Overview of biased light transport and light source minimization techniques

Kevin Streicher*
TU Wien

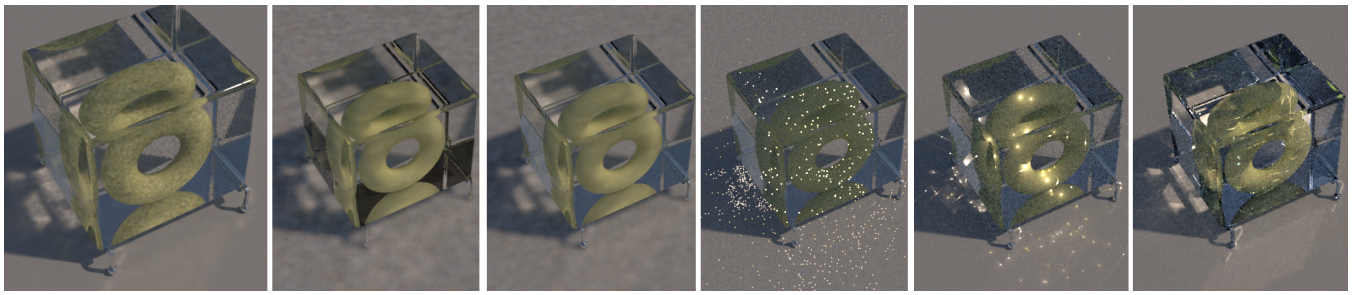

Figure 1: Photon mapping is a versatile biased rendering algorithm which can be used in conjunction of path tracing in form of hybrid algorithms to help rendering caustics and global illumination. Another strength of photon mapping is the high quality of quick preview images. All images were rendered with Mitsuba Renderer. The first three images are the results of standard, progressive and stochastic progressive photon mapping with 250.000 photons. The second three images are path tracing reference images rendered with bidirectional path tracing (16 spp), energy retribution path tracing (2 spp, 10 mutations per pixel) and path space metropolis light transport (32 spp). All images were rendered in about 8 seconds and in 512x512 pixel resolution.

Abstract

Light transport simulations are the key to physical correct rendering which allows photorealism by use of non-artistic but mathematically provable techniques. Several different state-of-the-art biased light transport algorithms are available and methods to improve efficiency and scalability are researched. Efficient global illumination algorithms for offline rendering are available both on CPU and GPU and real time solutions are also actively researched topics in the scientific community. Those algorithms often include techniques for temporal and spatial coherence, subsurface scattering and participating media. This state of the art report offers an introduction about bias and consistency to help understand the difference between widely used biased and unbiased rendering algorithms. Different photon mapping algorithms were used to render comparison images to visualize the strengths and weaknesses of the technique using the common torus scene which includes many difficult specular-diffuse-specular light paths.

Several recent extensions and improvements for both many-light rendering as well as recent accomplishments in the field of photon mapping are presented. A brief overview over various lightcut algorithms for efficient light clustering along with a recently published automatic light minimization technique is also provided. This report provides a survey over recent publications in these topics as well as ideas for possible further research directions.

CR Categories: I.3.7 [Computer graphics]: Three-Dimensional Graphics and Realism—Ray tracing;

Keywords: bias, consistency, biased light transport, many-light rendering, photon mapping, rendering, global illumination, light minimization, lightcuts, multidimensional lightcuts, bidirectional lightcuts, virtual point lights

1 Introduction

Photorealism in computer graphics is a common requirement and *physical based rendering (PBR)* allows this degree of correctness by simulation of light transport. This is why the term *light transport* is used synonymously for PBR. The goal is to create results that are indistinguishable from reality, which is especially important for *computer generated imagery (CGI)* in films where the final images can contain both real-world camera captures alongside CGI. PBR also allows the simulation of real-world lighting which is important for artists, architects, industry, product design and many other fields.

Several different physically based algorithms have been developed e.g. (in no particular order) Metropolis light transport introduced by Veach & Guibas [1997], distributed ray tracing by Cook [1984], path tracing by Kajiya [1986], bidirectional path tracing by Lafortune & Willems [1993], photon mapping by Jensen [2001] and building upon it progressive photon mapping by Hachisuka, Ogaki & Jensen [2008] and further stochastic progressive photon mapping by Hachisuka & Jensen [2009].

A general model for light transport is given by the *rendering equation* [Kajiya 1986]. This equation describes the amount of radiance leaving a point x into the direction ω. PBR algorithms essentially offer different ways to numerically solve this integral. The commonly used representation of the rendering equation is

$$L(x,\omega) = L_e(x,\omega) + \int_\Omega f_r(x,\omega',\omega) L(x,\omega')(\omega' \cdot n) d\omega', \qquad (1)$$

*e-mail: e1025890@student.tuwien.ac.at

where the radiance emitted by the point itself is $L_e(x,\omega)$. ω is the outgoing and ω' the incoming direction. $L(x,\omega')$ is the amount of radiance incoming at point x from direction ω'. Ω is the hemisphere of all possible incoming directions ω'. The *bidirectional reflectance distribution function (BRDF)* $f_r(x,\omega',\omega)$ defines the fraction of the radiance incoming from direction ω' that is reflected into direction ω at point x. The *bidirectional transmittance distribution function (BDTF)* describes the amount of light transmitted through the object into direction ω at any point x for any incoming direction ω'. The sum of both functions is referred to as the *bidirectional scattering distribution function (BSDF)*. For brevity, in our notation we only refer to the BRDF in all equations as it is commonly done in literature.

The rendering equation generalizes the reflected radiance at any point x. Usually, time and spatial coherence effects like motion blur, depth of field and others effects like subsurface scattering or spectral dispersion are important as well. Most state-of-the-art rendering algorithms are extensible to deal with those image features but light paths like *specular-diffuse-specular* (SDS) paths are very difficult to sample and some algorithms might not be able to take them into consideration. Light transport algorithms are also often not categorized and compared by image features but are classified by *consistency* and *bias*. Depending on the application one might also be interested in the general speed of convergence of specific light paths, the type of noise and image artefacts and memory requirements and a similar a categorization to this is given by Dachsbacher et al. [2014, p. 15] for many-light rendering methods.

Hachisuka [2013] explained bias and consistency in the context of Monte Carlo Integration. He also pointed out five common misconceptions about bias in light transport. One important misconception of those is that unbiased means more accurate. This is not necessarily true as one specific run of a biased algorithm could have a better result than one specific run of an unbiased algorithm. In the field of offline light transport users often require physical correctness and therefore require consistent algorithms. Bias can be accepted under some circumstances, and it usually yields a tradeoff between speed of convergence and bias. Unbiasedness does not mean that the algorithm converges to the correct solution and neither that it is more accurate that a biased one. In cases bias is accepted it should be as low as possible as we want the difference between the expected image and the ground truth to be minimal. We later provide the definition for both terms in this manuscript.

Even in theoretically unbiased algorithms it might be worth to accept bias. One example would be to naïvely shoot more rays into the direction of specular objects to improve the convergence of difficult, slowly converging light paths. Renderers which use multiple rendering algorithms in one rendering technique are called hybrid renderers. Often a biased algorithm like photon mapping is added to an unbiased path tracing framework to combine the best of both methods. Common hybrid global illumination renderers like mental ray, v-ray, light tracer are listed by Chaos Group [2014]. Other biased rendering algorithms would be irradiance caching by Ward [1994] or instant radiosity by Keller [1997]. An overview over radiance caching and its derivatives was given by Jarosz [2008, pp. 23–44] along with an introduction about Monte Carlo integration and density estimation.

This paper focuses on photon mapping variants and virtual point light based methods like lightcuts in the field of offline rendering. We have chosen photon mapping and many-light rendering algorithms as they are the recently most referred to state-of-the-art algorithms which are both *biased* and allow physically based rendering. Other important global illumination algorithms based on light transport matrices are matrix row-column sampling by Hašan [2007] and LightSlice by Ou and Pellacini [2011]. Dachsbacher et al. [2014] gave an overview over the latter two methods.

The second part of this paper focuses on light minimization techniques. The number of light sources usually directly affects the rendering time. Real world scenes with hundreds of light sources are not uncommon. For such huge globally illuminated scenes the film industry often uses supercomputers as presented by Wong []. Efficient algorithms for a large amount of light sources are therefore a requirement. Less light sources result in faster execution times [Podaras 2014]. Light clustering techniques for many-light rendering had been used in lightcuts, multidimensional lightcuts and bidirectional lightcuts by Walter et al. [2005; 2006; 2012]. Hašan et al. [2009] proposed an algorithm using virtual spherical point lights in many-light renderings with glossy surfaces. A GPU based out-of-core approach for many-light rendering was proposed by Wang et al. [2013].

2 Bias and Consistency

To understand the difference between biased and unbiased rendering algorithms it is important to understand the concept of bias and consistency in general.

We use the explanations of bias and consistency in context of Monte Carlo integration by Hachisuka [2013] and Crane [2006]. Any estimator fulfilling Equation (2) no matter how many samples are drawn is an unbiased estimator. This means that the error is expected to be zero:

$$E[f_N(X)] - I = 0, \tag{2}$$

where $f_N(X)$ is the Monte Carlo estimator for Equation (1). Any consistent estimator needs to fulfill Equation (3). This equation describes that the probability for the estimation error to be zero is equal to 1, i.e.,

$$lim_{N\to\inf}\left[\int_\Omega f(x)dx - f_N(X)\right] = 0. \tag{3}$$

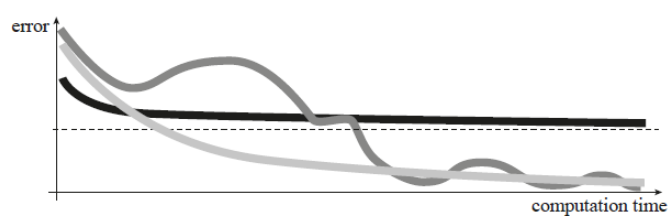

Figure 2: The convergence of the estimate of unbiased, biased, consistent and inconsistent algorithms. Dark grey represents a biased but consistent algorithm, which error is not monotonically decreasing but the error converges to zero as the estimate converges to the correct result. Light grey represents an unbiased and consistent algorithm, which error is monotonically decreasing and the algorithm also converges to the correct result. Black represents an unbiased but inconsistent algorithm, which error is monotonically decreasing but the algorithm does not converge to the correct result. As the algorithm does not increase a difference ε between the converged and the correct result remains. The image is taken from Keenan Crane© [2006].

The term consistency refers to the convergence of the algorithm to the correct result and a state-of-the-art algorithm in physically based rendering therefore is expected to be consistent. Consistency

analysis is a *limit* analysis of the algorithm and does not necessarily reflect the result achievable with memory, runtime and precision available. Under those restrictions and limited by modeling accuracy the result of an consistent algorithm still can be far from what can be seen with our own eyes. In an inconsistent algorithm the remaining error ε after convergence is still large enough to be visible. This error ε reflects all missed light paths and physical properties, e.g. missing reflections, subsurface scattering or missing participating media. A consistent algorithm does in fulfill the consistency definition in Equation 3 from the view of a mathematical analysis.

The monotonically decreasing error in unbiased algorithms allows to combine samples to improve the result. The combination of two unbiased samples is therefore more accurate than each single sample itself. This makes parallel rendering on completely independent machines possible. The bias in photon mapping is introduced by being limited in the number of stored photons. In implementations where density estimations are interpolated the interpolation does introduce another source of bias.

3 Photon Mapping

Algorithm 1 photon mapping

```
while photon map not full do                                  ▷ photon tracing
    for all lights l do
        trace photon p along ray
        if p hits surface s then
            if s is diffuse then
                store p in global photon map
            else
                reflect, transmit, absorb
                weight flux
                according to Russian roulette
            end if
        end if
    end for
    for all pixel p do                                        ▷ rendering
        trace ray from p into view direction until surface is hit
        estimate density using N nearest photons
    end for
end while
```

Photon Mapping is a two-pass global illumination rendering algorithm by Jensen and Christensen [1995]. In the standard implementation photon mapping is a biased and consistent algorithm. Algorithm 3 shows an high level description of the photon mapping algorithm. However, an in-depth explanation of photon mapping is given by Jensen [2002, pp. 15–58]. He describes photon emission, storage, rendering photon maps and the radiance estimates for photon mapping in detail as well as give an example implementation in c++. The strengths of photon mapping are the rendering of caustics, diffuse interreflection and participating media in complex scenes but its weaknesses are glossy surfaces and specular reflections which are usually rendered with distributed path tracing methods. The basic implementation uses a k-d tree.

In the first pass, a large number of photons is emitted from all light sources and their path through the scene is traced. A visualization of the photon tracing is given in Figure 3. Photons are stored on the first diffuse surface they hit. Each stored photon contains its position, power, phi-theta compressed incoming direction and a kd-tree flag. Photons are stored in one of several photon maps: the global photon map, the caustic photon map, and the volume photon map. Russian roulette depending on the BSDF is used to determine whether the photon is reflected, transmitted or stored. A photon can be stored once, multiple times or not at all if it is discarded.

The required number of shadow rays for direct illumination has a huge impact on global illumination algorithms. In algorithms where no shadow rays are used occlusion has to be computed as well and it usually is computation wise costly. To decrease the amount of shadow rays for photon mapping it is possible to use shadow photons to determine which areas are lit and which are in shadow. This approach can miss small occluders not hit by any shadow photons which is especially problematic when the overall number of photons is low. When photons hit a diffuse surface a shadow photon is traced along the original ray and this shadow photon is stored on all surfaces it hits. When during the rendering phase no shadow photon is found among the k nearest photons the point is considered lit and no shadow ray is shot.

Emission

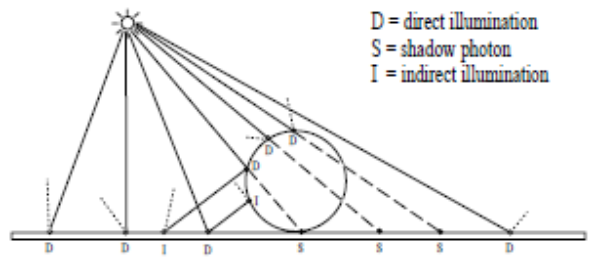

Figure 3: Photons are fired from each light source. They are stored on diffuse hit materials only as the probability to find a photon with matching incoming direction on a specular surface is zero. Their path through the scene is traced and photons are stored stochastically in one of the photon maps. Each photon carries a part of the light source radiant flux. Shadow photons are traced through opaque object to reduce the amount of shadow rays required during the rendering phase. The image was taken from Jensen© [2002].

All types of light sources are possible and the distribution of photon emission should reflect the intensities of all light sources. Point lights emit photons uniformly in all directions and most area lights emit photons accordingly to a photon emission function $f(\omega)$ dependent on the outgoing direction ω. All photons should carry equal flux to avoid wasting computation time with photons of low contribution.

This is very costly for sparse scenes as many photons will not hit any objects. Jensen [2002, pp. 15–58] suggests the use of projection maps to overcome this problem. The type of projection map has to be chosen according to the type of the light source, e.g. a point light source requires spherical projection, while a directional light requires a planar projection. A bounding volume for objects can be used for faster generation of the projection map but a trade-off between generation cost of the projection map and the cost for shooting wasted photons has to be made. After the generation of the projection map photons are shot only into active cells of the projection map. When projection maps are used it is required to scale the flux per photon accordingly as in,

$$P_{photon} = \frac{P_{light}}{n_e} \frac{\text{cells with object}}{\text{total number of cells}}, \tag{4}$$

where P_{photon} is the scaled flux. The first fraction in Equation 4 is the ratio of light source power to the total number of emitted photons. The second fraction is the ratio between active and total cells in the projection map. The use of projection maps is especially helpful for the separate generation of the caustic photon map as the distribution of specular objects is often very sparse.

The path of photons is usually terminated by *Russian roulette*. Russian roulette does help to keep the number of photons for each bounce similar as more photons are generated for reflection and transmission and an equal number of photons per path length is wanted. The decreased probability of survival on each interaction accounts for the increased number of photons. The drawback of Russian roulette is that it does increase the variance of the final image. When a sufficient number of photons are used, the output will converge to the correct result.

Storage

Photons are only stored on diffuse surfaces as the probability of finding a matching photon of incoming direction on a specular surface is zero and specular reflections are therefore rendered using distribution ray tracing. When a photon hits a specular material it will be stored in both the global as well as the caustic map on its first diffuse interaction.

For all diffuse paths the information about the photon-object interaction is stored in a global photon map which is a flat array during the tracing pass. For efficiency purposes the array is reorganized into a balanced k-d tree before the rendering phase. To include participating media ray marching and a *volume photon map* for photon-media interactions can be used..

Photon mapping differentiates between low resolution diffuse lighting and high resolution caustics. Photons reflected at a specular surface at least once are stored in a separate *caustic map*. The caustic map requires a higher density to avoid blurred edges in caustics. The density can be increased by using a projection map and shooting photons more likely into the direction of specular objects. Figure 3 shows two rendered images where the photon map is directly visualized and 100 and 500 photons are used in the radiance estimate.

A *balanced k-d tree* is used for storage. It is required to locate the k nearest photons very often and therefore this operation has to be optimized well. Jensen [2002] proposes a balancing algorithm similar to balancing algorithms for binary trees as well as how to compress the required memory for photon storage by precomputing phi-theta-encodings for incoming angles. Grid based photon maps were used by Purcell et al. [2003] and stochastic spatial hashing has been proven to be very efficient for progressive photon mapping by Hachisuka and Jensen [2010]. Pedersen [2013] has evaluated k-d tree based, grid-based, and stochastic hashing methods for progressive photon mapping on the CPU and was able to show how the latter are significantly faster than the k-d tree approach.

Rendering

In the second pass the stored photons are used to illuminate the scene. The radiance estimate for a point x is defined as in,

$$L_r(x,\overrightarrow{\omega}) \approx \frac{1}{\pi r^2}\sum_{p=1}^{N} f_r(x,\overrightarrow{\omega_p},\overrightarrow{\omega})\Delta\Phi_p(x,\overrightarrow{\omega_p}). \tag{5}$$

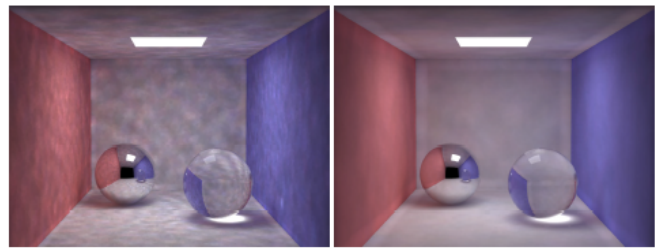

Figure 4: The results of a direct visualization from Jensen's© [2002] rendering of the Cornell box. For the left image used 100 and for the right image 500 photons were used in the radiance estimate. The false color bleeding is the result of using a sphere for locating the photons. Using a disc or an ellipsoid can decrease this problem.

Equation (5) estimates the radiance of N photons inside a sphere of radius r at a point x. This assumes that the surface is locally flat. Each photon carries the flux $\Delta\Phi_p(x,\overrightarrow{\omega_p})$ and its contribution is calculated using the BRDF $f_r(x,\overrightarrow{\omega_p},\overrightarrow{\omega})$ of the material hit. It would be possible to use other volumes than a sphere for the radiance estimation, but the area of an intersection between a plane and a sphere can be computed efficiently.

There are two ways to render photon maps. One possibility is to directly render the photon map as in Figure 3. This would be a global illumination algorithm but it is very inefficient. The other possibility is to use photon maps in hybrid algorithms. This means that not the whole rendering equation is estimated using photon maps. The integral is split and some parts are estimated with some other algorithm like distributed ray tracing.

Jensen [2002] splits the rendering equation into four part as follows,

$$\begin{aligned} L_r(x,\omega) &= \int_\Omega f_r(x,\omega',\omega)L_i(x,\omega')cos(\theta_i)d\omega_i' \\ &= \int_\Omega f_r(x,\omega',\omega)L_{i,l}(x,\omega')cos(\theta_i)d\omega_i' + \\ &\quad \int_\Omega f_{r,s}(x,\omega',\omega)(L_{i,c}(x,\omega')+L_{i_d}(x,\omega'))cos(\theta_i)d\omega_i' + \\ &\quad \int_\Omega f_{r,d}(x,\omega',\omega)L_{i,c}(x,\omega')cos(\theta_i)d\omega_i' + \\ &\quad \int_\Omega f_{r,d}(x,\omega',\omega)L_{i,d}(x,\omega')cos(\theta_i)d\omega_i'. \end{aligned} \tag{6}$$

The first integral in Equation (6) represents *direct* illumination, the second *specular and glossy* reflections, the third *caustics* and the fourth integral represents the *diffuse indirect* illumination. The integrand is the product of BRDF, incoming light, and a geometry term. Diffuse illumination and caustics are numerically integrated with photon map estimation. Monte Carlo ray tracing with importance sampling and the irradiance gradient caching scheme by Ward [1988] are used for specular and glossy reflections. Shadow rays are shot to determine the light source occlusion for direct lighting.

A hybrid algorithm is used because the required number of photons for good estimation of specular and glossy features is impractical. Final gathering is used to evaluate the multiple diffuse illumination. The photon map can help to find regions of interest and shoot more rays into directions where there are more photons. Keller [2000]

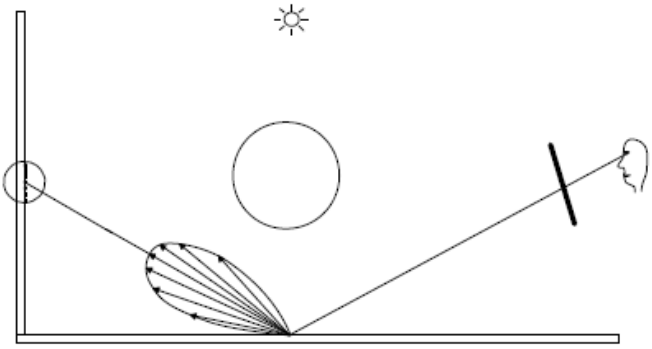

Figure 5: Importance sampling can be used for photon mapping to improve indirect illumination by shooting more rays according to the BRDFs. The image was taken from Jensen© [2002].

has published efficient importance sampling techniques for photon mapping and Suykens [2002] discusses visual importance and an estimation of the error bound. Visual importance can be used to estimate the required density in the photon map in each region of the scene and can be used in both the emission as well as the storage step.

A *final gathering* step is used to evaluate the diffuse interreflection. Final gather is a ray tracing scheme where samples over the hemisphere of each point are shot and photons or radiosity caches are used for the density estimation at each hit point of those rays. Naive final gather requires many rays for high quality renderings and Kato [2002, pp. 159–191] presents final gather techniques implemented in the distributed parallel renderer Kilauea. Among those he discusses sparse final gathering, hit point reprojection, caching final gather ray information, partial final gather re-shooting and in case of short distance final gather hit points a second pass final gather algorithm.

Problems

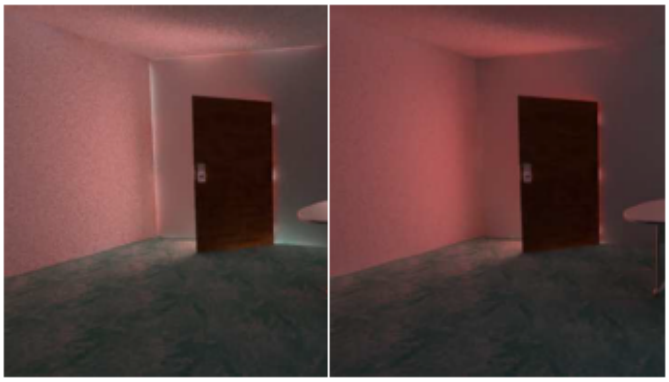

Figure 6: As the k nearest photons are selected even photons which are unable to illuminate a point x are included in the estimate. This leads to the wrong estimate. The left image shows light leakage from photons behind the wall. The right image shows how energy leakage can be reduced by storing photons in the global photon map on the second diffuse intersection instead of the first. This idea and both images were taken from the Metropolis photon sampling publication by Fan© [2005].

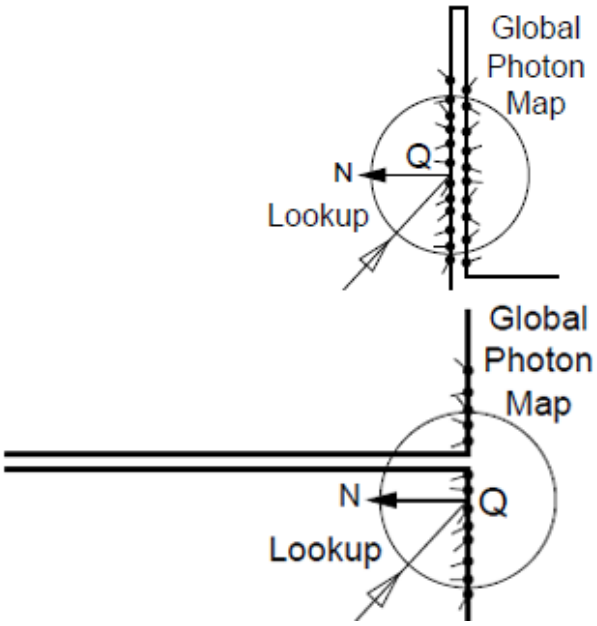

Figure 7: Both images visualize cases of light leakage. Photons which can not contribute to the point x are included in the estimation. The top case can be solved by storing the surface normal for each photon to differentiate the two surfaces. However, this does not help in the bottom case where the photons lie on surfaces with the same normal and are divided by a thin occluder. The images were taken from Kato© [2002, p 135].

The main problems of photon mapping are wrong density estimates because of light leakage and that the number of photons one can shoot and store is limited. Light leakage is the inclusion of photons into the estimate which can not contribute to the density because of occlusion. An example is given in Figure 3. Photons behind the wall are added to the density estimate as they are close to the point x on the visible side of the wall. One possibility to reduce this problem is the deformation of the sphere into an ellipsoid to include less wrong photons. Usually a disc instead of a volume is used to locate the photons but this does still include wrong photons. Other volumes and projected areas are possible but computation wise more expensive so a tradeoff between search cost and error has to be made. Another improvement is to store the surface normal which helps to identify photons on the opposite site but it does not help when they lie on the same surface and are divided by such a wall. This two cases were visualized by Kato [2002, pp. 122–193] as in Figure 3.

Another problem with photon mapping is that the density estimation can lead to blurry image features. This can be preferred over high frequency noise in case of diffuse global illumination which usually only changes slowly over the image but is problematic for caustics which often contain sharp edges. Good filtering techniques are needed to reduce this problem. Photons close to our point of interest x should be weighted strongly while photons further away should get less weight. Jensen [2002, p. 33] suggests to either use a cone or a Gaussian filter. More elaborate filtering techniques were introduced by Myszkowski [1997].

Photon mapping does not take the position of the camera into account and a lot of photons might not contribute to the final image. One way to reduce this problem is to use importance mapping by Soykens [2000]. The idea is to shoot importons from the viewer into the scene to identify important areas. An area is important

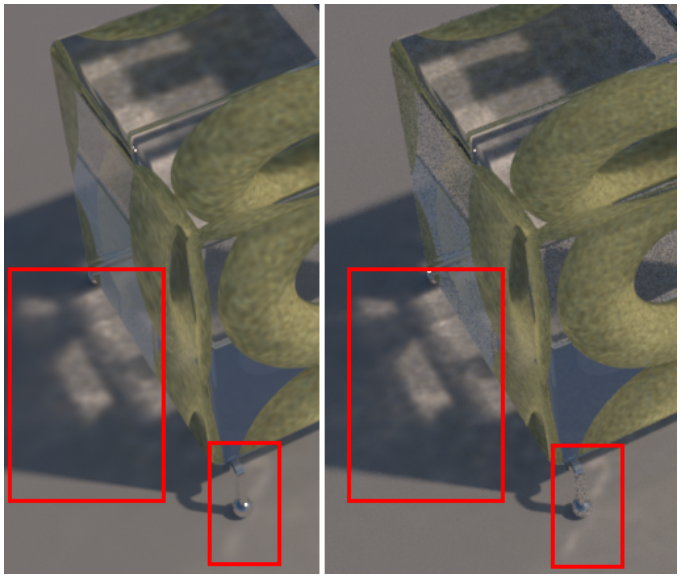

Figure 8: Both images were rendered with photon mapping with an 512^2 resolution in Mitsuba [2010]. The left image has less noise in the region of the floor visible through the cube. The reason for this is that the left image uses 32 spp instead of the 4 spp used in the right image. However, although the left image was rendered in 55 seconds and the right image in only 16 seconds we can see that the caustics in the right image are sharper than the blurry caustics in the left image. This was achieved using 500.000 photons in the right image instead of 250.000 in the left image. This means the photons used in each density estimation are expected to lie closer together which leads to a less blurry solution.

when light from photons in this area will reach the viewer. When importance mapping is used photons are only stored in case that the density of the photon map is too low and otherwise the energy is distributed among the nearest photons.

Over the years several photon mapping improvements were developed to tackle the problems and hindrances of the algorithm and they are important to include in an implementation ensure proper render times. It is therefore recommended to include visual importance by Suykens [2002, pp. 61–90] or Keller and Wald [2000]. The implementation of Wards [1988] irradiance caching is another by Jensen [2002, p. 41] as important considered improvements. The idea is to only calculate indirect illumination if an interpolation might be insufficient. Where possible a hybrid rendering algorithm might be the better choice than direct visualization of the photon map. Christensen [2002, pp. 93–121] discusses further improvements like frame coherence, iterative faster lookup for the n nearest neighbours, precomputed radiance estimates, unbiased radiance estimates, combining lookup results from several photon maps and importance drive photon tracing. Kato [2002, pp. 122–193] explains how to combine multiple photon maps for cases where one photon map does not fit into the memory, he explains the performance behaviour of parallel photon mapping as implemented in Kilaeua as well as improvements to the final gathering scheme like faster parallel processing and final gather estimation and reprojection. He also provides information about parallel photon mapping in Kilauea.

Transparent objects are difficult especially in cases where areas of the scene are mostly enclosed by transparent materials. Kato [2002, pp. 122–193] also describes a possibility to deal with such cases by partially ignoring transparent objects for photon mapping.

Overall photon mapping is a very common and capable state-of-the-art algorithm but it is rarely used with direct visualization of the photon map. In conjunction with ray tracing algorithms photon mapping can enhance difficult image parts like caustics. Although photon mapping is consistent in the limit in reality the algorithm is bound by the memory available for the number of photons which can be stored in the photon map.

To achieve the necessary speed needed for real time caustics with photon mapping Günther [2004] proposed an image space filtering method relying on box filters without any continuity tests. Larsen and Christensen [2004] presented a photon mapping algorithm relying on drawing points in the framebuffer and filtering using the GPU to achieve real-time frame rates for dynamic scenes. McGuire and Luebke [2009] presented an image based photon mapping algorithm which is able to render non trivial scenes at interactive frame rates using rasterization.

4 Progressive Photon Mapping

Algorithm 2 progressive photon mapping

```
for all pixel do                                   ▷ ray tracing pass
    trace ray until first diffuse hit point h
end for
while time is not up do                            ▷ photon tracing
    for all lights l do
        trace photon p along ray
        if p hits surface s then
            if s is diffuse then
                store p in global photon map
            else
                reflect, transmit, absorb
                weight flux
                according to Russian roulette
            end if
        end if
    end for
    for all hit points h do                        ▷ progressive radiance estimate
        decrease radius r
        calculate new radiance τ_M(x, ω⃗)
        recall stored radiance τ_N(x, ω⃗) for h
        update stored radiance
    end for
    discard photon map                             ▷ store radiance for h instead
end while
```

Standard photon mapping is unable to converge to the correct result with any arbitrary precision as the algorithm is limited by the available memory. *Progressive photon mapping (PPM)* by Hachisuka [2008] improves this by allowing arbitrary precision with a limited amount of memory. Algorithm 4 gives a high level overview of the algorithm.

PPM is a multipass rendering algorithm where the first pass is a ray and all subsequent passes are photon tracing passes. The memory limitation is bypassed as not all photons are stored and instead the estimation is improved with the new photons during each pass and then the photons are discarded.

After each photon tracing step the density is averaged over the area of the disc used to locate the photons as in Equation 7. As the radius of the disc needs to be reduced in each step to converge to a zero radius and allow arbitrary precision it is necessary to calculate the number of photons in the smaller disc of size $R(x) - dR(x)$ as in

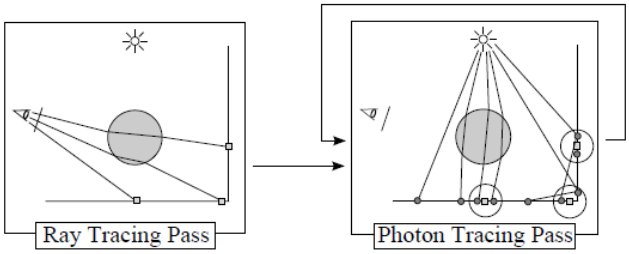

Figure 9: PPM uses one first ray tracing pass in which all hit points are stored until the first diffuse point to find all diffuse points visible in the image. Paths can be terminated by russian roulette. All subsequent passes are photon tracing passes. The image was taken from Hachisuka© [2008].

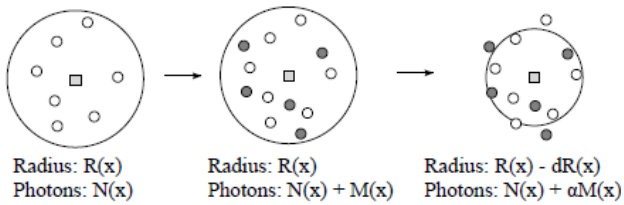

Figure 10: A visualization of the increase of photons and radius reduction in each step of the photon tracing passes as defined by Equation 7 and Equation 8. The increase of photons and the reduce of the radius are necessary to ensure that PPM is consistent. The image was taken from Hachisuka© [2008].

Equation 8. This assumes that the density in $R(x)$ is constant which requires the number of photons to increase by $\alpha M(x)$ with α as the rate of photons to keep in each step. During each step at each hit point the total flux is accumulated and stored as the *unnormalized flux*,

$$L(x,\omega) \approx \frac{1}{\pi R(x)^2} \frac{\tau(x,\omega)}{N_{emitted}}. \tag{7}$$

The idea is to reduce the current radius $R(x)$ from iteration to iteration by some value $dR(x)$. It is necessary to reduce the number of photons as the reduction of the radius leads to some photons dropping out as in,

$$\hat{N}(x) = \pi \hat{R}(x) \hat{d}(x) = \pi (R(x) - dR(x))^2 \hat{d}(x), \tag{8}$$

where $\hat{N}(x)$ is the new number of photons. This leads to the formula for the reduced radius as in,

$$\hat{R}(x) = R(x) - dR(x) = R(x) \sqrt{\frac{N(x) + \alpha M(x)}{N(x) + M(x)}}, \tag{9}$$

where $R(x)$ is the current radius, $\hat{R}(x)$ is the radius used in the next iteration and the number of photons should increase by a factor $\alpha M(x)$.

As the radius between two passes is reduced it is impossible to simply add up the unnormalized flux of the last and the current pass. To accredit for the radius changes the new number of photons is calculated as in Equation 8 and the new radius is calculated as in Equation 9. The radiance estimate for photon tracing passes is Equation 7. The BRDF is already multiplied into the unnormalized flux $\tau(x,\omega)$. We refer the interested reader to Hachisuka [2008, pp. 4–5] for a derivation of those formulas.

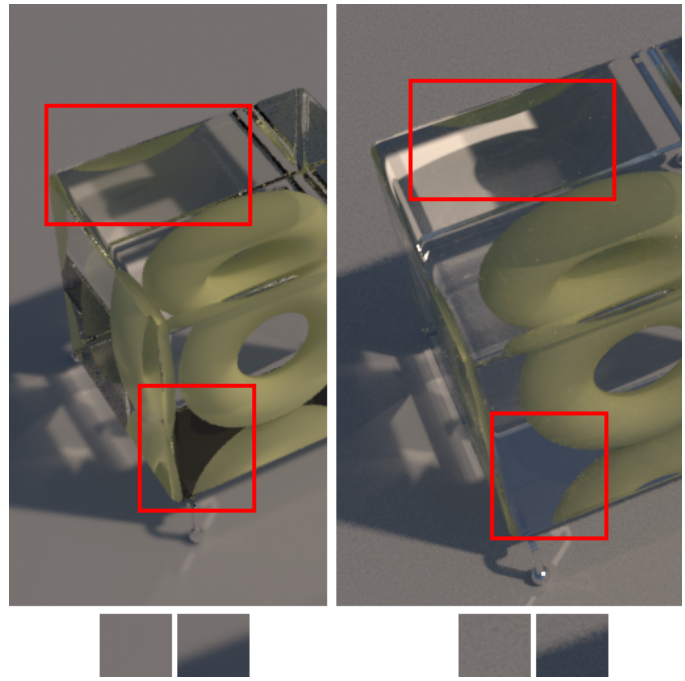

Figure 11: The result of the torus scene rendered with Mitsuba [2010] for 3 hours with a 512x512 pixel resolution. The left image is rendered with progressive photon mapping and the right image with path-space Metropolis light transport (32768 spp). PPM still misses some light paths in the marked areas and the caustics are not as sharp as those from the PSMLT rendering. However, the PPM rendering is less noisy because the noise is blurred.

As progressive photon mapping averages the normalized flux it can be expected to be converge slower than standard photon mapping because this requires more samples for an accurate representation. Progressive photon mapping is to prefer over photon mapping when not the rendering time but the memory is the limitation. However, it is not easy if not impossible to determine the required density and memory requirement for standard photon mapping beforehand such that the result image error is bounded by an arbitrary chosen ε.

Pedersen [2013] presented a parallel implementation of PPM on GPUs using the *memoryless* (stochastic) progressive photon mapping of Knaus and Zwicker [2011]. Knaus and Zwicker had shown that a radius reduction without the storage of local statistics is viable. They show that this is possible with a reformulation of progressive photon mapping where the expected average error as well as the variance converge to zero.

The idea of Knaus and Zwickers memoryless progressive photon mapping allows an increase of the variance of the error by a factor as given in Equation 10,

$$\frac{Var[e_{i+1}]}{Var[e_i]} = \frac{i+1}{i+\alpha}. \tag{10}$$

The parameter α with $0 < \alpha < 1$ defines how fast the variances increases. As the variance is inversely proportional to the square

radius as in Equation 11 it is possible to derive a formula for the radius reduction as in Equation 12, i.e.,

$$\frac{r_{i+1}^2}{r_i^2} = \frac{Var[e_i]}{Var[e_{i+1}} = \frac{i+\alpha}{i+1}, \tag{11}$$

$$r_i^2 = r_1^2 \left(\prod_{k=1}^{i-1} \frac{k+\alpha}{k} \right) \frac{1}{i}, \tag{12}$$

and the representation of their Monte Carlo estimate is given as

$$\overline{c}_N = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{p_e(x_i, \omega_i)} W(x_i, \omega_i)(L(x_i, \omega_i) + e_i), \tag{13}$$

$\overline{c}_N$ in Equation 13 is the pixel value after N samples (x_i, ω_i). The pair (x_i, ω_i) is the position and incoming direction of path the sample.

The inverse proportional relation of variance to squared radius in Equation 11 allows an explicit radius reduction as in Equation 12. They discuss several possibilities for the choice of the initial reference radius r_1, e.g. one global reference radius, a local reference radius dependent on the pixel footprint or a local reference radius dependent on the distance to the k-nearest neighbours.

The high level representation of their algorithm from Knaus and Zwicker [2011] is given as in Algorithm 4.

Algorithm 3 Memoryless progressive photon mapping as high level pseudo code from Knaus and Zwicker [2011]

```
i ← 0
while time is not up do
    generate photon map
    for all pixel do
        trace path from eye until diffuse surface is hit
        hit position and direction are (x_i, ω_i)
        path contribution is W(x_i, ω_i)
        path probability density is p_e(x_i, ω_i)
        get current radius r_i from reference r_1 and Eq. 11
        obtain radiance estimate L(x_i, ω_i) + e_i
        update pixel value from Eq. 13
    end for
end while
```

Their algorithm is essentially the same as a standard (stochastic) progressive photon mapper and therefore can be integrated easily into a photon mapper. The algorithm uses a different predetermined radius reduction which allows parallelism because no local statistics are required. They report render times and results which are almost the same as for stochastic progressive photon mapping.

Pedersen [2013] builds upon memoryless progressive and describes an implementation of it on the GPU. He also implemented a grid based and a hash table approach for photon storage on the GPU and showed that those two approaches are faster than the standard k-d tree based implementations on a GPU. We refer the reader to Knaus and Zwicker [2011] for the proof and derivation of the formulas for memoryless progressive photon mapping. They have shown that both the expected error as well as the variance of their method converges to zero and that the generated sequence of radii leads to the same sequence as PPM.

Kaplanyan and Dachsbacher [2013] presented an adaptive progressive photon mapping algorithm. They show that a unique *asymptotically* optimal α exists and that the initial choice of radius r_1 is crucial. A different approach for the choice of α was given by Fu and Jensen [2012]. A comparison of those two choices is still future work.

Liu [2014] showed how the importance photon shooting in both photon mapping and progressive photon mapping can be improved by using an adaptive method.

5 Stochastic Progressive Photon Mapping

Algorithm 4 progressive photon mapping

```
for all pixel do                                   ▷ ray tracing pass
    trace ray until first diffuse hit point h
end for
while time is not up do                            ▷ photon tracing
    for all lights l do
        trace photon p along ray
        if p hits surface s then
            if s is diffuse then
                store p in global photon map
            else
                reflect, transmit, absorb
                weight flux
                according to Russian roulette
            end if
        end if
    end for
    for all hit points h do                        ▷ progressive radiance estimate
        sample random point x⃗_i in search radius S
        decrease radius r
        calculate new radiance τ_{i+1}(S, ω⃗)
        update stored radiance for region S
    end for
    discard photon map                             ▷ store radiance for h instead
end while
```

Stochastic progressive photon mapping (SPPM) is an improvement of progressive photon mapping by Hachisuka [2009]. The main idea of SPPM is to use a distributed ray tracing pass after each photon tracing pass to estimate the density over the area around the hit point. The radius change as well as light estimation remains unchanged from ppm but instead of the fixed hit points x a region S with random sample point x_i is used. Those hit points x_i are the result from the distributed ray tracing pass. Algorithm 5 shows the pseudo-code for the stochastic progressive photon mapping algorithm.

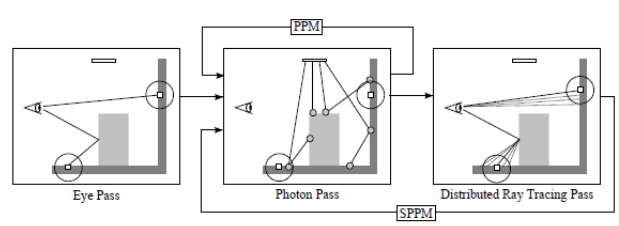

Figure 12: The idea of SPPM extends the PPM algorithm by using a distributed ray tracing step after each photon tracing step. While PPM uses the fixed hit points generated by the first ray tracing step the SPPM algorithm generates random samples into the direction of those points to generate a density estimation over the area around a hit point instead. The image was taken from Hachisuka© [2009].

The density estimate for an area S is given by

$$L(S,\omega) = \lim_{i\to\inf} \frac{\tau(S,\omega)}{N_{emitted}\pi R(S)^2}, \tag{14}$$

which it is essentially the same equation as for progressive photon mapping except that an area is evaluated instead of points. This simple change is achieved by using the hit points x_i generated during the distributed ray tracing pass as in Figure 5.

We refer the reader to Hachsiuka [2009] for the proof that this change still converges to the correct solution.

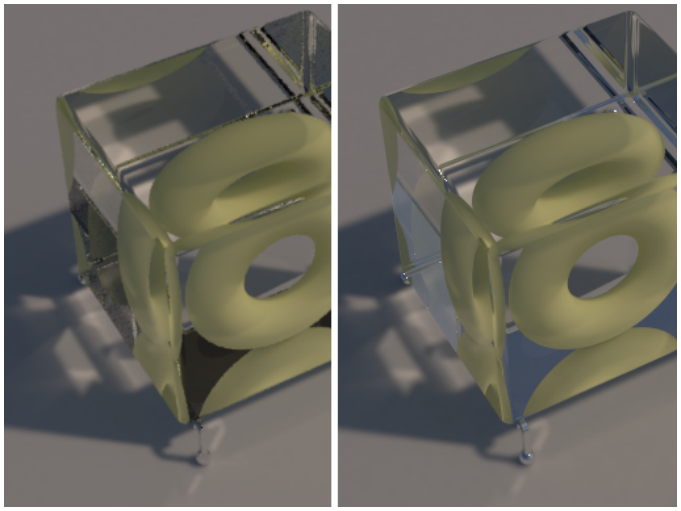

Figure 13: Both images were rendered with Mitsuba [2010]. The left one was rendered with PPM for 3 hours and the right image with SPPM for 25 minutes. The SPPM algorithm already had shown better convergence in difficult SDS areas inside the cube than the PPM algorithm.

Overall SPPM is simply an improvement to PPM and should be preferred for most scenes. It does take slightly longer to render because the distributed ray tracing adds another step which needs enough samples for good converges. As SPPM does estimate the density over a region it is well suited for effects like depth-of-field or motion blur effects.

Weiss and Grosch [2012] published an approach for animations called stochastic progressive photon mapping for dynamic scenes (DSPPM). The idea is to identify hit points and photons which are to static objects and reuse them. This is also done for photon paths where a photon path is split in case the photon hits a dynamic object.

6 Many-Light Rendering

Many-light rendering methods are algorithms for fast global illumination where virtual light sources are distributed in the scene and the illumination is calculated as the direct lighting of each of those *virtual light sources (VLS)*. An increasing number of light sources in general affects the rendering time negatively because every light source has to be treated separately. Algorithms to reduce the impact of the number of light sources therefore are required. This is not a problem for biased rendering in particular but for all rendering algorithms in general. In ray tracing and photon mapping algorithms shadow rays have to be shot to each light source. In photon mapping the number of photons needs to increase with more light sources to ensure sufficient density but the shadow photon approach can be expected to be even more important to reduce the number of shadow rays shot. For many-light rendering light reduction techniques might be even more interesting as the number of virtual light sources can be 100.000 and more.

Dachsbacher et al. [2014] published a survey about scalable realistic rendering with many-light methods. This second part of this paper focuses on providing a short excerpt about many-light renderings and to extend this to light minimization techniques to deal with a huge number of lights. Light minimization techniques focus on clustering lights to reduce the computational rendering effort or to turn off lights and redistribute their energy to other light sources. However, light minimization techniques can be expected to benefit all state-of-the-art rendering algorithms for scenes with many light sources.

The basis of many-light rendering methods is the instant radiosity method by Keller [1997]. The basic idea is to shoot light particles and create a *virtual point light source (VPL)* at every surface they hit.

Algorithm 5 A basic high level pseudo code for many-light rendering algorithms based on the description by Dachsbacher et al. [2014]

```
if phase == VPL generation then
    randomly choose primary light source S
    sample a random point x and direction ω by creating a VPL
    trace the ray x + tω. If it intersects a surface then create a VPL at the intersection location
    Decide randomly whether to reflect, transmit or stop using Russian Roulette.
end if
if phase == rendering then
    for all surface points do
        i ← 0
        while i < M do
            if illumination is not blocked then
                compute contribution of VPL[i]
            end if
            i ← i + 1
        end while
    end for
end if
```

Many-light rendering algorithms like instant radiosity have problems with difficult surfaces like glossy materials the same way as photon mapping algorithms as a huge number of VPL, respectively photons, are required for convergence. Similar to photon mapping where all photons should carry the same flux and should be stored in important regions the same requirements hold for VPL. Probability based rejection of unimportant VPLs by Georgiev and Slusallek [2010] is the idea to reject VPLs with low flux and normalize the carried flux accordingly to the rejection probability. An alternative to the VPL rejection is the Metropolis-Hastings sampling as suggested by Segovia et al. [2007]. Similar to the importance driven photon tracing a method to generate VPLs from the camera view was proposed as part of the bidirectional instant radiosity method by Segovia et al. [2006].

To efficiently be able to render high quality many-light images a huge number of VPL is required. Algorithms which are able to deal with millions of light sources efficiently are called scalable many-light algorithms. Dachsbacher et al. [2014] define scalable rendering algorithms in the context of many-light rendering as algorithms which cost increases slowly, or sub-linearly, with the num-

ber of used light sources. The common methods among these are lightcuts, multidimensional lightcuts and bidirectional lightcuts by Walter [2005; 2006; 2012], matrix row-column sampling by Hašan et al. [2007] and LightSlices by Ou and Pellacini [2011]. An approach based on genetic algorithms to automatically modify light source intensities and group light sources using a global error metric had been presented recently by Podaras [2014]. All those methods share the idea to select a sub set of light sources, group them and refine the selection based on an error metric.

Compared to photon mapping methods, less light sources have to be distributed than photons but their evaluation is more costly. While most VPL based methods are unable to render caustics and can produce singularities. Singularities are caused by the fact that the squared falloff term can blow up the light source contributions to arbitrarily large values. To alleviate this, most VPL based algorithms clamp the contribution at the cost of introducing bias.

Wang et al. [2013] presented an out-of-core many-light rendering algorithm able to handle scenes with 17.1 million triangles and 256 million lights. They modified lightcuts by Walter [2005] and implemented their version in an out-of-core framework.

7 Lightcuts

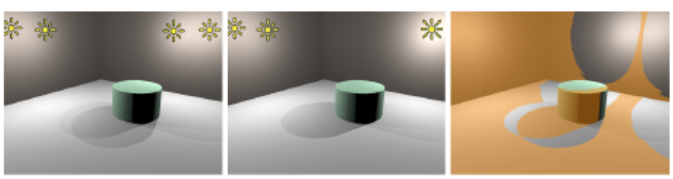

Figure 14: The image was taken from Walter© [2005] and shows the reference image with all four light sources (left), the result of one particular lightcut (middle) and the regions of low error (right).

Lightcuts had been the first scalable many-light rendering algorithm. The idea is to structure all lights in a binary tree and begin at the root node representing a cut where all light sources are grouped into one cluster. Each cluster is a combination of two light sources, two clusters or a combination thereof. An example scene and one lightcut is shown in Figure 7.

The algorithm iteratively refines the clusters by selecting the node with the highest relative error bound. This node is refined by replacing the node by its child nodes. A relative error bound of 2% is suggested. The selection of nodes in the light tree is called lightcut. Figure 7 shows three different lightcuts and the result rendering.

The binary light tree should cluster lights together which are as similar as possible. Therefore one separate tree for omnidirectional, oriented and directional point light sources is build. The building of the tree itself is not sublinear but as it has to be done only once per image or once per scene for static light sources the cost is insignificant.

8 Advanced Lightcuts

Reconstruction cuts are a way to increase the speed of lightcuts by computing them sparsely. The biggest cost of lightcuts are still shooting shadow rays. By using lightcuts sparsely and interpolation

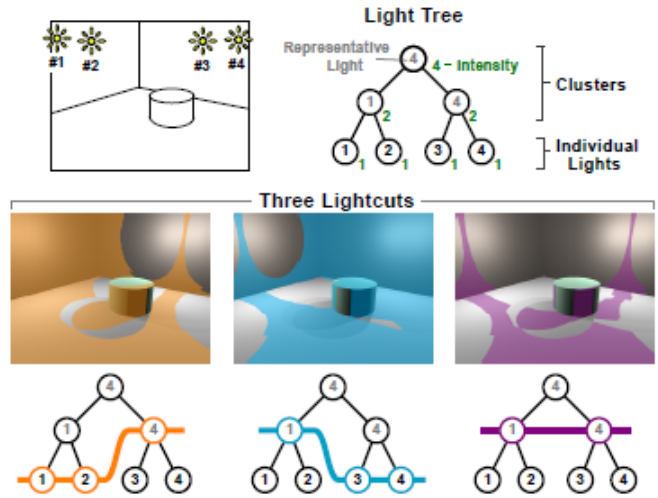

Figure 15: The top row shows the scene and the position of all four light sources. The second row shows the scene for three different lightcuts. The coloured areas are those where the difference to the scene with all four individual light is small. The third row shows the corresponding light clustering for each of the three lighcuts. The reference image is the left image in Figure 7. The image was taken from Walter© [2005].

between those lightcuts the amount of shadow rays can be significantly reduced. The image is separated into 16x16 tiles and lightcuts are calculated for the corners. Each block is subdivided if the samples at its corners do not match. The samples are considered a match if they hit the same type of material with surface normals which do not differ more than an angle α (e.g. 30 degrees). At each sample position a cone test is done to verify if the other points lie within the cone. If any of the points lies inside the cone of one of the other points it could be that the geometry of the surface shadows each other. In such cases further subdivision is required.

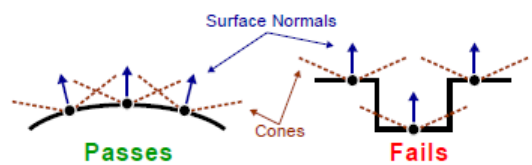

Figure 16: A cone test for two types of geometry. In the left case no point lies in the cone of the others and no further subdivision is required. In the right case the higher two points lie inside the cone of the lower point. This means the geometry possibly shadows the lower point and therefore subdivision is necessary. Image from Walter© [2005].

Multidimensioanl lightcuts by Walter et al. [2006] extend the idea of lightcuts to more dimensional integrals. This allows integration over time or the cameras aperture for motion blur and depth of field effects. The authors describe the multidimensional lightcuts as the integrals of radiance,

$$pixel = \int_{time} \int_{volume} \int_{aperture} \int_{pixelarea} L(x, omega), \qquad (15)$$

over time, volume, aperture and pixel area.

The primary light sources are discretized into a set of point lights $\mathbb{L}$ as for lightcuts. For each pixel a set of gather points $\mathbb{G}$ is generated and the pixel color is calculated as in,

$$pixel = \sum_{(j,i)\in G\times L} S_j M_{ji} G_{ji} V_{ji} \tau_{ji} I_{ji}, \tag{16}$$

where M is the material, G the geometry, V the visibility and I the intensity term. τ_{ji} is a binary decision variable which is one if the points exist at the same time and zero otherwise. As an evaluation over all pairs $(j,i) \in G \times \mathbb{L}$ is too expensive and an implicit product graph is used instead. The product tree is generated by generating a tree for all points $j \in \mathbb{G}$ and one tree for all points $i \in \mathbb{L}$. The product graph is the Cartesian product graph of those two trees. The main benefit of this approach is that only the two small gather and light trees are required instead of the larger product graph. This means that instead of millions of pairs only a small subset of several hundreds have to be evaluated on average. The VPL generation is the same as for lightcuts.

Bidirectional lightcuts by Walter et al. [2012] extends the methods for efficient evaluation of millions of light and gather pairs introduced in multidimensional lights cuts into a bidirectional framework. Instead of just using one bounce for eye rays during the generation of gather points several bounces are allowed. The gather points are referred to as *virtual sensor points (VPS)*. This allows them to render more light paths and material models like subsurface scattering which are strongly clamped in usual VPL methods. The bidirectional lightcuts are also very efficient for glossy surfaces as the countertop in Figure 8. The bidirectional lightcut algorithm could be included in other VPL algorithms but they suggest multidimensional lightcuts as this method is very efficient in evaluating only a tiny subset of all VPS and VPL clusters. With new weighting strategies they are able to stop paths earlier and to discard VPS with low contribution.

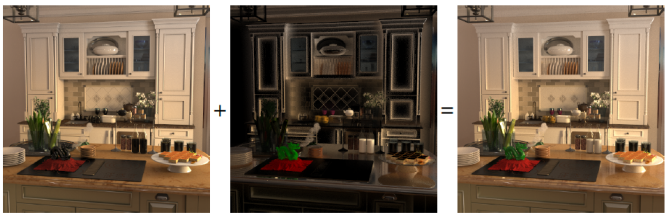

Figure 17: The composition of standard VPL and bidirectional paths which allows for paths including subsurface scattering which result in a high quality noise free image from bidirectional lightcuts (BDLC). The image was taken from Walter© [2012].

Progressive lightcuts on GPU by Davidovič et al [2012] choose a different approach to decrease the bias of lightcuts. They use a progressive relaxation scheme, which changes the clamping for singularities depending on the number of VPL.

9 Automatic Lighting Design

Podaras [2014] presented a light source minimization technique to reduce the number of light sources similar to lightcuts. Their algorithm does not require to build any additional data structure and supports full global illumination. All light sources are represented as entries in one vector. Either the light sources are represented by a binary variable which indicates if the light source shall be used or

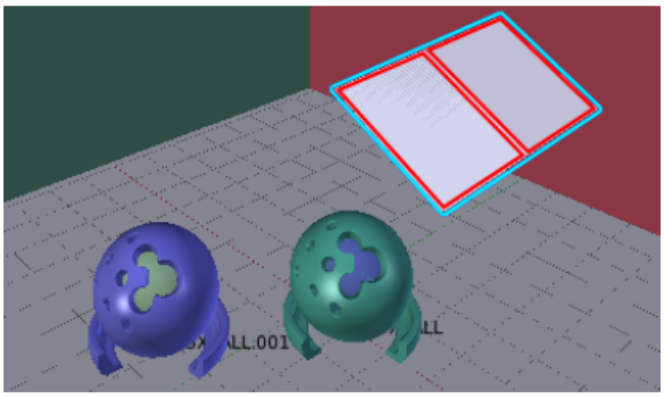

Figure 18: Example of two area light sources in red which are redundant with the larger area light source in cyan. The algorithm tries to disable unnecessary light sources and move their intensities to other light sources for a similar overall contribution. The algorithm does also work for arbitrarily placed light sources and is not limited to test scenes like the overlapping lighting setup. The image was taken from Podaras© [2014].

by the light source intensities. In each step of the used genetic algorithm a *mutation* or *crossover* is computed to search the solution space. A mutation is either to turn a light source on or off or in case of continuous search a small change of one light source intensities. A crossover is the combination of two light sources at a random index to create a new light source.

The *fitness* of the temporary solution is calculated as the *root-mean-square error (RMSE)* over all pixels between the current temporary image and one reference image but any error metric would be possible. The strength of their contribution lies in the automatic lighting design which does not require any prior knowledge about the scene or light sources. However, as any arbitrary image can be used as reference image it would be possible to allow user guidance, for example an artist drawing over the reference image to influence the result.

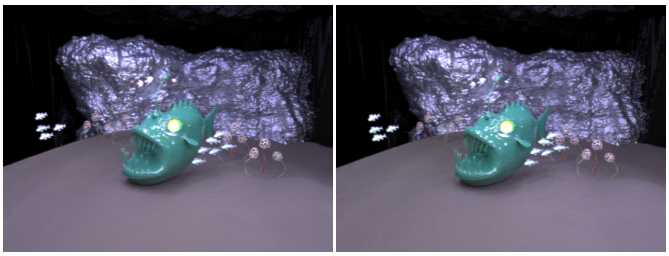

Figure 19: The left image was rendered with bidirectional path tracing using 100 light sources for one hour. The right image is the solution after 1000 iterations of their algorithm using only 58 lights. The light sources are mostly blue area lights placed around the scene. The algorithm does not only reduce singular light sources that do not contribute to the scene, but also recognizes many-for-many exchanges where a large amount of light sources can be substituted by a different, smaller subset. The image is courtesy of Podaras© [2014].

This technique requires algorithms for fast generation of high quality reference images. The evaluation of the algorithm for artistic guidance and not yet converged reference images still has to be done. In case that the reference image is not converged yet, but

a quick estimation is used as reference for the algorithm, then a robust error metric is required.

Figure 1 shows the results of stochastic progressive photon mapping after just a few seconds of rendering including caustics therefore SPPM might be a prime candidate to generate reference images which are used as input for the automatic lighting design algorithm.

10 Conclusion

The best algorithm for a scene does depend on many factors like geometry, materials and type of light paths. Caustics can be considered difficult to render where most earlier global illumination algorithms take a long time to converge or are unable to find such paths in reasonable time altogether. The high degree of parallelism on GPUs does allow very fast rendering algorithms but implementations are often more difficult because transfering information between the CPU and GPU is often the bottleneck in the rendering process.

Biased methods can result in very high quality images, though many of these methods can be improved with bias-suppression techniques. Bidirectional Lightcuts reduces the bias in lightcut-based methods and allow higher quality renderings. Many-light rendering algorithms might still have problems with caustics but with the introduction of bidirectional lightcuts subsurface scattering had become available for many-light rendering.

Many different approaches for biased light transport are currently researched. Both Dachsbacher et al. [2014, p. 14] as well as Walter et al. [2012, p. 3] suggested that the future of global illumination lies in the clever combination of existing techniques in form of hybrid algorithms. This trend has also affected available renderers in industry in form of hybrid renderers as listed by Chaos Group [2014]. The idea to combine the best of different approaches might sound obvious but it is not necessarily easy to find efficient and robust hybrid algorithms but it is likely that the future of efficient scalable light transport lies in some sort of hybrid algorithm.

11 Acknowledgments

Thanks to Wenzel Jakob and his wife who gave permission to use the torus scene.

References

CHAOS GROUP, 2014. Gi methods. `http://help.chaosgroup.com/vray/help/150SP1/gimethods.htm`. (Visited on 12/18/2014).

COOK, R. L., PORTER, T., AND CARPENTER, L. 1984. Distributed ray tracing. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '84, ACM, 137–145.

CRANE, K., 2006. Bias in rendering. http://www.cs.columbia.edu/ keenan/Projects/Other/BiasInRendering.pdf. (Visited on 11/24/2014).

DACHSBACHER, C., KIVNEK, J., HAAN, M., ARBREE, A., WALTER, B., AND NOVK, J. 2014. Scalable realistic rendering with many-light methods. *Computer Graphics Forum 33*, 1, 88–104.

DAVIDOVIČ, T., GEORGIEV, I., AND SLUSALLEK, P. 2012. Progressive lightcuts for gpu. In *ACM SIGGRAPH 2012 Talks*, ACM, New York, NY, USA, SIGGRAPH '12, ACM, 1:1–1:1.

FAN, S., CHENNEY, S., AND CHI LAI, Y. 2005. Metropolis photon sampling with optional user guidance. In *Rendering Techniques '05 (Proceedings of the 16th Eurographics Symposium on Rendering)*, Eurographics Association, Eurographics Association, 127–138.

FU, Z., AND JENSEN, H. W. 2012. Noise reduction for progressive photon mapping. In *ACM SIGGRAPH 2012 Talks*, ACM, New York, NY, USA, SIGGRAPH '12, ACM, 29:1–29:1.

GEORGIEV, I., AND SLUSALLEK, P. 2010. Simple and robust iterative importance sampling of virtual point lights. *Proceedings of Eurographics (short papers) 4*.

GÜNTHER, J., WALD, I., AND SLUSALLEK, P. 2004. Realtime caustics using distributed photon mapping. In *Proceedings of the Fifteenth Eurographics conference on Rendering Techniques*, Eurographics Association, 111–121.

HACHISUKA, T., AND JENSEN, H. W. 2009. Stochastic progressive photon mapping. In *ACM Transactions on Graphics*, vol. 28, ACM, 141.

HACHISUKA, T., AND JENSEN, H. W. 2010. Parallel progressive photon mapping on gpus. In *ACM SIGGRAPH ASIA 2010 Sketches*, ACM, 54.

HACHISUKA, T., OGAKI, S., AND JENSEN, H. W. 2008. Progressive photon mapping. In *ACM Transactions on Graphics*, vol. 27, ACM, 130.

HACHISUKA, T., 2013. Five common misconceptions about bias in light transport simulation, Nov. Accessed on 21st Nov 2014.

HAŠAN, M., PELLACINI, F., AND BALA, K. 2007. Matrix row-column sampling for the many-light problem. In *ACM Transactions on Graphics*, vol. 26, ACM, 26.

HAŠAN, M., KRIVANEK, J., WALTER, B., AND BALA, K. 2009. Virtual spherical lights for many-light rendering of glossy scenes. *ACM Transactions On Graphics, 2009, Vol.28(5)*.

JAKOB, W., 2010. Mitsuba renderer. http://www.mitsuba-renderer.org.

JAROSZ, W. 2008. *Efficient Monte Carlo Methods for Light Transport in Scattering Media*. PhD thesis, UC San Diego.

JENSEN, H. W., AND CHRISTENSEN, N. J. 1995. Photon maps in bidirectional monte carlo ray tracing of complex objects. *Computers & Graphics 19*, 2, 215–224.

JENSEN, H. W., CHRISTENSEN, P. H., KATO, T., AND SUYKENS, F. 2002. A practical guide to global illumination using ray tracing and photon mapping. In *ACM SIGGRAPH 2002 Course 43*, ACM.

JENSEN, H. W. 2001. *Realistic image synthesis using photon mapping*. AK Peters, Ltd.

KAJIYA, J. T. 1986. The rendering equation. In *ACM Siggraph Computer Graphics*, vol. 20, ACM, 143–150.

KAPLANYAN, A. S., AND DACHSBACHER, C. 2013. Adaptive progressive photon mapping. *ACM Trans. Graph. 32*, 2 (Apr.), 16:1–16:13.

KELLER, A., AND WALD, I. 2000. *Efficient importance sampling techniques for the photon map*. Fachbereich Informatik, Univ.

KELLER, A. 1997. Instant radiosity. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 49–56.

KNAUS, C., AND ZWICKER, M. 2011. Progressive photon mapping: A probabilistic approach. *ACM Transactions on Graphics 30*, 3, 25.

LAFORTUNE, E. P., AND WILLEMS, Y. D. 1993. Bi-directional path tracing. In *Proceedings of Computer Graphics*, vol. 93, Katholieke Universiteit Leuven, 145–153.

LARSEN, B. D., AND CHRISTENSEN, N. J. 2004. Simulating photon mapping for real-time applications. *Eurographics Symposium on Rendering*.

LIU, X.-D., AND ZHENG, C.-W. 2014. Adaptive importance photon shooting technique. *Computers & Graphics 38*, 0, 158 – 166.

MCGUIRE, M., AND LUEBKE, D. 2009. Hardware-accelerated global illumination by image space photon mapping. In *Proceedings of the Conference on High Performance Graphics 2009*, ACM, New York, NY, USA, HPG '09, ACM, 77–89.

MYSZKOWSKI, K. 1997. Lighting reconstruction using fast and adaptive density estimation techniques. In *Rendering Techniques 97*, J. Dorsey and P. Slusallek, Eds., Eurographics. Springer Vienna, 251–262.

OU, J., AND PELLACINI, F. 2011. Lightslice: matrix slice sampling for the many-lights problem. *ACM Transactions on Graphics 30*, 6, 179.

PEDERSEN, S. A., 2013. Progressive photon mapping on gpus.

PODARAS, S., 2014. Automated lighting design for photorealistic rendering.

PROCHE, B. 2000. *Rendering techniques 2000; proceedings of the Eurographics Workshop in Brno, Czech Republic, June 26-28, 2000*. Eurographics. Springer.

PURCELL, T. J., DONNER, C., CAMMARANO, M., JENSEN, H. W., AND HANRAHAN, P. 2003. Photon mapping on programmable graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, Eurographics Association, 41–50.

SEGOVIA, B., IEHL, J. C., MITANCHEY, R., AND PÉROCHE, B. 2006. Bidirectional instant radiosity. In *Rendering Techniques*, Lyon Research Center for Images and Intelligent Information Systems, 389–397.

SEGOVIA, B., IEHL, J. C., AND PÉROCHE, B. 2007. Metropolis instant radiosity. In *Computer Graphics Forum*, vol. 26, Wiley Online Library, 425–434.

VEACH, E., AND GUIBAS, L. J. 1997. Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 65–76.

WALTER, B., FERNANDEZ, S., ARBREE, A., BALA, K., DONIKIAN, M., AND GREENBERG, D. P. 2005. Lightcuts: A scalable approach to illumination. *ACM Trans. Graph. 24*, 3 (July), 1098–1107.

WALTER, B., ARBREE, A., BALA, K., AND GREENBERG, D. P. 2006. Multidimensional lightcuts. *ACM Trans. Graph. 25*, 3 (July), 1081–1088.

WALTER, B., KHUNGURN, P., AND BALA, K. 2012. Bidirectional lightcuts. *ACM Trans. Graph. 31*, 4 (July), 59:1–59:11.

WANG, R., HUO, Y., YUAN, Y., ZHOU, K., HUA, W., AND BAO, H. 2013. Gpu-based out-of-core many-lights rendering. *ACM Transactions On Graphics, 2013, Vol.32(6)*.

WARD, G. J., RUBINSTEIN, F. M., AND CLEAR, R. D. 1988. A ray tracing solution for diffuse interreflection. *ACM SIGGRAPH Computer Graphics 22*, 4, 85–92.

WARD, G. J. 1994. The radiance lighting simulation and rendering system. In *Proceedings of '94 SIGGRAPH conference*, SIGGRAPH '94, ACM.

WEISS, M., AND GROSCH, T. 2012. Stochastic progressive photon mapping for dynamic scenes. In *Computer Graphics Forum*, vol. 31, Wiley Online Library, 719–726.

WONG, W. Disney supercomputer renders big hero 6. `http://electronicdesign.com/blog/disney-supercomputer-renders-big-hero-6`. (Visited on 01/12/2015).