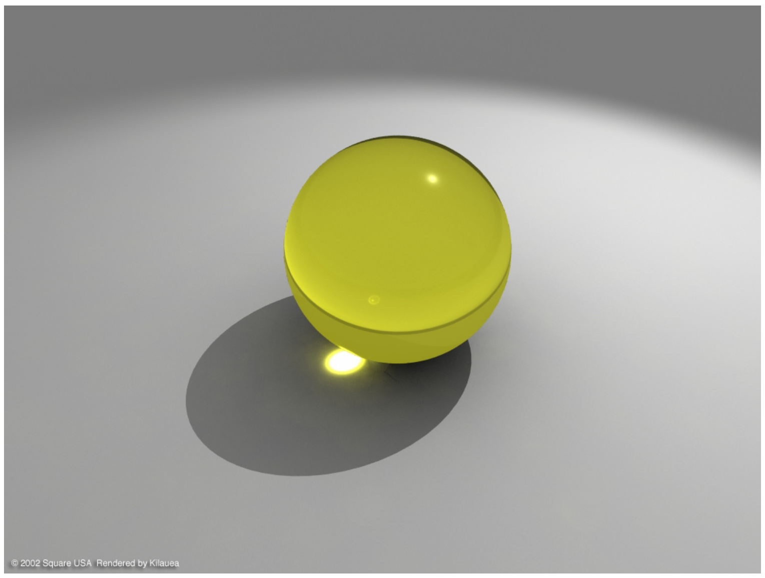


Figure 6.22: Sphere

Caustics with weak focus

However, some scenes require caustic effect with less light intensity. Figures 6.23 and 6.24 show such cases. Unlike the previous successful images, the caustic effect in these images are very noisy. Also, the edge of the caustic should be crisp, but is blurred in these samples.

To generate the crisp caustic effect, applying various filters when doing the caustic photon look-up may improve the result (see [Jensen01] about applying cone filter). However, applying edge enhancing filters only improves the edge of caustic and does not remove the noise in the caustic region. Dynamically changing the filter type depending on the region of the caustic may be possible, but coming up with an algorithm which works for every case is not so easy.

Solution: selective photon storing

To improve the quality of this caustic with less intensity, storing more caustic photons along with the filtering technique is necessary. The first thing to try is to increase the number of photons emitted. However, this may impose a memory capacity problem.

When a photon is emitted from the light source, there is a chance for it to be stored in the global photon map and/or the caustic photon map (or neither of them). Where it will be stored is only known after the photon tracing. In the previous case, only the number of total photons emitted from the light source is specified, and ended up with insufficient amount of caustic photons.

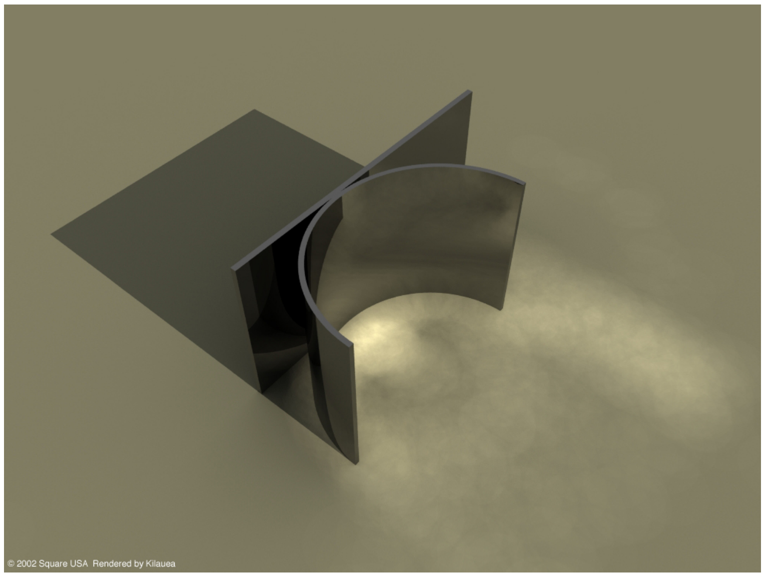


Figure 6.23: Caustic K (100,000 total photons, 926 caustic photons, 60,722 global photons)

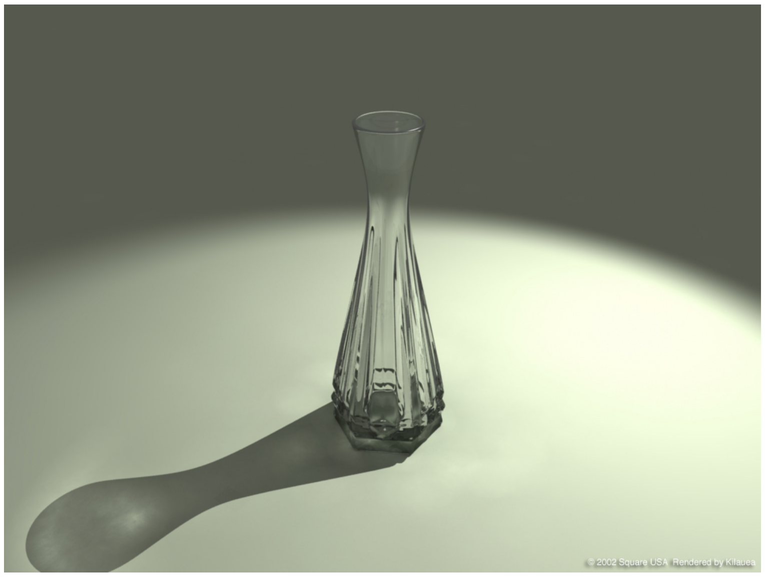


Figure 6.24: Vase (100,000 total photons, 7,157 caustic photons, 115,589 global photons)

If the number of emitted photons is increased ten times, then the amount of the caustic photons will be approximately ten times bigger. However, the amount of global photons will be ten times larger as well. This is a waste of memory, since the amount of global photons was sufficient for the quality of the image. The solution is to increase the number of caustic photons without increasing the number of global photons.

In order to achieve this, when the photon store to the global photon map occurs, only certain percentage of them are actually stored. All caustic photons are stored as usual. Simply reducing the amount of global photons will decrease the total power, so the power of global photons to be stored must be appropriately multiplied according to the percentage.

This causes the amount of caustic photons to increase compared to the global photons. The final amount of caustic photons can be controlled by the amount of total photons emitted from the light sources and the percentage to store global photons. This technique is truly powerful when rendering high quality caustics with less intensity.

Results

Figures 6.25, 6.26, and 6.27 are the sample images to show how the above technique improves the quality.

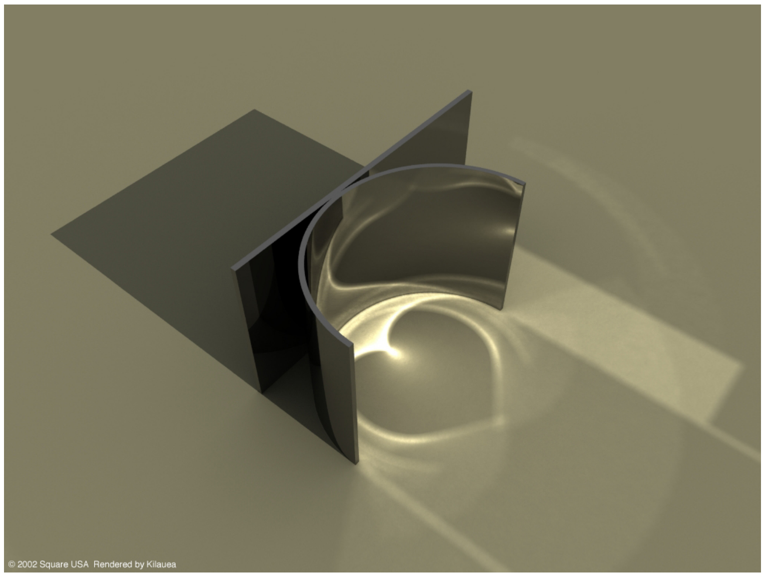


Figure 6.25: Caustic K (10,000,000 total photons, reduction ratio 95%, 92,324 cautic photons, 152,343 global photons)

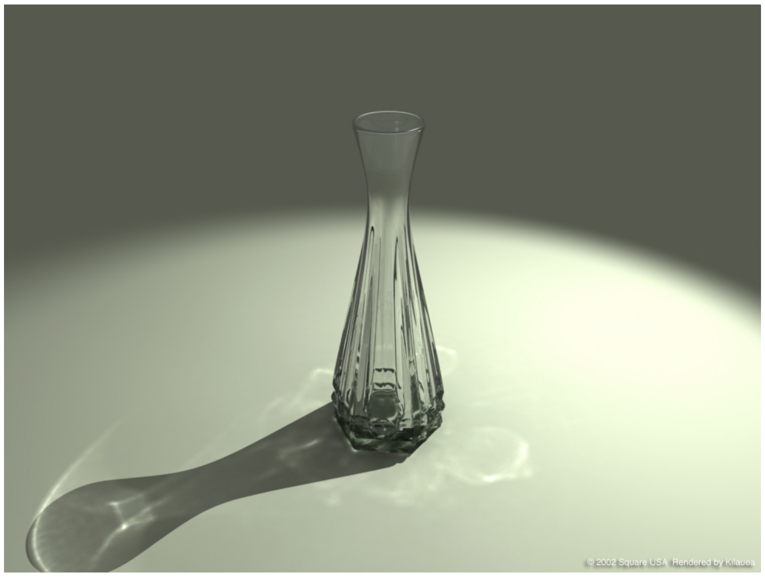


Figure 6.26: Vase (10,000,000 total photons, reduction ratio 99.5%, 701,103 caustic photons, 755,301 global photons)

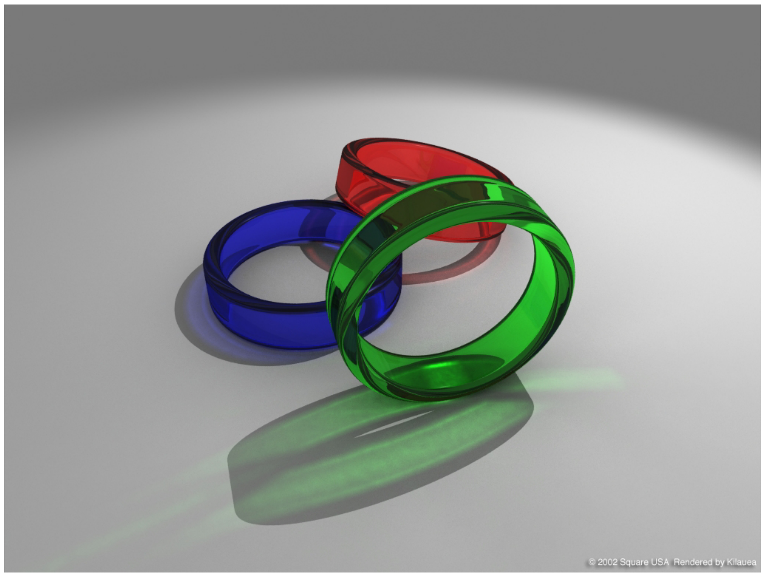


Figure 6.27: Rings (8,000,000 total photons, reduction ratio 99.5%, 894,317 caustic photons, 939,300 global photons)

6.2.3 Photon map rendering and transparent objects

How caustics are handled in the photon map rendering has another inherent problem related to transparent objects. To understand the problem, this section first explains how the operations related to the problem (photon tracing, photon storing, shadow calculation, and final gathering) are implemented in a typical photon map renderer [Jensen01]. Then the problem caused by this typical implementation and its solution are presented.

Photon tracing and storing

Photons are emitted from the light source and are traced through the scene. If a photon hits a surface with non-zero diffuse component, it is always stored in the photon map. When they hit an object, reflection or refraction (transmission) occurs at its surface. Depending on the property of the surface, photons get stored and reflected/refracted for further photon tracing. Photons eventually get absorbed at some surfaces or lost to the infinite distance.

On the diffuse surface, the photons are stored to two photon maps (global photon map and caustic photon map) depending on where they came from. Photons bounced from the diffuse objects are stored only to the global photon map. Photons from the mirror and glass objects are stored to both global photon map and caustic photon map. As a result, the caustic photon map becomes a subset of the global photon map – all photons in the caustic photon map exist in the global photon map.

The caustic photon map is looked up by the shading computation of the diffuse surfaces that direct illumination rays (rays from camera) hit during the final rendering. The global photon map is not referred at this point. Instead, it is looked up when the final gather rays hit the diffuse surfaces. The caustic photon map is not looked up in the final gather stage.

Because the global photon map includes all photons in the caustic photon map, the correct irradiance at the point final gather ray hit is obtained by looking up only the global photon map. This property speeds up the final gathering process.

Here is the step-by-step explanation of the direct illumination computation. When the direct illumination ray hits the point P, the shading computation is executed at P. In the computation, to determine if P is in the shadow, a shadow ray is shot to the light source. If P is found to be in the shadow, the direct illumination value is 0. The caustic photon map is looked up at P to find out total power of photons from the reflective and refractive surfaces.

Final gathering at P is performed to compute the irradiance from surrounding diffuse surfaces.

The reason for performing final gathering instead of just looking up the photon map is due to the precision. Basically the irradiance obtained from the photon map has low frequency noise and does not have enough precision to be used for the direct illumination computation. Thus, final gathering, which computes the irradiance from diffuse surfaces more accurately, is necessary. However, to compute the irradiance at the point where the final gather ray hit, not so much precision is required (except for some special cases, as mentioned in section 6.2.1) and therefore just a single look up to the global photon map suffices [Jensen01].

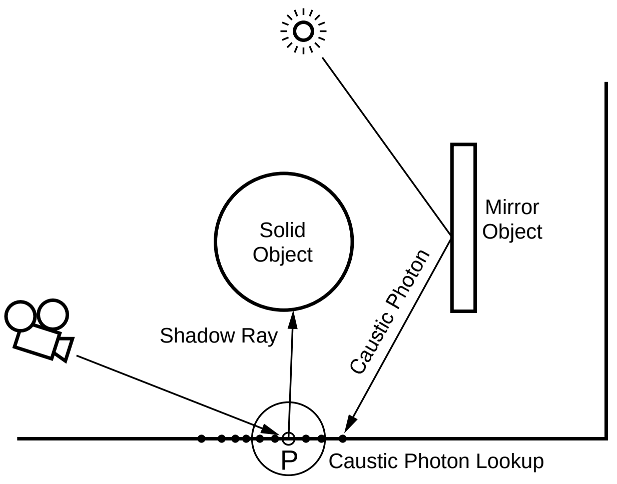


Figure 6.28: Shadow trace and caustic photon lookup

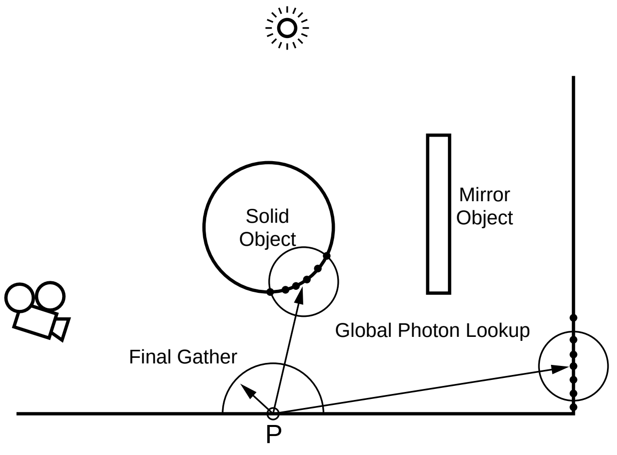


Figure 6.29: Final gather

Shadow computation with photon mapping

The shadow computation when doing a photon map rendering is generally processed in the direct illumination computation. More specifically, to determine that a point P on an object surface is in the shadow, a shadow ray is shot from P to a light source. If the ray hits an object, P is considered to be in the shadow. This process is done in the surface shading computation in the final rendering. This is basically the same as in the classical ray tracing. The merit of this conventional method is its relatively low computational cost for its accuracy.

The transparent objects must be considered carefully, however. Classical ray tracing cannot correctly compute the shadow from objects with some transparency. To fake it, the shadow ray tracing from P is forced to continue on its path when it

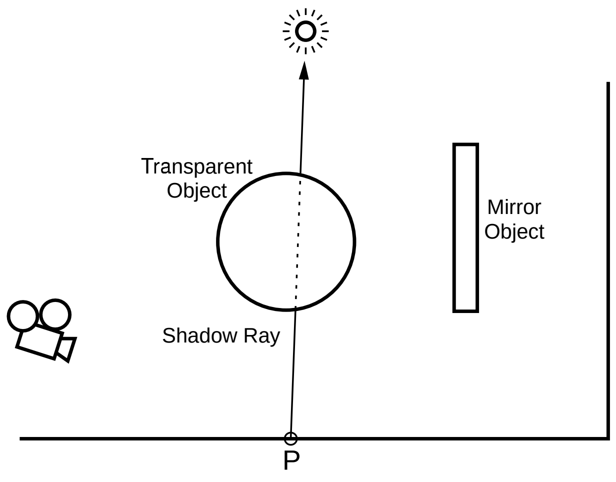


Figure 6.30: Fake shadow

hits a transparent object (figure 6.30). After the ray reaches the light source, the light contribution is calculated from the transparency of objects that the shadown ray went through.

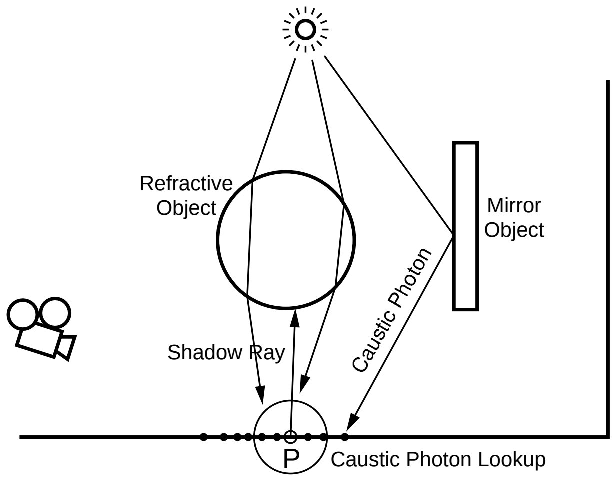


Figure 6.31: Caustic shadow

The photon map rendering provides a more accurate solution. When the shadow ray from P hits an object, the tracing is discontinued, no matter what transparency the object has. This determines that P is in the shadow area, and the direct illumination value at P for this light source becomes 0. In reality, if the object which the shadow ray hit had some transparency, P should be somewhat illuminated by the light. P could also be illuminated by reflection from mirror objects. To correctly handle these cases, the caustic photon map is looked up at P (figure 6.31). Caustic photons which traveled through transparent objects and reflected from mirror surfaces contribute to the illumination at P. This allows the computation of complex light behaviors from transparent and reflective objects. This in turn means that, for correct global illumination computation, photons which refracted and transmitted through transparent objects and reflected from mirror objects must be stored in the caustic photon map, and the final direct illumination shading must correctly reference them.

Final gathering

To summarize, the computation at P at the final rendering stage can be broken down to three processes: computing direct illumination shading color and shadow, and final gathering. The final gathering is required to compute the power from surrounding diffuse surfaces (figure 6.32).

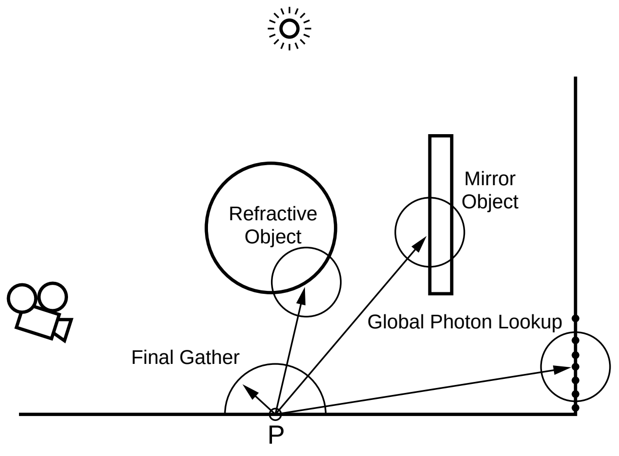


Figure 6.32: Final gathering

Final gather ray tracing is in principle the same as ray tracing. However, there are two things to watch out. First, when the final gather ray hits an extremely close diffuse surface, secondary final gathering must be executed to obtain enough precision (see section 6.2.1). Second, when the final gather ray hits transparent or mirror objects, the tracing is discontinued and 0 is returned as a result.

Problem with photon mapping and transparency

In theory, photon mapping seems to be able to correctly render scenes with transparent objects. However, creating images like figures 6.33 and 6.34 is not as easy as it seems.

From the view of the photon map algorithm, these scenes contain critical issues regarding how photons which passed through the transparent object should

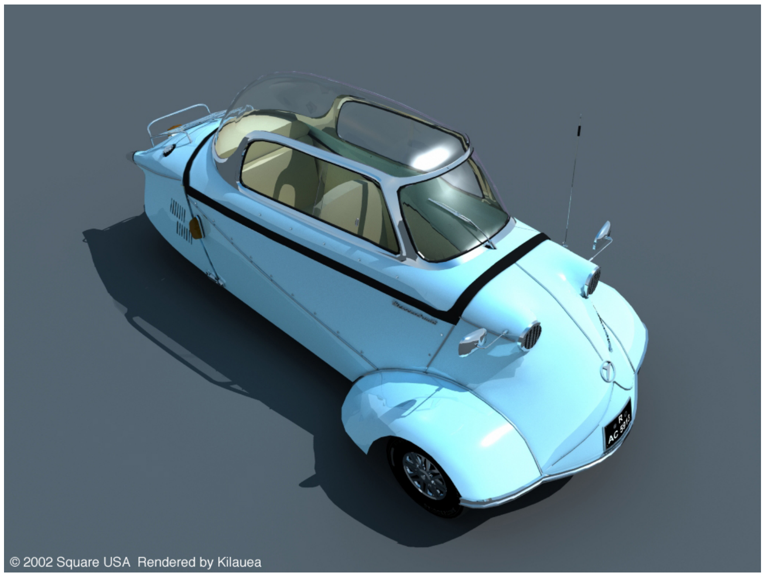


Figure 6.33: Messerschmidt

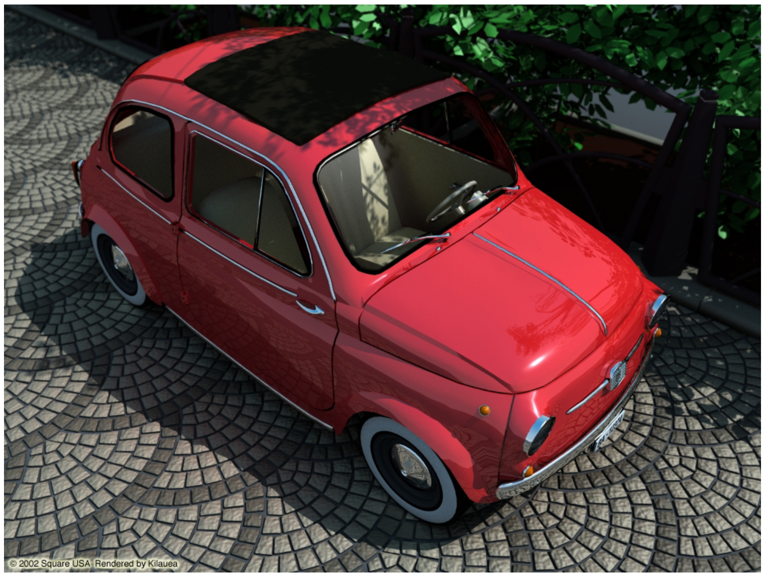


Figure 6.34: Fiat 500L

be handled. The global illumination computation of the light coming in through windshield landing on the seat raises a very complex problem.

Let us first consider what happens during photon tracing and storing. A photon which reached the windshield will be refracted and transmitted through the glass, entering the inside of the car and eventually hitting the seat. Because the photon is from a transparent object, it is stored to the caustic photon map as well as the global photon map (figure 6.35).

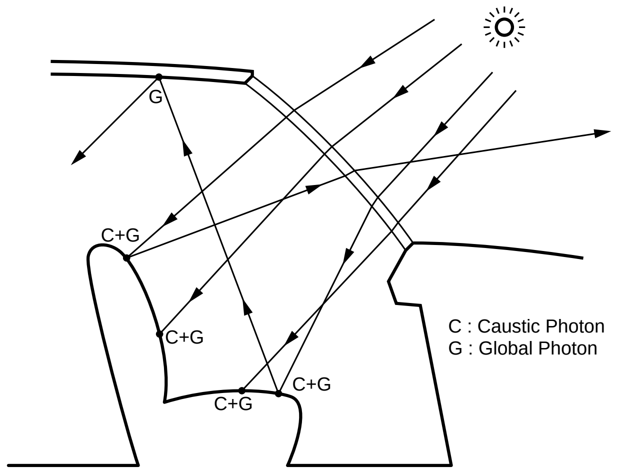


Figure 6.35: Photon storing and glass window

The car is completely confined and has no light source inside. Thus the only possible light source is through glass. In other words, the interior of the car will only be illuminated by caustic photons.

Now let us have a look at how the shading computation takes place inside the car. Various lighting effects such as the color bleeding from diffuse surfaces in the car, shadow cast from the roof of the car, and the global illumination effects from outside the car are all expected to happen inside the car.

At point P on the seat, a shadow ray is shot to the light source outside to determine if it is in the shadow. Because the car is completely confined, the ray will always hit some object. Therefore the direct illumination values to all light sources outside the car will be zero. The caustic photons are supposed to prevent the interior from getting pitch black in such cases. By looking up the caustic photon map at P, the total power of photons reflected from or transmitted through some objects and reaching P can be computed. The blue light from the sky and green light from the trees going through the glass to reach inside the car will all be considered (figure 6.36).

However, this caustic look-up does not give adequate image quality in practice. This caustic effect is basically the same as the weakly focused caustics mentioned in section 6.2.2, and has the same problem. Rendering sharp-edged or flat, noiseless caustics is extremely difficult, and some improvements may be obtained by the method introduced in 6.2.2.

For the color bleeding effects, the final gather value (irradiance value from diffuse surfaces) must be computed. Final gather rays from P will always hit objects in the car. When the ray hits a diffuse surface in the car again at Q, the global photon map is looked up at Q as usual, and there will be no problem regarding the

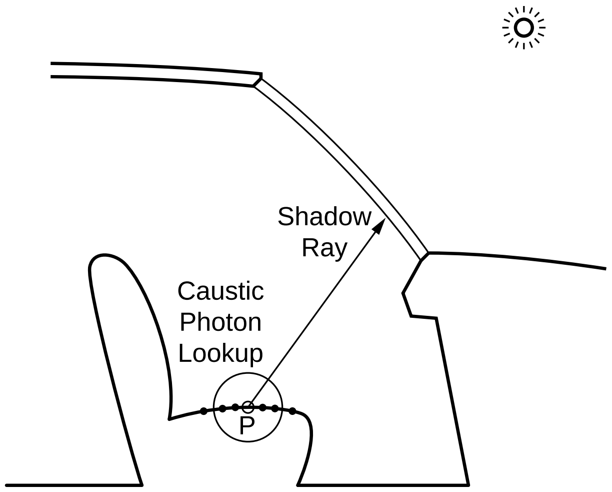


Figure 6.36: Shadow and caustic look-up

precision.

However, there is a great chance for the final gather ray to hit the glass. The ray tracing is discontinued, and the global photon map is looked up. Because the glass has no diffuse component, no photons are stored and the result becomes 0.

This does not mean that the interior of the car is incorrectly ignoring the light from outside. The caustic photons are supposed to provide correct illumination effects inside the car. It is just that getting enough precision from these caustic photons is extremely difficult.

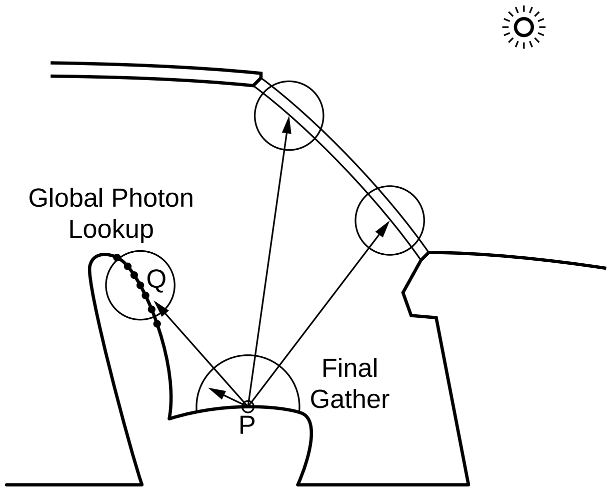


Figure 6.37: Final gather and glass window

The images in figures 6.38 and 6.39 suffer from this problem.

The following sections propose two techniques to overcome this problem. The

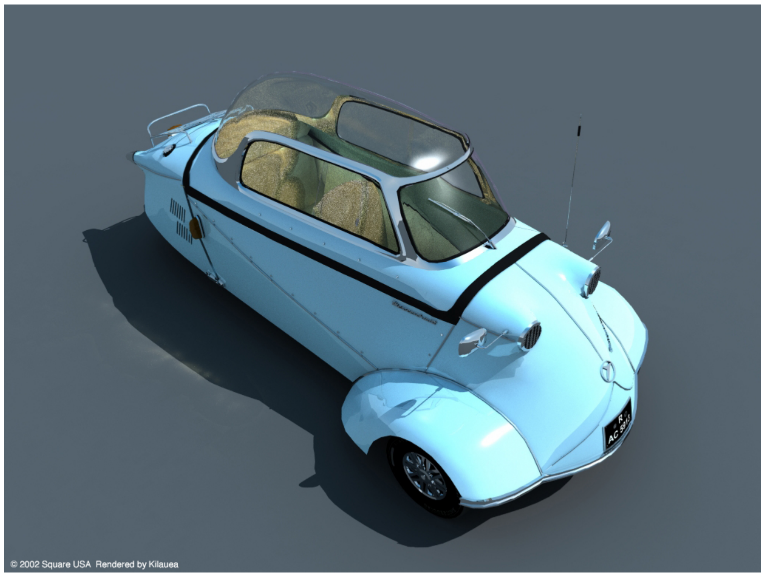


Figure 6.38: Messerschmidt (2,000,000 total photons, reduction ratio 99%, 285,269 caustic photons, 395,823 global photons)

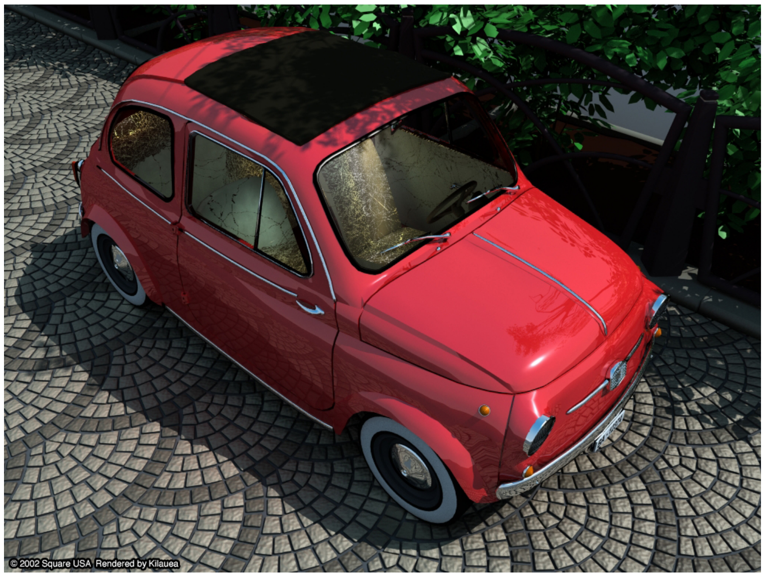


Figure 6.39: Fiat 500L (500,000 total photons, reduction ratio 0%, 12,018 caustic photons, 807,132 global photons)

first one is imperfect, but is very simple and practically usable in many situations. The second one can handle tinted glass, which is not possible in the first method.

Solution A: completely ignoring glass

The problem is essentially caused by how the photons passing through the glass surfaces are handled. If the glass is thick and the thickness is inconsistent, the complex behavior of light must be correctly computed. In this case, not much can be done other than to use the caustic photon map in a straightforward manner. The problem basically boils down to increasing the precision of the irradiance obtained from the caustic photon map.

However, if the glass has constant thickness and is thin enough to ignore its refraction, the simplest yet effective method is to just ignore its existence in the global illumination computation. If the glass window did not actually exist, the scene would be rendered with perfect quality. Then, let us just remove it from global illumination point of view.

Because there is no light source in the car, all photons come from outside. Previously, all incoming photons passed through the glass surfaces, but now the glass is not visible from photons and thus will directly hit objects in the car. These photons did not come through glass surfaces, so they are only stored in the global photon map. Some of them will further bounce in the car and get stored in the global photon map at where they hit. The caustic photons stored by passing through the glass window in the previous case are completely gone (figure 6.40).

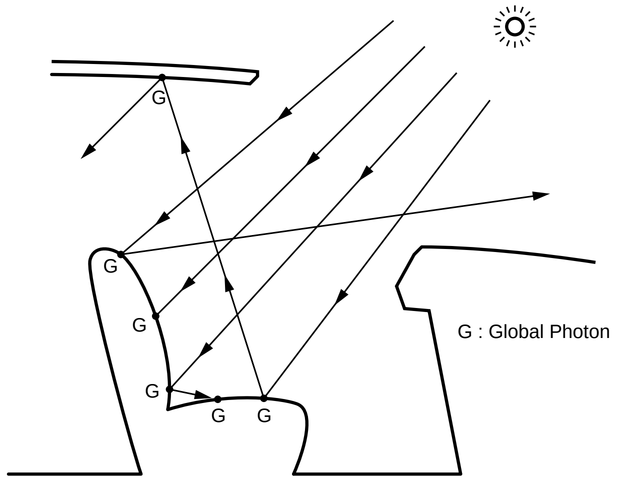


Figure 6.40: Solution A: photon store

The direct illumination computation at P on the object in the car starts by computing the surface shading color at P. At this stage, shadow rays from P are shot to light sources. Previously, shadow ray tracings were discontinued as soon as they hit the glass windows. Since the glass windows are ignored now, the light sources outside can be directly considered for the illumination at P. Shadows in the car will be rendered with sufficient quality (figure 6.41).

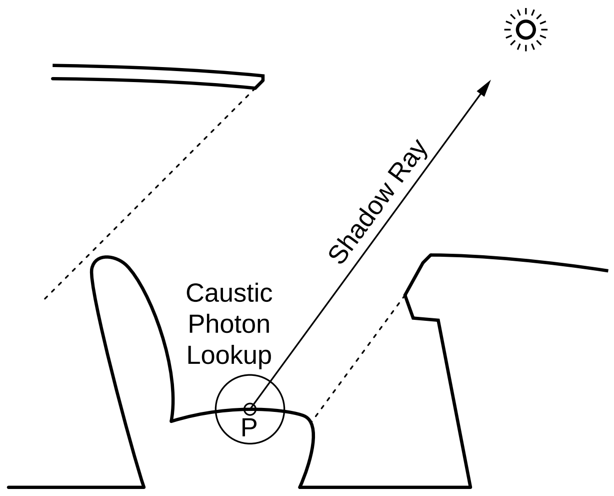


Figure 6.41: Solution A: shadow trace & cautic lookup

Then the caustic photon map at P is looked up. However, almost no caustic photons are expected to be found around P. This is unlike the previous case where every photon coming into the car through the glass is stored in the caustic photon map. The precision of the caustic photons at P is no longer significant.

At last, the final gathering at P is executed. Many final gather rays will hit the glass, but the tracing is not discontinued anymore. Instead, they will continue their path outside the car, eventually hitting something or picking up the color of the sky. The global photon map is looked up at where they hit to determine the irradiance, and the final gather value at P will be determined (figure 6.42).

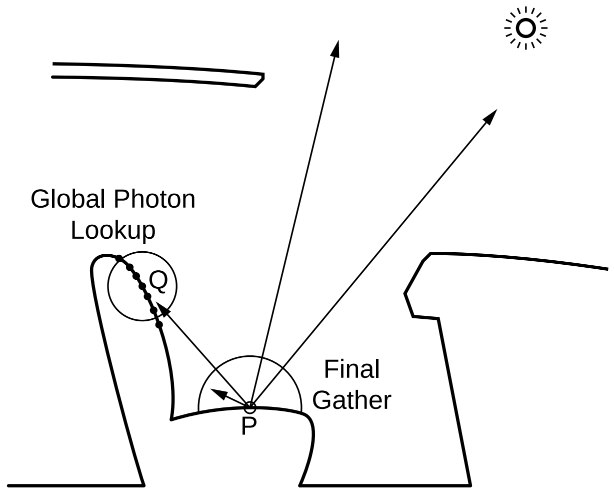


Figure 6.42: Solution A: final gather

Solution B: conditionally ignoring glass

The previous solution simply ignored the glass. The solution presented here considers refraction and transmission through glass in the photon tracing, allowing a little more complex simulation. However, the behavior of refraction and transmission are simplified in the final rendering stage. Therefore, this solution is not strictly computing the global illumination. Compared to solution A, this solution has a little more flexibility in the possible lighting effects such as tinted glass.

Because there are no light sources in the car, all photons must pass through the glass window to get inside the car. This time, the photon tracing correctly computes the refraction and the transmission when going through the glass. Thus, photons passing through the glass will pick up its color. In a regular photon map rendering, the photons coming through the glass window and hitting diffuse surfaces in the car are stored in both global and caustic photon map. However, in this solution, they are only stored to the global photo map. Photons bouncing in the car are treated as in the regular photon mapping (figure 6.43).

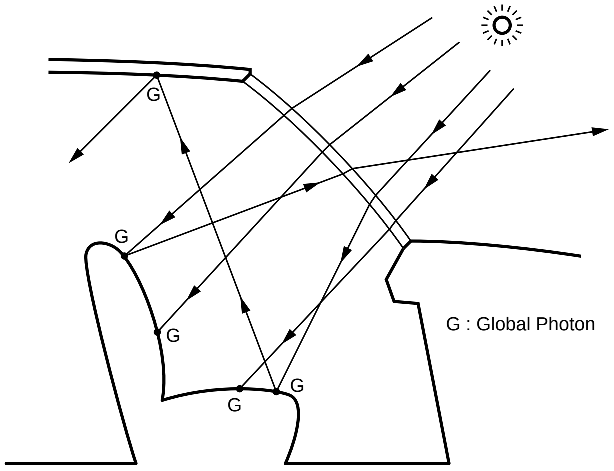


Figure 6.43: Solution B: photon store

In the final rendering stage, the glass is not totally ignored, but rays will simply go through them. During the direct illumination computation, shadow rays are shot at the light sources. When the shadow ray hits the glass, it goes straight through the glass. Because of this, the light outside the car can be considered in the direct illumination computation at a point P in the car. To get the effect of the tinted glass, the color of the light source may be modified according the color of the glass when the ray goes through the glass. This method also achieves high quality shadows in the car.

Unlike in the straightforward application of photon mapping, the interior of the car will not be filled by caustic photon map because this solution skips the the storing of photons coming through the glass window to the caustic photon map in

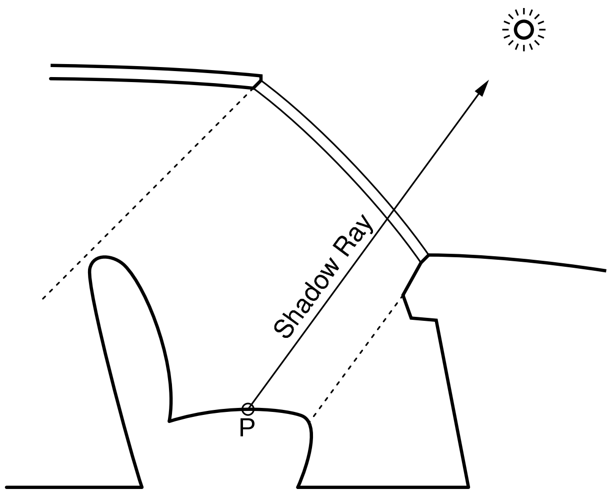


Figure 6.44: Solution B: shadow ray

the photon tracing stage. Because the illumination inside the car does not depend on the caustic photons, the precision of caustic photon map is practically negligible.

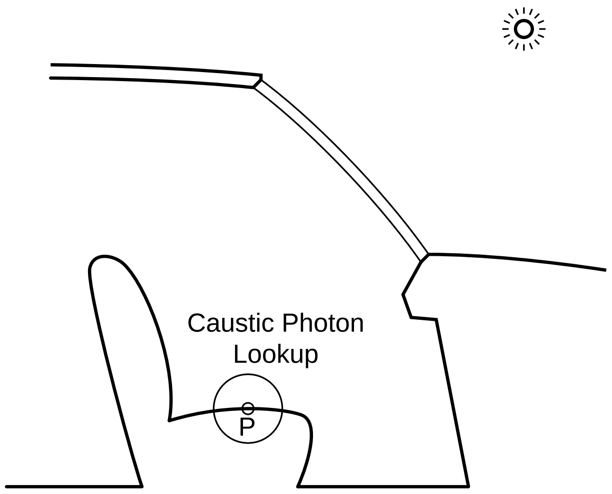


Figure 6.45: Solution B: caustic look-up

Final gather rays will go straight through the glass, just like shadow rays. The color of the glass must be considered when going through the glass. The ray will continue on its path until it hits an object, where the global photon map is looked up to compute the irradiance, while considering the color of the glass. When the final gather ray hits another object in the car, then the regular final gathering procedures are taken to compute the irradiance.

Because photon tracing is correctly refracted but shadow rays and final gather rays are not refracted, this solution is inconsistent and not strictly correct. Also,

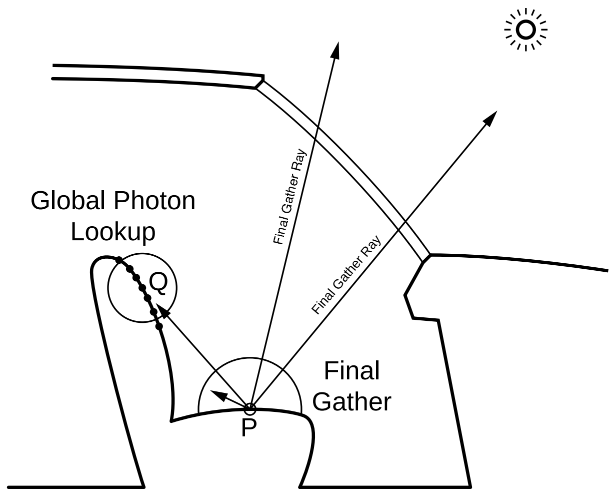


Figure 6.46: Solution B: final gather

because this method relies on shadow ray tracing for the shadow and direct illumination component computation, only the light sources directly visible are evaluated. In other words, illumination from reflection is neglected. Solution A suffers from the same limitation. Nevertheless, if applied in the correct situation, these solutions are fairly good workarounds for the caustic photon map problem.

Rendering results

Image 6.47 is created using solution A. Image 6.48 uses solution B to tint the glass.

6.3 Speeding up global illumination

The global illumination computation using Monte Carlo ray tracing is computationally very expensive. This is somewhat relieved by the use of the photon map, but still further speed-up is mandatory for the production use, considering today's processor speeds. Kilauea's approach was to obtain the required computational power from massively parallel computing, but even by using such processing power, algorithmic improvements to the global illumination computation was demanded.

The speed-up methodology in Kilauea is categorized to two fields: parallel processing and improving global illumination computation algorithm.

6.3.1 Speed-up in parallel processing

First, the viability of the parallel photon map implementation is discussed. Please refer to [S2000Course40] [Chalmers] [Kilauea] for more in-depth description of Kilauea's parallel architecture.

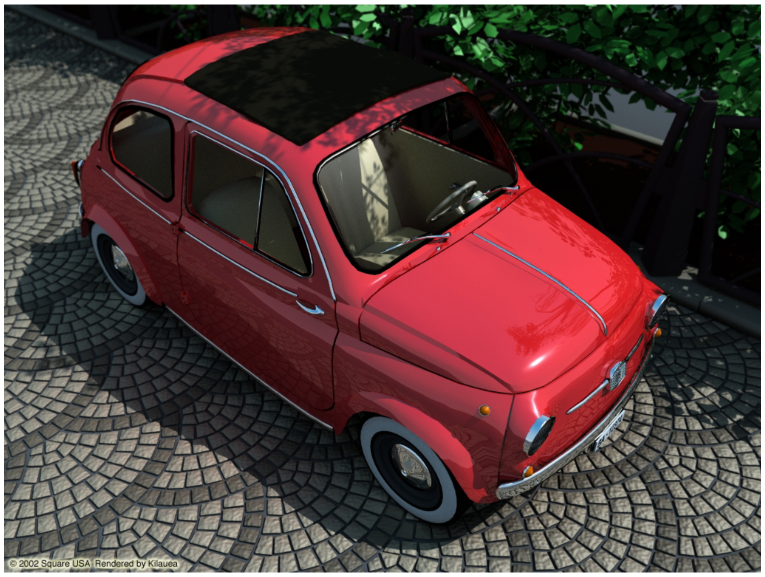


Figure 6.47: Fiat 500L

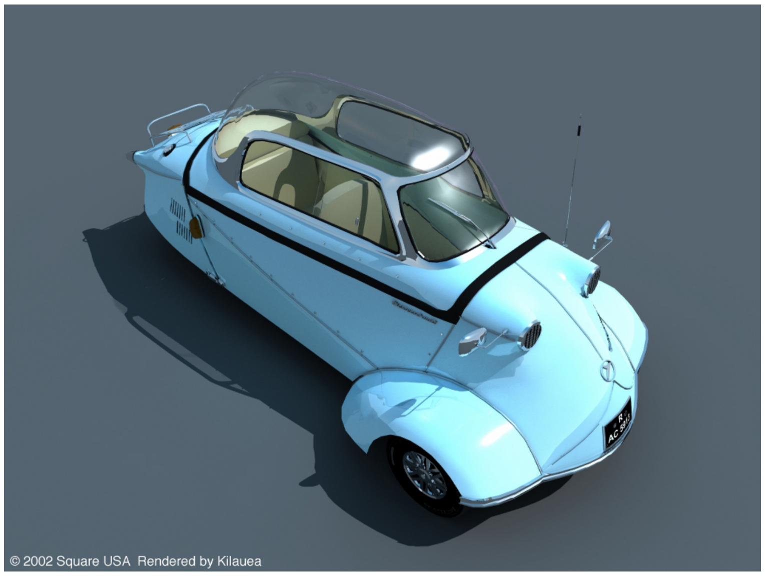


Figure 6.48: Messerschmidt

Parallel photon tracing and storing

In the process of photon tracing and storing to the photon map, each individual photon emitted from light sources is independent from each other, and thus can be completely processed in parallel. This property of the photon tracing and storing makes it suitable for parallel processing on multiple machines and multiple CPUs.

By simply letting each CPU to grab photon data from the pool of photons generated by the light sources in the scene, the CPU load will be balanced (figure 6.49).

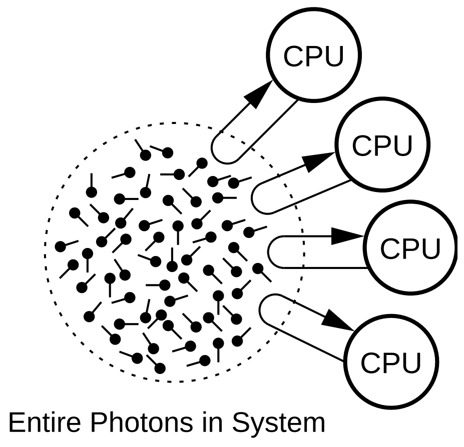


Figure 6.49: Parallel Photon Tracing

Kilauea is designed to use multiple machines. In such parallel environment, the creation of the photon map needs some careful consideration. Kilauea creates two types of photon map, depending on the resulting size of the photon map.

1. Photon map is small enough to be stored in one machine
2. Photon map is too large and is distributed to several machines

The photon tracing computations are distributed to available CPUs randomly at runtime. In the first case, when the photon store occurs, because the photon must be stored on all photon maps on all machines, its copies are distributed to all machines. Doing so on all machines results in identical photon maps on all machines in the end. In the second case, similar photon storing principle can be applied to create the photon map. Please refer to SIGGRAPH 2002 Practical Parallel Rendering course notes for more on this topic. The photon distribution inflicts some overhead as the number of machines increases, but compared to the total computational cost in the Kilauea rendering, the overhead is acceptable.

When about 700,000 photons are stored, the total size of the photon map becomes approximately 35 MB. The transfer of this data to another node over 100 BaseT network takes 5 to 6 seconds. If the entire rendering takes 10 minutes, this overhead is acceptable.

Parallel irradiance precalculation

In the final photon map generation process, the irradiance values at the position of photons are precalculated to speed up the final rendering pass.

Each precalculation is an independent procedure for every photon stored in the photon map. Thus, this stage is suitable for parallel processing, yielding adequate parallel performance[S2000Course38].

In the case where Kilauea uses multiple machines and the copies of the photon map exist independently on all machines, the maximum parallel performance is expected when all CPUs perform the precalculation on different photons, and later gather the results. However, there will be an overhead for distributing the precalculation results to all other machines, just like in the photon storing.

Parallel photon look-up

Each shading computation in Kilauea is processed in parallel independently. Each shading computation at each sample point does not interfere with the others, and the order that they are processed is determined at runtime. However, one shading computation is basically processed sequentially. The entire Kilauea rendering consists of numerous shading computations, each of which having high granularity. Therefore, by considering one shading computation as the unit of parallel processing, effective parallel performance and load balancing over multiple CPUs are accomplished. In other words, trying to process one shading computation in parallel is a trivial matter which does not improve the performance.

Because the photon look-up operation is invoked from the shaders, the look-up will be executed in parallel. Thus, the photon map should be accessible in parallel. Because the photon look-up is a read-only operation, accessing the photon map from multiple threads running on multiple CPUs at the same time is not a problem. Obviously, the local memory required for the look-up operation must be managed individually for each thread.

As previously mentioned, two types of photon map are created in Kilauea, depending on its size. If the photon map is huge and had to be distributed to multiple machines, the look-up operation becomes very complex, and will suffer from a severe message passing overhead (more on this topic in SIGGRAPH 2001 Practical Parallel Rendering course notes [S2000Course40]). However, if the entire copies of the photon map can be placed on all machines, the look-up operation only has to reference the photon map in its machine, and thus will exhibit a superb parallel performance.

Effectiveness of parallel photon mapping

Photon mapping and parallel processing work very well together, and also the parallel photon map implementation is quite straightforward. For the parallel implementation of the photon map, parallelization of shading and ray tracing is mandatory. However, if there is a working parallel ray tracer, then adding the photon map capability should not be a problem.

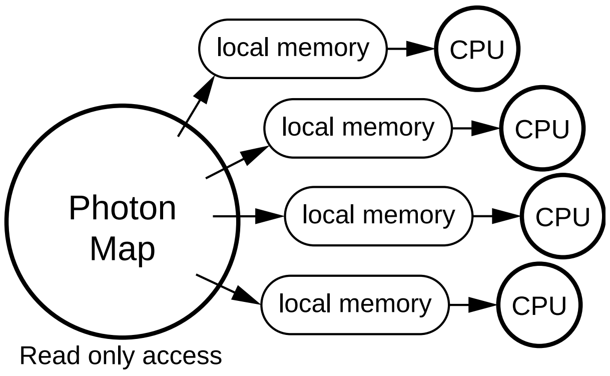


Figure 6.50: Parallel photon look-up

When multiple machines are connected to process photon mapping in parallel, stored photons and irradiance precalculation results must be transferred, inflicting some communication overhead. However, even in such distributed environment, photon map essentially consists of independent elements and is an appropriate data structure for parallel processing.

In the shared memory parallel environment, there will be no communication overhead and the optimal parallel performance is easily achieved. This is one of the strengths of photon mapping.

6.3.2 Final gather estimation

This section discusses the *final gather estimation*, an algorithmic improvement to the global illumination computation, and its parallel implementation ideas.

Observation of global illumination

The profiling of the Kilauea rendering indicates that most of the computation time is spent on final gathering. The motivation of the final gather estimation is to speed up the final gathering stage. The starting point of the algorithm is the irradiance caching idea proposed by [Ward88] and [Jensen01].

To achieve one of the objectives of Kilauea to render extremely complex scenes, the scene distribution to multiple machines and ray tracing in such environment must always be taken into account. For this reason, Kilauea is designed to be a parallel renderer based on message passing, and this message passing will always be an overhead compared to sequential ray tracers. The overhead must be tolerated for the purpose of rendering complex scenes. This in turn means that ray tracing is relatively expensive in Kilauea compared to sequential ray tracers. The principle of Kilauea is to cover this overhead by allowing massively parallel rendering with high scalability. However, an innovative idea to drastically reduce the number of final gather rays is still necessary. This approach to reduce final gather rays also conforms to Kilauea's objective to render complex scenes, because the ray tracing computation becomes more expensive as the scene size increases.

There are two approaches in reducing the final gather rays.

1. Process final gathering itself in some intelligent way and reduce the emitted final gather rays per one final gathering

2. Reduce the number of sample points where final gathering is performed

In Kilauea, how final gather rays are shot at a point on a surface depends on how the shader for this surface is implemented. Shooting more final gather rays is a double-edged sword: the computation time is sacrificed for better rendering quality. If efficient hemisphere sampling can be achieved to compute accurate final gather values with less final gather ray tracing, the total number of ray tracing to render an image is reduced. This is the first idea.

On the other hand, when the the change of final gather values over a certain surface is observed, the gradient curve is very smooth in most areas. With this characteristic in mind, if the final gathering is performed only at key positions, the total number of ray tracing is again drastically reduced. This implies that doing final gathering for every screen space sampling position is mostly redundant. This is the second idea.

In the early development stage, various sampling algorithms were tested and implemented into the shader to realize the first idea, but only a few percent speed-up was accomplished. For a drastic speed-up, implementation of the second idea was crucial.

Basic methodology

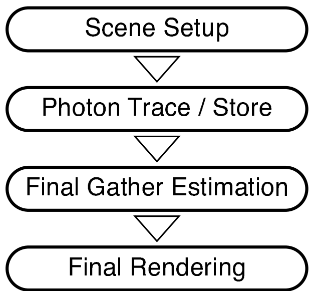


Figure 6.51: Multi-pass rendering

Currently, for computing one image, Kilauea performs a multi-pass rendering [Kilauea] (figure 6.51). First stage is the photon tracing. Next, the final gather estimation is executed, and then the final rendering pass computes the image. In the final gather estimation stage, the final gather values (irradiance from diffuse surfaces, to be more precise) at object surfaces in the scene are computed. Because this final gather estimation pass is independent, the screen space can be sampled optimally just for final gather computation, drastically reducing the total number of final gather rays in total.

The final gather values in the scene are examined based on how the scene appears on the screen. That is, such information as the area after it is projected to the screen and the complexity inside pixels are considered. This simplifies the elimination of occluded regions and incorporating the subpixel complexity in the computational logic of the following rendering stage. In other words, the precision of final gather values is controlled by view-dependent analysis and unnecessary computation can be excluded as much as possible.

Final gathering at some surface appearing in a certain pixel is controlled by analyzing the gradient of the final gather values over the screen space. If the final gather values change smoothly at a certain region in the screen, very little final gather sample points are needed. If the values differ significantly over the region, many final gather sample points must be placed on the screen region where the surface is projected.

If the irradiance change over a region is very smooth, the final gather sample points are placed only sparsely and a simple screen space based interpolation calculates the final gather values within this region. Final gather values computed this way are stored in the 2D image according to their screen space locations. The final rendering stage simply refer to this image to obtain the final gather value, without actually shooting final gather rays.

If for some reasons, the final gather estimation cannot or somehow discontinue its effort in obtaining the final gather values, then Kilauea simply switches back to regular final gathering.

Parallel processing methodology

Kilauea is designed to use the processing power of multiple machines, and the final gather estimation must also be processed in parallel.

As mentioned in the previous section, the final gather estimation depends heavily on the screen space analysis, requiring somewhat different parallel processing approach from other tasks in Kilauea. For the final gather estimation, the screen is simply divided into small rectangular buckets. Each CPU retrieves buckets and computes final gather values within them in parallel.

For the load balancing purpose, the granularity of the parallel processing unit should be high enough – that is, the bucket size should be small enough. For instance, when using 16 CPUs to render a 1024x768 image, a bucket size of 64×64 is enough to keep CPUs busy most of the time. If higher granularity is desired, the bucket size can be adjusted accordingly to keep the acceptable parallel perfor-

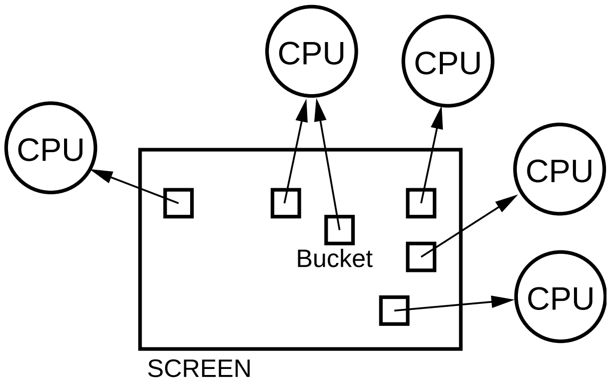


Figure 6.52: Parallel Bucket Processing

mance.

Screen space mask

The following test scenes from figure 6.53 and 6.54 are used for the explanation.

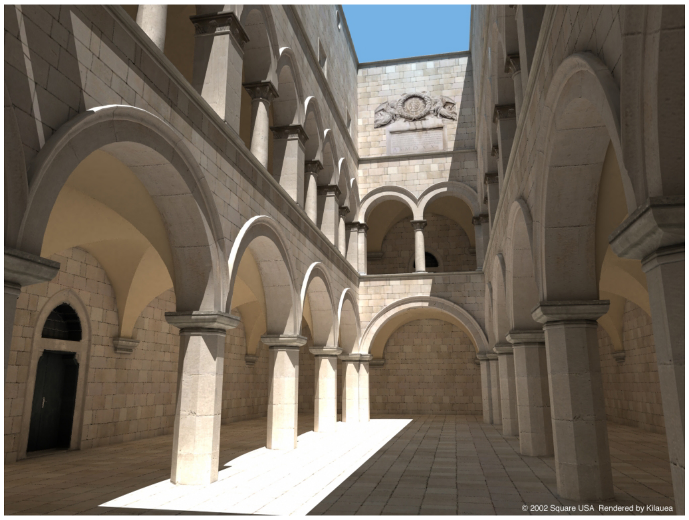


Figure 6.53: Sponza

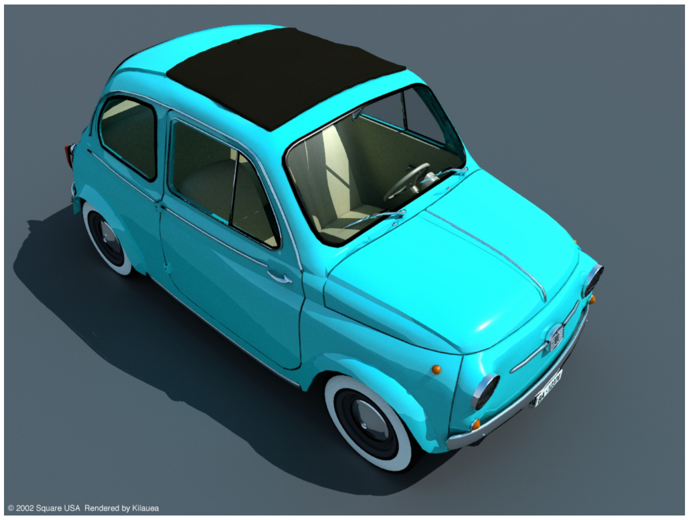


Figure 6.54: Fiat500L simple

As stated in the previous section, the final gather estimation request is received as a rectangular bucket in the screen space, and the final gather values at visible surfaces in all pixels of this bucket are computed. The final gather estimation process first receives the bucket and creates the screen space mask for this region. The purpose of this mask is to determine the pixels unsuited for the final gather estimation just by doing a screen space analysis. The pixels where attempting final gather estimation is apparently futile are called *critical pixels*, and other pixels are called *safe pixels* by convention.

First, all pixels in the bucket are ray traced with point sampling. At the surface that the primary ray hit, its position, normal, and the direct illumination color are computed. The critical and safe pixels are determined according to these data. The final gather estimation will only be applied to the safe pixels later on.

Also, by evaluating the change in direct illumination results, the region to be supersampled in the final rendering is determined. The final rendering stage generates sampling schedule according to the prediction made here. The region requiring high supersampling such as the silhouettes of objects and shadows are effectively determined for every pixel, contributing to the speed-up.

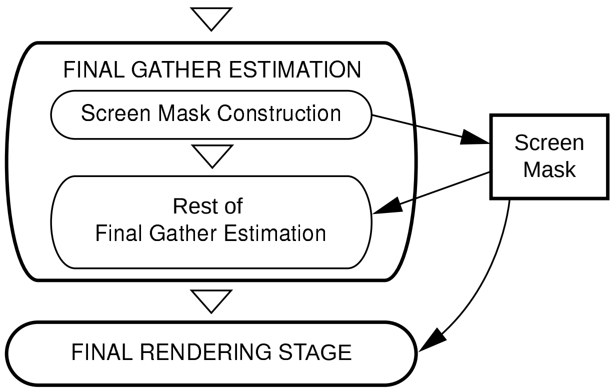


Figure 6.55: Screen mask in use

The screen space mask generation basically searches for flat areas with large spatiality, where the final gather values can be estimated by simple interpolation.

For the critical pixel detection, normals and positions of neighboring pixels are compared. If the difference in the normals or the distances from the camera to the object is over a certain threshold, the region is considered to contain some significant geometrical change where the final gather estimation is ineffective, and thus is considered as the critical pixel.

After processing all pixels for this comparison with neighboring pixels to determine the critical pixels, the region with critical pixels turns out to be where the silhouettes of objects or the uneven surfaces are located. The safe pixels, on the other hand, are in the flat regions where the final gather estimation is expected to work effectively.

The accuracy of the screen space mask depends on the thresholds for the differences in normals and distances from camera. The optimal values for these parameters vary depending on the scene. Kilauea allows users to modify these thresholds to obtain more accuracy.

The results of screen space masks are shown in the figure 6.56 and figure 6.56.

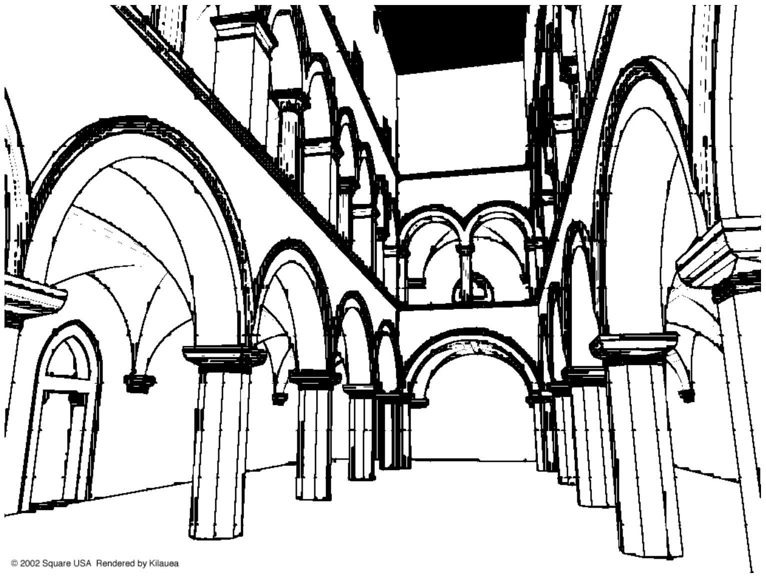


Figure 6.56: Sponza screen mask

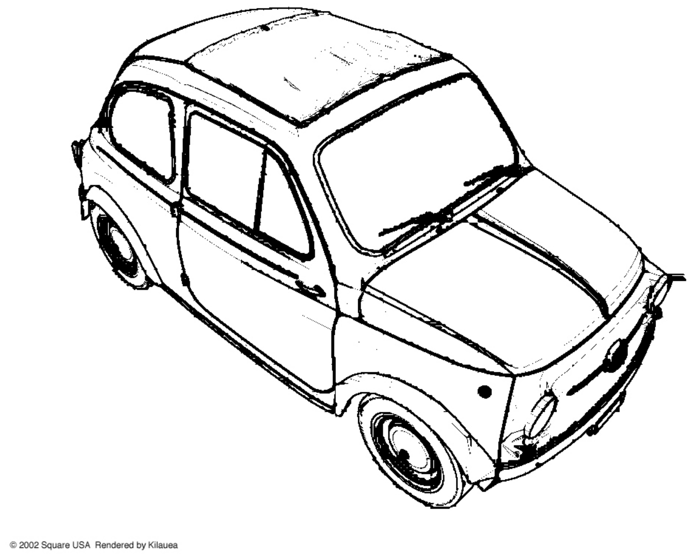


Figure 6.57: Fiat 500L screen mask

Sparse final gathering

The actual final gather values are still unknown after the generation of the screen space mask. To find them out, final gathering sample points are sparsely chosen over the screen space in a uniform manner and the final gatherings take place at these points.

The distance between final gather sample points is a scene dependent parameter set by users. For most scenes, 32 or 16 pixels seems to work without any problems. Smaller distance results in a better final gather value precision, but slower processing speed.

Kilauea totally depends on the shader implementation for how the final gather rays are shot. For most purposes, adaptive hemisphere sampling approach mentioned earlier is used. The final gather estimation process, however, is independent from how the actual final gather rays are shot and computed.

At each final gather sample point, the final gather value is computed from final gather rays. Also, the *harmonic mean distance* [Ward88] is computed from the distances that the final gather rays traveled. These values are stored in the bucket's pixel information.

Here is the image showing the location of the sparse final gather sample points with white dots. Note that there are no dots on the critical pixels, and the final gather estimation is only executed on the safe pixels.

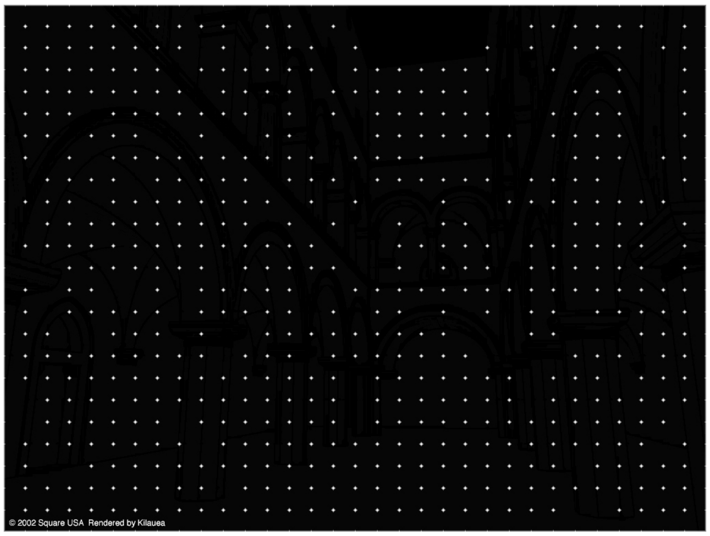


Figure 6.58: Sponza sparse final gathering

Adaptive subdivision

From the sparse final gather values, more precise final gather values in all safe pixels must somehow be interpolated. The algorithm here is again a screen space

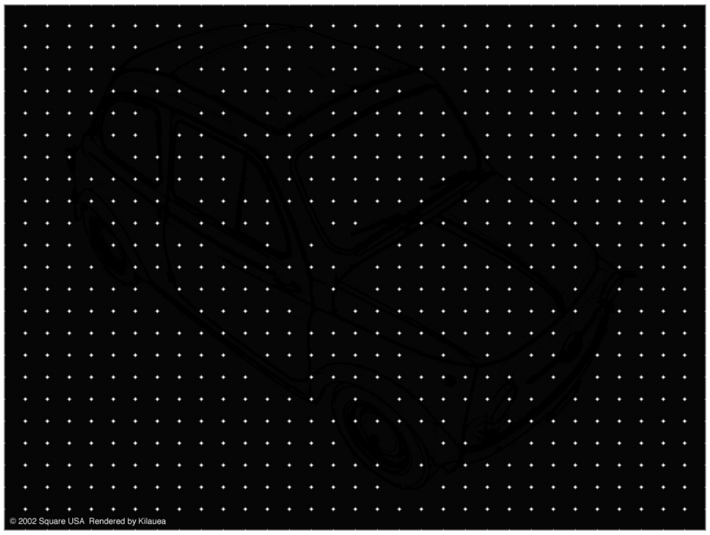


Figure 6.59: Fiat 500L sparse final gathering

analysis.

If the distance between neighboring sparse final gather sample points is 16, the bucket is broken down to tiles of 16×16.

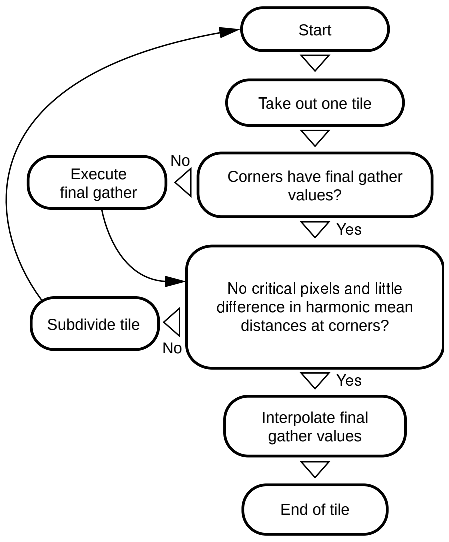

Start
Take out one tile
Corners have final gather values?
No
Execute final gather
Yes
No critical pixels and little difference in harmonic mean distances at corners?
No
Subdivide tile
Yes
Interpolate final gather values
End of tile

Figure 6.60: Processing Tiles

Within each tile, if all four corners have already executed final gathering and all pixels within the tile are safe pixels, the feasibility of simple interpolation to compute the final gather values is evaluated according to the harmonic mean distances of the four corners. The harmonic mean distance of the final gather rays from a certain point represents the spatiality at this point [Ward88]. If the area was found to be spatially large by evaluating the harmonic mean distances at the four corners, the final gather values within the tile should vary smoothly, and thus the area is safe to be interpolated. Kilauea does a simple bilinear interpolation here instead of more complex interpolation with the consideration of 3D positions because a simple interpolation was found to give sufficient quality. After the interpolation, all final gather values for this tile is computed, which ends the final gather estimation process for this tile.

If, from the analysis of the harmonic mean distances, the tile was found to be in a spatially small area and simple interpolation is not feasible, the tile is further subdivided into more tiles. Additional final gathering is executed at the corners where they have not done final gathering yet. The harmonic mean distances are evaluated for the subtiles, and the process continues recursively.

If the critical pixels exist in the tile, it is also subdivided. If the new corners are not critical pixels and have not done final gathering yet, final gatherings are performed at those pixels.

The tiles overlap with neighboring tiles by one pixel. Because of this, many final gather computations can just reference the previous results from neighboring tiles.

During the course of this adaptive subdivision, some tiles will complete their computation by interpolation in the early stage of subdivision. Some tiles may require subdivision up to the size of one pixel.

However, for most cases, the subdivision to the size of one pixel is overdoing the final gathering. The size of the minimum subdivision tile size is a user-defined parameter in Kilauea. The size of 3×3 seems to work reasonably well in most scenes. This implies that in the safe pixel region, only final gathering for every one pixel results in sufficient quality.

After the final gather estimation process, some safe pixels might remain unresolved. They are simply final gathered as usual in the final rendering stage.

Figures 6.61 and 6.62 show the results of the adaptive subdivision. The positions where further final gatherings were performed are plotted in white dots.

Here are the stats for these images:

- Sponza

  Total pixels: $1024 \times 768 = 786,432$

  Safe pixels: 615,995 (78.33%)

  Critical pixels: 170,437 (21.67%)

  Sparse FG sample points: 650

  Adaptive subdiv FG sample points: 23,689

  Total FG sample points: 24,339

- Fiat 500L

  Total pixels: $1024 \times 768 = 786,432$

  Safe pixels: 718,887 (91.41%)

  Critical pixels: 67,525 (8.59%)

  Sparse FG sample points: 744

  Adaptive subdiv FG sample points: 16,762

  Total FG sample points: 17,506

The percentage of final gatherings performed at safe pixels is 3.95% for Sponza and 2.43% for Fiat 500L.

Figures 6.63 and 6.64 show final gather values computed by final gather estimation.

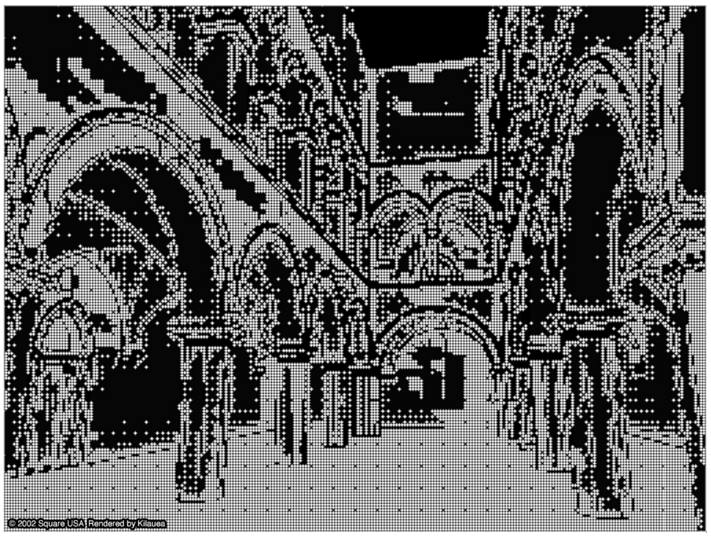


Figure 6.61: Sponza: adaptive subdivision

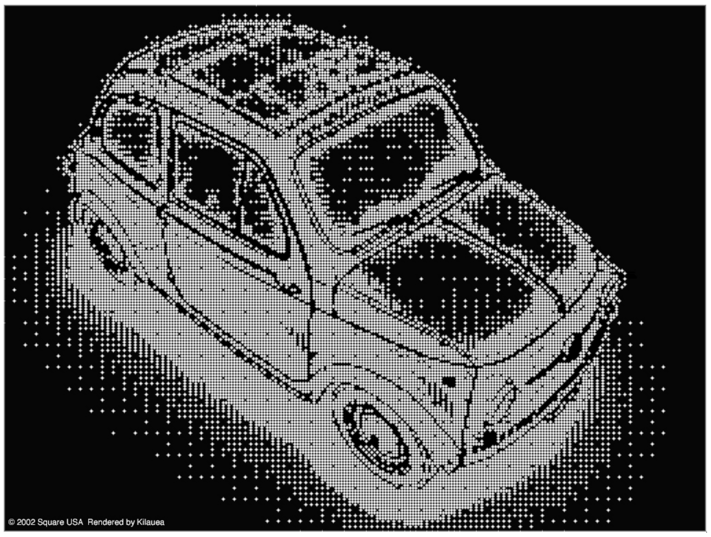


Figure 6.62: Fiat 500L: adaptive subdivision

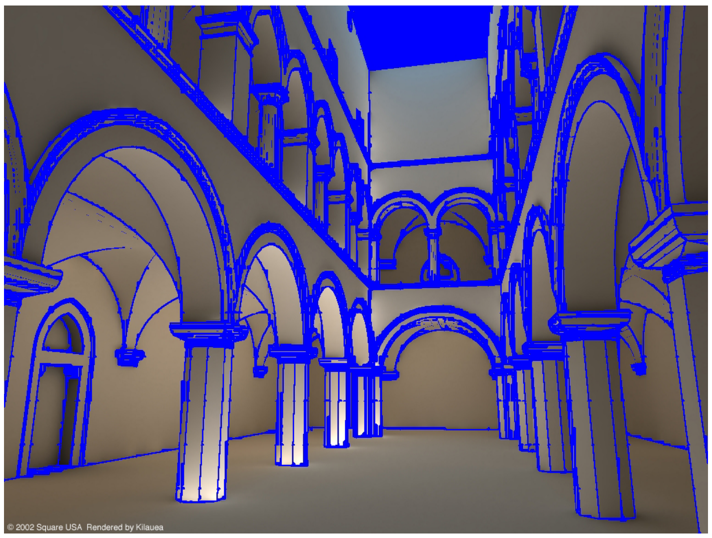


Figure 6.63: Sponza: final gather estimation

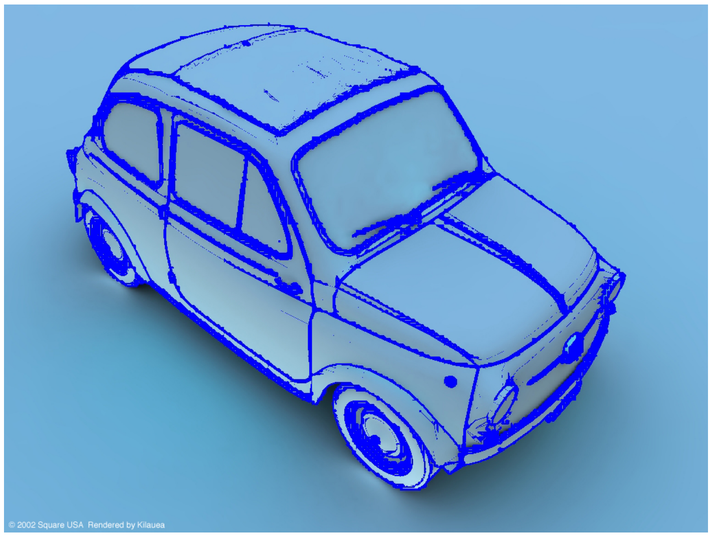


Figure 6.64: Fiat 500L: final gather estimation

Consideration of final gather estimation

There are two important parameters to control the quality of final gather estimation. One is used to detect critical pixels, and the other is used in analyzing the spatiality from harmonic mean distances.

The default parameters are set conservatively. That is, the default behavior leans toward stability and quality over speed. Without tuning the parameters, Kilauea may not render at its fastest possible speed, but it makes sure that no quality problems will occur.

The tuning of these parameters is very subtle. If the parameter leans toward speed-up little more than the optimal value, degradation of the image quality appears as low frequency noise. Human eyes tend to be very sensitive to low frequency noise, and the case where the user picks speed over this low frequency noise is highly unlikely. Thus, the right values have to be chosen every time.

The low frequency noise is mostly caused by the low precision in the harmonic mean distance computation. To obtain accurate harmonic mean distances, the number of final gather rays must be increased, which contradicts to the premise of final gather estimation.

With sparse final gathering and adaptive subdivision, the number of final gather rays is remarkably reduced. However, even further optimization is demanded.

6.3.3 Final gather reprojection

Section 6.3.2 described the final gather estimation, the basic logic behind Kilauea's global illumination speed-up. Final gather reprojection presented here attempts to optimize the final gather estimation by reducing final gather rays even further.

Concept

The motivation here is to reduce the number of final gather rays further, while keeping the final gather estimation algorithm.

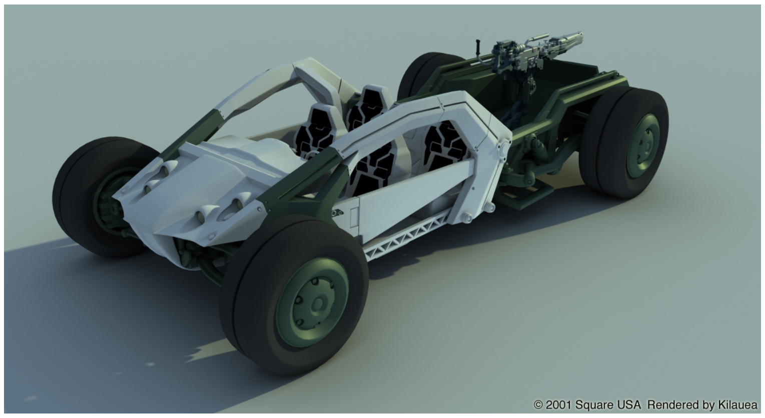


Figure 6.65: nJeep1

This image 6.65 is the result from final gather estimation, and this image 6.66 shows the final gather values in color for every bucket computed in the final gather estimation. Let us consider two final gather sample points in this image, A and B, and a point C which appears to be in the middle of them in the screen space. If the

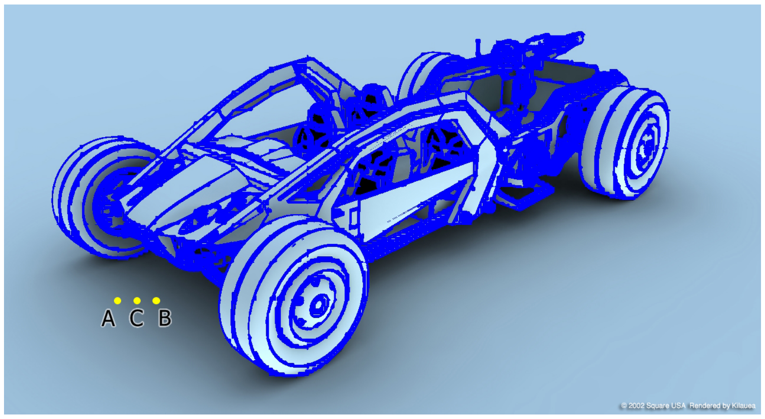


Figure 6.66: nJeep1 final gather estimation

final gatherings at A and B are known, is there a way to predict the final gathering result at C?

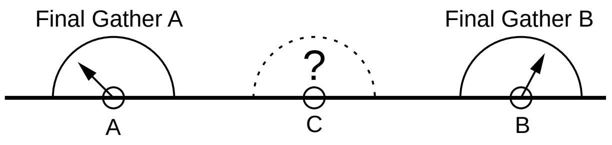


Figure 6.67: Estimate final gather values at C from values at A and B

The images 6.68 and 6.69 show how the final gatherings actually appear at A and B. The final gathering with the resolution of 65×65 is performed at each point, and the final gathering results are reconstructed into an image as if it were from the fisheye camera shot. The final gather value is, in a sense, simply the average color of this image.

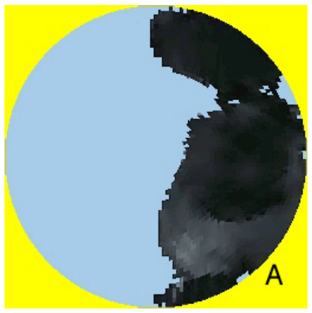


Figure 6.68: Final gather at A

In the adaptive subdivision explained in section 6.3.2, final gathering is needed

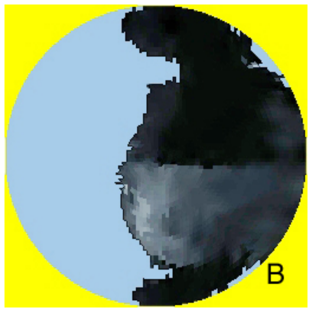


Figure 6.69: Final gather at B

at each corner of the subdivided tiles. If A and B were the corners of the original tile, C would be one of the corners of the subtile on the edge made by A and B. By following the algorithm of adaptive subdivision, the final gathering at C must be performed.

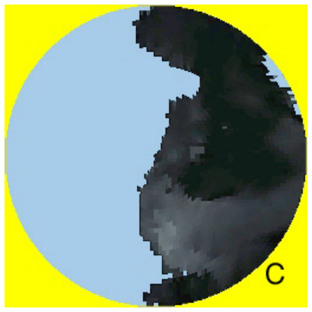


Figure 6.70: Final gather at C

The image 6.70 shows how the final gathering at C appears. The image looks very much like the final gathering images at A and B – in fact, the image at C appears to be somewhere in the middle of images at A and B. If the final gather ray information at C can be estimated using final gather ray information at A and B, the number of final gather rays can be reduced drastically. Another good thing about this idea is that it could be easily incorporated into the final gather estimation algorithm.

However, simple 2D image processing to create the intermediate final gather information is not practical because the results from final gathering is computed by ray tracing the scene, which correctly considers the location of objects in the scene. Instead of a 2D solution, how the hit positions of the final gather rays from A and B will appear at C must be computed. This in turn is a reprojection of final gather hit points from A and B to the final gather space at C.

The final gather hit point from A may not be visible from C because it is obstructed by the final gather hit point from B (figure 6.72). This implies that an

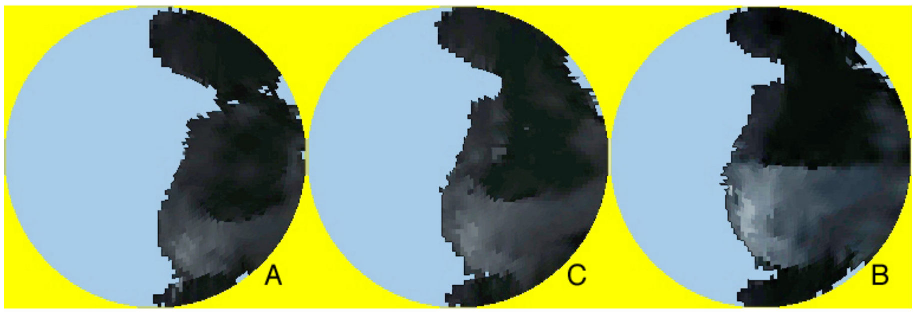


Figure 6.71: Estimate final gather at C by image processing

algorithm like Z-buffer must be incorporated in the reprojection process.

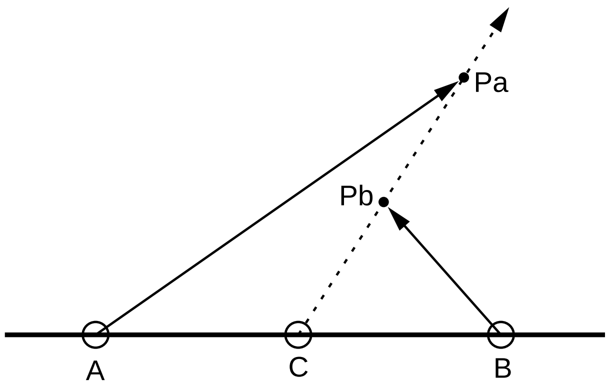


Figure 6.72: Necessity of Z-buffer

Caching final gather ray information

In the regular final gathering, the color and the distance to the hit point computed by final gather ray tracing are accumulated and each ray information is not stored individually. However, for the final gather reprojection, because the final gather ray information from a certain point are reused in another, all final gather ray information must be stored. This includes all final gather information resulting from the sparse final gathering and adaptive subdivision in the final gather estimation process.

One might concern about the memory consumption of the final gather ray information. Single final gather ray takes up 54 bytes, including the hit point and its normal, irradiance, and parameters for debugging purposes. For one final gather sample point, adaptive final gathering is performed with 65×65 precision, ending up shooting about 1000 final gather rays. In this case, the memory usage for this sample point is approximately 53 Kbytes. In a 64×64 bucket, if the final gath-

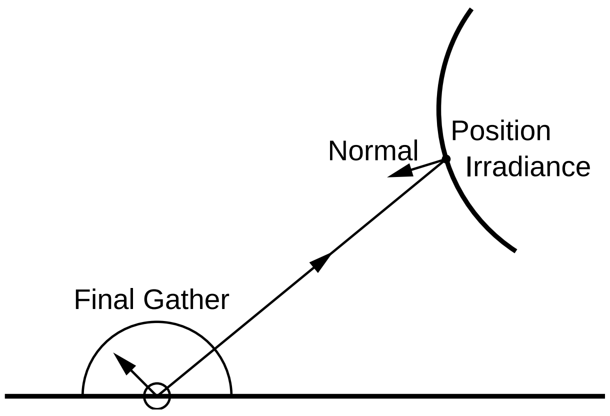


Figure 6.73: Final gather ray information

ering was performed every two pixels (which is close to the worst scenario), the number of pixels which performed final gathering is about 500. The total amount of memory to store this much final gather information is about 25 MB. This is not that much, considering that machines used for Kilauea development have 1 GB of memory.

In reality, the memory usage depends on the number of final gather sample points, and mostly turns out to be under half of the predicted value. To save memory further, the bucket size can be reduced to, say, 32×32, and the required memory will be approximately quartered.

This final gather ray information is created frequently, and released all at once after one bucket is processed. Specialized memory management mechanism for this final gather ray information promises the optimal memory usage.

Hit point reprojection

This section explains how the final gather ray information at C is reprojected from the information at A and B.

Numerous final gather rays have been already shot from both A and B, and those results (final gather ray hit position and normal at the hit position) are already stored in the memory.

The final gather information at C is stored in the 2D array. The final gather information basically represents the image captured when the hemisphere above the point on the surface is viewed. The previously shown images of the final gather information were distorted like a view from a fisheye lens, to stress the principle of hemisphere sampling. For the ease of implementation, this distortion is transformed so that the hemisphere sampling points can be placed in the 2D array. This array, without fisheye distortion, is shown in figure 6.74.

The resolution of the 2D array at C is determined by the final gathering resolu-

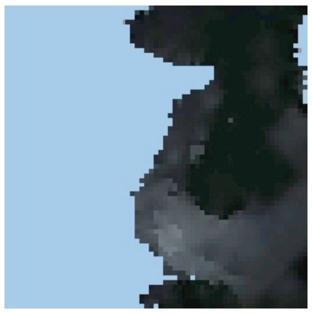


Figure 6.74: Final gather image at C

tion at A and B. If 65×65 final gatherings are performed at A and B, the resolution of the array at C is also 65×65.

Let us consider one final gather ray from A in figure 6.75. The corresponding ray information contains the hit position P and the color at that position. How P appears from C needs to be computed.

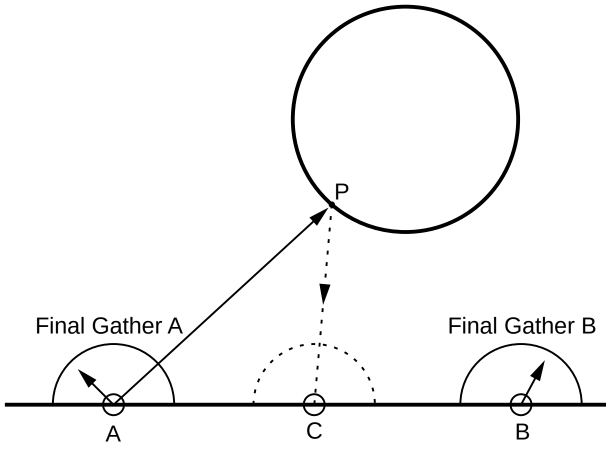


Figure 6.75: Reprojecting P

In the final gathering process, Kilauea uses a local coordinate system on the surface. This coordinate system is uniquely defined by two basis vectors computed from the normal at the position to final gather. P is first transformed to the local coordinate at C. Then, the corresponding location in the 2D array is computed from the local coordinates of P. The mapping to the 2D array uses concentric coordinates. The irradiance is then stored in the 2D array [Shirley00]. One thing to note about this process is that the transformation is done to the point data.

For all final gather rays in A and B, the above transformation is performed. The final gather hit position to be transformed may be invisible from C because it is under the “horizon”, as in figure 6.76. Such hit position is discarded from the reprojection process at C.

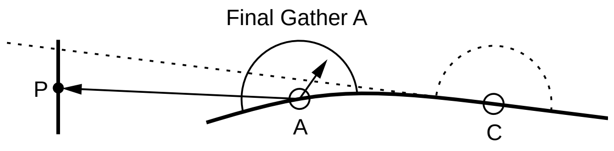


Figure 6.76: Horizon problem

Some final gather rays to be transformed might not have hit any objects. Those rays are treated to have the infinite distance. After the projection process for all final gather rays at A and B are complete, the 2D array in C ends up being mostly filled. By taking the average of all irradiance values in the array, the final gather value at C is computed without shooting any final gather rays.

However, there is no guarantee that the array will always be filled. For example, if some area is visible from C but not from A and B, the values in the array corresponding to the direction of the area are never filled (figure 6.77).

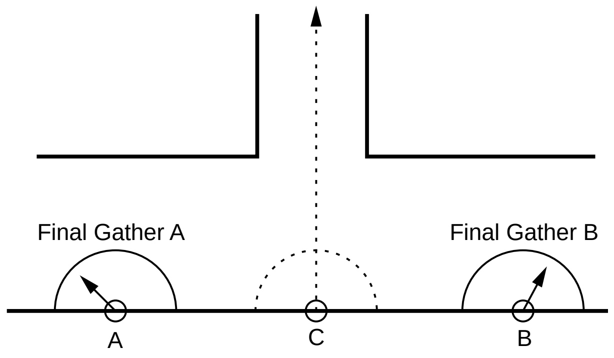


Figure 6.77: Hidden area problem

There is also a case where the projected point P is actually hidden from C (figure 6.78). In this case, C will know that there is an object to the direction of P, but will not be able to determine the irradiance.

By using more final gather sample points for the reprojection, there will be more chance to avoid the problems mentioned above. In fact, Kilauea uses neighboring four final gather sample points in order to fill the 2D array by reprojection as much as possible.

Z-buffer in reprojection

For the transformation of final gather ray information from A and B to C, the occlusion must be considered somehow. For instance, final gather ray hit points,

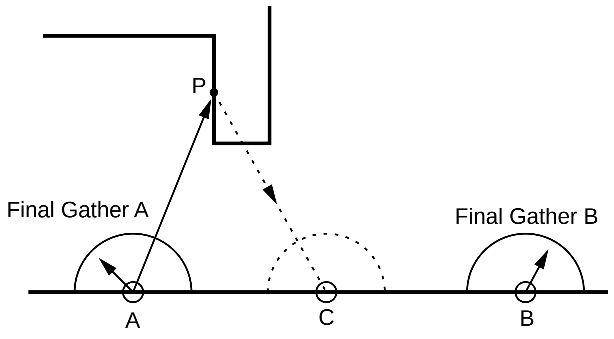


Figure 6.78: Back face problem

P_a and P_b, may be projected onto the same point from point C (figure 6.79). Z-buffer is used to correctly pick the closer hit point.

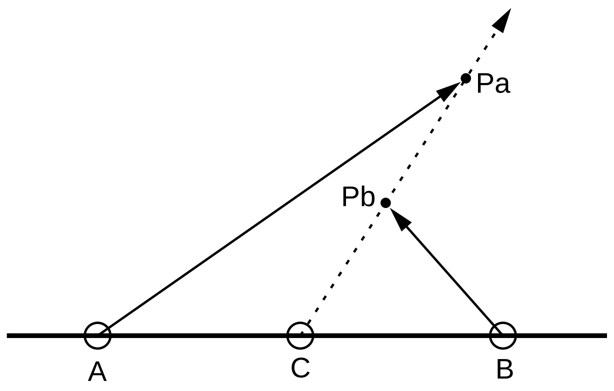


Figure 6.79: Need for Z-buffer

The final gather ray tracing of the photon map method terminates when the ray hits an object, even if it is transparent. That is, all objects are treated to be opaque. This property is convenient for incorporation of Z-buffer, since it cannot handle transparency easily.

The resolution of Z-buffer is determined by the resolution of final gathering, as explained in section 6.3.3. Because the final accuracy demanded at C is the same as neighboring final gather sample points, there is no need to have any higher resolution. In the current implementation of Kilauea, the final gathering resolution is fixed per rendering frame, and thus the Z-buffer resolution becomes constant throughout the rendering. This property greatly simplifies the memory management.

When writing into the Z-buffer after the reprojection, the distance that the ray traveled is used as Z. If all pixels in the Z-buffer is filled, the final gather value can be computed by averaging the irradiance values in the Z-buffer.

Area after reprojection

A problem occurs after the reprojection because the final gather hit point does not have a size. As illustrated in figure 6.80, the ray hit points from A on ceiling should be occluded by the ray hit points from B on the object in front, but some ray information from A may pass through and incorrectly affect the final gather value at C.

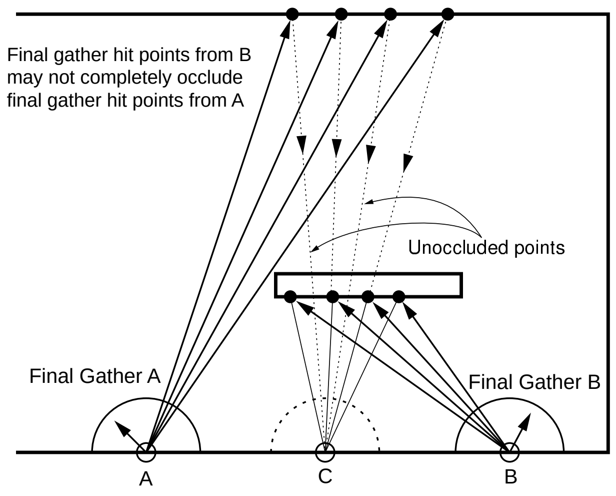

Final gather hit points from B
may not completely occlude
final gather hit points from A
Unoccluded points
Final Gather A
Final Gather B
A
C
B

Figure 6.80: Need for Z-buffer

This problem evidently occurs when the distance of the final gather ray is reduced in the reprojection and the resulting Z-buffer has many holes. This effect is nearly negligible when the final gather sample points are in a spatially large region and are relatively close to each other. However, in the final gather estimation, the analysis of the harmonic mean distances should cause the reprojection in such optimal regions to be skipped from the first place. Thus, the final gather reprojection must carefully consider the size of the point after the transform and correctly handle occlusion.

If the final gather hit point is projected to have shorter final gather distance (figure 6.81), the point is enlarged and its value is distributed to neighboring elements in the Z-buffer. The correct enlargement of the final gather hit points is difficult because final gathering is essentially just point sampling and has no area. The approach described below is fake, yet sufficiently accurate and fast.

If the distance the final gather ray traveled is halved, the area in the Z-buffer is quadrupled – four pixels in the Z-buffer are filled. The region to fill is a circle centered around the reprojected point. Implementation-wise, because the low resolution does not allow precise filling of the circle, pixels are randomly plotted depending on the area in the circle. These pixels are set with the same value.

If the final gather hit point is projected to have longer final gather distance

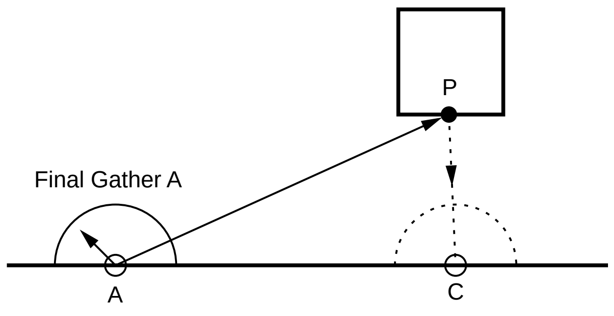


Figure 6.81: Shorter final gather distance

(figure 6.82), it is just treated as a point and the above procedures are all skipped.

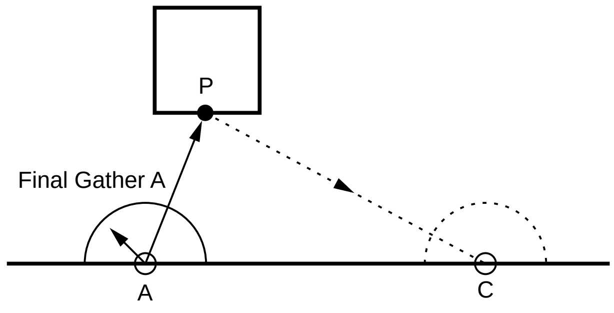


Figure 6.82: Longer final gather distance

Unresolved pixels

After going through the reprojection procedures described so far, the Z-buffer will still have unresolved pixels. If the space in the final gather directions corresponding to the location of the unresolved pixels is not visible from the original final gather sample points, filling those pixels just by reprojection is impossible.

However, if the surrounding pixels have valid values, the unresolved pixels are filled by interpolation from their neighbors. The obtained results are usually not far off from the correct final gather results, because such dispersed holes in the Z-buffer are most likely the result of the low sampling rate, not because those pixels were actually not seen from the original final gather sample points. because such dispersed holes in the Z-buffer are due to the low sampling rate and not the result of those pixels actually unseen from the original final gather sample points.

If the unresolved pixels in the Z-buffer are cluttered in one area, then interpolating the values within this area is dangerous. Unless the values for all unresolved pixels can be obtained somehow, the final gather value computation gives up on reprojection and switches back to regular final gathering. This conservative approach keeps the robustness of the algorithm.

Section 6.3.3 discusses partial final gather ray shooting to fill the unresolved pixels.

Problem: Aliasing in Z-buffer

With the discussion so far, the final gather value can be computed by reprojecting the final gather information into a Z-buffer and taking the average of the colors in the buffer. If the precision of the reprojection is insufficient, low frequency noise usually appears in the final image. Unfortunately, the reprojection algorithm needs more careful attention to the precision for obtaining better quality in many scenes.

The image 6.83 shows the final gather reprojection result at a point in the scene in figure 6.84. The circled region in the image heavily suffers from the aliasing problem caused by thin bars. Image 6.85 is the high resolution version, showing how the image 6.83 should appear. Image 6.86 has the original resolution and is computed from the high resolution image. For computing the final gather values, the precision in the last image is mandatory.

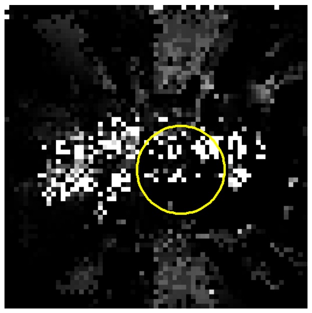


Figure 6.83: Aliasing in Z-buffer

The solution is not as easy as just increasing the resolution of the whole Z-buffer. The data to be written in the Z-buffer are the ray data from neighboring final gather sample points, which are essentially just point data. More information must be computed to fill the high-resolution Z-buffer, which in turn might end up slower than brute force final gathering. The aliasing must be suppressed by a clever technique which minimizes the computational cost and memory usage.

The examination of the image suffering from aliasing indicates that the problem is concentrated in the area with significant change in color. Generally, the aliasing is somewhat relieved by supersampling this problematic region. The reprojected final gather hit points may be overlapping in a pixel in the problematic region, but discarded by a simple Z comparison. To prevent important ray information from getting discarded, one might consider taking the average color for that pixel instead of doing Z comparison. However, treating occlusion correctly is extremely important, as explained in section 6.3.3.

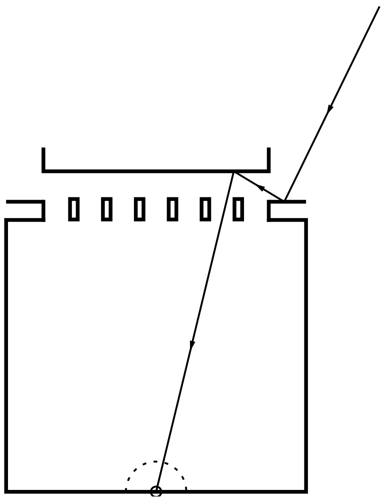


Figure 6.84: Aliasing scene

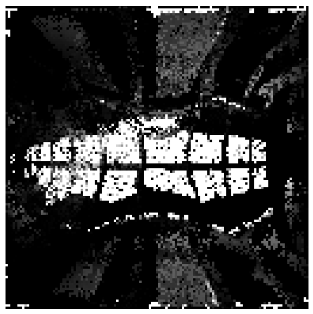


Figure 6.85: High resolution Z-buffer

Thoughtless increase of the resolution may also defeat the purpose of Z-buffer because the reprojected data will only be plotted sparsely in the buffer, without any overlap. There will be no Z comparison as a result, and the occlusion will not be solved at all.

Simple Z-buffer is sufficient in most pixels, and they should stick to the current methodology for computational cost. Therefore, the mechanism to switch between Z-buffering and supersampling depending on the difference in the color of neighboring pixels is necessary.

Solution: Multi-resolutional Z-buffer

One idea to achieve the switching of Z-buffering and supersampling is to keep all the final gather information stored in their corresponding pixel in the Z-buffer. This way, the algorithm may later choose to do full supersampling using all stored information or just do Z comparison.

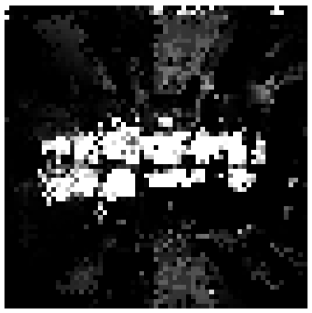


Figure 6.86: Antialiased Z-buffer

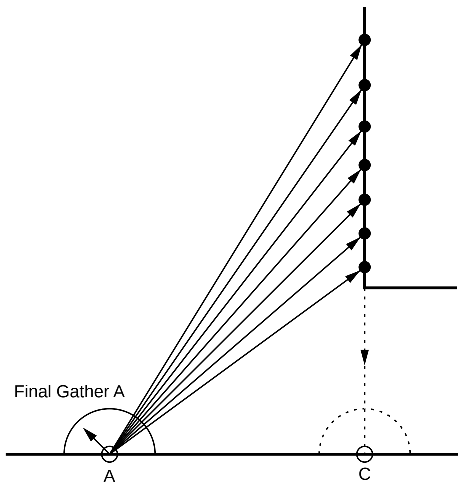


Figure 6.87: Projection to same coordinates

However, in extreme cases as illustrated in figure 6.87, many final gather hit points may end up projected to the same Z-buffer pixel and end up storing too many data in it. The mechanism to discard some apparently unnecessary data must be incorporated. Also, variable storage size per pixel complicates the memory management. The simplicity of fixed-size Z-buffer memory management is lost in this method.

To compensate the problem, Kilauea uses multi-resolutional Z-buffer, meaning that each pixel in Z-buffer has some subpixel resolution. The resolution of the Z-buffer is determined as in the previous way; if the neighboring final gather sample points used 65×65, Z-buffer resolution is also 65×65. To prepare for possible supersampling, each pixel is given 4×4 or 8×8 subpixel resolution.

8×8 supersampling is more than enough for antialiasing of edges, and is set as the maximum Z-buffer subpixel resolution. Now that the maximum amount of needed memory is known, the efficient memory management of the Z-buffer is a simple task.

The reprojected final gather data are stored in the subpixels of the Z-buffer. In the end, some final gather data may end up getting discarded by Z comparison, but the subpixel resolution allows many of them to be intact. Because Z comparison is performed at each subpixel, the occlusion is correctly considered.

After the reprojection, the following steps are taken to determine whether a pixel should perform Z comparison or supersampling. First, for each pixel in the Z-buffer, the closest ray information is chosen, and used as the color for this pixel. This yields the same image as from simple Z-buffering. Then, for each pixels in this image, surrounding pixel colors are compared. If the change in color is over a certain threshold, the average of all colors in the subpixels becomes the value for this pixel. This is essentially doing supersampling for this pixel. The pixels under the threshold simply use the result from Z-buffering.

After all Z-buffer pixels are processed, the final gather value is at last computed by averaging colors of all pixels in the Z-buffer.

In this multi-resolutional Z-buffer algorithm, the computational cost for processing the pixels which were determined to use Z-buffering increased only slightly, compared to the previous straightforward implementation of Z-buffer. For those pixels which were determined to use supersampling, because the pixels have to always compute the color using Z-buffer, the computational cost is higher. However, the supersampling is only operated on minority of pixels with significant change in color.

To conclude, the multi-resolutional Z-buffering effectively achieved anti-aliasing with minimum computational cost and efficient memory usage.

The white dots in the images 6.88 and 6.89 show the positions where full final gathering had to be executed in the course of uniformly spaced final gathering and adaptive subdivision. The number of dots apparently decreased compared to the previous images.

From these results, the amount of the final gathering in the region of safe pixels are reduced to 1.99% for Sponza and 0.76% for Fiat 500L. Without final gather reprojection, the reduction rates are 3.95% for Sponza and 2.43% for Fiat 500L.

The images 6.90 and 6.91 shows the final gather values computed with final gather reprojection turned on. The quality of images 6.63 and 6.64 not using final gather reprojection is still retained.

Partial final gather reshooting

As pointed out in section 6.3.3, the final gather reprojection may not be able to resolve all pixels in the Z-buffer. If one or more unresolved pixels are found, the algorithm just gives up and falls back to regular final gathering.

However, because many pixels in the Z-buffer already have valid values, starting all over from scratch seems to be wasteful. If only appropriate final gather rays can be shot to the direction corresponding to the location of unresolved pixels,

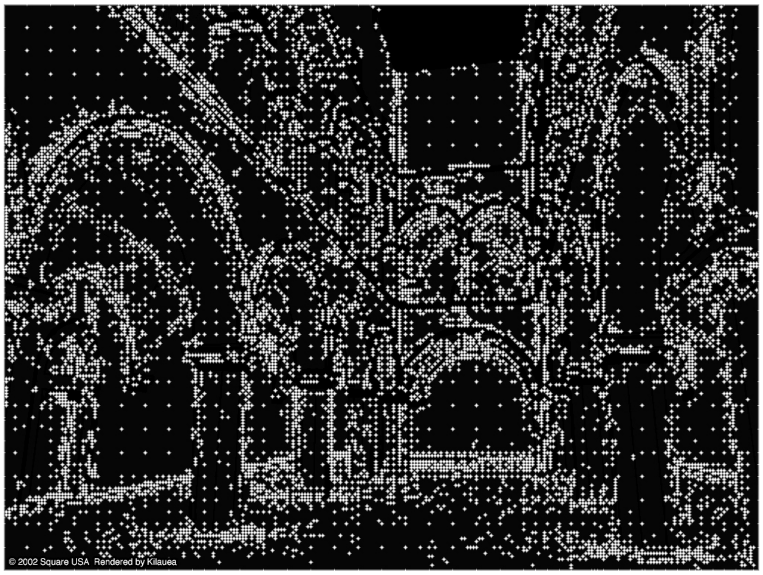


Figure 6.88: Sponza final gather reprojection

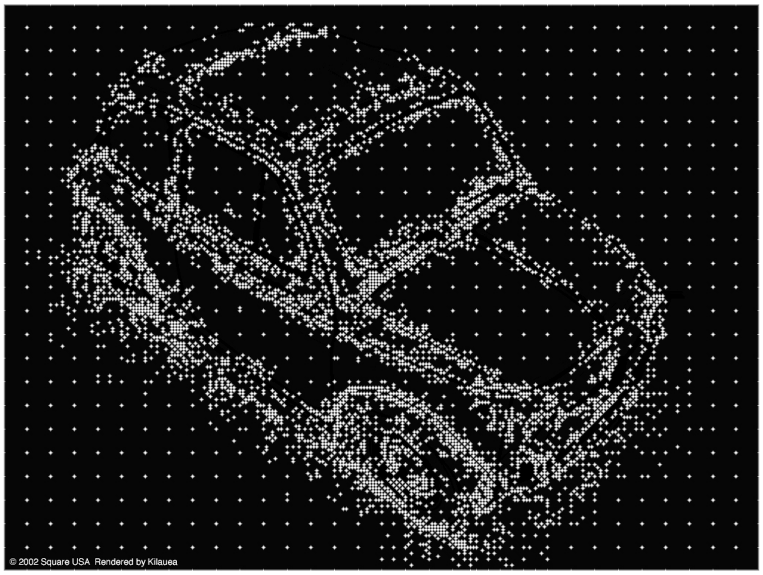


Figure 6.89: Fiat 500L final gather reprojection

even further reduction of final gather rays can be achieved. This is implemented in Kilauea by slightly modifying the final gathering logic.

In order to partially shoot final gather rays, the Z-buffer with unresolved pixels is saved, and then the final gathering is started. Before the ray is shot, Z-buffer is checked to see if it already has the irradiance corresponding to the direction of the ray. If so, then shooting of the ray is skipped and the irradiance from Z-buffer is

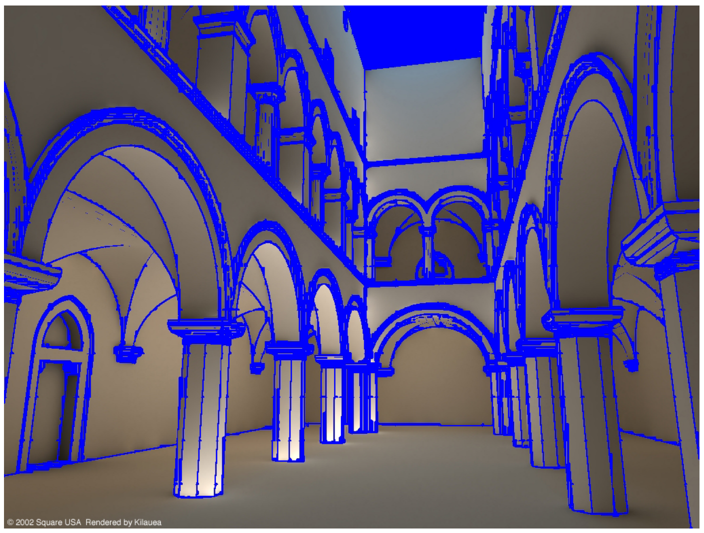


Figure 6.90: Sponza final gather estimation w/ reprojection

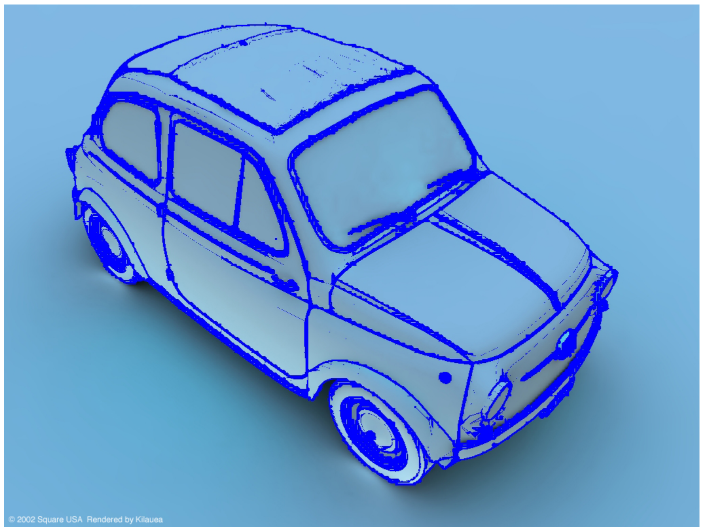


Figure 6.91: Fiat 500L final gather estimation w/ reprojection

used instead. If the corresponding pixel in Z-buffer is unresolved, the ray is shot and the resulting irradiance is written in the Z-buffer.

This partial final gather reshooting approach successfully achieves further final gather ray reduction. The following numbers show how many rays were required to compute the sample images using this approach.

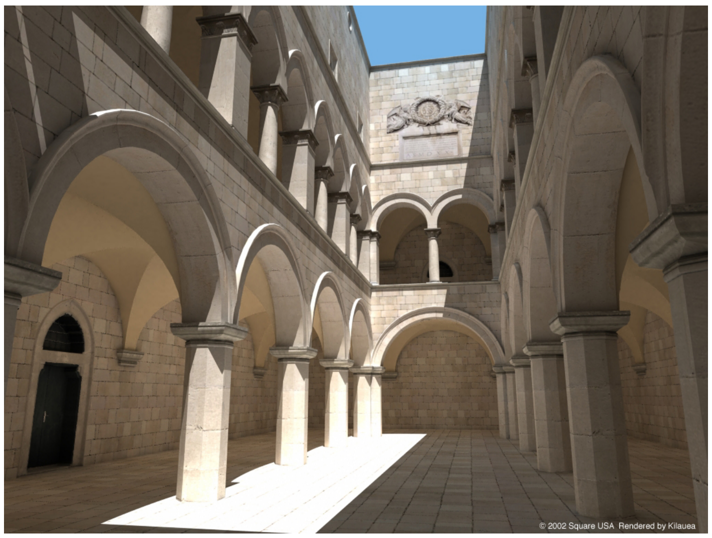


Figure 6.92: Sponza final result

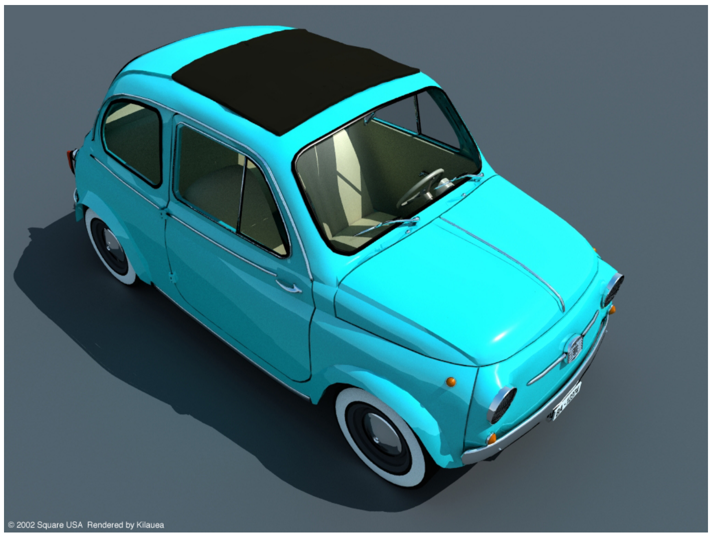


Figure 6.93: Fiat 500L final result

- Sponza

  Total pixels: 786,432

  Safe pixels: 612,973

  Final gather sample points: 12,223

  Total final gather rays: 14,862,569

- Fiat 500L

Total pixels: 786,432

Safe pixels: 718,907

Final gather sample points: 5,533

Total final gather rays: 2,252,958

In the safe pixels, Sponza required 18.9 final gather rays per pixel, and Fiat required only 3.1 final gather rays per pixel. This is a drastic speed-up, considering that the image quality is equivalent to the result from brute-force rendering with 65×65 final gather rays per pixel.

Consideration of final gather reprojection

Final gather reprojection, in conjunction with final gather estimation, achieved high quality renderings while using minimal number of rays. For regions identified as safe pixels, the quality of 65×65 final gathering is attained. The critical pixels, however, need to do brute force final gathering. Improving the speed in the critical region is a future work.

Final gather reprojection itself is computationally expensive, but it is found to be effective in most test cases. This implies that straightforward final gathering in Kilauea is even slower. Because of the message passing overhead, Kilauea's ray tracing is relatively slow. Final gather reprojection successfully covers this weakness by remarkably reducing the number of final gather rays.

For very simple scenes, final gather reprojection may turn out to be slower than simple final gathering. To compensate, final gather reprojection can be turned off as the user desires.

However, for many complex scenes, final gather reprojection works extremely well. Because the cost for ray tracing becomes larger as the scene becomes more complex, and the cost for final gather reprojection roughly stays constant for the change in the scene complexity, the final gather reprojection is especially powerful on complex scenes. For a massively parallel renderer based on message passing like Kilauea, the property that the final gather reprojection process can always be executed independently is an additional benefit.

Final gather reprojection computation involves repetitive computation to fixed size data, which makes it suitable for hardware implementation. If portion of the final gather reprojection algorithm can be implemented in hardware, the global illumination rendering may undergo a breakthrough in speed-up.

Final gather estimation cache

The final gather estimation computes final gather values for pixels in the screen. The values are only computed in safe pixels, and for critical pixels, the null flags are set to indicate that there are no values.

With the resolution of 1024×768, the size of this final gather estimation results turns out to be about 8MB to 10MB. The size of data depends on the resolution and the ratio of safe pixels and critical pixels. To avoid this data from consuming too much memory for high resolution renderings, the data is cached to a local disk. This file is referred from the shader in the final rendering stage.

For more description on each stages in multi-pass rendering in Kilauea, the data written out by the final gather estimation stage, and how this data file is used, please refer to [Kilauea].

Further improvements

The final gather estimation is only applied to the surface that the primary ray hit. The surfaces seen as a result of reflection or transmission are fully final gathered. Applying the same principle to those surfaces is not a problem at all – it is just that Kilauea first limited its final gather estimation implementation to the surface directly seen from camera for simplicity. Now that the effectiveness of final gather estimation is proven, the support for reflection/transmission is ready to be included.

6.4 Conclusions

The first half of this course note covered handy tips and tricks when introducing a photon map renderer into actual production. The problems related to caustics are especially difficult to troubleshoot without the correct understanding of photon map and global illumination. Even with the thorough understanding, parameter tuning requires strong logical reasoning skills, not just some gut feeling. Perhaps more simple and intuitive quality control mechanism built into the renderer is demanded.

The latter half focused on Kilauea's approach in speeding up the final gathering stage of the photon map rendering. Global illumination computation evidently requires more computational power than the conventional local illumination computation. Final gather estimation and final gather reprojection effectively cut down the number of final gather rays needed, while maintaining the quality of images. In the future, even further speed-up techniques will need to be developed.

The creation of global illumination scenes does not work in the same way as the creation of direct illumination scenes. The production pipeline based on direct illumination model relies on the fact that all components of the scene can be controlled independently. This principle does not apply in global illumination, where every component affect each other. The staff members working in the direct illumination based production pipeline and the developers of the global illumination renderer need to address this issue in order to come up with a totally new workflow.

From the pure technical stand point in writing renderers, global illumination computation to correctly simulate light and produce high quality images is one of the most promising future directions. Monte Carlo ray tracing along with photon mapping has a great potential, especially considering its extensibility and parallel performance.

Currently, Kilauea had just passed through the experimental stage and was undergoing beta testing. Further improvements such as introducing SSE instructions to many low-level repetitive operations still remain, and it is truly unfortunate that the Kilauea Research Project is discontinued at this point.

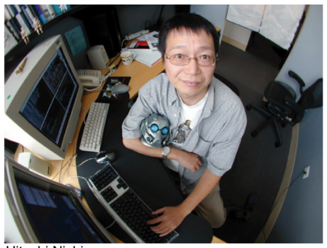
6.5 Acknowledgements

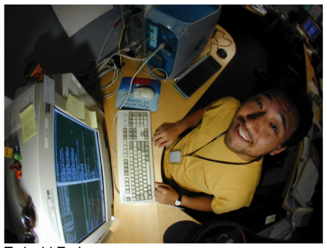
We show our sincere gratitude to Kazuyuki Hashimoto, Kaveh Kardan, and all members of Square USA R&D division for provided us with numerous advice and ideas during four years in the Kilauea Research Project. This course note would not have completed without the assistance of Junichi Kimura of Square USA R&D division.

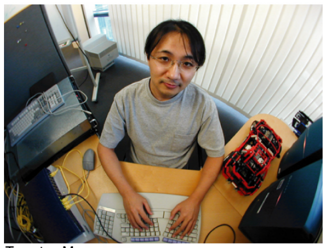
Many thanks to Alan Chalmers of the University of Bristol for advice on parallel rendering, and Henrik Wann Jensen of Stanford University for advice on global illumination and photon mapping.

Sponza model was provided by the courtesy of Marko Dabrovic. Many test-suite models are from De ESPONA 3D Encyclopedia 2000 Edition.

Toshi Kato


Hitoshi Nishimura


Tadashi Endo


Tamotsu Maruyama

Jun Saito

Motohisa Adachi

Chapter 7

References and further reading

This section lists the references referenced in these course notes plus additional background material relevant to the photon map method. The material is divided into three groups: the photon map method, ray tracing and photon tracing and data-structures with focus on kd-trees. Each part is in chronological order with annotations. In addition we have listed a number of animations rendered with photon maps and finally we have provided a more detailed list of relevant background literature.

The photon map method

[Jensen95a] Henrik Wann Jensen and Niels Jørgen Christensen.
"Photon maps in Bidirectional Monte Carlo Ray Tracing of Complex Objects".
Computers & Graphics **19** (2), pages 215–224, 1995.
The first paper describing the photon map. The paper suggested the use of a mixture of photon maps and illumination maps, where photon maps would be used for complex surfaces such as fractals.

[Jensen95b] Henrik Wann Jensen.
"Importance driven path tracing using the photon map".
Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering), pages 326–335. Springer Verlag, 1995.
Introduces the use of photons for importance sampling in path tracing. By combining the knowledge of the incoming flux with the BRDF it is possible to get better results using fewer sample rays.

[Jensen95c] Henrik Wann Jensen and Niels Jørgen Christensen.
"Efficiently Rendering Shadows using the Photon Maps".

In *Proceedings of Compugraphics'95*, pages 285–291, Alvor, December 1995.
Introduces the use of shadow photons for an approximate classification of the light source visibility in a scene.

[Jensen96a] Henrik Wann Jensen.
"Rendering caustics on non-Lambertian surfaces".
Proceedings of Graphics Interface'96, pages 116-121, Toronto, May 1996 (also selected for publication in Computer Graphics Forum, volume 16, number 1, pages 57–64, March 1997). Extension of the photon map method to render caustics on non-Lambertian surfaces ranging from diffuse to glossy.

[Jensen96b] Henrik Wann Jensen.
"Global illumination using photon maps".
Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering), pages 21–30. Springer Verlag, 1996.
Presents the global illumination algorithm using photon maps. A caustic and a global photon map is used to optimize the rendering of global illumination including the simulation of caustics.

[Jensen96c] Henrik Wann Jensen.
The photon map in global illumination.
Ph.D. dissertation, Technical University of Denmark, September 1996.
An in-depth description of the photon map method based on the presentations in the published photon map papers.

[Christensen97] Per H. Christensen.
"Global illumination for professional 3D animation, visualization, and special effects" (invited paper).
Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering), pages 321–326. Springer Verlag, 1997.
Describes the requirements of a global illumination method in a commercial environment, and motivates the choice of the photon map method.

[Myszkowski97] Karol Myszkowski.
"Lighting reconstruction using fast and adaptive density estimation techniques".

Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering), pages 321–326. Springer Verlag, 1997.
Efficient techniques for filtering and visualizing photons.

[Slusallek98] Philipp Slusallek, Mark Stamminger, Wolfgang Heidrich, J.-C. Popp, and Hans-Peter Seidel.
"Composite Lighting Simulations with Lighting Network".
IEEE Computer Graphics & Applications, 18(2), pages 22-31, March/April 1998.
Describes a framework in which the photon map can be integrated into a radiosity simulation.

[Peter98] Ingmar Peter and Georg Pietrek.
"Importance driven construction of photon maps."
Rendering Techniques '98 (Proceedings of the Ninth Eurographics Workshop on Rendering), pages 269–280. Springer Verlag, 1998.
Use of importance to focus the photons where they contribute most to the visible solution. This requires an initial importance (or "importons") tracing pass from the camera before the photon tracing pass from the light sources.

[Jensen98] Henrik Wann Jensen and Per H. Christensen.
"Efficient simulation of light transport in scenes with participating media using photon maps".
Proceedings of SIGGRAPH 98, pages 311–320. ACM, 1998.
Extension of the photon map method to simulate global illumination in scenes with participating media.

[Lange98] Thorsten Lange and Georg Pietrek.
"Rendering Participating Media using the Photon Map".
Technical Report no. 678, University of Dortmund, 1998.
Also describes the extension of the photon map method to simulate global illumination in the presence of participating media.

[Jensen99] Henrik Wann Jensen, Justin Legakis and Julie Dorsey.
"Rendering of Wet Materials".
Proceedings of the Tenth Eurographics Workshop on Rendering, pages 281-290, Granada, June 1999.
Simulates subsurface scattering using the volume photon map in order to render wet materials.

[Dorsey99] Julie Dorsey, Alan Edelman, Henrik Wann Jensen, Justin Legakis and Hans Køhling Pedersen.
"Modeling and Rendering of Weathered Stone".
Proceedings of SIGGRAPH 99, pages 223–234, 1999.
Describes rendering of volumetric weathering effects in stone based on subsurface scattering optimized using the volume photon map.

[Christensen99] Per H. Christensen
"Faster Photon Map Global Illumination".
Journal of Graphics Tools, 4(3), pages 1–10, 1999.
Introduces precomputed irradiance values per photon for faster look-ups.

[Jensen00] Henrik Wann Jensen.
"Parallel Global Illumination using Photon Mapping".
In *SIGGRAPH'2000, Course 30*, New Orleans, July 2000.
Describes how to implement the photon mapping algorithm to take advantage of multi-processor/multi-host computers.

[PMAPCourse] SIGGRAPH 2000 Course Note.
"A Practical Guide to Global Illumination Using Photon Maps".
Previous SIGGRAPH course on photon mapping.

[Suykens00] Frank Suykens and Yves Willems.
"Density control for photon maps".
Rendering Techniques 2000 (Proceedings of the Eleventh Eurographics Workshop on Rendering), pp. 11–22. Springer-Verlag, 2000.
Introduces techniques for limiting the density of the photons in order to get a better distribution of photons. Also presents ideas for using visual importance to construct higher quality photon maps.

[RPK] Ph. Bekaert and F. Suykens.
RenderPark, a physically based rendering tool.
K.U. Leuven, http://www.renderpark.be, 1996-2001.
An open-source renderer that supports photon mapping.

[Jensen01] Henrik Wann Jensen.
Realistic Image Synthesis using Photon Mapping.
AK Peters, 2001
An in-depth book describing photon mapping, all the theory,

and all the practical details. Includes an implementation of the photon map data structure.

Ray tracing and photon tracing

[Whitted80] Turner Whitted.
"An improved illumination model for shaded display".
Communications of the ACM, volume 23, number 6, pages 343–349. ACM, June 1975.
The classic ray tracing paper.

[Arvo86] James Arvo.
"Backward ray tracing".
Developments in ray tracing, SIGGRAPH 86 seminar notes. ACM, August 1986.
Introduces light ray tracing and illumination maps for computing caustics.

[Glassner89] Andrew S. Glassner.
An introduction to ray tracing.
Academic Press, 1989.
The standard reference on ray tracing. Still a pleasure to read.

[Shirley91] Peter Shirley.
Physically Based Lightning Calculations for Computer Graphics.
Ph.d. thesis, University of Illinois at Urbana-Champaign, 1991.
Good overview of Monte Carlo ray tracing. Also presents one of the first practical multi-pass global illumination methods.

[Chen91] Eric Shenchang Chen, Holly E. Rushmeier, Gavin Miller, and Douglas Turner.
"A progressive multi-pass method for global illumination".
Proceedings of SIGGRAPH 91, pages 164–174. ACM, 1991.
One of the first multi-pass global illumination methods. Uses illumination maps for caustics, radiosity for indirect light and path tracing for rendering.

[Ward92] Gregory Ward and Paul Heckbert.
"Irradiance gradients".
Third Eurographics Workshop on Rendering, pages 85–98. Eurographics, 1992.
Describes the irradiance gradients method which is used for the final gathering step of the photon map method.

[Pattanaik93] Sumant N. Pattanaik.
”Computational Methods for Global Illumination and Visualisation of Complex 3D Environments”.
Ph.d. Thesis, Birla Institute of Technology & Science, 1993
Introduces particle tracing where photons are emitted from the light sources and stored in a mesh.

[Rushmeier93] Holly Rushmeier, Ch. Patterson and A. Veerasamy.
”Geometric Simplification for Indirect Illumination Calculations”.
Proceedings of Graphics Interface ’93, pages 35-55, 1994.
Introduces the concept of geometry simplification for the radiosity step of multipass global illumination computations.

[Glassner95] Andrew S. Glassner.
Principles of digital image sythesis.
Morgan Kaufmann, 1995.
Gives an excellent overview of the entire field of image synthesis. Of particular interest here is the description of Monte Carlo photon tracing and Russian roulette.

[Lafortune96] Eric P. Lafortune.
Mathematical Models and Monte Carlo Algorithms for Physcially Based Rendering.
Ph.d. thesis, Katholieke University, Leuven, Belgium 1996.
Good overview of Monte Carlo ray tracing techniques including bidirectional path tracing.

[Dutre96] Philip Dutré and Yves D. Willems.
Mathematical Frameworks and Monte Carlo Algorithms for Global Illumination in Compute Graphics.
Ph.d. thesis, Katholieke Universiteit Leuven, 1996.
Another fine overview of Monte Carlo ray tracing and photon tracing.

[Ward98] Gregory Ward Larson and Rob Shakespeare.
Rendering with Radiance — the art and science of lighting visualization.
Morgan Kaufmann, 1998.
An entire book dedicated to the excellent Radiance renderer with many practical examples.

Datastructures

[Bentley75] Jon L. Bentley.
"Multidimensional binary search trees used for associative searching".
Communications of the ACM, volume 18, number 9, pages 509–517. ACM, 1975.
First paper on the kd-tree datastructure.

[Preparata85] Franco P. Preparata and Michael Ian Shamos.
Computational Geometry An Introduction, Springer-Verlag, 1985

[Cormen89] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest.
Introduction to algorithms.
MIT Press, 1989.
Good overview of algorithms including the heap-datastructure.

[Sedgewick92] Robert Sedgewick.
Algorithms in C++.
Addison-Wesley, 1992.
Also good description of the heap structure, and algorithms for the median search (used in the balancing algorithm).

Other references

These are additional useful references for ray tracing based rendering methods.

[Ansi86] American National Standard Institute.
"Nomenclature and Definitions for Illumination Engineering".
ANSI report, ANSI/IES RP-16-1986, 1986.

[Arvo90] James Arvo and David Kirk.
"Particle Transport and Image Synthesis".
Computer Graphics **24** (4), pages 53–66, 1990.

[Aupperle93] Larry Aupperle and Pat Hanrahan:
"A Hierarchicah Illumination Algorithm for Surfaces with Glossy Reflection".
Computer Graphics, pages 155–162, 1993.

[Bentley79a] Jon Louis Bentley.
"Multidimensional Binary Search Trees in Database Applica-

tions".
IEEE Trans. on Soft. Eng. **5** (4), pages 333–340, July 1979.

[Bentley79b] Jon Louis Bentley and Jerome H. Friedman.
"Data Structures for Range Searching".
Computing Surveys **11** (4), pages 397–409, 1979.

[Bentley80] Jon Louis Bentley, Bruce W. Weide, and Andrew C. Yao.
"Optimal Expected-Time Algorithm for Closest Point Problems".
ACM Trans. on Math. Soft., **6** (4), pages 563–580, dec. 1980.

[Chalmers] Alan Chalmers et al. "Practical Parallel Rendering". ISBN: 1-56881-179-9, A K Peters, 2002.

[Christensen93] Per Henrik Christensen, David Salesin and Tony DeRose.
"A Continuous Adjoint Formulaion for Radiance Transport".
Fourth Eurographics Workshop on Rendering, pages 95–104, 1993

[Christensen95] Per Henrik Christensen.
Hierarchical Techniques for Glossy Global Illumination.
PhD thesis, Seattle, Washington, 1995.

[Collins94] Steven Collins.
"Adaptive Splatting for Specular to Diffuse Light Transport".
In *Proceedings of the 5th Eurographics Workshop on Rendering*, pages 119–135, Darmstadt 1994.

[Cook84] Robert L. Cook.
"Distributed Ray Tracing".
Computer Graphics **18** (3), pages 137–145, 1984.

[Cook86] Robert L. Cook.
"Stochastic Sampling in Computer Graphics".
ACM Transactions on Graphics **5** (1), pages 51–72, Jan. 1986.

[Dutre94] Philip Dutré and Yves D. Willems.
"Importance-driven Monte Carlo Light Tracing".
In *proceedings of 5. Eurographics Workshop on Rendering*, pages 185–194, Darmstadt 1994.

[Dutre95] Philip Dutre and Yves D. Willems.
"Potential-Driven Monte Carlo Particle Tracing for Diffuse Environments with Adaptive Probability Density Functions".

In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95*, pages 306–315, New York, NY, 1995. Springer-Verlag.

[Ebert94] David Ebert, Ken Musgrave, Darwyn Peachey, Ken Perlin and Steve Worley.
Texturing and Modeling: A Procedural Approach.
Academic Press, October 1994.

[Goral84] Cindy Goral, Kenneth Torrance, Donald Greenberg, Bennet Battaile. "Modeling the Interaction of Light Between Diffuse Surfaces". *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, number 3, pages 213-222, July 1984, Minneapolis, Minnesota.

[Gritz96] Larry Gritz and J. K. Hahn.
"BMRT: A Global Illumination Implementation of the RenderMan Standard".
Journal of Graphics Tools, Vol. 1, No. 3, pages 29-47, 1996.

[Hall88] Roy Hall.
Illumination and Color Computer Generated Imagery.
Springer-Verlag, 1988

[Heckbert90] Paul S. Heckbert.
"Adaptive Radiosity Textures for Bidirectional Ray Tracing".
Computer Graphics **24** (4), pages 145–154, 1990.

[Horowitz93] Ellis Horowitz, Sartaj Sahni and Susan Anderson-Freed.
Fundamentals of Data Structures in C, Computer Science Press, 1993

[Igehy99] Homan Igehy. Tracing ray differentials. *Computer Graphics*, 33(Annual Conference Series):179–186, 1999.

[Jensen93] Henrik Wann Jensen.
Global Illumination using Bidirectional Monte Carlo Ray Tracing.
M.Sc. thesis, Technical University of Denmark (in Danish), 1993.

[Jensen95f] Henrik Wann Jensen and Niels Jørgen Christensen.
"Optimizing Path Tracing using Noise Reduction Filters".
In *Proceedings of WSCG 95*, pages 134–142, Plzen 1995.

[Kajiya86] James T. Kajiya.
“The Rendering Equation”.
Computer Graphics **20** (4), pages 143–149, 1986.

[Kalos86] M. Kalos and P. Whitlock. *Monte Carlo Methods, Volume 1: Basics*. J. Wiley, New York, 1986.

[Keller96] Alexander Keller.
“Quasi-Monte Carlo Radiosity”.
In proceedings of *7th Eurographics Workshop on Rendering*, pages 102–111, Porto 1996.

[Keller00] Alexander Keller and Ingo Wald.
“Efficient Importance Sampling Techniques for the Photon Map”.
In *Vision Modelling and Visualization 2000*, pages 271–279, Saarbruecken, Germany, 2000.

[Kilauea] Kilauea, SquareUSA’s rendering software with photon maps.
`http://www.squareusa.com/kilauea/`.

[Kopp99] Nathan Kopp.
Personal communication.

[Lafortune93] Eric P. Lafortune and Yves D. Willems.
“Bidirectional Path Tracing”.
In *Proceedings of CompuGraphics*, pages 95–104, 1993.

[MegaPov00] A free ray tracer that supports photon maps.
Source code and examples are available at:
`http://www.nathan.kopp.com/patched.htm`,
Mar. 2000

[Nicodemus77] F. E. Nicodemus, J. C. Richmond, J. J. Hsia. I. W. Ginsberg and T. Limperis:
“Geometric Considerations and Nomenclature for Reflectance”.
National Bureau of Standards, 1977

[Niederreiter92] Harald Niederreiter.
Random Number Generation and Quasi-Monte Carlo Methods, SIAM, 1992.

[Pattanaik95] S. N. Pattanaik and S. P. Mudur.
“Adjoint equations and random walks for illumination computation”.
ACM Transactions on Graphics, 14(1):77–102, January 1995.

[Pavicic90] Mark J. Pavicic.
"Convenient Anti-Aliasing Filters that Minimize Bumpy Sampling".
In *Graphics Gems I*, eds. Andrew S. Glassner, pages 144–146, 1990.

[Pharr97] Matt Pharr, Craig Kolb, Reid Gershbein, Pat Hanrahan. "Rendering Complex Scenes with Memory-Coherent Ray Tracing". *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 101-108, August 1997, Los Angels, California.

[Rubinstein81] Reuven Y. Rubinstein.
Simulation and the Monte Carlo Method.
John Wiley & Sons, 1981.

[Rushmeier88] Holly Rushmeier.
Realistic Image Synthesis for Scenes with Radiatively Participating Media.
Ph.d. thesis, Cornell University, 1988.

[Schlick93] Christophe Schlick.
"Customizable Reflectance Model for Everyday Rendering". In *proceedings of 4. Eurographics Workshop on Rendering*, pages 73-84, Paris 1993.

[Shirley90] Peter Shirley.
"A Ray Tracing Method for Illumination Calculation in Diffuse-Specular Scenes". *Proceedings of Graphics Interface '90*, pages 205–212, 1990.

[Shirley92] Peter Shirley.
"Nonuniform Random Point Sets via Warping".
Graphics Gems III (David Kirk ed.), Academic Press, pages 80–83, 1992.

[Shirley95] Peter Shirley; Bretton Wade; Phillip Hubbard; David Zareski; Bruce Walter and Donald P. Greenberg.
"Global Illumination via Density Estimation".
In "Rendering Techniques '95". Eds. P.M. Hanrahan and W. Purgathofer, *Springer-Verlag*, pages 219-230, 1995.

[Shirley96] Peter Shirley; C. Wang and Kurt. Zimmerman.
"Monte Carlo Techniques for Direct Lighting Calculations".
ACM Transactions on Graphics **15** (1), 1996.

[Shirley00] Peter Shirley. “Realistic Ray Tracing”. ISBN: 1-56881-110-1, A K Peters, 2000.

[S2000Course38] SIGGRAPH 2001 Course Note. “A Practical Guide to Global Illumination Using Photon Mapping”.

[S2000Course40] SIGGRAPH 2001 Course Note. “Parallel Rendering and the Quest for Realism: The ’Kilauea’ Massively Parallel Ray Tracer”.

[Kilauea] SIGGRAPH 2002 Course Note. “The ’Kilauea’ Massively Parallel Global Illumination Renderer”.

[Silverman86] B.W. Silverman.
Density Estimation for Statistics and Data Analysis.
Chapmann and Hall, New York, NY, 1986.

[Smits92] Brian E. Smits, James R. Arvo, and David H. Salesin.
“An importance-driven radiosity algorithm”.
Computer Graphics, 26(2):273–282, July 1992.

[Suykens01] Frank Suykens and Yves D. Willems.
“Path Differentials and Applications”.
In ”Rendering Techniques 2001 (Proceedings of the Twelfth Eurographics Workshop on Rendering”, Eds. S.J. Gortler and K. Myszkowski. *Springer-Verlag*, pages 257–268, Londen, UK, 2001

[Suykens01TR] Frank Suykens and Yves D. Willems.
“Path differentials and applications”.
Technical Report CW307, Department of Computer Science, Katholieke Universiteit Leuven, Leuven, Belgium, May 2001.

[Tamstorf97] Rasmus Tamstorf and Henrik Wann Jensen.
“Adaptive Sampling and Bias Estimation in Path Tracing”.
In ”Rendering Techniques ’97”. Eds. J. Dorsey and Ph. Slusallek. *Springer-Verlag*, pages 285–295, 1997.

[Veach94] Eric Veach and Leonidas Guibas.
“Bidirectional Estimators for Light Transport”.
In *Proceedings of the 5th Eurographics Workshop on Rendering*, pages 147–162, 1994.

[Veach95] Eric Veach and Leonidas Guibas.
“Optimally Combinig Sampling Techniques for Monte Carlo Rendering”.
Computer Graphics **29** (4), pages 419–428, 1995.

[Veach97] Eric Veach and Leonidas Guibas.
"Metropolis Light Transport".
Computer Graphics **31** (3), pages 65–76, 1997.

[Volevich99] Vladimir Volevich, Karol Myszkowski, Andrei Khodulev and Edward A. Kopylov.
"Perceptually-Informed Progressive Global Illumination Solution".
Technical Report 99-1-002, University of Aizu, Japan, 1999.

[Ward88] Greg Ward, Francis M. Rubinstein, and Robert D. Clear.
"A Ray Tracing Solution for Diffuse Interreflection".
Computer Graphics **22** (4), pages 85-92, 1988.

[Ward91] Greg Ward.
"Real pixels".
In *Graphics Gems II*, James Arvo (ed.), *Academic Press*, pages 80-83, 1991.

[Zimmerman98] Kurt Zimmerman.
Density Prediction for Importance Sampling in Realistic Image Synthesis.
Ph.d. thesis, Indiana University, 1998.

This paper is a slightly extended version of the paper in *Rendering Techniques '96 (Proceedings of the Seventh Eurographics Workshop on Rendering)*, pages 21–30, 1996

Global Illumination using Photon Maps

Henrik Wann Jensen

Department of Graphical Communication
The Technical University of Denmark
`hwj@gk.dtu.dk, http://www.gk.dtu.dk/~hwj`

Abstract

This paper presents a two pass global illumination method based on the concept of photon maps. It represents a significant improvement of a previously described approach both with respect to speed, accuracy and versatility. In the first pass two photon maps are created by emitting packets of energy (photons) from the light sources and storing these as they hit surfaces within the scene. We use one high resolution caustics photon map to render caustics that are visualized directly and one low resolution photon map that is used during the rendering step. The scene is rendered using a distribution ray tracing algorithm optimized by using the information in the photon maps. Shadow photons are used to render shadows more efficiently and the directional information in the photon map is used to generate optimized sampling directions and to limit the recursion in the distribution ray tracer by providing an estimate of the radiance on all surfaces with the exception of specular and highly glossy surfaces.

The results presented demonstrate global illumination in scenes containing procedural objects and surfaces with diffuse and glossy reflection models. The implementation is also compared with the Radiance program.

Key words: Global Illumination, Photon Maps, Monte Carlo Ray Tracing

1 Introduction

Simulating global illumination in general environments is a complex task. Currently the most successful approaches combine radiosity and ray tracing [24, 5, 34]. Even though ray tracing has been extended with Monte Carlo techniques [14, 19, 27, 29, 32] and radiosity has been extended with directional capabilities [13, 26, 3, 11, 6] neither of the two methods precludes the use of the other. In general Monte Carlo ray tracing is very time consuming and gives noisy results while radiosity uses a lot of memory to store directional information and it cannot handle specular reflection properly.

Most radiosity implementations use a simplified ray tracing algorithm to render the result in order to simulate specular reflections seen by the eye [28, 26, 25, 6]. Shirley [24] noticed that the ray tracing method could be used to render shadows as well since radiosity has problems at discontinuities. He also introduced the use of light ray tracing [1] to render caustics. Chen et al. [5] went even further and used path tracing to render all diffuse reflections seen directly by the eye in order to eliminate all visible artifacts from the radiosity algorithm. They only used the radiosity algorithm to model soft indirect illumination. Rushmeier et al. [22] concluded that the radiosity solution could be simplified since the path tracing algorithm would hide most of the artifacts in it and they introduced geometric simplification in which the radiosity algorithm is performed on a simple geometric approximation of the original model. Their motivation was the fact that radiosity becomes very time and memory consuming as the number of surfaces in the model grows.

This paper introduces a two pass method in which we simplify the representation of the illumination instead of simplifying the geometry. We obtain this simplification by using the photon map introduced in [15]. We combine the extensions to the photon map presented in recent papers [16, 17, 18] in order to render the scene more efficiently. The photon map is used to generate optimized sampling directions, to reduce the number of shadow rays, to render caustics and to limit the number of reflections traced as the scene is rendered with distribution ray tracing.

2 Overview of the Method

The first pass in the method is constructing the photon map by emitting photons from the light sources in the model and storing these in the photon map as they hit surfaces. The result is a large number of photon hits stored within the scene. This information can be seen as a rough representation of the light within the model.

Ward [29, 32] uses a comparable strategy storing irradiance values at surface points. Our approach does however differ significantly in several aspects. The creation of the photon map is light driven and it supplements the eye-driven rendering step very well. Effects like caustics that are very difficult to compute using traditional Monte Carlo ray tracing are easily obtained with the photon map. Furthermore we store incoming flux (photons) which is much simpler and less accurate than irradiance values. Our motivation for doing so is that we obtain a very flexible environment with a lot of useful information that can be applied in the rendering step. The use of photons allows us to estimate surface radiance at surfaces with arbitrary BRDF's. The information can also be applied in an unbiased fashion to optimize the rendering step. The photons can be used to compute improved control variates [20] or as demonstrated in [16] to generate optimized sampling directions.

Our rendering engine is a distribution ray tracer in which rays are traced from the eye into the scene. The information from the photon map is applied during rendering in two different ways. We distinguish between situations where we need an accurate computation and situations in which an approximate estimate can be applied. As the rays are traced through several reflections their contribution to the final pixel radiance becomes lower and we apply the approximate estimate which for all surfaces equals a radiance estimate obtained from the photon map. For highly glossy surfaces we do however trace additional sample rays since reasonable radiance estimates for these surfaces require a large number of photons. The accurate computation is applied at surfaces seen directly by the eye or via a few specular reflections. This computation is performed using importance sampling where the information about the incoming flux is integrated with the BRDF to provide optimized sampling directions. Furthermore we use information about shadow photons to reduce the number of shadow rays. Accurate computation of caustics is done by visualizing a radiance estimate obtained using a separate caustics photon map which has a high density of photons.

3 Pass 1: Constructing the Photon Maps

The photon maps are constructed by emitting a large number of photons (packets of energy) from the light sources in the scene. Each photon is traced through the scene using a method similar to path tracing. Every time a photon hits a surface it is stored within the photon map and Russian roulette [2] is used to determine whether the photon is absorbed or reflected. The new direction of a reflected photon is computed using the BRDF of the surface.

Unlike previous implementations we use two photon maps: A *caustics photon map* and a *global photon map*. The caustics photon map is used only to store photons corresponding to caustics and it is created by emitting photons towards the specular objects in the scene and storing these as they hit diffuse surfaces. Caustics are rendered by visualizing a radiance estimate based on the caustics photon map directly and this requires a high density of photons.

The global photon map is used as a rough approximation of the light/flux within the scene and it is created by emitting photons towards all objects. It is not visu-

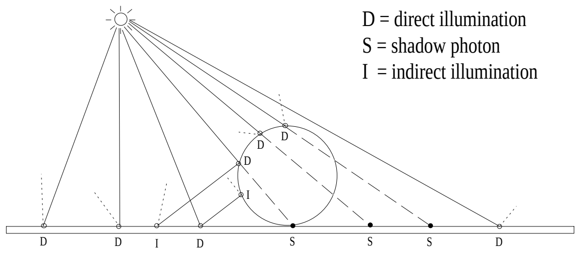


Figure 1: The photons in the global photon map are classified to optimize the rendering of shadows

alized directly and therefore it does not require the same precision as the caustics photon map. We use the extension presented in [17] and create shadow photons by tracing rays with origin at the light source through the entire scene. At the first intersection point a normal photon is stored and at the following intersection points we store shadow photons. These shadow photons are used during the rendering step to reduce the number of shadow rays (see figure 1).

The fact that we have two separate photon maps has improved both the speed, reduced the memory requirements and improved the accuracy of the method. Rendering caustics is faster since the caustics photon map contains only photons related to caustics. Locating photons in the global photon map is also faster since it has fewer photons and these photons have energy levels that are more similar since it does not contain the mixture of caustics photons with high density and low energy and normal photons with low density and high energy. This significantly improves the accuracy of the radiance estimate.

The photons are stored in a balanced kd-tree [4]. This data-structure is both compact and efficient. The fact that the tree is balanced guarantees that the time it takes to locate M photons in a tree with N photons is $O(M \cdot \log_2(N))$. In practice the search is much more efficient since the photons are located in the same parts of the tree. The use of a balanced kd-tree makes the rendering more efficient as demonstrated in [18] but just as important it reduces the memory requirements for each photon hit and allows us to represent each photon using only 20 bytes.

4 Pass 2: Rendering

The final image is rendered using Monte Carlo ray tracing in which the pixel radiance is computed by averaging a number of sample estimates. Each sample consists of tracing a ray from the eye through the pixel into the scene. The radiance returned by each ray is computed at the first surface intersected by the ray and it equals the surface radiance, $L_s(\mathbf{x}, \Psi_r)$, leaving the point of intersection, $\mathbf{x}$, in the direction, Ψ_r, of the ray. $L_s(\mathbf{x}, \Psi_r)$ is computed using the rendering equation [14]:

$$L_s(\mathbf{x}, \Psi_r) = L_e(\mathbf{x}, \Psi_r) + \int\limits_{\Omega} f_r(\mathbf{x}, \Psi_i; \Psi_r) L_i(\mathbf{x}, \Psi_i) cos\theta_i \, d\omega_i \tag{1}$$

Where L_e is radiance emitted by the surface, L_i is the incoming radiance in the direction Ψ_i, f_r is the BRDF and Ω is the sphere of incoming directions. L_e is taken directly from the surface definition and needs no further calculation. The value of the integral, L_r, depends on the radiance values in the rest of the scene and it can be solved directly using Monte Carlo techniques like path tracing. This is however a very expensive method and a more efficient approach can be obtained by using the photon map in combination with our knowledge of the BRDF and the incoming radiance.

The rendering equation (1) can be split into a sum of several components. We omit the position and direction parameters for clarity, and express L_r as

$$\begin{aligned} L_r \;=\; & \int_\Omega f_r L_{i,l} \cos\theta_i \, d\omega_i + \\ & \int_\Omega f_{r,s} (L_{i,c} + L_{i,d}) \cos\theta_i \, d\omega_i + \\ & \int_\Omega f_{r,d} L_{i,c} \cos\theta_i \, d\omega_i + \\ & \int_\Omega f_{r,d} L_{i,d} \cos\theta_i \, d\omega_i \end{aligned} \tag{2}$$

where

$$f_r = f_{r,s} + f_{r,d} \quad \text{and} \quad L_i = L_{i,l} + L_{i,c} + L_{i,d}$$

In this equation the incoming radiance has been split into contributions from the light sources, $L_{i,l}$, contributions from the light sources via specular reflection (caustics), $L_{i,c}$ and indirect soft illumination, $L_{i,d}$ (light which has been reflected diffusely at least once). The BRDF has been separated into a diffuse part, $f_{r,d}$, and a specular part, $f_{r,s}$. The diffuse part represents all reflection models from Lambertian to slightly glossy while the specular part are highly glossy and ideal specular reflection models (examples are presented in section 6).

Equation 2 is used to compute the radiance leaving a surface. In the following sections we discuss the evaluation of each of the parts in the equation in more detail. We distinguish between two different evaluations of the integrals: An accurate and an approximate.

We use the accurate computation if the surface is seen directly by the eye or perhaps via a few specular reflections. We also use the accurate computation if the distance between the ray origin and the intersection point is below a small threshold value — otherwise we might risk inaccurate colour bleeding effects in corners. The approximate evaluation is used if the ray intersecting the surface has been reflected diffusely since it left the eye or if the weight of the ray is low (it contributes only little to the pixel radiance).

4.1 Direct Illumination

The first term in (2) represents the contribution via direct illumination by the light sources. This term is normally computed by sending shadow rays towards all light sources to check for visibility. We compute the contribution differently depending on whether we need an accurate or an approximate evaluation.

In the accurate evaluation of the contribution we use the observation that most scenes have large areas that are either fully illuminated or in shadow. We can use the information in the photon map to identify these areas in order to avoid using shadow rays. We only use shadow rays in situations where the nearest photons in the global photon map contains a mixture of direct illumination photons and shadow photons or if the number of illumination and shadow photons located is too low. This strategy is described in more detail in [17].

The approximate evaluation is simply the radiance estimate obtained from the global photon map (no shadow rays or light source evaluations are used).

4.2 Specular Reflection

The second term in (2) is radiance reflected of specular and highly glossy surfaces. This value is computed using standard Monte Carlo ray tracing. By using importance sampling based on the BRDF the computation can is most cases be done using only a limited number of sample rays.

4.3 Caustics

The third term in (2) represents caustics on diffuse and slightly glossy surfaces. We evaluate this term using the information in the caustics photon map (see section 5). We never compute caustics via Monte Carlo sampling since this is almost impossible in most situations. This means that the radiance estimate based on the caustics photon map is visualized directly and this is the reason why the number of photons in the caustics photon map must be high.

4.4 Soft Indirect Illumination

The fourth term in (2) is incoming light which has been reflected diffusely at least once since it left the light source. This light is then reflected diffusely by the surface (using $f_{r,d}$) and consequently the resulting illumination is very "soft".

The approximate evaluation of this integral is the radiance estimate based on the global photon map (see section 5).

In the accurate evaluation we use importance sampling to compute the indirect illumination. As described in [16] we combine the information in the photon map with the BRDF in order to generate optimized sampling directions. At Lambertian surfaces we also use the irradiance gradient caching scheme [32]. This means that we only compute indirect illumination on Lambertian surfaces if this information cannot be interpolated from previously computed values.

5 Estimating Radiance using the Photon Map

The information in the photon map can be used to compute the radiance leaving a surface in a given direction. Since the incoming direction is stored with each photon we can integrate the information with any BRDF. In practice the approximation is limited to surfaces ranging from Lambertian to slightly glossy. To compute radiance leaving highly glossy surfaces a very large number of photons is needed. There is nothing in our algorithm preventing this approach. We have however found that highly glossy surfaces can be treated efficiently using Monte Carlo ray tracing and we use this strategy in order to limit the memory requirements.

To compute the radiance, L_r, leaving an intersection point $\mathbf{x}$ at a surface with BRDF f_r, we locate the N photons with the shortest distance to $\mathbf{x}$. Based on the assumption that each photon p represents flux $\Delta\Phi_p$ arriving at $\mathbf{x}$ from direction $\Psi_{i,p}$ we can integrate the information into the rendering equation as follows

$$L_r(\mathbf{x},\Psi_r) = \int_\Omega f_r(\mathbf{x},\Psi_r,\Psi_i)\frac{d^2\Phi_i(\mathbf{x},\Psi_i)}{dA\,d\omega_i}\,d\omega_i \approx \sum_{p=1}^{N} f_r(\mathbf{x},\Psi_r,\Psi_{i,p})\frac{\Delta\Phi_p(\mathbf{x},\Psi_{i,p})}{\pi r^2} \qquad (3)$$

We use the same approximation of ΔA as [15] where a sphere centered at $\mathbf{x}$ is expanded until it contains N photons and has radius r. ΔA is then approximated as πr^2.

An alternative could be using a sphere of a fixed size and use all the photons within this sphere. We have tested this technique and it improves the estimate slightly since ΔA is kept constant. It does however fail in scenes with a high variation in the density of the photons since it either gives bad estimates in areas with few photons or blurry estimates in areas with a high photon density. We have

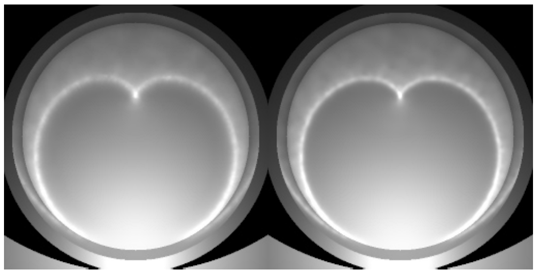


Figure 2: The effect of the cone filter. The left image is unfiltered and the right image is filtered using the cone with 1 as the filter constant

considered a number of adaptive strategies for computing the necessary size of the sphere based on the local photon density. We did however find that the payoff with respect to quality could not compensate for the extra computing time.

In situations where the density of the photons is too low the radiance estimation strategy can give blurry results. To compensate for this situation we have successfully applied a cone-filter to the estimate. In the cone-filter a weight is attached to each photon based on the distance, d_p, between $\mathbf{x}$ and the photon p. This weight is:

$$w_p = max(0, 1 - d/(kr)) \tag{4}$$

where k is a filter constant characterizing the filter. To normalize the filter we need some knowledge on the distribution of the photons. Since we use a sphere to locate the photons it would be natural to assume that the distribution of the photons is 3 dimensional and related to the sphere. However, photons are stored at surfaces which are 2 dimensional. Furthermore the area estimate is also based on the assumption that photons are located on a surface. Our normalization is therefore based on a 2d-distribution of the photons and it becomes $1 - \frac{2}{3k}$. The filtered radiance estimate can thus be expressed as

$$L_r(\mathbf{x}, \Psi_r) \approx \frac{\sum_{p=1}^{N} f_r(\mathbf{x}, \Psi_r, \Psi_{i,p}) \Delta\Phi_p(\mathbf{x}, \Psi_{i,p}) w_p}{(1 - \frac{2}{3k})\pi r^2} \tag{5}$$

In figure 2 we have showed the effect of the cone filter as it is applied to the well known cardioid caustic. We used only 12000 photons to render this caustic and the result is that the traditional radiance estimate looks blurry. Applying the cone filter significantly reduces this blur. In the figure we use a filter constant $k = 1$. This value generally works very well.

6 Results and Discussion

We have implemented the two pass method in a program called `MIRO` on a 100MHz Pentium PC with 32MB RAM running Linux.

Our first test scene is the *museum* shown in fig. 3. It has 5000 normal objects (spheres, triangles etc.) and 1 procedural object (the sphere flake). All important combinations of light reflections can be found in this scene. We have caustics from the glass sphere onto the rough surface, caustics from the glossy cylinder on the wall and also on the procedural sphere flake object. The sphere flake object has been rendered using Schlick's reflection model [23] with a diffuse-specular parameter of 0.1. Other important reflections include the colour bleeding effect between the walls, the glossy reflection of the metallic teapot (using Ward's anisotropic model [31]), the transmission of light through the glass sphere and the specular reflection in the floor and the teapot. The scene is illuminated by two small spherical area light sources. We used 289.000 photons in the caustics photon map and 165.000 photons in the global photon map to render this scene. This corresponds to approx. 9MB memory. The image was rendered in the resolution 1280x960 and the rendering time was 51 min.The photon map was constructed in 5 min. The most time consuming part of the scene is the computation of reflection of the teapot into the glossy cylinder since the number of reflections traced by each ray is not limited by using the photon map.

To demonstrate how the photon map actually works we have visualized the radiance estimate directly in fig. 4. The radiance estimate is shown for all diffuse surfaces (include the sphere flake) and it is based only on the 165.000 photons in the global photon map. An average of 80 photons have been used to estimate the radiance per surface intersection. Notice how all important types of reflections are included even though they are blurry. The blur in the photon map is actually an advantage since it reduces noise in the final gathering step where Monte Carlo sampling is used to render the initial reflections accurately.

Our second test scene shown in fig. 5 demonstrates the looks of a caustic from a

Scene	Resolution	Caustic photons	Global photons	Pass 1	Rendering
Diffuse Cornell Box	1280x960	21.162	286.489	67 sec.	8 min
Diffuse Cornell Box [**R**]	1280x960	-	-	-	60 min
Glossy Cornell Box	2560x1920	0	382.598	56 sec.	50 min
Glossy Cornell Box [**R**]	5120x3840	-	-	-	360 min
The Museum	1280x960	389.755	165.791	298 sec.	51 min
The Cognac Glass	1280x960	224.316	3095	27 min.	65 min

Table 1: Rendering statistics. [**R**] indicates the images rendered with Radiance

cognac glass onto a rough (fractal) surface approximated by 500.000 triangles. The reflection model for the surface is Schlick's reflection model with a diffuse-specular component of 0.6 — we found that this value makes the sand look more realistic. The caustic was created using 224.000 photons. These photons represent both the red caustic and the illumination of the surface below the cognac glass. If this model is rendered without caustics the surface below the cognac glass would be black. We also measured the advantage of using shadow photons in this scene and by using only 216 shadow photons ($\approx$ 5KB extra information) we were able to reduce the number of shadow rays with more than 70 %

To test the performance of our method we have compared it with the Radiance program — a superb global illumination program developed over the last 10 years by Greg Ward [33]. It is based on a significantly optimized Monte Carlo ray tracing scheme and it performs very well even compared with newer hierarchical radiosity techniques [6].

We have used our photon map implementation and Radiance to render two variations of the *Cornell box*: One in which the floor is Lambertian and one in which the floor is highly glossy (using the Anisotropic reflection model). We adjusted the parameters in both programs in order to obtain good quality within a reasonable time. The rendering times are shown in table 1.

The version with the diffuse floor was rendered in 1280x960 with both programs and as it can be seen from the table. With photon maps the rendering time is 6 times faster than Radiance. The primary reason is that we avoid the recursive sampling of the indirect illumination and the fact that we use fewer shadow rays to sample the area light source. In this scene the average depth of the image sampling rays is very close to 2 since most of the radiance computations beyond the first diffuse reflection is handled by the photon map.

The Cornell box with the glossy floor was rendered in 2560x1920 with `MIRO` and 5120x3840 with Radiance. We had to increase the resolution in Radiance since it uses path tracing to render glossy surfaces. Both images have been reduced to the resolution 640x480 and this means that Radiance uses 64 samples per pixel (ie. 64 samples to sample the indirect illumination on the first glossy surfaces seen through a pixel). `MIRO` uses distribution ray tracing and the only reason why we increased the resolution was in order to obtain the same level of anti-aliasing as Radiance. The distribution ray tracer in `MIRO` spawns a maximum of 6 sample rays at the glossy surface. Combined with the 16 samples per pixel this gives a maximum of 96 samples used to compute the indirect illumination on the first glossy surface seen through a pixel. This also means that the glossy surface looks less noisy in the version rendered by `MIRO`. The two images are shown in fig. 6 and fig. 7 and as we can see they look very similar. There is a slight differences in the overall illumination caused by different tone reproduction functions (gamma correction). The timing results in table 1 shows that MIRO renders the glossy Cornell box approximately 7 times faster than Radiance. The number of pixels is 4 times higher in the Radiance version but this number cannot be used directly due to the different sampling schemes used by the two programs. It is more correct to look at the number of samples spawned at the glossy surface since this is the actual reason why rendering these images takes so relatively long.

In table 1 we have collected some statistics showing the memory and time required to render the test images. As it can be seen the number of photons used is in the range 200.000-500.000. We have not carefully optimized these numbers since the rendering time is only affected slightly by the number of photons. Instead we use an appropriate number of photons and for our test images we have found that 2-500.000 photons gives nice results. In more complex scenes it would probably be necessary to use a higher number of photons. It is however important to notice that the necessary number of photons is not directly related to the number of objects in the scenes. It is instead related to the complexity of the flux within the scene. We would probably be able to render the Cornell box with detailed stone walls made of millions of triangles using the same number of photons as we did with the simple Cornell box. If we render a scene with too few photons we get low frequency noise in the caustics - this kind of noise is less disturbing than the high frequency noise that is normally seen in Monte Carlo ray tracing algorithms. The effect on the remaining parts of the illumination is more subtle and it depends on the rendering parameters. But if too few photons are used it means that we have to use more sample rays to compute the indirect illumination and we might get "poor statistics" in the radiance estimates which in our implementation results in recursive Monte Carlo sampling.

We believe that the results can be improved even further by using the photon map more intelligently. As an example we might use the photons to answer questions regarding the number of samples necessary to use for a pixel. Another interesting use of the photon map would be reclassification of light sources as done in [5]. The photon map could also be used to represent flux within participating media. This should be straightforward to implement since nothing prevents photons from being stored within a volume.

Currently we have only rendered scenes containing a few light sources (less than 10). Rendering scenes with many light sources makes the use of photon maps more complicated since naive emission of photons from every light source will generate a very large number of photons. We might use some kind of radiosity-like importance to distribute the photons more intelligently within the scene. It would be very interesting to make a 3-pass method in which an initial simple ray tracing pass is used to generate importance information that can be used when emitting the photons. This might also help answering the difficult question of the necessary number of photons. The current strategy is just to use enough photons (what the available memory permits).

A very important aspect of the photon map is the fact that it is easy to integrate into existing ray tracing programs since it only requires the existence of intersection routines for each object. The scene does not have to be tessellated and the photon map structure is completely separated from the geometric representation. The photon map code can be provided in a separate module that contains the necessary functions (e.g. a function that given a position and a surface definition returns the radiance in a given direction).

7 Conclusion

We have presented a general two-pass global illumination method based on photon maps. We integrate information from an accurate caustics photon map and a less accurate global photon map into a distribution ray tracer. Caustics are rendered by visualizing a radiance estimate from the caustics photon map directly. The information in the global photon map is used to generate optimized sampling directions, to reduce the number of shadow rays and to limit the number of reflections traced by providing an approximate radiance estimate.

We have used the method to simulate global illumination in scenes containing procedural objects and surfaces with diffuse and glossy reflection. Comparisons with existing global illumination techniques indicate that the photon map provides an efficient environment for global illumination.

8 Acknowledgment

The author wishes to thank Greg Ward who provided invaluable help with the Radiance program. Thanks also to Niels Jørgen Christensen and the reviewers for their helpful comments.

References

[1] Arvo, James: "Backward Ray Tracing". *Developments in Ray Tracing. ACM Siggraph Course Notes* **12**, pp. 259-263, 1986.

[2] Arvo, James and David Kirk: "Particle Transport and Image Synthesis". *Computer Graphics* **24** (4), pp. 53-66, 1990.

[3] Aupperle, Larry and Pat Hanrahan: "A Hierarchical Illumination Algorithm for Surfaces with Glossy Reflection". *Computer Graphics* **27** (4), pp. 53-66, 1993.

[4] Bentley, Jon Louis: "Multidimensional Binary Search Trees Used for Associative Searching". *Comm. of the ACM* **18** (9), pp. 509-517, 1975.

[5] Chen, Eric Shenchang; Holly E. Rushmeier, Gavin Miller and Douglass Turner: "A Progressive Multi-Pass Method for Global Illumination". *Computer Graphics* **25** (4), pp. 164-174, 1991.

[6] Christensen, Per Henrik: *"Hierarchical Techniques for Glossy Global Illumination"*. Ph.d. thesis, University of Washington, 1995.

[7] Collins, Steven: "Adaptive Splatting for Specular to Diffuse Light Transport". In *proceedings of 5. Eurographics Workshop on Rendering*, pp. 119-135, Darmstadt 1994.

[8] Cook, Robert L.: "Distributed Ray Tracing". *Computer Graphics* **18** (3), pp. 137-145, 1984.

[9] Glassner, Andrew S.: *"Principles of Digital Images Synthesis"*. Morgan Kaufmann Publishers Inc., 1995.

[10] Goral, Cindy M.; Kenneth E. Torrance; Donald P. Greenberg and Benneth Battaile: "Modeling the Interaction of Light Between Diffuse Surfaces". Computer Graphics **18**, pp. 213-222, 1984.

[11] Gortler, Steven J.; Peter Schröder; Michael F. Cohen and Pat Hanrahan: "Wavelet Radiosity". *Computer Graphics* **27** (4), pp. 221-230, 1993.

[12] Heckbert, Paul S.: "Adaptive Radiosity Textures for Bidirectional Ray Tracing". *Computer Graphics* **24** (4), pp. 145-154, 1990.

[13] Immel, David S.; Michael F. Cohen and Donald P. Greenberg: "A Radiosity Method for Non-Diffuse Environments". Computer Graphics **20** (4), pp. 133-142, 1986.

[14] Kajiya, James T.: "The Rendering Equation". *Computer Graphics* **20** (4), pp. 143-149, 1986.

[15] Jensen, Henrik Wann and Niels Jørgen Christensen: "Photon maps in Bidirectional Monte Carlo Ray Tracing of Complex Objects". *Computers and Graphics* **19** (2), pp. 215-224, 1995.

[16] Jensen, Henrik Wann: "Importance Driven Path Tracing using the Photon Map". In "Rendering Techniques '95". Eds. P.M. Hanrahan and W. Purgathofer, *Springer-Verlag*, pp. 326-335, 1995.

[17] Jensen, Henrik Wann and Niels Jørgen Christensen: "Efficiently Rendering Shadows using the Photon Map". In *Proceedings of Compugraphics 95'*, pp. 285-291, 1995.

[18] Jensen, Henrik Wann: "Rendering Caustics on non-Lambertian Surfaces". To be presented at *Graphics Interface '96*, Toronto 1996.

[19] Lafortune, Eric P.; Yves D. Willems: "Bidirectional Path Tracing". *Proceedings of CompuGraphics*, pp. 95-104, 1993.

[20] Lafortune, Eric P.; Yves D. Willems: "The Ambient Term as a Variance Reducing Technique for Monte Carlo Ray Tracing". In *proceedings of 5. Eurographics Workshop on Rendering*, pp. 163-171, Darmstadt 1994.

[21] Lafortune, Eric P.: "*Mathematical Models and Monte Carlo Algorithms for Physcially Based Rendering*". Ph.d. thesis, Katholieke University, Leuven, Belgium 1995.

[22] Rushmeier, Holly; Ch. Patterson and A. Veerasamy: "Geometric Simplification for Indirect Illumination Calculations". *Proceedings of Graphics Interface '93*, pp. 35-55, 1994.

[23] Schlick, Christophe: "A Customizable Reflectance Model for Everyday Rendering". In *proceedings of 4. Eurographics Workshop on Rendering*, pp. 73-84, Paris 1993.

[24] Shirley, Peter: "A Ray Tracing Method for Illumination Calculation in Diffuse-Specular Scenes". *Proceedings of Graphics Interface '90*, pp. 205-212, 1990.

[25] Shirley, Peter; Bretton Wade; Phillip Hubbard; David Zareski; Bruce Walter and Donald P. Greenberg: "Global Illumination via Density Estimation". In "Rendering Techniques '95". Eds. P.M. Hanrahan and W. Purgathofer, *Springer-Verlag*, pp. 219-230, 1995.

[26] Sillion, François X.; James R. Arvo; Stephen H. Westin and Donald P. Greenberg: "A Global Illumination Solution for General Reflectance Distributions". *Computer Graphics* **25** (4), pp. 187-196, 1991.

[27] Veach, Eric and Leonidas Guibas: "Optimally Combinig Sampling Techniques for Monte Carlo Rendering". *Computer Graphics* **29** (4), pp. 419-428, 1995.

[28] Wallace, John R.; Michael F. Cohen and Donald P. Greenberg: "A Two-Pass Solution to the Rendering Equation: A Synthesis of Ray Tracing and Radiosity Methods". *Computer Graphics* **21** (4), pp. 311-320, 1987.

[29] Ward, Gregory J.; Francis M. Rubinstein and Robert D. Clear: "A Ray Tracing Solution for Diffuse Interreflection". *Computer Graphics* **22** (4), pp. 85-92, 1988.

[30] Ward, Greg: "Real pixels". In *Graphics Gems II*, James Arvo (ed.), *Academic Press*, pp. 80-83, 1991.

[31] Ward, Gregory J.: "Measuring and Modeling Anisotropic Reflection". *Computer Graphics* **26** (2), pp. 265-272, 1992.

[32] Ward, Gregory J. and Paul S. Heckbert: "Irradiance Gradients". *In Proceedings of the Third Eurographics Workshop on Rendering*, pp. 85-98, Bristol 1992.

[33] Ward, Gregory J.: "The RADIANCE Lighting Simulation and Rendering System". *Computer Graphics* **28** (4), pp. 459-472, 1994.

[34] Zimmerman, Kurt and Peter Shirley: "A Two-Pass Solution to the Rendering Equation with a Source Visibility Preprocess". In "Rendering Techniques '95". Eds. P.M. Hanrahan and W. Purgathofer, *Springer-Verlag*, pp. 284-295, 1995.

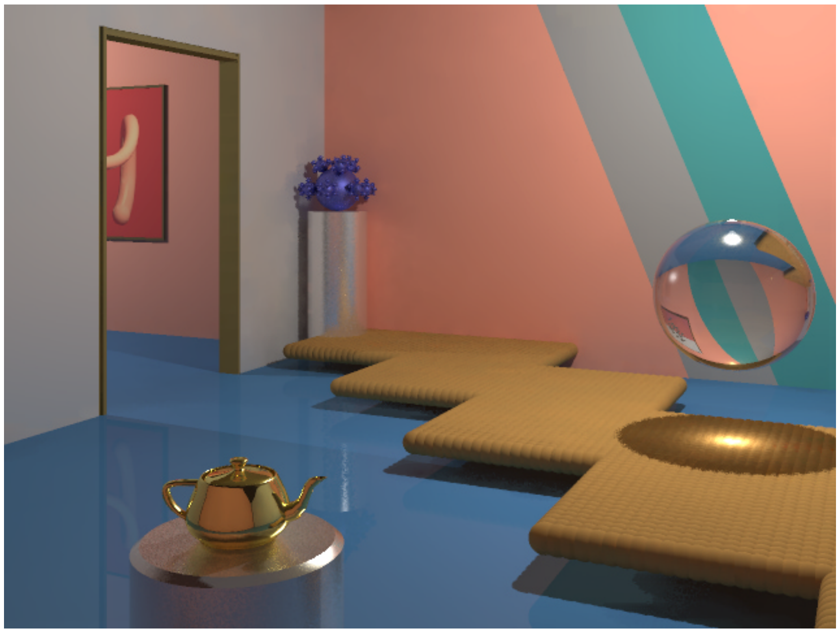


Figure 3: The Museum scene

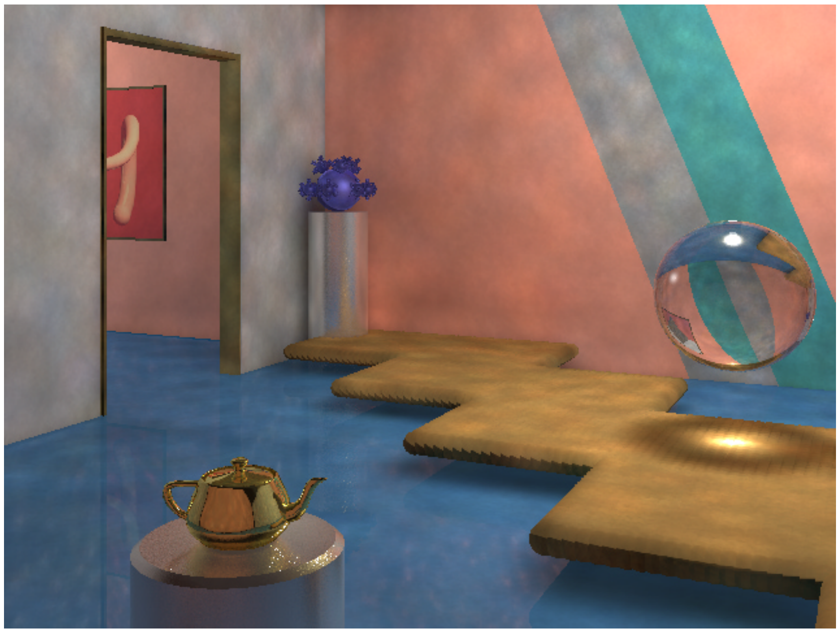


Figure 4: Direct visualization of the global photon map in the Museum scene

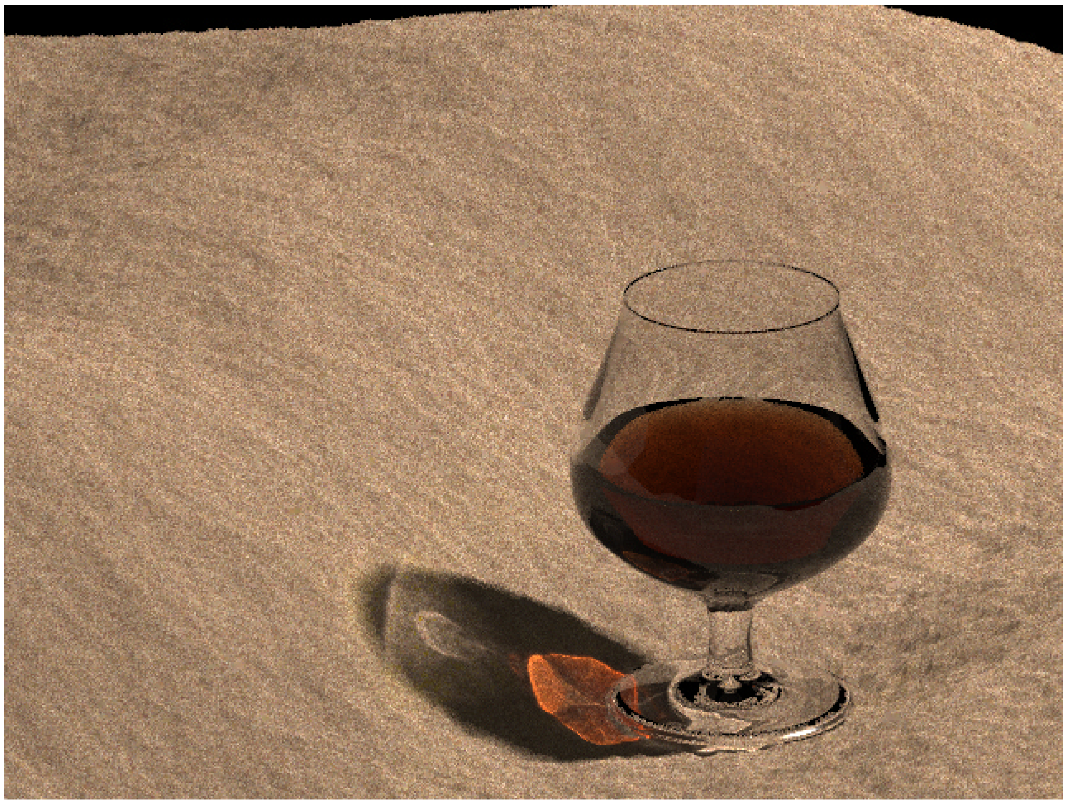


Figure 5: A Cognac glass on a fractal surface

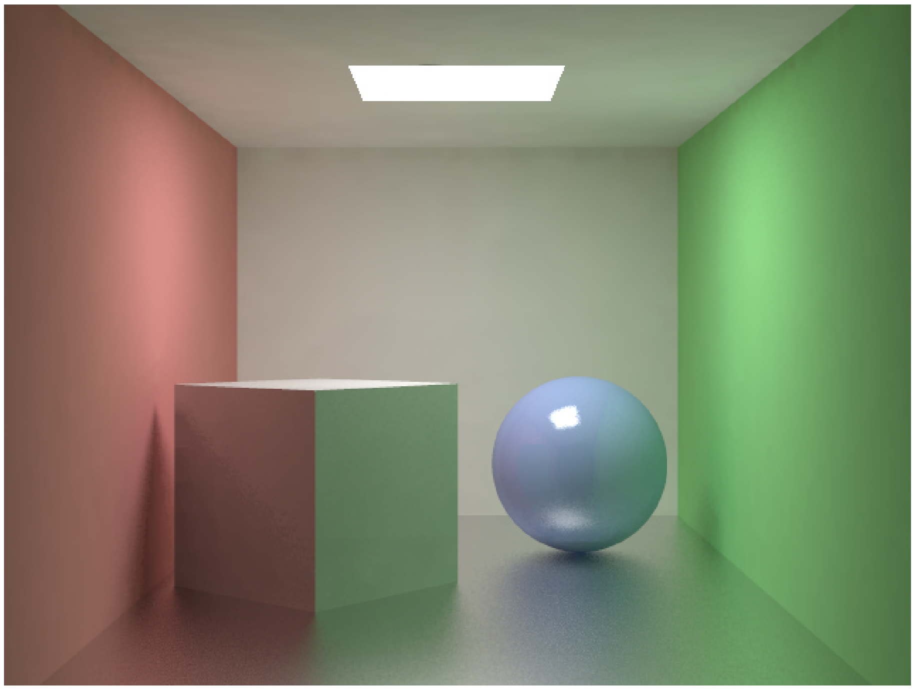


Figure 6: The glossy Cornell box rendered with photon maps

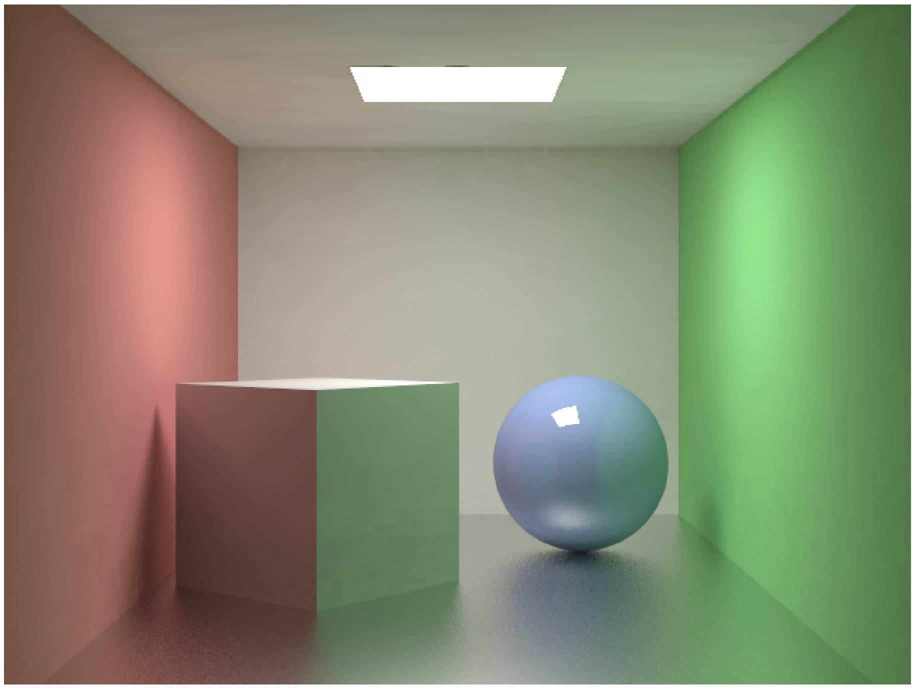


Figure 7: The glossy Cornell box rendered with Radiance

This paper appeared in *Proceedings of Graphics Interface '96*, pages 116-121, 1996

Rendering Caustics on Non-Lambertian Surfaces

Henrik Wann Jensen

Department of Graphical Communication
The Technical University of Denmark
`hwj@gk.dtu.dk, http://www.gk.dtu.dk/~hwj`

Abstract

This paper presents a new technique for rendering caustics on non-Lambertian surfaces. The method is based on an extension of the photon map which removes previous restrictions limiting the usage to Lambertian surfaces. We add information about the incoming direction to the photons and this allows us to combine the photon map with arbitrary reflectance functions. Furthermore we introduce balancing of the photon map which not only reduces the memory requirements but also significantly reduces the rendering time. We have used the method to render caustics on surfaces with reflectance functions varying from Lambertian to glossy specular.

Keywords: Caustics, Photon Map, Ray Tracing, Rendering.

1 Introduction

Caustics provides some of the most spectacular patterns of light in nature. Caustics are formed when light reflected from or transmitted through a specular surfaces strikes a diffuse surface. An example is the caustic formed as light shines through a glass of wine onto a table.

In traditional ray tracing [22] diffuse surfaces are only illuminated by the light sources. Caustics which are indirect illumination on the diffuse surfaces are not rendered at all. Even the stochastic ray tracing methods [5, 8] cannot render caustics properly. In order to integrate the computation of caustics into ray tracing it is necessary to compute illumination from light transmitted via specular surfaces. This computation is in most situations very complex and it has been solved only for a simple class of specular objects (ie. polygons [20]). Mitchell et al. [12] has presented a very comprehensive technique and their method is capable of handling caustics from implicit surfaces. The method is unfortunately very complex and also very time consuming.

Arvo [1] extended the standard ray tracing algorithm by introducing a preprocessing step in which caustics are computed. This preprocessing step uses backward ray tracing (also known as light ray tracing, illumination ray tracing and photon tracing) in which packets of energy (photons) are emitted from the light sources in the scene towards the specular surfaces. Each photon is reflected by the specular surfaces and stored on the Lambertian surfaces. The main problem with this approach is computing the intensity (radiance) of the caustics. This value depends upon the number of photons per surface area. Arvo solved this problem by using illumination maps which is an empty texture map divided into a large number of small area-elements. As photons hit a surface the energy is registered at the appropriate area-elements. In this way the caustics are created as textures on the Lambertian surfaces within the scene. A problem with this approach is the fact that a large number of photons must be used to eliminate noise in the caustics. Heckbert [7] introduced a method that adaptively subdivided the illumination map into area elements (rexes) with a size corresponding to the local density of the photon-hits. Chen et al. [3] and Collins [4] use a fixed illumination map. To eliminate noise they use different filter-kernels to spread the energy from each photon onto several area-elements. Jensen et al. [9] stored all photon-hits explicitly in a photon map and avoided using the illumination map. Instead they introduced a new technique for estimating the number of photons per area by looking only on the distribution of photons within the scene. Their method is capable of handling complex objects (ie. procedurally defined objects) and the estimate is less prone to noise since it can be seen as a low-pass filter. The photon map does however require large amounts of memory in complex scenes.

In scenes with simple objects it is possible to avoid the photon based approach. If the specular objects are polyhedral backwards beam tracing [14, 15, 21] can be used to render caustics on the Lambertian surfaces. With backwards beam tracing the illumination map can be replaced by caustic polygons that represent illumination from caustics on Lambertian surfaces.

In bidirectional path tracing [11, 18] the rendering of caustics is significantly improved compared to traditional path tracing [8]. The method is however still purely stochastic and it still requires a large amount of sample rays to produce results that are not too noisy.

The most popular techniques today are clearly the backward ray tracing techniques as introduced by Arvo. These methods are often faster and more general than other approaches and they are often used in global illumination techniques [3, 9, 16] to render caustics. They do unfortunately have one significant drawback - they are limited to Lambertian surfaces. In many situations this is not a problem. However, within global illumination where accurate rendering is important the Lambertian assumption does not always produces satisfying results. It is however difficult to eliminate the Lambertian assumption since it removes the view independence of the caustics and therefore complicates the storage of irradiance on the surfaces.

In this paper we present a technique in which we extend the photon map in order to store irradiance on surfaces with reflection functions that are non-Lambertian. We achieve this by extending the information stored with each photon with the incoming direction of the photon. This allows us to combine the photons with general bidirectional reflectance distribution functions. We present results which demonstrate rendering of caustics on surfaces with reflection functions ranging from Lambertian to almost glossy specular.

2 The Photon Map

The photon map represents a rough distribution of light throughout the scene. It is created by emitting a large number of photons from the light sources into the scene. In [9] the photon map is used not only to simulate caustics but all kinds of illumination. We are only interested in caustics and we therefore construct a *caustics* photon map specifically aimed at rendering caustics.

The caustics photon map is constructed by emitting a large number of photons towards all the specular objects within the scene. Each time a photon hits a surface two things happen. If the surface is diffuse the photon is stored in the photon map, and if the surface has a specular component Russian roulette is used to determine whether the photon should be reflected specularly or absorbed. In this way we obtain an unbiased solution without have to trace each photon through an infinite number of specular reflections.

Every photon is stored within the photon map. As [9] we use a kd-tree [2] to store the photons. While rendering the scene we need a data-structure that allows us to quickly locate photons within a given volume. Furthermore we need a very compact data-structure since we want to be able to use millions of photons. This makes the kd-tree a natural choice. In [9] the kd-tree was build on the fly as photons intersected the surfaces within the scene. This strategy can very easily result in a skew kd-tree that no longer has optimal search times. Searching is performed very often during rendering and we have found that balancing the kd-tree before actually rendering the scene significantly reduces the rendering time in most scenes. The balancing algorithm is performed after all the photons have been emitted. The photons are stored in a linked list of large arrays (each array having 65536 photons). The balancing algorithm manipulates this data structure directly in order to avoid having two copies of the photon map in memory. The balancing algorithm converts the unordered list of photons into a balanced kd-tree by recursively selecting the root node among the data-set as the median element in the direction which represents the largest interval. The existing data structure can be reused since we use a heap structure to represent the balanced tree. This completely eliminates the need for child-pointers. Further information on how to balance kd-trees can be found in [2].

As mentioned a large number of photons might be used in the photon map and it is necessary to use a compact representation. We have decided to use the following representation in which each photon only uses 20 bytes:

```
struct photon {
  float position[3];
  rgbe energy;
  char theta,phi; // incoming direction
  short flags;
  }
```

This representation is actually more compact than the one presented in [9] even though we have added information about the incoming direction. The use of a heap-like data-structure eliminates the need of two child pointers which would otherwise increase the memory requirements for each photon with 8 bytes (40%). The energy is represented as 3 floats packed into 4 bytes using the technique described in [19].

3 Rendering Caustics with the Photon Map

In standard ray tracing diffuse surfaces are only illuminated by the light sources. By introducing the photon map we have photons representing energy from caustics deposited on these diffuse surfaces. To render the caustics we need to extract radiance information from the photon map. This means that we must compute the density of the photons on all area-elements within the scene. Assuming that we have an intersection point $\mathbf{x}$ with normal $\vec{n}$ and an outgoing direction Ψ_r in which we want to compute the radiance, L_r, by using the photon map. L_r can be expressed as

$$L_r(\mathbf{x}, \Psi_r) = \int\limits_{\text{all } \Psi_i} f_r(\mathbf{x}, \Psi_r, \Psi_i) L_i(\mathbf{x}, \Psi_i) |\vec{n} \cdot \Psi_i| \, d\omega_i \tag{1}$$

where L_i is the incoming radiance from the direction Ψ_i, and f_r is the bidirectional reflectance distribution function.

To compute the contribution L_i we locate the N photons with the shortest distance to $\mathbf{x}$. If we assume that each photon p represents a packet of energy (flux) $\Delta\Phi_p$ arriving at $\mathbf{x}$ from direction $\Psi_{i,p}$ then it is possible to integrate the information into equation 1 as follows

$$\begin{aligned} L_r(\mathbf{x}, \Psi_r) &= \int\limits_{\text{all } \Psi_i} f_r(\mathbf{x}, \Psi_r, \Psi_i) \frac{d^2\Phi_i(\mathbf{x}, \Psi_i)}{dA \, d\omega_i} \, d\omega_i \\ &\approx \sum_{p=1}^{N} f_r(\mathbf{x}, \Psi_r, \Psi_{i,p}) \frac{\Delta\Phi_p(\mathbf{x}, \Psi_{i,p})}{\Delta A} \qquad (2) \end{aligned}$$

We use the same approximation of ΔA as [9]. That is we take a sphere centered at $\mathbf{x}$ and expand it until it contains N photons and has radius r. ΔA is then approximated as

$$\Delta A = \pi r^2 \tag{3}$$

and we can rewrite equation 2 as

$$L_r(\mathbf{x}, \Psi_r) \approx \frac{1}{\pi r^2} \sum_{p=1}^{N} f_r(\mathbf{x}, \Psi_r, \Psi_{i,p}) \Delta\Phi_p(\mathbf{x}, \Psi_{i,p}) \tag{4}$$

4 A Non-Lambertian Reflection Model

In order to test the algorithm we need a reflection model capable of simulating non-Lambertian reflection. Several comprehensive models exists for the purpose of accurately simulating the physical behavior of different materials. However we are only interested in a simple model that can be used to validate our algorithm. A suitable model was presented by Schlick in [13]. This model is simple and it has the very nice property that it provides a continuous transition from Lambertian reflection to glossy specular reflection. We omit the usage of anisotropic reflection (even though nothing in our model prevents us from simulating anisotropy) and use the following BRDF:

$$f_r = \frac{1 - G(v)G(v')}{\pi} + \frac{G(v)G(v')}{4\pi v v'} \left(\frac{\alpha}{(1 + \alpha t^2 - t^2)^2} \right) \tag{5}$$

where the function G represents a geometrical self-shadowing factor:

$$G(v) = \frac{v}{\alpha - \alpha v + v} \tag{6}$$

and $t = \frac{\vec{n}\cdot(\Psi_r+\Psi_i)}{|\vec{n}\cdot(\Psi_r+\Psi_i)|}$, $v = \vec{n} \cdot \Psi_r$, $v' = \vec{n} \cdot \Psi_i$ and α is the diffuse-specular factor varying from 1 (diffuse) to 0 (specular).

To test our algorithm with this reflection model we only have to modify one parameter, α. This value determines whether the surface is diffuse or specular. Our results in the following section refers to this value as the diffuse-specular component.

5 Results and Discussion

We have implemented and tested our rendering algorithm on a Silicon Graphics Onyx computer with 1GB RAM. Since our representation of the photon map is quite memory efficient we never needed the 1GB memory. In general we rendered caustics using approx. 5-10MB for the photon map. Only in extreme cases where the caustic is rendered on surfaces with a reflection function approaching glossy specular did we need a large number of photons. In these cases we used approx. 10-30MB of memory for the photon map.

Our first test case is shown in figure 1. This is the standard model used to illustrate caustics. The cardioid-shaped caustic is formed by placing a light source on the edge of a cylinder which has a reflective inner side.

The incoming direction of the light at the edge of the cardioid equals the tangent to the cardioid. This information is quite useful when we remove the Lambertian assumption from the receiving surface. It allows us to predict how the caustic should look as the surface becomes more glossy. Figure 1 contains 4 rendered images showing how the caustic looks as we change the diffuse-specular component of the surface from 1 to 0.01. As expected the intensity of the caustic is reduced mostly in those parts where the incoming direction of the light differs mostly from the incoming direction of the viewing ray. We used approx. 340.000 photons in all the images corresponding to 7 MB of memory. The quality of the images can be improved slightly by using more photons. The images have been rendered in 320x240 with 4 samples per pixel and the rendering time for the images was (from left to right) 22, 24, 27 and 42 seconds - just ray tracing the images takes 7 seconds. The rendering time increases as the surface becomes more glossy and the only reason for this is the fact that the image sampling algorithm requires more rays to render the caustic on the glossy surface. Using a fixed number of samples per pixel would make the rendering time the same for all the images.

Our second test case (figure 2) is a simple scene demonstrating what happens with the caustic from a glass sphere as the receiving surfaces become glossy. As we can see the shape of the caustics is no longer oval but curved. In this scene we had to use approx. 250.000 photons to obtain a nice caustic — using fewer photons makes the caustic look more blurred. The image was rendered in 640x480 with 4 samples per pixel and the rendering time was 182 seconds.

Our third test case (figure 3) is a more complex scene in which we benefited from usage of a non-Lambertian reflection model. It is a glass of cognac on a sand-surface. The sand is a fractal surface (with $2 \cdot 1024^2$ triangles) on which we have produced a synthetic sand-texture. We have used a diffuse-specular factor of 0.6 - using a Lambertian approximation makes the sand look more unnatural and flat. The caustic in this image was rendered using approx. 350.000 photons. The image was rendered in 26 minutes in the resolution 640x480 with 4 samples per pixel. Notice how the red-looking caustic is formed as light is transmitted through several layers of glass and cognac. The intensity of each photon is modified using Beer's law as the photon is transmitted through a dielectric media.

In the following table we have collected some statistics of the resources required to render the images:

Image	Photons	Preprocess	Rendering
Figure 1[a]	336.191	8 min.	22-42 s.
Figure 2	250.677	58 s.	182 s.
Figure 2[b]	250.677	58 s.	378 s.
Figure 2[c]	5.036.126	19 min.	276 s.
Figure 2[d]	5.036.126	19 min.	1380 s.
Figure 3	352.497	15 min.	26 min.
Figure 3[e]	352.497	15 min.	42 min.

[a] Applies to all images in figure 1

[b] Without balancing the photon map

[c] A reference image demonstrating how the rendering time is (un)affected by the number of photons used

[d] The same as 2[c] but with an unbalanced photon map

[e] Without balancing the photon map

As shown in the table we also rendered figure 2 and figure 3 without balancing the photon map and this clearly affected the rendering times in particular as the number of photons increased. The rendering times were almost doubled with the unbalanced version. We also examined how the rendering times were affected as more photons were added and we rendered figure 2 using 5.0 million photons corresponding to a data-structure of almost 100 MB. Naturally this increased the preprocessing time due to the extra photons emitted from the light source. The time used in the balancing algorithm were less than a minute. As we can see from the table the balancing algorithm reduces the rendering time with more than 75 % In general we have noticed that rendering time with the balanced photon map is only slightly affected as the number of photons is increased. This is particularly important in situations where high quality is required or in situations where large parts of the scene are illuminated by caustics. It also makes the photon map easier to use since the primary parameter becomes the amount of memory available.

Currently the user must specify both how many photons should be generated at the light sources and how many photons N to use in the radiance computation (equation 4). It would be nice to have an adaptive method that based upon the local density of the photons determined how many photons to use in the estimate. In general we have found that it is quite easy to predict good values for the two parameters. If the parameters are badly chosen the caustic will either become too blurred or too noisy.

The computed caustics are completely view-independent (even image independent). We do not need an initial ray tracing pass to determine the "bucket-size" as the illumination map based approaches. This also means that if very complex caustics are being visualized the user needs to adjust the number of photons used according to the desired resolution of the display. Another solution is just to always use enough photons if the memory permits it. As we have shown balancing the photon map (kd-tree) almost eliminates the dependence of the rendering time on the number of photons.

The rendering times could also be reduced even more by optimizing the integration of the photon map with Schlick's reflection model. Since we only have a discrete set of directions we could benefit from lookup tables and save a lot of vector computations.

The next step is integration of the method into a global illumination algorithm and extending the use of the photon map to other kinds of indirect illumination as in [9].

6 Conclusion

We have presented a new algorithm for rendering caustics on non-Lambertian surfaces. The method is based on an extension of the photon map and it renders caustics on procedurally defined surfaces. By balancing the photon map data structure we improve the rendering time and reduce memory requirements. The resulting method is fast and general and our test-images demonstrate that it is possible to achieve good results using only a limited amount of photons. The method is therefore useful in existing global illumination techniques in which caustics can be computed separately.

7 Acknowledgment

Thanks to Niels Jørgen Christensen, the reviewers and to Per Christensen and Martin Grabenstein for their helpful comments.

References

[1] Arvo, James: "Backward Ray Tracing". *Developments in Ray Tracing. ACM Siggraph Course Notes* **12**, pp. 259-263, 1986

[2] Bentley, Jon Louis: "Multidimensional Binary Search Trees Used for Associative Searching". *Comm. of the ACM* **18** (9), pp. 509-517, 1975

[3] Chen, Eric Shenchang; Holly E. Rushmeier, Gavin Miller and Douglass Turner: "A Progressive Multi-Pass Method for Global Illumination". *Computer Graphics* **25** (4), pp. 164-174, 1991

[4] Collins, Steven: "Adaptive Splatting for Specular to Diffuse Light Transport". In *proceedings of 5. Eurographics Workshop on Rendering*, pp. 119-135, Darmstadt 1994

[5] Cook, Robert L.: "Distributed Ray Tracing". *Computer Graphics* **18** (3), pp. 137-145, 1984

[6] Glassner, Andrew S.: "Principles of Digital Images Synthesis". *Morgan Kaufmann Publishers Inc.* 1995.

[7] Heckbert, Paul S.: "Adaptive Radiosity Textures for Bidirectional Ray Tracing". *Computer Graphics* **24** (4), pp. 145-154, 1990

[8] Kajiya, James T.: "The Rendering Equation". *Computer Graphics* **20** (4), pp. 143-149, 1986

[9] Jensen, Henrik Wann and Niels Jørgen Christensen: "Photon maps in Bidirectional Monte Carlo Ray Tracing of Complex Objects". *Computers and Graphics* **19** (2), pp. 215-224, 1995

[10] Jensen, Henrik Wann: "Importance Driven Path Tracing using the Photon Map". In "Rendering Techniques '95". Eds. P.M. Hanrahan and W. Purgathofer, *Springer-Verlag*, pp. 326-335, 1995

[11] Lafortune, Eric P.; Yves D. Willems: "Bidirectional Path Tracing". *Proceedings of CompuGraphics*, pp. 95-104, 1993

[12] Mitchell, Don and Pat Hanrahan: "Illumination from Curved Reflectors". *Computer Graphics* **26** (4), pp. 283-291, 1992

[13] Schlick, Christophe: "A Customizable Reflectance Model for Everyday Rendering". In *proceedings of 4. Eurographics Workshop on Rendering*, pp. 73-84, Paris 1993

[14] Shinya, Mikio; Tokiichiro Takahashi and Seiichiro Naito: "Principles and Applications of Pencil Tracing". *Computer Graphics* **21** (4), pp. 45-54, 1987

[15] Shinya, Mikio; Takafumi Saito and Tokiichiro Takahashi: "Rendering Techniques for Transparent Objects". *Proceedings of Graphics Interface '89*, pp. 173-182, 1989

[16] Shirley, Peter: "A Ray Tracing Method for Illumination Calculation in Diffuse-Specular Scenes". *Proceedings of Graphics Interface '90*, pp. 205-212, 1990

[17] Shirley, Peter; Bretton Wade; Phillip Hubbard; David Zareski; Bruce Walter and Donald P. Greenberg: "Global Illumination via Density Estimation". In "Rendering Techniques '95". Eds. P.M. Hanrahan and W. Purgathofer, *Springer-Verlag*, pp. 219-230, 1995

[18] Veach, Eric and Leonidas Guibas: "Bidirectional Estimators for Light Transport". In *Proceedings of 5. Eurographics Workshop on Rendering*, pp. 147-162, Darmstadt 1994

[19] Ward, Greg: "Real pixels". In *Graphics Gems II*, James Arvo (ed.), *Academic Press*, pp. 80-83, 1991

[20] Ward, Greg: "The RADIANCE Lighting Simulation System". In *Global Illumination.* ACM Siggraph Course Notes **18**, 1992

[21] Watt, Mark: "Light-Water Interaction using Backward Beam Tracing". *Computer Graphics* **24** (4), pp. 377-385, 1990

[22] Whitted, Turner: "An Improved Illumination Model for Computer Graphics". *Comm. of the ACM* **23** (6), pp. 343-349, 1980.

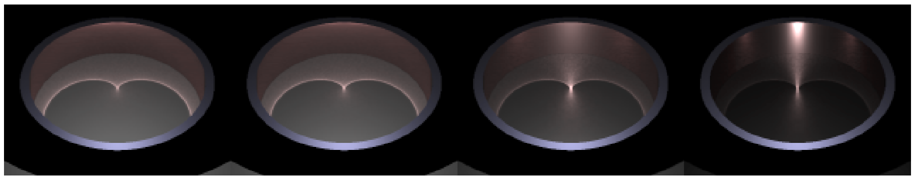


Figure 1: Four images demonstrating the looks of the cardioid created as light is reflected inside a cylinder-ring. From left to right the diffuse-specular component α is 1.0, 0.5, 0.1 and 0.01.

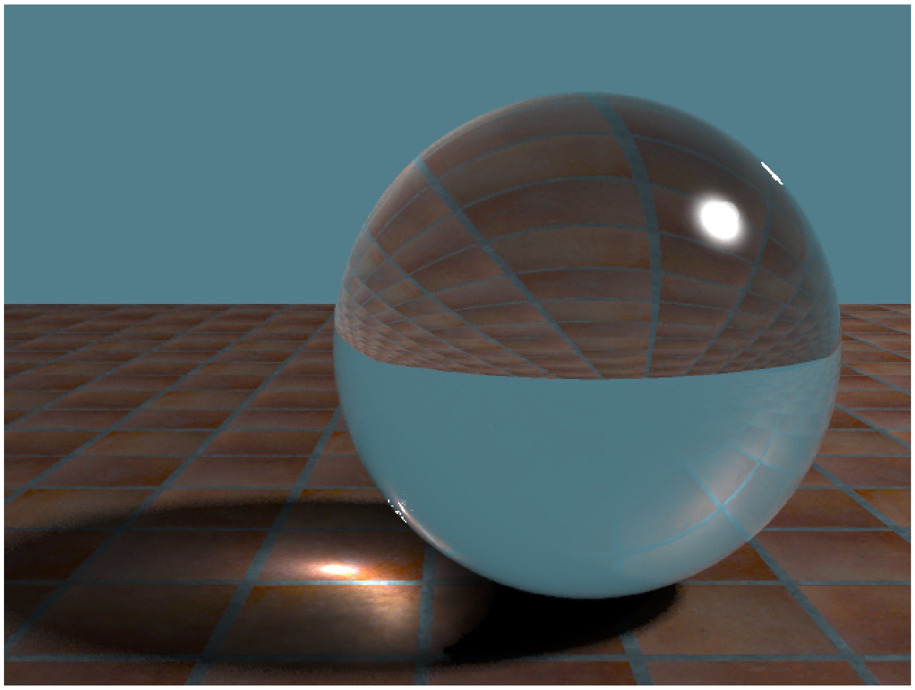


Figure 2: A caustic from a glass sphere onto a glossy stone surface with a diffuse-specular component $\alpha = 0.1$. Notice how the caustic becomes curved instead of oval as it would on a Lambertian surface.

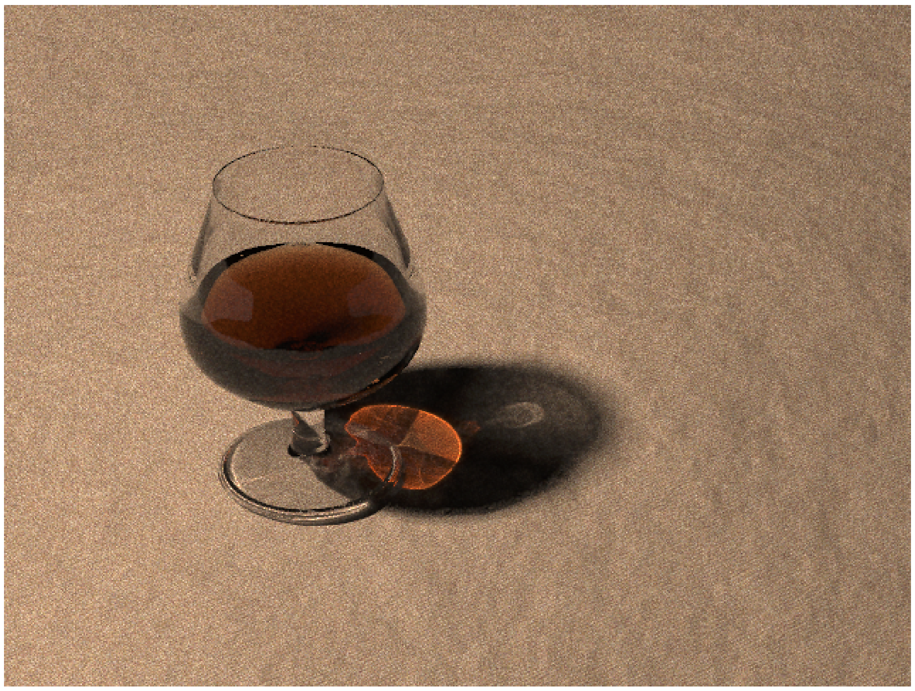


Figure 3: A glass of cognac on a sand-surface. The sand is a fractal surface with a synthetic sand-texture. The diffuse-specular component of the surface is $\alpha = 0.6$ and this value improves the realism of the sand compared to a Lambertian approximation.

Density Control for Photon Maps

Frank Suykens, Yves D. Willems

Department of Computer Science, K.U. Leuven, Belgium
franks,ydw@cs.kuleuven.ac.be

Abstract. The photon map method allows efficient computation of global illumination in general scenes. Individual photon hits, generated using Monte Carlo particle tracing, are stored in the maps and form a geometry independent representation of the illumination. Two important issues with the photon map are memory requirements to store the photons and the question how many photons are needed for an accurate representation of illumination in a certain scene. In this paper we introduce a method to control the density of photon maps by storing photons selectively based on a local required density criterion. This reduces memory usage significantly since in unimportant or over-dense regions less photons are stored. Results for caustic photon maps and global photon maps representing full illumination show a decrease in number of photons of a factor of 2 to 5. The required density states how accurate the photon map should be at a certain location and determines how many photons are needed in total. We also derive such a criterion based on a novel path-importance-based first pass, taking some steps towards solving the difficult 'how many photons' question.

1 Introduction

Computing global illumination solutions for general scenes is a difficult job. Scenes can be very complex, and the materials used can have arbitrary reflection and refraction properties.

Pure Monte Carlo methods, like path tracing[11] or bidirectional path tracing[12, 20], are capable of computing light transport for such general scenes. They do not store any information in the scene and are therefore capable of rendering very complex geometry. However, not storing the illumination means that it has to be recomputed every time when needed. This can be very inefficient for example for multiple indirect reflections and caustics (or caustics seen through a mirror for bidirectional path tracing).

Two-pass and multi-pass methods address this problem by computing and storing illumination in the scene in one or more preprocessing passes. A final Monte Carlo (or ray-tracing) pass can use this illumination information. In this context, radiosity and light maps[1] have often been combined with ray-tracing[3, 16] or (bidirectional) path tracing[19].

An interesting two-pass algorithm and storage method that uses photon maps was proposed by Jensen[6, 7, 10]. In an initial Monte Carlo particle tracing pass, a number of photons are traced through the scene and stored individually in a kd-tree. To reconstruct radiance at a certain point the nearest photons are located and used in a radiance estimate. Usually a separate high-density caustic photon map for direct visualization is used next to a global photon map for efficiently computing indirect light. The next section gives a more detailed overview of this algorithm.

An advantage of this method is that the storage is independent of the geometry, but rather dependent on the illumination complexity. The method accommodates general material properties and includes all possible global illumination effects.

Disadvantages are the large memory requirements for storing the photons and the more expensive radiance reconstruction (e.g. compared to radiosity) because the nearest photons must be located among all the stored photons. Another difficulty with photon maps is that it's hard to know how many photons should be used for a particular scene in order to get a sufficient accuracy. In most current implementations this number is set by the user, and dependent on his know-how of photon maps and global illumination.

In this paper we introduce density control for photon maps. Photons are only stored in a photon map when a certain required density criterion is not yet met, otherwise their energy is distributed among the nearest neighbors. This approach has two main advantages:

- The density of photon maps can be controlled locally in the scene. Less photons are stored in over-dense or unimportant regions. This can reduce memory requirements quite effectively.
- Introducing the concept of required density, offers an interesting framework for error control in photon maps. Since the density of photon maps is related to their accuracy, a high density should be chosen for important regions in the scene. The required density can be chosen arbitrarily and can be based on principles like view importance, relative error, visual masking by textures, ...

The method is applied first to caustic photon maps and a simple convergence criterion for these maps is presented. A second application uses a novel view-importance-based first pass to determine required densities for global photon maps. The required densities are large for important parts of the scene and the real density of the global map matches closely due to the selective storage.

Both approaches significantly decrease memory requirements and, maybe even more important, they take steps to letting a user choose a more scene-independent accuracy, rather than the number of photons. However many interesting extensions are possible within this framework, and further research is needed to put it to full use.

Another approach to control the density of photon maps, was presented by Peter and Pietrek [15]. They use an importance map to guide more photons to visually important regions in the scene. The photon map is then used for importance sampling in a standard path tracing pass. Their resulting photon maps however contain a mixture of high- and low-powered photons, which can result in higher variance when reconstructing illumination[8]. Our approach offers a more localized control over the density and results in a more smoothly varying photon energy.

The next section describes the current photon map algorithm in more detail establishing some terms and notation. Section 3 describes the new method for selectively storing the photons in the photon map. Section 4 proposes a required density heuristic for the global photon map and the results obtained. Section 5 presents conclusions.

2 Photon Maps

This section briefly describes the standard photon map method as presented by Jensen in [6]. More detailed information can be found in his paper(s).

It is a two-pass method, where in the first pass a high-density caustic map and a lower density global photon map are constructed by tracing particles or photons from the light sources. Photons are only stored on diffuse (D) and (slightly) glossy surfaces (G). The caustic map contains photons that have been reflected specularly (S) a number of times, storing ($L(S)^+(D|G)$) paths. The global photon map stores an approximation to the full global illumination solution ($L(D|G|S)^*(D|G)$ paths).

For each photon i the energy or flux Φ_i, the incoming direction ω_i and the location p_i are stored. From the photon map, reflected radiance L_r can be estimated at a position x on a surface in a direction ω by locating the nearest M photons around x:

$$L_r(x,\omega) \approx \tilde{L}_r(x,\omega) = \sum_{i=1}^{M} f_r(x,\omega_i,\omega)\frac{\Phi_i}{\pi r_M^2(x)}$$

with f_r the BRDF and $r_M(x)$ the maximum distance between x and its M nearest photons. This corresponds to nearest neighbor density estimation[17], where a sphere is expanded to find the nearest neighbors. To efficiently find nearby photons, they are stored in a (possibly balanced) kd-tree.

The second pass is a stochastic ray-tracing pass, that uses the photon maps in different ways. Suppose a path is traced from the eye e and hits a surface in x. In x direct illumination is computed and the caustic map is visualized directly, because caustics are hard to compute using stochastic ray-tracing. For other indirect illumination, scattered rays are spawn say to a point y on another surface. If a diffuse or glossy bounce is made in x, the radiance approximation using the global photon map is used in y. For a specular bounce the same procedure is repeated for y as was done for x.

Advantages of the photon map method are:

- The method includes all possible global illumination effects.
- Storage of photon maps is independent of geometry, allowing it to be applied to very complex scenes.

Our current implementation of the photon map method includes both the caustic and global photon map. For the radiance estimates the number of photons M was set to 50. Some differences with Jensen's method are that we don't (yet) balance our kd-tree, use Ward's irradiance caching [21] for indirect diffuse illumination nor use importance sampling for directions based on the global photon map [5]. These are all optimizations for the second rendering pass to speed up computations. Our rendering pass therefore can take several hours to compute a final image. However in this paper we focus on the construction of the photon maps, and our techniques can be combined with these optimizations without major change.

3 Selective Storage of Photons

When constructing photon maps, the photons are emitted from the light sources according to the emitted radiance distribution. Scattering on surfaces is performed according to the physics involved (proportional to BRDF times cosine). As a result the flux of each photon is the same in case of a single wavelength [8]. Differences can occur when using multiple wavelengths at once (e.g. an RGB color per photon), or when sampling the exact underlying physics is not possible (e.g. analytical sampling of BRDF times cosine impossible).

This way of constructing the maps results in a photon density that follows the illumination intensity in the scene. Brightly lit regions correspond to a high density of the map. A high density also corresponds to a high accuracy when estimating radiance.

However, in very bright regions (e.g. the center part of caustics) the density might be much higher than needed, while other parts (e.g. outer part of caustics, visually important parts of the scene) can have a lower density. To increase the density for these parts, more photons must be shot, but a large percentage will again be stored in the already bright regions.

This observation has lead to the basis of our method: Photons are still generated as before, but storage is controlled using a local required density criterion. If the density is already sufficient, the photon is not stored but its power is distributed over previously stored nearby photons. The next section describes this more formally while section 3.2 describes application to caustics.

3.1 A method for selective storage and redistribution

Suppose we have traced a new photon k to a position x on a surface. Suppose also we can evaluate a certain required density $D_r(x)$ that gives us a measure of how dense the photon map must be at x for accurate reconstruction. Note that in our current method the required density is only dependent on the position, which is sufficient for storage on diffuse and not too glossy surfaces. However if desired, it is possible to adapt all proposed methods to take the incoming directions of the photons into account.

To determine whether or not we want to keep the photon, we estimate the current photon map density $D_c(x)$. This can be done by locating the M nearest photons and evaluating:

$$D_c(x) = \frac{M}{\pi r_M^2(x)}$$

An acceptance probability P_{acc} can now be defined as a function of the density ratio: $s(x) = (D_c(x)/D_r(x))$. For P_{acc} we have tried a step function ($s(x) \leq 1$ accept, otherwise distribute) and a translated cosine, both with good results.

If the photon is accepted it is stored in the photon map, otherwise it's power must be accounted for somehow to keep the global flux in the map consistent with the flux emitted from the light sources.

One simple (and unbiased) way to do this, is to modify the power of accepted photons by $1/P_{acc}$[1], which corresponds to a form of Russian Roulette. However this can lead to huge photon powers and we noticed a significant variance increase in the reconstruction.

Better results were obtained by distributing the photon power over its nearest neighbors. This can be justified as follows:

If we would have stored the photon k then reconstruction of radiance using M+1 photons at x would be:

$$\tilde{L}_r(x, \omega) = \frac{\sum_{i=1}^{M} f_r(x, \omega_i, \omega)\Phi_i + f_r(x, \omega_k, \omega)\Phi_k}{\pi r_M^2(x)}$$

Note that $r_M(x)$ without k stored is equal to $r_{M+1}(x)$ when k is stored, since k is located in x.

Now since we don't store the photon the power of the other photons must be adjusted, so that the reconstruction in x would deliver the same result:

$$\tilde{L}_r(x, \omega) = \frac{\sum_{i=1}^{M} f_r(x, \omega_i, \omega)(\Phi_i + \Delta\Phi_{k,i})}{\pi r_M^2(x)}$$

Different choices for $\Delta\Phi_{k,i}$ can be made depending on f_r and the distance of x to photon i:

[1]For this method the acceptance probability may never become zero

- $f_r(x, \omega_i, \omega)$: To get an equal reconstruction $\tilde{L}_r(x, \omega)$ in x, $\Delta\Phi_{k,i}$ should be zero when f_r is zero because these photons do not contribute to $\tilde{L}_r$. Currently the angle between ω_i and the normal n_x in x is used to determine whether $\Delta\Phi_{k,i}$ should be zero (i.e. for a non-transparent material, $\Delta\Phi_{k,i} = 0$ when $cos(\omega_i, n_x) \leq 0$). [2]
- Distance to x: The distribution of the photon power can be seen as applying a low-pass filter (or as splatting). The dependence of $\Delta\Phi_{k,i}$ on the distance to x determines the filter kernel. We distribute the power equally over the affected photons to keep the photon powers homogeneous which, as said, is beneficial for the reconstruction.

So to summarize, we choose:

$$\forall i, cos(\omega_i, n_x) > 0 : \Delta\Phi_{k,i} = \Phi_k / M'$$

with M' the number of photons that have a cosine > 0.

Of course the radiance estimate at other locations than x, will give a slightly modified result. But since the current density is high enough anyway, this averaging can be expected not to introduce artifacts (if the required density does not change too abruptly).

Note that the selective storage requires estimation of the map density during its construction. We store the photons directly in an unbalanced kd-tree so that the lookup is efficient. Before the final rendering pass this tree can be balanced for even faster access.

We now have a method to control the density of photon maps based on a required density D_r. This density can be chosen arbitrarily, depending on the application, providing a flexible density control framework.

3.2 Application: Caustic maps

The selective storage was first tested on a caustic map. In this application the required density D_r for the caustic map is simply chosen to be constant (currently set manually). Caustic photons are then traced through the scene as usual, but the stored density will be limited.

One useful observation is that if the illumination at a certain location is much larger than the caustic map contribution alone, the caustic map does not have to be as accurate since relative errors will be small. From this observation we derived a simple convergence criterion for the caustic map at x: The ratio D_c/D_r is multiplied by a factor dependent on the relative contribution of the caustic radiance $\tilde{L}_c$ compared to the global radiance $\tilde{L}_g$ (which also includes the caustic radiance):

$$C(x) = \frac{D_c(x)}{D_r(x)} F(\tilde{L}_g / \tilde{L}_c)$$

If $C(x) > 1$ the caustic map can be considered converged at this point. Currently F is chosen by hand, and good results were obtained by taking a power of two or three for F. For the radiance reconstructions ($\tilde{L}_g$ and $\tilde{L}_c$) an outgoing direction must be known. We chose it equal to the normal in x. For diffuse surfaces this direction is unimportant, but for glossy materials other choices may be better.

[2] Another approach could be to choose a larger delta for photons with a direction similar to the distributed photon. This might be better for non-diffuse brdf's but at the cost of a less smoothly varying photon power.

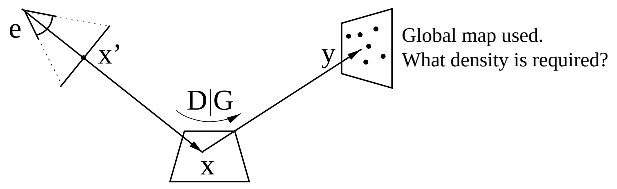


Fig. 1. Illumination from diffuse or glossy reflection is computed using a radiance estimate at y from the global photon map.

Figure 5 shows a scene with a glass egg lit by two light sources. Caustics can be seen to the left and right of the egg. For this image D_r was set to 80000 photons/m^2. There are about 92.000 photons stored in the caustic map using density control. Without this control 190.000 photons would have been stored, while there are no differences in the resulting images. Constructing the density controlled caustic map required 145 seconds (SGI Octane, 195MHz R10000), while constructing the normal caustic map (190.000 photons) required 90 seconds. The extra time is spent in the look-ups needed for the density estimates. This overhead, however, is not a problem since the final rendering is orders of magnitudes slower.

Figure 6 visualizes $C(x)$ (set to $(\tilde{L}_g/\tilde{L}_c)^3$) for this scene. Blue color indicates that $D_c \geq D_r$. This is were the memory is gained. Green color indicates that $D_c < D_r$ but $C(x) > 1$, which means the caustic map is considered converged. Red color means that more photons are needed. Some variance near this regions can be seen in the image.

We also continued the caustic map construction until all regions were converged. This required about 195.000 photons, while without density control 730.000 photons would be stored. So a factor of 2 to 4 can be gained easily (Actually an arbitrary factor can be gained since the number of photons using density control is limited.)

4 A required density estimate based on importance

The selective storage allows the photon map density to be adapted to a certain local required density criterion. Many choices, depending on the application, are possible and in this section we develop a three-pass view-importance-based method that gives a heuristic for the required density of global photon maps. The main idea is that when the final stochastic ray-tracing pass is computed for a certain camera position, the global map density should be high were the contribution to the error of a pixel in the image might be large.

View-importance has been used in many other algorithms, for example to drive a refinement criterion[18] or to direct more paths to important parts of the scene[14, 2]. Lischinski et. al.[13] presented a hierarchical radiosity method with error-driven refinement (also using importance) that considered the effect of interactions on the *total error*.

Our method is derived for eye paths exy of length two (see figure 1), meaning that in x a diffuse or glossy bounce has taken place and the global map is used in y. Extension to longer paths is discussed further on.

4.1 Importance and error

When an image is rendered, the estimated flux Φ_j (or average radiance) of a pixel j will exhibit a certain error E_j, that should be limited depending on the desired image quality. The error E_j is an integral over points x' on the pixel j of the error ϵ on the incoming radiance L:

$$E_j = \int_{A_j} dA_j W_e(e \to x')\epsilon(e \leftarrow x')$$

W_e is the emitted potential[3]. It includes a weighting factor for the camera model used in order not to burden the equations. Note that x' can be replaced by x in this equation.

This equation can be expanded to an integral over surfaces A_y [4] where the global map is used to approximate radiance:

$$E_j = \int_{A_j} dA_j W_e(e \to x) \int_{A_y} dA_y f_r(e,x,y)G(x,y)\epsilon(y \to x) \qquad (1)$$

$\epsilon(y \to x)$ is the error made by approximating $L(y \to x)$ with a global map estimate $\tilde{L}(y \to x)$ and $G(x,y)$ is the geometric term: $cos(n_x, x \to y)cos(n_y, y \to x)/||y - x||^2$. This equation relates the pixel error to the reconstruction error using the global map at indirectly visible surfaces. Note that E_j does not represent the total error of the pixel because direct light, caustic maps and specular reflected light are not taken into account.

There is a close relation between $\epsilon(y \to x)$ and the density of the photon map (higher density, lower error). This relationship is discussed in section 4.2. Now we want to find a suitable value for ϵ throughout the whole scene, so that we can derive a required density for constructing the global map.

A first assumption is that the error $\epsilon(y \to x)$ is independent of the direction in y. This is no problem for diffuse surfaces but may break down for highly glossy materials. An $\epsilon(y)$ can now be defined as the reconstruction error of the global map in any direction. The error contribution $\delta_j(y)$ made by $\epsilon(y)$ to the total pixel error E_j can now be written as:

$$\delta_j(y) = \frac{dE_j}{dA_y} = \int_{A_j} dA_j W_e(e \to x) f_r(e,x,y)G(x,y)\epsilon(y) \qquad (2)$$

We want the error $\epsilon(y)$ to be small if it contributes more to the error of the pixel. Note that this corresponds exactly to the notion of importance. Choosing a constant allowable error δ for any y and j [5], an upper bound for the error in y on a surface is now a minimum over all pixels j of δ divided by the importance of j in y (so that the maximum error in y is determined by the pixel with maximum importance):

$$\epsilon(y) \leq MIN_j \left(\frac{\delta}{\int_{A_j} dA_j W_e(e \to x) f_r(e,x,y)G(x,y)} \right) \qquad (3)$$

[3]As importance corresponds to flux, potential corresponds to radiance.

[4]The integral is expressed over the area measure dA_y, since the density of the photon maps is also expressed in terms of area.

[5]Actually $E_j = \int_{A_y} dA_y \delta_j(y) = \delta A_y$ with A_y the area seen indirectly through pixel j. This area can be conservatively approximated by for instance the total area of the scene, giving a fixed relationship between the pixel flux error E_j and δ.

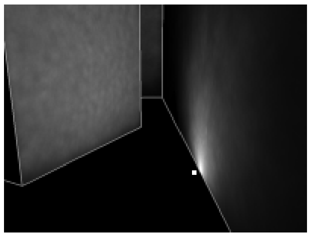


Fig. 2. Importance of a single pixel (indicated by the white dot) for surfaces indirectly visible through the pixel.

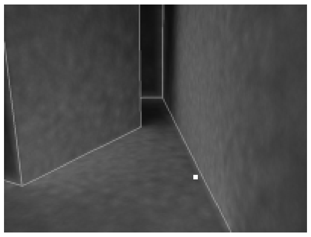


Fig. 3. Importance of the whole screen for surfaces indirectly visible through the screen. Single pixel importances are underestimated.

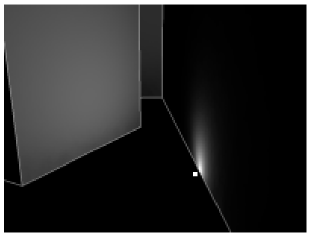


Fig. 4. Path based importance approximation for a pixel. Note the similarity with the real pixel importance.

To compute this upper bound the importance solution for every pixel j is needed separately. Since this is infeasible, most importance based algorithms choose to compute importance for the whole screen at once ($A_j \rightarrow A_{screen}$). This, however, averages the pixel importances, underestimating the typical localized importance of these pixels. Figure 2 and 3 show an example of this difference. In figure 2 the indirect importance of the indicated pixel is visualized, while figure 3 shows the total screen importance. The importance of the wall near the pixel is too low because it is unimportant for many other pixels.

We take a different approach by reducing the pixels to an infinite small size. An upper bound can now be expressed as a minimum over a screen position x' (removing the integral over A_j):

$$\epsilon(y) \leq MIN_{x'} \left(\frac{\delta}{A_{pix} W_e(e \rightarrow x) f_r(e, x, y) G(x, y)} \right) \tag{4}$$

Note that we include a pixel area factor (A_{pix}) to get results of equal magnitude as in equation (3). It can also be viewed as if the integral in (3) was approximated by one sample x'.

In order to be able to evaluate $\epsilon(y)$ during photon map construction, we introduce a first pass that shoots particles (or importons [15]) from the eye into the scene and stores the upper bound at hit points on the indirectly visible surfaces.

It is important to note that equation (4) does not contain any integrals anymore, so shooting the importons is not a form of Monte Carlo integration, but an arbitrary way to sample a 4D function dependent on x' and y. The generating pdf's are not taken into account and reconstruction of the function can be done by interpolation. This function can be evaluated for the individual paths and we will refer to it as the *path-importance function*.

Since we need the minimum over x' we do not need to store the x' position. $\epsilon(y)$ can be reconstructed by locating nearby importons and *taking the minimum of their values* (A possible variant could take the distance to y into account to get a smoother function).

Figure 4 shows a reconstruction of the maximum importance ($\sim 1/\epsilon$) over x' for one pixel of interest. A good match is obtained with the pixel importance shown in figure 2.

The incoming direction of importons is currently ignored, but could be used for a directional maximum error estimate.

4.2 Error and Density

$\epsilon(y)$ gives us an estimate of the upper error bound allowed in y. To derive the required density $D_r(y)$ at y we need to find the relationship between ϵ and D_r.

This is a difficult question and currently we assume them to be inverse proportional to each other:

$$D_r(y) = \frac{C}{\epsilon(y)}$$

If the allowable error is large, a lower density can be tolerated. The constant C is combined with the other constants in ϵ, into one *accuracy parameter* α of the algorithm, so that:

$$D_r(y) = \alpha \cdot MAX_{x'}(W_e(e \to x) f_r(e, x, y) G(x, y))$$

The parameter α determines the number of stored photons needed in the scene.

This parameter is currently chosen manually and a more detailed analysis of the error vs. density relation is definitely necessary. However, most important is that the constant(s) will be virtually independent of the scene (or at least easily derived from it), since the allowable pixel error and the relation local error - local density are isolated into a manageable form.

4.3 Longer paths

The allowable reconstruction error $\epsilon(y)$ was derived for paths of length two, assuming a diffuse or glossy bounce in x. However it can happen that the global map is used after more than one bounce (e.g. if a specular surface is hit first). Longer paths (e.g. $exzy$) introduce extra integrals over intermediate vertices (z) into the error bound equation.

Accurate evaluation of these integrals for every importance path would be infeasible, and therefore we believe more in a crude but conservative approximation, so that a path-importance function can easily be evaluated for longer individual paths also.

Two common cases where longer paths occur are:

- **Specular bounces:** Integrals introduced by specular bounces are very sharply peaked and can be approximated by a single sample. However convergence or divergence of light rays is not taken into account when using importance functions based on single paths. An interesting approach here would be to incorporate ray differentials, a way to track pixel footprints during ray tracing[4], into the importance evaluation. Our current implementation does not yet deal with specular surfaces in the importance-driven required density case.
- **Corners:** As can be seen in figure 4, pixels near a corner require a very high density on the adjacent wall, because a small area on this wall has a big influence on the pixel error. Jensen recognized this problem and uses an additional bounce when the distance between two path vertices is too small [6]. We have experimented with a crude approximation to the integral one such bounce introduces. A detailed analysis would lead too far, but results were promising and the required density in corners is effectively reduced.

Currently we are investigating techniques to generalize the single-path-importance evaluations so that, when the required density would be too high, automatically extra bounces are introduced. The distinction between specular, glossy or diffuse or corner distance thresholds would not be needed anymore. We believe generalizing ray differentials to arbitrary brdf's would be interesting for this but, as Igehy says, this is still an open question[4].

4.4 Results

We implemented the importance-based required density method into RenderPark[6] and tested it on a number of scenes.

The accuracy parameter α was chosen by hand (10000 was used) for one scene and, as was expected, also gave good results for other scenes. Scenes contained diffuse, some glossy but no specular surfaces. We tested relatively simple scenes, but the method is expected to scale just as well as the standard photon map method.

In a first pass the required density was computed for a certain view using the path-importance-based technique. Figure 8 shows a false-color overview of the computed required density. Blue corresponds to a low density while red is a very high density (a log scale is used). A minimum required density is used if the importance is too low. This explains the constant blue color in unimportant parts of the scene. Due to the glossy desk-pad a higher density is required on the cylinder and parts of the wall. About 80.000 importons were stored in the map, but this number could have been reduced, since many of them corresponded to a lower required density than our minimum.

Using the computed required density, a global photon map was constructed using density control. Figure 9 shows the actual density of the global photon map used to render the image in figure 7. This global map contained only 40.400 photons, while without importance based density control 290.000 photons would have been stored.

Note that the glossy reflections on the desk-pad are using the global map directly from the cylinder and wall. The density there is adequate for these reflections.

Some interesting points learned from the experiments were:

- Not all regions have enough photons according to the required density. However the image shows no artifacts, so probably our accuracy is even set a bit too high or the density-error relationship should be tuned.
- Since we emit photons in the same way as with the standard photon map method, dark regions can require a large number of photons to be traced before they contain enough photons. However the storage remains limited whereas in the standard method this regional density would be infeasible. Importance sampling when emitting photons as in [15] might also be useful to guide photons where more density is needed. However the time to construct the photon maps is in any case still much smaller than the time taken by the final rendering pass.
- We noticed some bias in the reconstruction when a very low-density region with higher powered photons meets a very high-density region, similar to the blurring of caustic borders when using the caustic map. Jensen's technique to reduce caustic blurring[9] could also be used here, or one could try to get a smoother transition from low to high densities. However, it seemed not necessary for the images we rendered.

A few more general remarks about the method:

- The importance-based derivation of the required storage accuracy could also be used for other algorithms, typically other final gathering approaches.
- A similar method could be derived to estimate the required density for the directly visualized caustic map. However the handling of specular bounces should be further developed first.
- As with any importance-based method, the view-independence of, in this case, the global photon map is lost.

[6]RenderPark is a physically based rendering system available at `www.cs.kuleuven.ac.be/~graphics`

5 Conclusion

This paper presented a method to control the density of photon maps. A technique was introduced to selectively store photons in a photon map, while ensuring a correct illumination representation in the map. Storage can be significantly reduced in over-dense or unimportant regions. Benefits of the original method, like the ability to handle complex geometry and materials are preserved.

The decision to store photons is based on a required density criterion, which provides a flexible framework for density control of photon maps. We have applied it to caustic maps and derived a convergence criterion for these maps. Storage gains of a factor 2 and more were obtained.

Also a novel path-based-importance first pass was used to derive the required density for global maps. Dense storage was only necessary in important parts of the scene, leading to less photons in the map. An accuracy parameter can be specified rather than determining the number of photons in the map. This takes us a step closer to answering the 'how many photons' question, a step that might even be more important than the storage reduction.

Although good results were obtained much more research is needed to tune the density control framework and to put it to full use. We assume that the error of reconstruction is simply proportional to the inverse density of the map. However much more advanced relations should be investigated using more scene information, for example:

- The actual illumination could be taken into account. For smoothly varying illumination a high density may not be necessary for a high accuracy or on the other hand a very bright area might have a significant influence although importance is low. Note that these approaches would require a simultaneous construction of illumination and required density maps, since they need each others information.
- Visual masking by high frequency textures or color saturation after tone mapping can reduce the needed density.
- Storage on glossy surfaces could greatly benefit from a directional required density criterion.

Currently we use a fixed number of nearest photons in the reconstruction. A very interesting extension would be to include this number into the framework, for example by adjusting it when the required density is not reached somewhere.

Some other points for future research are the generalization of the path-based importance to arbitrary paths and the use of importance sampling to guide photons to regions were more density is needed. Extending the density control to volume photon maps [10] for participating media is also possible.

Acknowledgments

This research was supported by FWO Grant #G.0263.97

References

1. James R. Arvo. Backward Ray Tracing. In *ACM SIGGRAPH '86 Course Notes - Developments in Ray Tracing*, volume 12, August 1986.
2. Philip Dutre and Yves D. Willems. Potential-Driven Monte Carlo Particle Tracing for Diffuse Environments with Adaptive Probability Density Functions. In P. M. Hanrahan and

W. Purgathofer, editors, *Rendering Techniques '95*, pages 306–315, New York, NY, 1995. Springer-Verlag.

3. Paul S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. *Computer Graphics*, 24(4):145–154, August 1990.
4. Homan Igehy. Tracing ray differentials. *Computer Graphics*, 33(Annual Conference Series):179–186, 1999.
5. Henrik Wann Jensen. Importance driven path tracing using the photon map. In P. M. Hanrahan and W. Purgathofer, editors, *Rendering Techniques '95*, pages 326–335 326–335. Springer-Verlag, New York, 1995. also in Proceedings of the Sixth Eurographics Rendering Workshop, 1995 (Rendering Techniques '95, Springer Verlag).
6. Henrik Wann Jensen. Global illumination using photon maps. In Xavier Pueyo and Peter Schröder, editors, *Eurographics Rendering Workshop 1996*, pages 21–30, New York City, NY, June 1996. Eurographics, Springer Wien. ISBN 3-211-82883-4.
7. Henrik Wann Jensen. Rendering caustics on non-lambertian surfaces. *Computer Graphics Forum*, 16(1):57–64, 1997. ISSN 0167-7055.
8. Henrik Wann Jensen. A practical guide to global illumination using photon maps. In *ACM SIGGRAPH '99 Course Notes - Unpublished*, chapter 7, pages 1–72. 1999.
9. Henrik Wann Jensen and Niels Jørgen Christensen. Photon maps in bidirectional Monte Carlo ray tracing of complex objects. *Computers and Graphics*, 19(2):215–224, March–April 1995.
10. Henrik Wann Jensen and Per H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. In *Computer Graphics (ACM SIGGRAPH '98 Proceedings)*, pages 311–320, 1998. r.
11. J. T. Kajiya. The rendering equation. In *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 143–150, August 1986.
12. E. P. Lafortune and Y. D. Willems. A theoretical framework for physically based rendering. *Computer Graphics Forum*, 13(2):97–107, June 1994.
13. Dani Lischinski, Brian Smits, and Donald P. Greenberg. Bounds and Error Estimates for Radiosity. In *Computer Graphics Proceedings, Annual Conference Series, 1994 (ACM SIGGRAPH '94 Proceedings)*, pages 67–74, 1994.
14. S. N. Pattanaik and S. P. Mudur. Adjoint equations and random walks for illumination computation. *ACM Transactions on Graphics*, 14(1):77–102, January 1995.
15. Ingmar Peter and Georg Pietrek. Importance driven construction of photon maps. In G. Drettakis and N. Max, editors, *Rendering Techniques '98 (Proceedings of Eurographics Rendering Workshop '98)*, pages 269–280, New York, NY, 1998. Springer Wien.
16. Peter Shirley. A ray tracing method for illumination calculation in diffuse specular scenes. In *Proceedings of Graphics Interface '90*, pages 205–212, Toronto, Ontario, May 1990.
17. B.W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapmann and Hall, New York, NY, 1986.
18. Brian E. Smits, James R. Arvo, and David H. Salesin. An importance-driven radiosity algorithm. *Computer Graphics*, 26(2):273–282, July 1992.
19. F. Suykens and Y. D. Willems. Weighted multipass methods for global illumination. In *Computer Graphics Forum (Proc. Eurographics '99)*, volume 18, pages C–209–C–220, September 1999.
20. Eric Veach and Leonidas J. Guibas. Optimally Combining Sampling Techniques for Monte Carlo Rendering. In *Computer Graphics Proceedings, Annual Conference Series, 1995 (ACM SIGGRAPH '95 Proceedings)*, pages 419–428, 1995.
21. Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A Ray Tracing Solution for Diffuse Interreflection. In *Computer Graphics (ACM SIGGRAPH '88 Proceedings)*, volume 22, pages 85–92, August 1988.

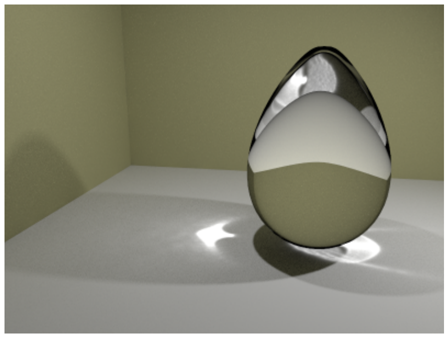


Fig. 5. Caustics cast by an egg illuminated by two light sources. About half the number of photons were stored compared to the standard photon map method, without visible differences.

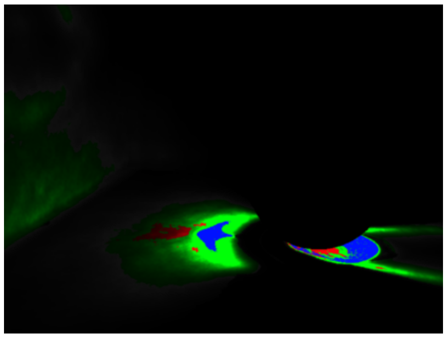


Fig. 6. Visualization of the caustic map convergence criterion. (Blue: Required density reached, Green: required density not reached, but other illumination high enough to mask the errors, Red: density too low.

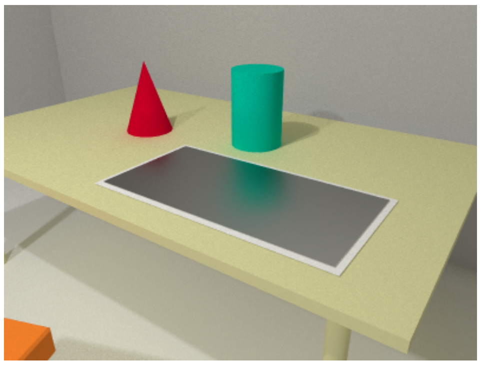


Fig. 7. Image computed using a global photon map constructed with importance driven density control. For the glossy reflection on the pad, the global map was used directly on the reflected surfaces.

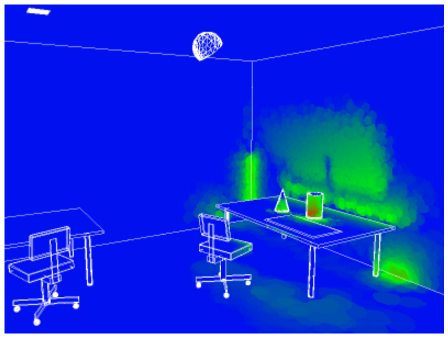


Fig. 8. Required density determined by a path based importance function stored in an importance map. The above image shows the view of the camera.

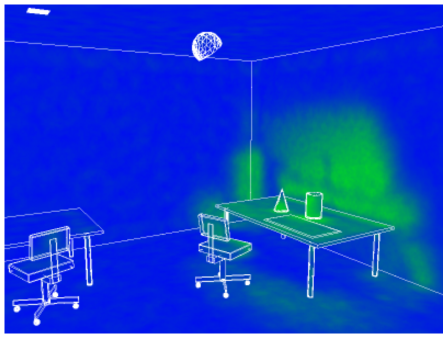


Fig. 9. Actual density of the global photon map used to generate the above image. About 5 times less photons were stored in the global map than would be stored with a standard photon map method.