Efficient Approximations for Global Illumination

by

Romain Pacanowski
romain.pacanowski@etu.u-bordeaux1.fr

under the supervision of

Xavier Granier
granier@labri.fr

January 2005

Abstract

"An Approximate Global Illumination System for Computer Generated Films" presents for the first time a complete rendering framework with Global Illumination capabilities used to produce Shrek 2. Although this article does not introduce significant innovations for the research area, the introduced framework allows artists to easily incorporate Global Illumination effects and quickly gives them visual feedback. To achieve this amazing feat, Dreamworks and PDI have modified existing acceleration techniques and made geometrical and physical simplifications. This document reviews them and presents alternative or complementary published techniques that may also be used as efficient simplifications.

1 Introduction

Around us, the world contains many light phenomena and computers are used to simulate them. For the past twenty years, Global Illumination has been a research area for computer graphics and its goal is to simulate all light phenomena in order to compute realistic images. Realistic images are synthetic images indisguinshtable from real world images such as photographs. The input for a Global Illumination algorithm is a description of the 3D world, called scene, of the viewer observing the scene, of the material of all elements in the scene and of the light sources.

Without Global Illumination technique, illumination looks too flat or too synthetic. In some cases such as illustrated by figure 1, the scene cannot be rendered without Global Illumination technique.

Since light and material interactions are physical complex phenomena, Global Illumination techniques are complex to implement and involve a rather slow process. This is why, for a long time, production renderers have preferred to ignore it or fake it. In a production context, art direction cannot wait for hours (or even days) for small or invisible changes in an image that will be seen during 5 seconds of the whole movie.

Before presenting the Global Illumination that has been added in PDI/DW renderer as well as the optimizations and simplifications that were made, we will briefly present some fundamentals of Global Illumination.

2 Preliminary : Fundamentals of Global Illumination

The final goal of Global Illumination algorithms is to compute images that accurately reproduce the appearance of objects. Since the object's appearance is strongly correlated to its material (see 2.2) and the light present in the scene, physical quantities (see 2.1) that describe and modelise light's behavior need to be introduced.

Note that these quantities are also necessary to formulate the main problem that Global Illumination must solve as we will see it in the section 2.3.

2.1 Radiometry

Radiometry is the area of study involved in the physical measurement of light. The most important radiometric quantities are:

1. Radiant Power or Flux Φ, expressed in Watt (W), represents how much energy flows from/through a surface per unit time.

2. Irradiance E, expressed in Watt per square meter (Wm^{-2}), represents the energy of light arriving on a surface, per unit surface area.

3. Radiosity B, expressed in Watt per square meter (Wm^{-2}), represents the energy of light leaving a surface per unit surface area.

4. Radiance L is the flux per unit projected area per unit solid angle ($Wsr^{-1}m^{-2}$), it represents how much power arrives/leaves from a point on a surface per unit projected area and per unit solid angle (see figure 2).

Note that radiance is a five dimensional quantity varying with position and direction. We use the same notation as [12] to differentiate incident from exitant radiance.

1. $L(x \rightarrow \Theta)$ represents the radiance leaving point x in direction Θ.

2. $L(x \leftarrow \Theta)$ represents the radiance arriving at point x from direction Θ.

Radiance is the most important quantity for computer graphics because radiance measures/captures exactly the appearance (the color) of the object. Radiance is what our eyes, sensors or cameras perceive and it is invariant along straight paths. Therefore, Global Illumination algorithms' goal is to compute radiance for every pixel in the image.

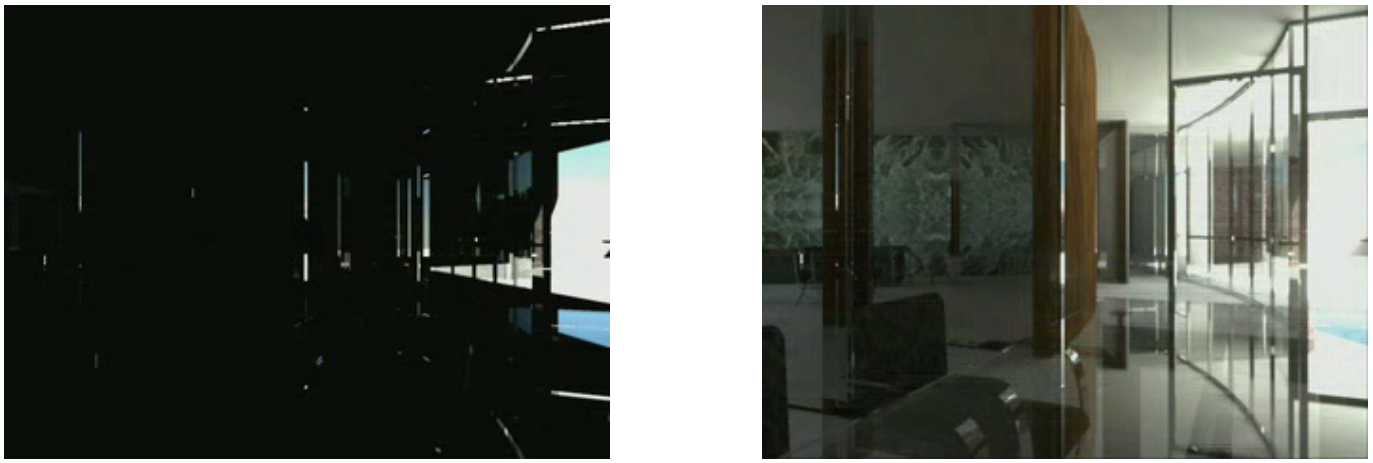

Figure 1: *A complex scene rendered with a classical ray tracer (left picture) and rendered with Global Illumination technique (right picture).*

The following relationships between the previous four quantities can be derived:

$$\Phi = \int_A \int_\Omega L(x \to \Theta)\, \cos\Theta\, d\omega_\Theta\, dA_x \quad (1)$$

$$E(x) = \int_\Omega L(x \leftarrow \Theta)\, \cos\Theta\, d\omega_\Theta \quad (2)$$

$$B(x) = \int_\Omega L(x \to \Theta)\, \cos\Theta\, d\omega_\Theta. \quad (3)$$

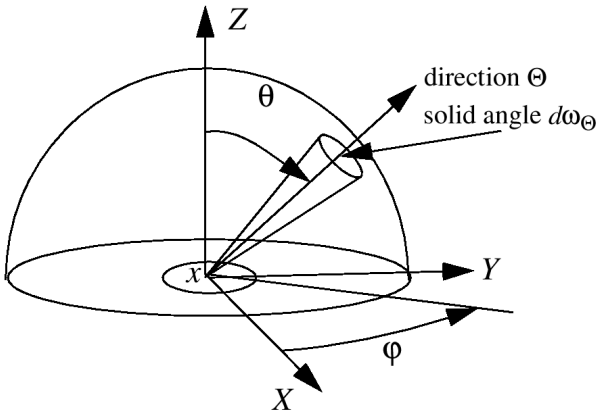

Figure 2: *Illustration of the solid angle and differiential solid angle.*

2.2 Light interaction with materials

In general, light enters a material at some point P_1 with a given direction Θ_1 at some time t_1 with a wavelength λ_1 and exit from it at a point P_2 with a direction Θ_2 at time t_2 with a wavelength λ_2.

Most Global Illumination algorithms suppose that t_1 equals t_2, thus ignoring phosphorescence effects, and also suppose that λ_1 equals λ_2, thus ignoring fluorescence effects.

Based on these assumptions the Bidirectional Scattering Surface Reflection Distribution Function (BSSRDF see [20]) modelises the behaviour of light arriving at some point P with a given direction Θ. Unfortunately, this is an eight-dimensional function that is too costly to be used. Another assumption commonly made is to consider that light arrives in and leaves the surface from the same point ($P_1 = P_2$). This simplifies the BSSRDF in a so called Bidirectional Reflection Surface Distribution (BRDF see [12]) defined over the entire sphere.

The BRDF (see figures 3 and 4) is a positive value, a four-dimensional reciprocal quantity defined at each point on a surface as:

$$\textbf{BRDF: } f_r(x, \Theta_i \to \Theta_r) = \frac{dL(x \to \Theta_r)}{dE(x \leftarrow \Theta_i)} = \frac{dL(x \to \Theta_r)}{L(x \leftarrow \Theta_i)\cos\theta_i d\omega_{\Theta_i}}$$

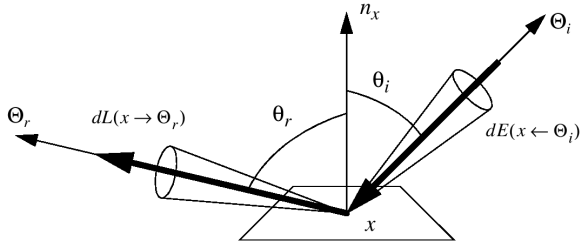

Figure 3: *Illustration of the BRDF quantity.*

$$f_r(x, \Psi \leftrightarrow \Theta) = \frac{dL(x \to \Theta)}{dE(x \leftarrow \Psi)} = \frac{dL(x \to \Theta)}{L(x \to \Psi)\, \cos(N_x, \Psi)\, d\omega_\Psi} \quad (4)$$

where

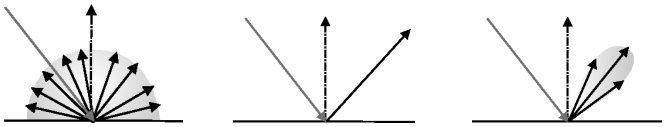

Figure 4: *Illustration of different types of BRDFs. From left to right : uniform diffuse also called Lambertian BRDF; pure specular (mirror) BRDF and glossy BRDF.*

1. $d\omega_\Psi$ is the differential solid angle centered around Ψ direction
2. N_x is the surface's normal at point x .

BRDF should be energy conservative. This means that no energy can be created when light interacts with material. The total amount of light leaving at some surface's point must be less or equal to the total amount of light arriving in the same point. Note that for computation reasons the BRDF can be split into two parts resulting in the following equation:

$$f_r(x, \Psi \leftrightarrow \Theta) = f_{r,S}(\Psi \leftrightarrow \Theta) + f_{r,D}(\Psi \leftrightarrow \Theta)$$

where

1. $f_{r,S}$ represents the specular/glossy term (not necessarily a perfect specular reflection)
2. $f_{r,D}$ represents the diffuse term (not a necessarily Lambertian reflection) .

One important consequence is that knowing the irradiance at a point x, we may compute the radiance with the BRDF:

$$L(x \rightarrow \Theta) = \int_{\Omega_x} f_r(x, \Psi \leftrightarrow \Theta)\, dE(x \leftarrow \Psi). \quad (5)$$

2.3 The Rendering Equation [21]

The Rendering Equation is a precise and elegant formulation of the Global Illumination problem. It has been introduced by Kajiya [21] who also introduced, in the same article, a method to solve it (i.e., to render an image with a full global illumination solution).

The hemispherical formulation of the rendering equation will be presented below. Other formulations may be found in [12].

Energy law conservation gives us that:

$$L_(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + L_r(x \rightarrow \Theta) \quad (6)$$

where

- $L_e(x \rightarrow \Theta)$ is the emitted radiance by the surface at x and in the outgoing direction Θ
- $L_r(x \rightarrow \Theta)$ is the reflected radiance by the surface at x in direction Θ .

Although it appears very simple, this equation is in fact highly complex because after using the fact that radiance is invariant along a straight line, and using equation (4), we obtain:

$$L_(x \rightarrow \Theta) = L_e(x \rightarrow \Theta) + \int_{\Omega_x} L(x \leftarrow \Psi)\, f_r(x, \Psi \rightarrow \Theta)\, \cos(N_x, \Psi)\, d\omega_\Psi. \quad (7)$$

This last equation is called the Rendering Equation. Radiance appears both on the left side and on the right side making analytical resolution impossible in practice. The equation is highly recursive and the dimension of the Global Illumination problem is infinite!

The main problem in this equation is to evaluate $L(x \leftarrow \Psi)$ (and thus the irradiance) which is unknown. Since radiance is invariant along straight paths we can rewrite $L(x \leftarrow \Psi)$ as:

$$L(x \leftarrow \Psi) = L(r(x, \Psi) \rightarrow -\Psi)$$

where $r(x, \Psi)$ is the closest intersection from a ray leaving x in Ψ direction. At this point, we once again have to evaluate the radiance and thus we have a recursive procedure to evaluate $L(x \leftarrow \Psi)$. The stopping condition occurs when the ray hits a light source. The method used to evaluate $L(x \leftarrow \Psi)$ is called light-gathering algorithm.

2.4 Regular Expressions for Light Paths

A common way to describe the different light paths has been introduced by Heckbert [17]. By using Regular Expressions, all light paths may be summed up in the following regular expression:

$$L(S|D)^*E \quad (8)$$

where

- L represents a Light source
- E represents the eye of the observer
- S represents a specular reflection
- D represents a diffuse reflection
- $(k)^*$ means zero or more k events
- $(r|p)$ means an r or p event
- $(k)^+$ one or more k events.

Suykens [22] extends this notation by introducing a G term for glossy reflections illustrated by figure 11. Equation (8) becomes:

$$L(S|D|G)^*E. \tag{9}$$

Therefore, a full Global Illumination method must handle all light paths and some of them are illustrated by figures 5, 6 and 7.

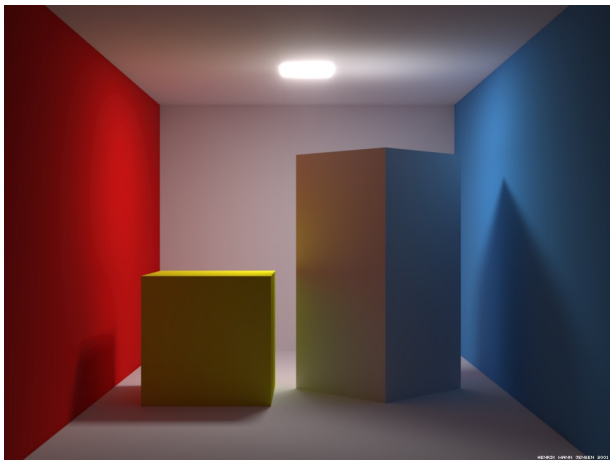

Figure 5: *Illustration of diffuse interreflections represented by $L(D^+)E$ paths. Note the blue appearance of the right box's right face due to the reflection of light on the blue wall.*

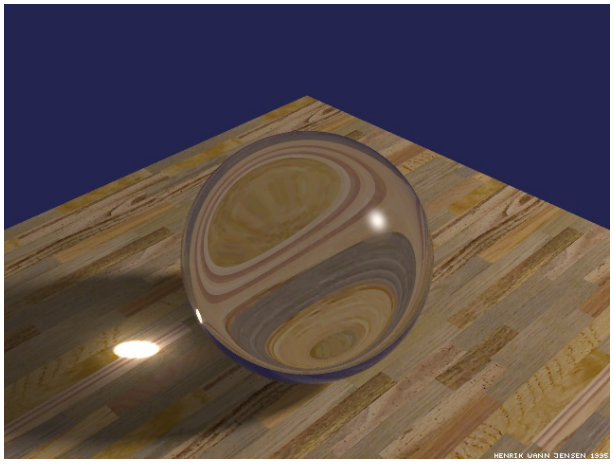

Figure 6: *Illustratation the direct caustic phenomena represented by $L(S^+)DE$ paths. The refractive sphere focuses light on specific area on the table.*

2.5 Classical Ray Tracing

Global Illumination methods are subdivided into two classes: determinist methods and stochastic methods. Stochastic methods extend the classical ray tracing algorithm while determinist methods may use it in the final rendering step.

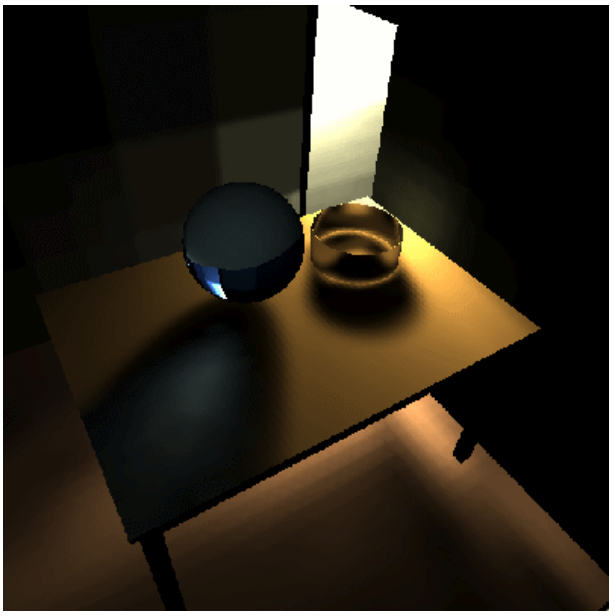

Figure 7: *Illustration the indirect direct caustic phenomena represented by $L(S^+)(D^+)DE$ paths. Light coming from another room is concentrated through a blue refractive sphere which produces a blue caustic on the table.*

Classical ray tracing has been introduced by Whitted [41] as backward ray tracing. It is a simple recursive procedure that traces rays backward from eye to light sources as illustrated by figure 8. For each pixel, a ray is launched through it and tested for intersection against the objects in the scene. If an intersection is found, local illumination is computed at the intersection point. The stop condition of the recursion is fixed by the user (bias is therefore introduced) or it is stopped when a ray hits a diffuse surface or even if a ray does not hit anything. In this last case, a background color is assigned to the pixel. If recursion is limited to 1 then the ray tracing algorithm is called ray casting and performs only visibility. Classical ray tracing allows to simulate phenomena such as perfect mirror reflection or refraction. Therefore it simulates only $LD^?S^*E$ light paths. Note that computing ray-object intersections is the most costly part of the ray tracing algorithm even if intersection acceleration techniques exist (more details in [13] and in [33]).

Distributed Ray Tracing is a direct extension to the classical ray tracing algorithm made by Cook *et al.* [8] to simulate soft glossy reflections (see figure 11) and shadows. Soft shadows are related to the presence of a penumbra region around the umbra region. Figure 10 illustrates soft shadows while figure 9 illus-

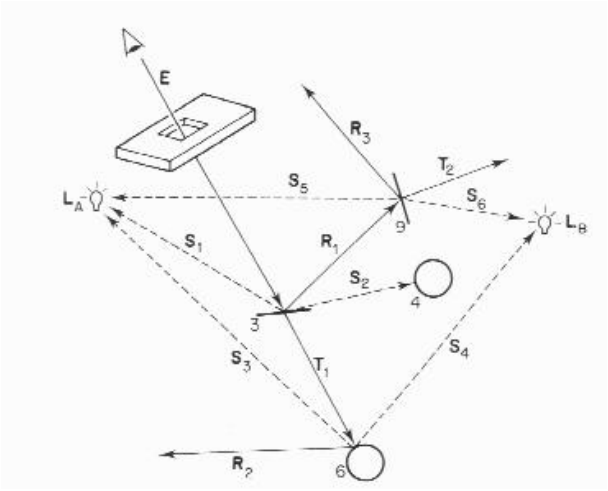

Figure 8: *Classical ray tracing algorithm. S denotes a shadow ray, R a reflected ray , T a transmitted (refracted) ray, E the eye position and L the light position.*

trates hard shadows.

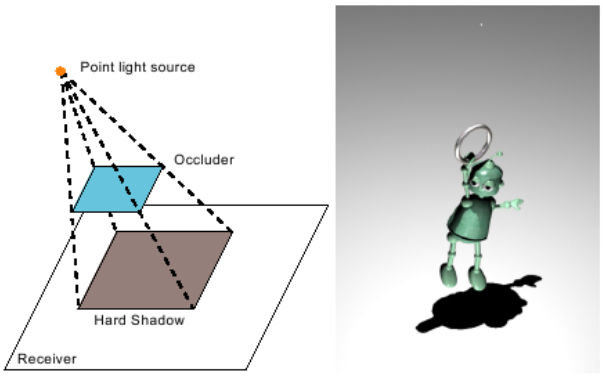

Figure 9: *Taken from [16]. Hard shadows.*

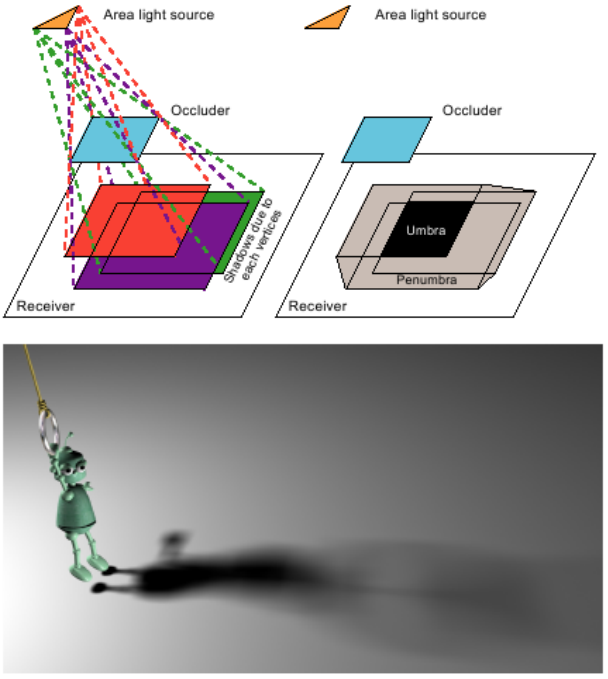

Figure 10: *Taken from [16]. Soft shadows.*

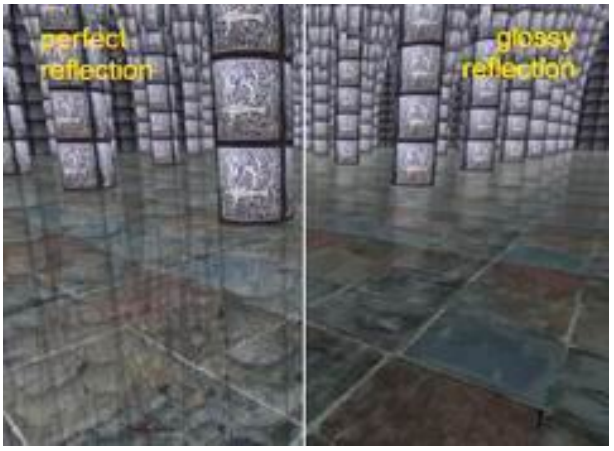

Figure 11: *Scene rendered on the left side with perfect (mirror) reflection and on the right side with glossy reflection.*

3 Simplified Lighting Model and Optimisation Techniques

3.1 Accelerating Radiance Computation

As explained in the previous section, radiance computation is recursive and thus very time consuming. It seems logical to try to accelerate the radiance computation and more precisely the light-gathering part. We will first briefly review some popular accelerating methods and then introduce Tabellion et al [37].

3.1.1 Previous Approaches

Popular methods that are "Map based" try to accelerate the light gathering algorithm by including a first pass in the Global Illumination algorithm where light is deposed on a map. This map is then used to evaluate the term $L(x \leftarrow \Psi)$.

Illumination Map [3] Illumination Map introduced by Arvo [3] is a technique that stores illumination value in a texture map. It is the first method which was able to render caustics.

During a first pass, photons emanating from lights

are deposed on the texture map. After the first pass, the texture map contains the irradiance of the model. During the second pass (rendering pass), the illumination map is used to estimate irradiance.

More precisely, each photon is not stored in the illumination map but its power is accumulated for some local region. Knowing the area of the region gives immediatly an estimate of the photons' density and then of the irradiance. This approach is a histogram approach.

The main problem with those maps is that they are correlated to the geometry of the surface they are associated with. Therefore they can be very difficult to use even if a parametrization exists. In addition it is very difficult to choose the resolution of the map and furthermore the incoming direction of the photons is not stored. As a result illumination maps are restricted to lambertian surfaces.

Photon Map [20] Photon Map is an extension of illumination maps in 3D. Photon maps are not geometry correlated and consequently can be used with very complex models. Photon Map is the main structure used in the photon mapping algorithm that is a stochastic Global Illumination method. The first pass of the algorithm builds the photon map and the second pass uses a path-tracing algorithm combined with the photon map to render the scene. In the last development two maps are built: one for caustics and another for indirect illumination.

The photon map data structure is 3D kd-tree which is balanced for efficiency reasons. The kd-tree allows to quickly find, for a given photon, its neighbours, in 3D space. During the first pass, photons from light sources are emitted. A photon touching an object can be reflected, transmitted or absorbed according to the material properties of the hit surface. Only photons that hit diffuse surfaces will be stored (in fact their power and their incoming direction are stored). Note that unlike Illumination Map the Photon Map uses a kernel density approach (more details in Bruce Walter's PhD thesis [38]) that operates directly on elements, to estimate the photons density. This approach allows filtering the estimate with a local kernel operator (cone or gaussian kernel filters for example).

Finally, the classical ray tracing algorithm is used for specular surfaces. Optimizations such as Russian Roulette or Projection Maps are used to quickly compute the Photon Maps and to avoid the introduction of a bias.

Unfortunately the main bottleneck of photon mapping is still the light gathering part of it. When Radiance needs to be computed for a given position x the procedure used can be seen as an expanding sphere around x that grows until N photons are found or until a maximal radius R is reached (R and N are user defined parameters).

Radiosity Textures [17] Radiosity Textures have been introduced and used by Heckbert to include diffuse interreflections in the classical ray-tracing algorithm. It is a hybrid approach that combines radiosity and classical ray tracing in a three pass method:

1. size pass records screen size information for each Radiosity Texture.

2. light pass progressively traces rays from light sources to depose photons on diffuse surfaces and to construct the Radiosity textures.

3. eye pass is an adaptive supersampling ray tracing pass that computes the image using the textures (see below).

The size pass should be used to compute the resolution of the Radiosity Textures called Adaptive radiosity Textures or Rexes. Their resolution is computed according to their screen size. Rexes are implemented as quadtrees and store for diffuse surfaces only the number of photons and their power.

Since the photons are stored in a texture a surface parametrization is needed to know where to store the photon in the associated Rex. Thus the Rex is a density estimator. In Heckbert's article a histogram approach is used to estimate the density instead of a kernel approach which is better.

Furthermore, the light pass uses adaptive sampling to concentrate rays on visible portions of the scenes and radiosity is also adaptively resolved according to each surface's projection. Although radiosity is a view-independent quantity, the radiosity textures computed in this method are view-dependent.

Finally, in Heckbert's article the size pass has not been implemented and the rendering method is limited to LS^*DS^*E paths and therefore does not include all possible light paths.

3.1.2 Tabellion et al. [37] Radiosity Map

Tabellion et al. make a clear choice not to completely solve the Rendering Equation. Their Renderer concentrates its efforts on the most important parts of the scene which will be seen in the image.

Their strategy to accelerate radiance computation is mainly based on Adaptive Radiosity Textures [17]

where they add some simplifications. Those simplifications are needed to improve the user workflow because using Radiosity Textures is still a time-consuming method.

Firstly, they limit the recursion of indirect illumination to one. For each surface they map a texture image, called Radiosity Map, in which each texel (a texture element) contains a radiance value. This value is computed during a preprocessing step by directly calling the shader that computes the local illumination from the texel associated point in 3D space. This is only possible because the recursion of indirect illumination has been limited to one. More precisely, texel's value is computed by inverting its four corners resulting in a quadrilateral area on the surface of which shading, texturing anti-aliasing is performed.

Like in [17], the resolution of the texture is computed by default according to the surface's size in screen space. This means that far points will have a low resolution (and thus automatically noise filtering) texture. However, with extreme camera motions, this method fails to provide a good resolution.

3.2 Approximate Lighting Model

3.2.1 Dreamworks lighting model

Tabellion et al. [37] limit their renderer to take only into account diffuse intereflections for indirect illumination. Obviously caustics or glossy reflections cannot be rendered and are left to other specific methods.

With this approximation, when computing the radiance estimate for indirect illumination, the shader uses only the diffuse component ($f_{r,D}$) of the BRDF. Of course, this is too limiting for perfect specular surfaces directly seen by the camera. Therefore the indirect diffuse illumination interacts with an arbitrary BRDF during the final shading pass (eye pass).

The rendering equation used is then:

$$L(x \rightarrow \Theta) = L_e + L_{direct\,ill.} + \sum_{\lambda=R,G,B} f_r(x, \Theta'_\lambda \rightarrow \Theta) L_\lambda(x \leftarrow \Theta'_\lambda) \, |\overrightarrow{n} \cdot \Theta'_\lambda|.$$

The approximation made, appears more clearly the way the incident radiance $L_\lambda(x \leftarrow \Theta')$ is computed:

$$L_\lambda(x \leftarrow \Theta') = \frac{E(x, \overrightarrow{n})}{|\overrightarrow{n} \cdot \Theta'|} \tag{10}$$

where $E(x, \overrightarrow{n})$ is the extrapolated irradiance computed with irradiance caching scheme (see 3.3.2) and Θ' is the incoming dominant light direction derived using weights ω_i computed during the irradiance caching construction.

3.2.2 Average incoming light direction using Light Vectors

The approach used by Tabellion is very close to the approach used with Light Vectors that have been introduced by Zaninetti et al. [42] and improved in [31].

The Light Vector is derived from Arvo's Irradiance Jacobian [4]. It is a data structure that may be seen as a virtual point light source simulating the incident energy at point x.

Like in Photon Mapping, the computation of L_r (cf equation 6) is split into separate components:

1. direct component
2. specular component
3. caustic component
4. indirect component.

For each part of the illumination described above, a separate Light Vector is used and Light Vectors are stored, like in Photon Mapping, in a separate kd-tree allowing fast neighbourhoods' search.

Our main interest relies on the Indirect Light Vector (ILV) that approximates the indirect lighting. An ILV at point x contains:

1. n components according to the color model used (n=3 for RGB model)
2. $\overrightarrow{D}$ normalized average direction of incident light
3. P irradiance value
4. quantities used to interpolate the ILV:
   (a) The gradient vector
   (b) The tangent coordinate system to the surface at point x
   (c) Arithmetic average distance to the other objects in the scene.

$\overrightarrow{D}$ and P are gathered in the following equation:

$$L_{indirect}(x, \overrightarrow{\omega}_r) = P \, f_{rd}(x, \overrightarrow{D} \rightarrow \overrightarrow{\omega}_r). \tag{11}$$

$L_{indirect}(x, \overrightarrow{\omega}_r)$ is computed with a path tracing algorithm by sampling the hemisphere centered on x. As previously stated this is very time-consuming and Zaninetti et al. introduced a bias by limiting the recursiveness of the path-tracing algorithm while saving computational time. While sampling the hemisphere they also compute for a small overhead $\overrightarrow{D}$ value. The ILV is then fully defined.

Furthermore, equation (11) is more robust than Tabellion's equation (10).

3.3 Irradiance Caching improvements

Irradiance Caching is a well known method introduced by Ward [40] to accelerate the computation of indirection illumination. It is also used in the Photon Mapping algorithm. Irradiance computation is time consuming because it is computed by uniformly sampling the incident radiance over the hemisphere requiring many ray tracing computations.

A good observation made by Ward is that illumination varies slowly on a continuous region. Penumbra and Umbra regions or caustics introduce discontinuities but if we limit ourselves to lambertian surfaces, we avoid discontinuity events. Therefore irradiance caching only works with Lambertian surfaces. Ward's main idea is to compute irradiance only at some locations in the scene and to interpolate it for the remaining locations.

Formally, the Irradiance E at a given point x with a given normal $\overrightarrow{n}$ is approximated by the following equation:

$$E(x, \overrightarrow{n}) \approx \frac{\sum_i \omega_i E_i(x_i)}{\sum_i \omega_i} \tag{12}$$

where ω_i are weights. The way the weights are computed depends on the error function (see below).

3.3.1 Classical Irradiance Caching

The decision to interpolate or compute irradiance at a location is based on the weights value.

In Ward's method weights are computed according to the following equation:

$$\omega_i(x, \overrightarrow{n}) = \frac{1}{\epsilon_i(x, \overrightarrow{n})}$$

where $\epsilon_i(x, \overrightarrow{n})$ is the error function.
Ward's error function depends on:

1. R_0 harmonic mean distance to the object seen from x
2. $\|x - x_i\|$, euclidian distance between already computed irradiance sample and x
3. Normals' variation between x and x_i.

Ward also suggests that a weight is used only if $\omega_i > \frac{1}{a}$ where a is a user defined parameter. The higher the weight the better the estimate. Therefore, equation 12 becomes:

$$E(x, \overrightarrow{n}) \approx \frac{\sum_{i,\omega_i > \frac{1}{a}} \omega_i E_i(x_i)}{\sum_{i,\omega_i > \frac{1}{a}} \omega_i(x_i, \overrightarrow{n})}. \tag{13}$$

If there are not enough samples (i.e., all ω_i are $< \frac{1}{a}$) then a new irradiance value is computed.

Computing weights can be very long and some optimisation techniques have been elaborated in order to make the irradiance cache algorithm more practical.

Irradiance gradients have been introduced later by Ward and Heckbert [39] to improve the rendering quality. One gradient vector modelizes the orientation rate of change and another modelizes the position rate of change. Using those gradients improves the estimated irradiance and smoother images are rendered.

3.3.2 Irradiance Caching Coherence (ICC)

Dreamworks Renderer modifies the classical error function to take into account the visual importance of the point x for which irradiance is needed. The visual importance term introduced is related to the projected pixel area by using the solid angle of the pixel through which x is seen.

Their error function also includes a unique user parameter K that controls the accuracy of the algorithm.

Every irradiance value is stored in a cache with corresponding geometric information and irradiance gradient vectors.

The main advantage of their new error function is that sampling density is automatically adjusted. This avoids over sampling the irradiance in corner areas. Furthermore their method avoids sampling patterns that are too dense or too sparse as illustrated by figure 12.

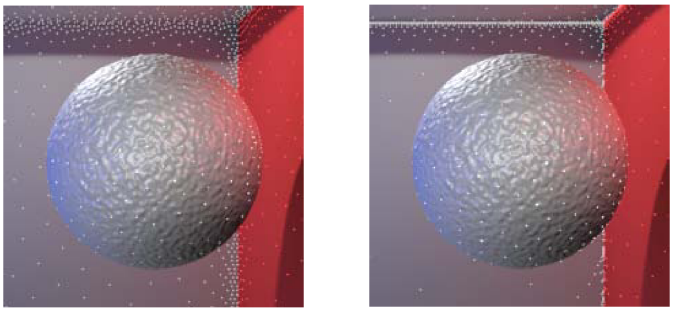

Figure 12: *Taken from [37]. Left: Irradiance sampling frequency with PDI error function. Right: Irradiance sampling frequency with Ward error function.*

Finally, weights are computed using the following equation:

$$\omega_i(x, \overrightarrow{n}) = 1 - \epsilon_i(x, \overrightarrow{n})$$

and the Irradiance equation used only takes into account samples with a positive weight. Also notice

that dominant incoming light direction is computed, weighted and stored in the cache when looping over irradiance samples whose weight is positive. This procedure is very similar to the one used for the computation of $\overrightarrow{D}$ in Light Vector's technique.

3.3.3 Another Irradiance Caching Scheme used by Light Vectors

The Irradiance Caching Scheme used by Zaninetti [42] is different to the previous one. Zaninetti et al. use a caching scheme to accelerate ILV computations.

Because the mapping used to sample the hemisphere is different (not stratified) from Ward's and Heckbert's, the irradiance gradients derived are not the same. Note that the gradient computed for each ILV is computed in the same loop as $L_{indirect}$ and therefore the extra-computation needed is negligible.

The error function used is very close to Ward's except that:

1. Two user-defined accuracy tolerance thresholds are used. One for the change of the gradient and the other for surface's curvature change.
2. the cached sample and the current point must belong to objects having the same BRDF.

Here it is not the irradiance value at point x that is directly interpolated but the Indirect Light Vector.

The main contribution using ILV instead of Irradiance Caching is not to be restricted to Lambertian surfaces because an average incoming direction is also stored in the ILV. The direction is firstly computed without weights. Weights only apply to ILVs interpolation.

4 Simplified Geometry for Global Illumination

Highly detailed geometry is required to modelize complex object's shape. Unfortunately if we increase the number of polygons/triangles to describe the model we also increase the rendering time when using ray tracing. This is why simplified geometry methods have been developed to reduce the ray-intersection computation time while preserving highly detailed geometry for the rendering. First we will briefly present Tabellion's approach based on micro-polygons and then we will present an alternative method: Points.

4.1 Tabellion's Approach

A common technique called displacement mapping allows to modelize very complex geometry.

This technique allows to add geometric detail to a surface geometry. The key idea behind displacement mapping is that surfaces in the real world are never totally flat but have some bumps.

Displacement mapping is a method to present surface details by defining an offset (displacement) from a base surface. It is different from Bump Mapping because the geometry and the normal are perturbed. The basic idea is very simple : the base surface is perturbed along its normals using displacement values specified in the displacement map.

Historically there are two families of implementation : one using scan line renderer with micropolygons and the other one using ray tracing algorithm. The necessity to use ray tracing with micropolygons is straightforward in order to use Global Illumination methods and therefore to include shadows. In fact scan line methods do not typically include shadows.

The main problem when using micro-polygons is the huge amount of geometry produced due to the small size of them. Computing the intersection between many (more than 1 million) micro-polygons and a ray could be very lengthy or even impossible due to the nature of the ray tracing algorithm that requires random access to all geometry data at the same time ! The main problem with micro-polygons and ray tracing is that most of the time all micropolygons geometry won't fit into memory or that it is too costly to allocate such a huge amount of memory.

One approach to resolve this problem is to use the fact that rays are spatially coherent for a given pixels' neighborhood. Therefore the geometry cache introduced by Pharr and Pat Hanrahan [28] enables the rendering of highly complex scenes while using a limited amount of memory.

Of course, this only solves the memory problem and does not solve the computation time required to compute ray micro-polygons intersection.

Tabellion et al. [37] used recent optimization techniques [7] that are based on a multiresolution geometry. Multiresolution geometry may be seen as a level of details where the same object has many representations. For example in [7] they compute three approximations of the same geometry: coarse, medium and fine tessellation.

Inspired by this approach Tabellion et al. use a simplified version of the micro-polyons geometry to accelerate ray tracing intersection but use the full geometry to compute the shading.

4.2 Alternative Approach using Points

Other simplified geometry techniques exist. In this section we will see how points have been a research interest area for almost 20 years and how they recently became very popular and how they can be used in Global Illumination algorithm

4.2.1 Overview of Points

Since the first pioneer report by Levoy et al. [23], where they show how to reconstruct a continous surface from sampled points and how to render it with anti-aliasing, points have renewed interest since 1998 with Grossman's article [15]. Another reason for this renewed interest is that 3D scanners have become very popular and allow to acquire very complex geometry from real world objects. The most advanced method used, presented in the last Siggraph Point course is the one developed by Matusik [25]. Finally points "renaissance" is also due to the constant growing complexity of polygons/triangles. Managing a complex growing connectivity while polygon/triangle occupies only a few pixels in screen space have brought researchers to question the utility of triangles as a fundamental display primitive.

Points are nonuniform samples of the surface. A points' cloud describes the surface geometry, including position and normals, and the appearance of the surface (reflectance property or a BRDF at each point). This means that Points discretize appearance and geometry at the same rate but points suffer from the loss of connectivity information.

As illustrated by figure 13 there are two main issues with Points:

- acquisition of points with reflectance properties and efficient storage of the acquired data.
- reconstruction and rendering surface from points.

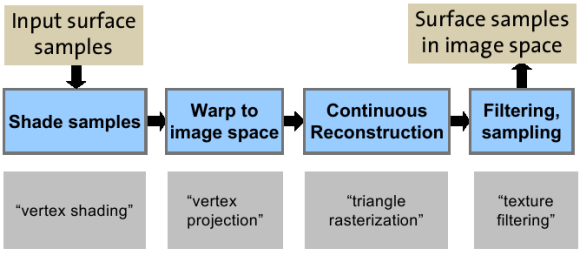

Figure 13: *Taken from Siggraph2004 Course, the rendering point pipeline and its equivalent step for the triangle rendering pipeline.*

Reconstruction problems appear because of the connectivity loss when using point sampled geometry. Good reconstruction must solve tears or hole problems as quickly as possible. There are two main approaches for reconstuction techniques: object space and screen space. We will briefly review reconstruction techniques in screen space. Reconstruction in object space are presented further.

As pointed out by Grossman et al. [15], when a point sample is projected into image space and then converted to screen space it hits a pixel. Unfortunately all pixels belonging to the same surface are not necessarily hit by the points sampled for that surface: holes may appear. Grossman et al. [15] use hierarchical Z-Buffer to detect gaps and to fill them.

Splatting technique has been introduced by Levoy et al. [23] but the main contribution has recently been from Zwicker et al. [43] after a first approach in [27] where Pfister et al. introduced the first complete point (called Surfel) rendering framework with a hierarchical representation, texture filtering, visibility calculations and image reconstruction. In order to fill holes, they use symetric radial gaussian filters centered on each hole or the supersampling procedure.

Finally, in [43], Zwicker et al. make a new step in point rendering quality with an improved splatting technique that introduces a new EWA([19] and [18]) formulation for irregular point data. Their new technique handles correctly hidden surface removal and transparency as illustrated by figure 14.

Figure 14: *Taken from [43] Complex point sampled semi-transparent object.*

Multiresolution Data Structures Once point samples have been recovered they must be organized efficiently. We will compare two approaches that were published in 2000 at Siggraph Conference.

The Layered Depth Cube Tree Pfister et al. [27] use a Layered Depth Cube (LDC) Tree to store the acquired point samples. LDC is inspired by Chang et al. [6] Layered Depth Image (LDI) Tree and Shade et al. [32] who have introduced LDI.

An LDI is simply a view of the scene from a camera but multiple pixels are stored along each line of sight. It is a bidimensionnal array composed by layered depth pixel where a layered depth pixel contains multiple (the number of layers) depth pixels. A depth pixel contains visual attributes such as the color and of course a floating value corresponding to its depth. Three orthogonal LDIs are gathered to form an LDC [24].

In [27] ray casting is used to create the three orthogonal LDIs and therefore the corresponding LDC. Contrary to Chang et al. [6] who organize LDI in a hierarchical tree in which they adaptatively select the best LDI according to the desired sampling rate, Pfister et al. [32] organize their LDIs in an octree where each node contains an LDC (called block).

QSplat [29] For the Michelangelo's project Rusinkiewicz and Levoy created a multiresolution point rendering system nammed QSplat. The data structure used to store the geometry can handle very large meshes or point clouds (such as 127 mio input points correspondig to 760MB of data and up to 8 mio rendered points). It is a bounding sphere hierarchy constructed as a preprocess and written onto disk. Each node of the tree contains the sphere center and radius and optionally a color.

Stamminger et al. [9] improve the QSplat data structure in so-called "Sequential Point Trees". Instead of a hierarchical traversal rendering algorithm. The hierarchical points representation is broken into sequential segments to accelerate the rendering that is done in the GPU. Their structure also allows smooth transition between points and triangles representation according to the distance viewer.

4.2.2 Radiosity using Points

Until recently, no article had been published where points were used with a determinist method.

Determinist methods, also called Radiosity based methods, have been introduced before the Rendering Equation by [14]. In fact Radiosity methods do not solve the complete Rendering Equation but only a part of it and therefore take only into account diffuse interreflections that correspond to LD^*E paths.

These assumptions simplify the Rendering Equation in a so called Radiosity equation:

$$B_i = B_{ei} + \rho_i \sum_j F_{ij}\, B_j. \tag{14}$$

Details on how this equation is derived may be found in Sillion et al. [34] or in Dutre et al. [12].

This equation shows that the radiosity B_i at a patch i is the sum of the self-emitted radiosity B_{ei} and the fraction of irradiance at i that gets reflected. The F_{ij} is the so-called patch-to-patch form factor and represents the fraction of the irradiance on i originates at j or the fraction of power emitted by i that lands on j. Radiosity is a determinist method because the light interactions are computed with a finite number of polygons called patches here. A classical radiosity algorithm contains the following step:

1. scene discretization in patches
2. Form factors computation
3. Numerical solve of the radiosity system defined by equation (14)
4. Display of the solution.

Note that the solution of Radiosity system is view-independent and that the solution is the equilibrum of the light exchange between surfaces. To solve the radiosity system is not very complex and iterative numerical method such as Jacobi or Gauss-Seidel converge after relatively few iterations.

Before returning to our points we should highlight the two main problems in the radiosity algorithm.

Patches should be small enough to capture illumination variation (near soft shadows regions for example) otherwise visual artefacts will appear but their number should not be too large because the memory needed to store them would dramatically increase.

Form factors also need to be stored but the main problem is to compute them. As explained in [34] simple cases allow to compute analytically the form factor between two patches. Otherwise, each form factor requires the solution of non-trivial four-dimensional integral with singularity where the distance between two patches tends to zero and the integrand may also have discontinuities due to visibility changes.

When using points we are facing a significant problem: there are no patches at all! Yamamoto et al. [11] address this problem by computing an *effective area* between two surfels defined by a point, a normal and

a radius. This effective area attempts to take into account discs overlapping that necessary to avoid holes in the sampled geometry.

Once effective areas have been computed they are organised in hierarchical structure of bounding spheres. Form factors are then computed using effective areas and the image is rendered using the Stamminger [9] method to compute diffuse interreflections.

Atlas of Discoids Basically a Surfel is a point with a radius and a normal at this point. Thus it also defines a disc. As previously stated during the acquisition step, those discs overlap each other to avoid holes. Michelin et al. [35] have developed a method to construct an implicit surface from a set of discs that overlapped. This set is called Atlas of Discoids. The basic idea is to sketch the model with a set of discs that can overlap each other. Furthermore overlaps are welcome with the discoids technique. Although the paper in which discoids are introduced does not mention the use of this technique in point sample geometry. The link and the utility of such a technique seems obvious.

The implicit surface is defined where superposing areas appear. In the simple case of planar surfaces that overlap the superposition area corresponds to their intersection. Of course this simple case is not the general one. Therefore, if Δ is the subset of discoids overlapping each other the implicit surface defined by this subset according to Michelin et al. is:

$$Z(M) = \sum_{i \,|\, D_i \in \Delta} \beta_i(M_i)\, z_i(M) = 0$$

where

- β_i is a distance function between M_i and the center of the discoid D_i. It varies continously from the center of the discoid to the border of it from 1 to 0.

- z_i algebraic distance between M and its orthogonal projection on discoid D_i.

Note that β_i functions affect the shape and the continuity of the surface in the overlapping area therefore only β_i functions have to be chosen ! With this method Michelin et al. [36] have also reformulated the Radiosity equation in a new function basis with local support that allows them to render scene with a Radiosity technique without artifacts (such as discontinuities across a surface) that appear with the classical Radiosity method.

4.2.3 Stochastic Methods using Points

The problem using points with stochastic methods can be formulated this way: how to compute the intersection between a ray and a point and how to make sure that the computed intersection is view-independent ?

[30] Jensen et al. choose not to compute a view-independent intersection. After all, global illumination algorithm are not view-independent even if ray - polygon intersection normally are !

Instead of trying to reconstruct a surface from the point sampled geometry Jensen et al. derives a view-dependent intersection procedure, based on cylinders infered by surfels, whose result gives the point intersection and its normal. This can be view as a screen space reconstruction procedure.

On the other hand [1] Adamson et al. introduce a new way to efficiently compute the intersection of a ray and point set surfaces. This is done by reconstructing the surface in object space with a method called Moving Least Squares (MLS).

MLS [10] based approach MLS is binded to a projection procedure that allows to reconstruct a smooth surface from unscattered data. This projection procedure is fully described in [2]. The result is an infinitely smooth and manifold surface controlled by a parameter h that allows to smooth out the noise. As illustrated by figure 15, for each projected point a local reference domain/plane H and a local bivariate polynomial g approximation is needed to reconstruct the surface called point set surface.

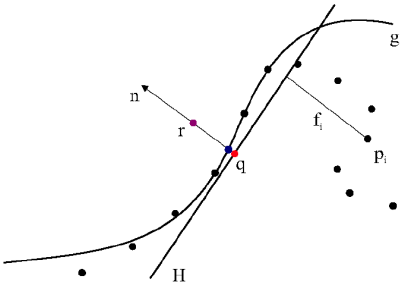

Figure 15: *Taken from [2] MLS Projection.*

It seems obvious not to use a global scale factor h with non uniform point samples. If h is too large, too much smoothing is done for a high density area. If h is too low, as described in [2], numerical instabilities may occur in a sparsely sampled region. Extensions to adaptatively compute h have been made in [5] and in [26].

Finally implicit surfaces driven by discoids are also candidates to stochastic methods since Michelin et al. [35] derive a procedure to compute the intersection between a ray and an atlas of discoids. Unfortunately it seems that no article was published in which discoids or MLS surfaces are used with a stochastic method.

5 Conclusion

In this document we have covered the optimizations and the approximations made by Tabellion et al. [37] in order to add Global Illumination capabilities to their rendering framework. We have also pointed out other published techniques that can be used to accelerate the computation of the Rendering Equation or to simplify the geometry. Light Vector may be a valid approach to take into account specular reflections with indirect illumination and Points, as geometrical simplification may also be an interessting line of research instead of micro-polygons.

References

[1] Anders Adamson and Marc Alexa. Ray tracing point set surfaces. In *SMI '03: Proceedings of the Shape Modeling International 2003*, page 272. IEEE Computer Society, 2003.

[2] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, 2003.

[3] James Arvo. Backward ray tracing. In *SIGGRAPH 86 Course Notes, Volume 12*, August 1986.

[4] James Arvo. The irradiance jacobian for partially occluded polyhedral sources. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 343–350. ACM Press, 1994.

[5] Mario Botsch, Andreas Wiratanaya, and Leif Kobbelt. Efficient high quality rendering of point sampled geometry. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 53–64. Eurographics Association, 2002.

[6] Chun-Fa Chang, Gary Bishop, and Anselmo Lastra. Ldi tree: A hierarchical representation for image-based rendering. In Alyn Rockwood, editor, *Siggraph 1999, Computer Graphics Proceedings*, pages 291–298, Los Angeles, 1999. Addison Wesley Longman.

[7] Per H. Christensen, David M. Laur, Julian Fong, Wayne L. Wooten, and Dana Batali. Ray differentials and multiresolution geometry caching for distribution ray tracing in complex scenes. In *Computer Graphics Forum (Eurographics 2003 Conference Proceedings*, pages 543–552. Blackwell Publishers, Septemner 2003.

[8] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 137–145. ACM Press, 1984.

[9] Carsten Dachsbacher, Christian Vogelgsang, and Marc Stamminger. Sequential point trees. *ACM Trans. Graph.*, 22(3):657–662, 2003.

[10] Lévin David. Mesh-independent surface interpolation. In *Geometric Modeling for Scientific Visualization*, pages 37–49. Edited by Brunnett, Hamann and Mueller, Springer-Verlag, 2003.

[11] Yoshinori Dobashi, Tsuyoshi Yamamoto, and Tomoyuki Nishita. Radiosity for point-sampled geometry. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference on (PG'04)*, pages 152–159. IEEE Computer Society, 2004.

[12] Philip Dutre, Kavita Bala, and Philippe Bekaert. *Advanced Global Illumination*. A. K. Peters, Ltd., 2002.

[13] Andrew S. Glassner. *An Introduction to Ray tracing*. Morgan Kaufmann, 1989.

[14] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the interaction of light between diffuse surfaces. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 213–222. ACM Press, 1984.

[15] J. P. Grossman and William J. Dally. Point sample rendering. pages 181–192. Springer-Verlag, August 1998.

[16] Jean-Marc Hasenfratz, Marc Lapierre, Nicolas Holzschuch, and François Sillion. A survey of real-time soft shadows algorithms. In *Eurographics*. Eurographics, Eurographics, 2003. State-of-the-Art Report.

[17] Frank Heckbert. Adaptive radiosity textures for bidirectional ray tracing. In *ACM SIGGRAPH Proceedings*, volume 24, pages 145–154, August 1990.

[18] Paul Heckbert. Survey of texture mapping. In *IEEE Computer Graphics and Applications*, November 1986.

[19] Paul Heckbert. Fundamental of texture mapping and image warping. In *Master's Thesis. University of California. Berkeley*, 1989.

[20] Henrik Wann Jensen. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., 2001.

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

[22] Frank Suykens De Laet. *On Robust Monte Carlo Algorithms for Multi-pass Global Illumination*. PhD thesis, Departement of Computer Science, Kathoelike Universiteit Leuven, September 2002.

[23] M. Levoy and T. Whitted. The use of points as a display primitive. Technical report, University of North Carolina, January 1985.

[24] Dani Lischinski and Ari Rappoport. Image-based rendering for non-diffuse synthetic scenes. In *Proceedings of the 9th Eurographics workshop on Rendering*, pages 301–314, 1998.

[25] Wojciech Matusik, Hanspeter Pfister, Addy Ngan, Paul Beardsley, Remo Ziegler, and Leonard McMillan. Image-based 3d photography using opacity hulls. *ACM Trans. Graph.*, 21(3):427–437, 2002.

[26] Mark Pauly, Markus Gross, and Leif P. Kobbelt. Efficient simplification of point-sampled surfaces. In *VIS '02: Proceedings of the conference on Visualization '02*. IEEE Computer Society, 2002.

[27] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surfels: surface elements as rendering primitives. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 335–342. ACM Press/Addison-Wesley Publishing Co., 2000.

[28] Matt Pharr and Pat Hanrahan. Geometry caching for ray-tracing displacement maps. In Xavier Pueyo and Peter Schröder, editors, *Eurographics Rendering Workshop 1996*, pages 31–40, New York City, NY, 1996. Springer Wien.

[29] Szymon Rusinkiewicz and Marc Levoy. Qsplat: a multiresolution point rendering system for large meshes. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 343–352. ACM Press/Addison-Wesley Publishing Co., 2000.

[30] Gernot Schaufler and Henrik Wann Jensen. Ray tracing point sampled geometry. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 319–328. Springer-Verlag, 2000.

[31] Xavier Serpaggi and Bernard Péroche. An adaptive method for indirect illumination using light vectors. In A. Chalmers and T.-M. Rhyne, editors, *Eurographics 2001 Proceedings*, volume 20(3), pages 278–287. Blackwell Publishing, 2001.

[32] Jonathan Shade, Steven Gortler, Li wei He, and Richard Szeliski. Layered depth images. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 231–242. ACM Press, 1998.

[33] Peter Sherley. *Realistic Ray Tracing*. AK Peters, 2003.

[34] François Xavier Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann Publishers, Inc., 1994.

[35] Michelin Sylvain, Arquès Didier, and Piranda Benoît. Modelisation of implicit surfaces driven by an atlas of discoids. In *GraphiCon'2000*, pages 256–261, August 2000.

[36] Michelin Sylvain, Arquès Didier, and Piranda Benoît. Overlapping radiosity: Using a new function base with local disk support. In *8th International Conference in Central Europe on Computer Graphics and Visualization*, volume 2, pages 236–243, February 2000.

[37] Eric Tabellion and Arnauld Lamorlette. An approximate global illumination system for computer generated films. *ACM Trans. Graph.*, 23(3):469–476, 2004.

[38] Bruce Walter. *Density Estimation Techniques for Global Illumination*. PhD thesis, Cornell University, 1998.

[39] Gregory J. Ward and Paul Heckbert. Irradiance gradients. In *Third Eurographics Workshop on Rendering*, pages 85–98, Bristol, UK, 1992.

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

[41] Turner Whitted. An improved illumination model for shaded display. In *Communications of the ACM 23(6)*, May June 1980.

[42] Jacques Zaninetti, Xavier Serpaggi, and Bernard Péroche. A vector approach for global illumination in ray tracing. In *Computer Graphics Forum*, volume 17, September 1998.

[43] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Surface splatting. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378. ACM Press, 2001.