Approximation of Global Illumination in Real-Time

Christian Freude[*]
Vienna University of Technology

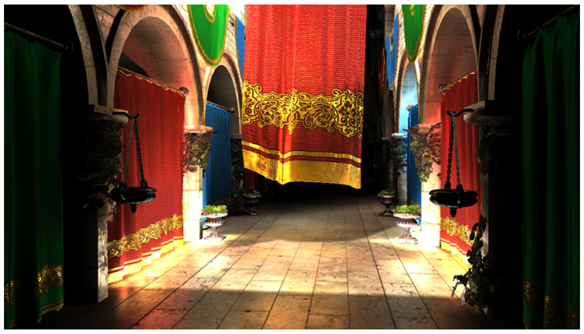

Figure 1: A scene rendered using the method described in the paper called 'Interactive Indirect Illumination Using Voxel Cone Tracing' (see Section 3.2.2). Image part taken from Crassin et al. [2011].

Abstract

In computer graphics global illumination is used to generate realistic images. Rendering an image using full global illumination is usually a time consuming task. However, in recent years several methods have been developed to approximate global illumination in real-time. In this paper a variety of such methods, which are able to approximate global illumination in dynamic scenes at real-time frame rates, are discussed. At first the motivation and a general introduction to global illumination is given. Then different methods regrading ambient occlusion and indirect illumination are explained and discussed.

CR Categories: I.3.0 [COMPUTER GRAPHICS]: General; I.3.6 [COMPUTER GRAPHICS]: Methodology and Techniques;

Keywords: real-time, global illumination

1 Introduction

Global illumination stands for algorithms and methods which calculate the light propagation in a scene [Bengtsson 2010]. Such algorithms are able to produce realistic images, which are often indistinguishable from real photographs. One very important characteristic of such images is indirect illumination. This means that not only direct light from light sources is simulated, but also indirect light. Such light bounces off surfaces which do not actively emit light.

Many global illumination methods are able to produce accurate physically based simulations of the light transport in a scene. Images rendered with such methods look very realistic. Figure 2 shows two rendered images of the same scene. The left image was rendered by only taking local illumination into account, whereas the right image was rendered using full global illumination. In the right image colour bleeding, caused by indirect light from the coloured walls, and caustics, caused by refraction in the glass sphere, can be seen.

Dutré et al. [2006] mentioned many different fields which can benefit from such realistic images. For example, architects can visualise the lighting conditions in planned buildings and interiors. Such previsualisations can use different light setups or daytimes and can be presented to clients. Additionally, film and visual effects can produce even more realistic images in order to fool the audience into thinking that the computer generated content is real.

However, physically accurate simulation of all lighting effects is not always necessary. For example, many computer games only use very simple lighting models. In fact, often only direct illumination is rendered, or low-frequency indirect illumination is precalculated. Only in recent years methods like *ambient occlusion* or more advanced indirect light approximations have been used to enhance the rendered images. Ritschel et al. [2012] stated that although indirect light is important, it only has to be plausible and not really physical accurate. So it is often sufficient to approximate only low frequency indirect light and a small number of light bounces.

This leads to the goal of this paper which is to present methods which only approximate global illumination and therefore are able to render images in real-time. Another requirement is that they are, to some extent, able to handle dynamic scenes. In the context of games, for example, this is often a desirable feature.

2 Illumination in Rendering

In this section fundamental concepts regarding global illumination, such as the rendering equation, BRDFs and the light transportation notation, are discussed. Furthermore, two common offline methods to compute global illumination are described.

[*]e-mail: e0728278@student.tuwien.ac.at

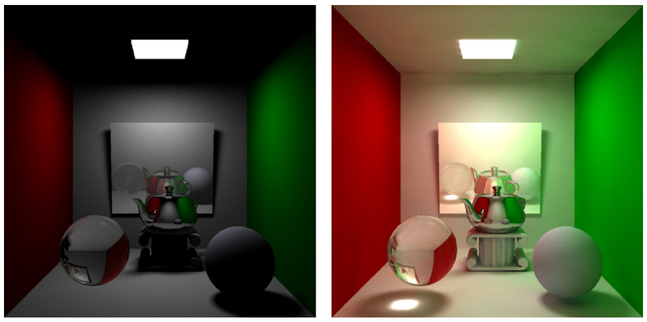

Figure 2: Comparison between direct illumination only (left) and full global illumination (right). In the left image no indirect illumination or caustics are visible. However, in the right image colour bleeding, caused by indirect light, and caustics, caused by refractions, can be seen. Images taken from Knecht [2009].

2.1 Rendering equation

To solve the problem of global illumination one has to evaluate the so called rendering equation. This task can be accomplished using a variety of methods. The basics of two, commonly used approaches, are described in Section 2.4.

The rendering equation can be seen in Equation 1 [Knecht 2009]. It describes the light which leaves a certain point in a scene in a certain direction [Bengtsson 2010].

$$L_o(p,\omega_o) = L_e(p,\omega_o) + \int_\Omega \rho(p,\omega_i,\omega_o)L_i(p,\omega_i)cos\theta d\omega_i \quad (1)$$

- $L_o(p,\omega_o)$: expresses the light that leaves point p in direction ω_o (outgoing light)
- $L_e(p,\omega_o)$: stands for the emitted light from point p in direction ω_o (emitted light)
- $\rho(p,\omega_i,\omega_o)$: describes how much of the light that arrives at point p from direction ω_i is leaving in direction ω_r (often a BRDF is used)
- $L_i(p,\omega_i)$: stands for the light which arrives at point p, coming from direction ω_i (incoming light)
- $cos\theta$: here θ is the angle between the normal of point p and ω_i (due to Lambert's cosine law)
- Ω : in case a BRDF is used, it stands for the normal aligned hemisphere over p

An illustration of the rendering equation, with slightly different notation, can be seen in Figure 3.

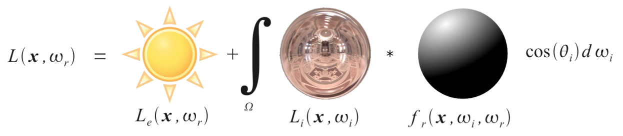
$L(\boldsymbol{x},\omega_r)$
=
+
$\int_\Omega$
*
$\cos(\theta_i)d\omega_i$
$L_e(\boldsymbol{x},\omega_r)$
$L_i(\boldsymbol{x},\omega_i)$
$f_r(\boldsymbol{x},\omega_i,\omega_r)$

Figure 3: Illustration of the rendering equation. Image taken from Bengtsson [2010].

An important fact is that the L_i term (Equation 1) of the right side is a L_o term of another point. Due to this recursive nature, the rendering equation is hard to solve [Knecht 2009].

2.2 BRDF

The term $\rho(p,\omega_i,\omega_o)$ of the rendering equation mainly determines the appearance of surfaces in the rendered image. In order to model a surface, and its interaction with light, often a so called bidirectional reflectance distribution function (BRDF) is used. It describes how the light which arrives at a surface point is scattered over its hemisphere. Some, usually less complex, BRDFs make the assumption that the distribution does not depend on the location. Such simplified BRDFs return the amount of reflected light, only in respect to an incoming and an outgoing direction. They ignore the actual position on the surface. BRDFs can be described analytically or acquired by measuring real materials. The latter produces a lot of data [Knecht 2009].

2.3 Light Transport Notation

The so-called light transport notation is used to describe the path of a light ray and the occurring light bounces during his way through a scene [Heckbert 1990]. Using the letters L, D, S and E a path is encoded in a string. Such a path starts at L, the light source, and then interacts diffuse (D) or specular (S) at some locations with the scene, until it reaches the Eye (E). All possible paths of a ray can be described using the regular expression $L(S|D)^*E$. Using such a notation it is also possible to encode the capabilities of a rendering method [Knecht 2009]. An illustration of different light paths and their notation can be seen in Figure 4.

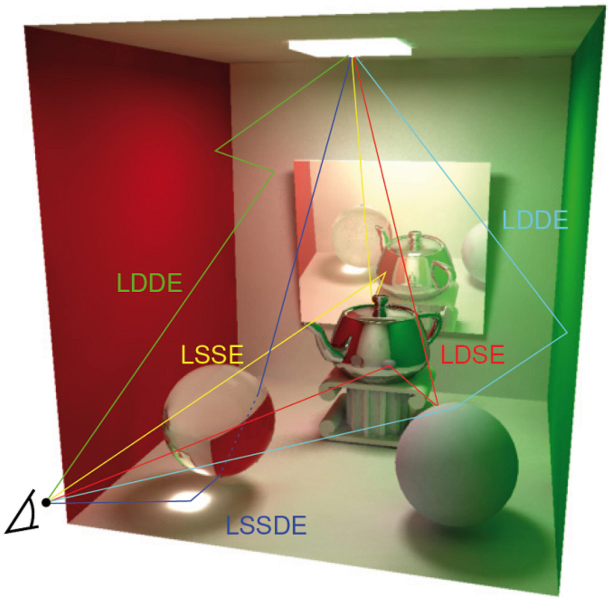

Figure 4: Illustration of the light transport notation. For example, the green and the light blue path are both labelled as LDDE, since they start at the light source and hit two diffuse surfaces until reaching the eye. Image taken from Knecht [2009].

2.4 Global illumination

There are two major approaches for calculating global illumination renderings. One is called path tracing, and the other is called radiosity. Both have different advantages and exist in many variations. Their basic principles are described in this section.

Path tracing essentially traces the path of light rays through the scene. The basic idea is that for each pixel of the virtual camera's image plane the algorithm casts multiple rays into the scene. At each intersection with the scene a ray may get reflected or refracted. Each ray is traced along one single path until it reaches a light source. Those paths which hit the light source contribute to the pixels colour value [Watt 2002]. An advantage is that advanced variations of this method are able to generate realistic global illumination renderings. The drawback is that, in general, the rendering time is high. Furthermore, the calculated global illumination is only valid for the used viewpoint.

The second major approach, called **radiosity**, basically consists of four steps. At first the scene geometry is discretized into little patches. Each patch reflects the light from other patches, and in case of a light source also emits light. Then so-called form factors are computed for all pairs of patches. They represent the geometric relation of those patches and determine how much each patch reflects the light of another patch. In order to distribute the energy (light) of the emitting patches over the entire scene, the method then calculates the so-called radiosity value for each patch. This is done by solving a system of linear equations, which include the previously calculated form factors. In the last step the values for each patch are interpolated and displayed, e.g by using a rasterizer. An advantage is that the form factors only depend on the geometry and are therefore valid for arbitrary viewpoints. Changes of light intensities only require to re-solve the equation system. Due to those properties, a walk-through of a static scene is, in respect to the global illumination solution, inexpensive. The disadvantage is that basic radiosity only computes diffuse lighting, but no specular effects [Watt 2002; Dutré et al. 2006]

3 Real-time methods

Path tracing and *radiosity* are two common approaches for offline (not real-time) rendering. Although radiosity is to some extend real-time capable, and path tracing has become much faster in recent years via GPU acceleration, both approaches in general are not able to render dynamic scenes in real-time. For real-time rendering of global illumination in dynamic scenes, various approximations and simplified models are used. In many cases only a limited number of diffuse light bounces are calculated, or occlusion in case of indirect light is ignored. Such methods, which approximate global illumination in real-time, are presented and discussed in this section.

One approach often used for real-time applications is light mapping. Precomputed illumination, using any kind of method, is stored in so-called light maps. These maps are then simply texture mapped onto the objects in the scene to give the impression of illuminated surfaces [Knecht 2009]. Due to the fact that light mapping is fairly simple and quite old, although still used nowadays, it is only mentioned and not described in detail.

In Section 3.1 methods for ambient occlusion (AO) and algorithms which are based upon this technique are discussed. Section 3.2 presents more sophisticated approaches which approximate indirect illumination.

3.1 Ambient occlusion

A first step towards global illumination and enhancing a rendered image is a technique called ambient occlusion (AO). The basic idea is to compute how much (uniformly distributed) light, from the environment, would hit a surface point. This is done by integrating the occlusion in every direction of the surrounding hemisphere, which actually solves the inverted problem [Knecht 2009]. According to this degree of occlusion a surface point is darkened. This approximates the fact that less light would hit the surface based on the amount of occlusion.

3.1.1 Temporal Screen-Space Ambient Occlusion

Mattausch et al. [2011] presented a method which improves screen space ambient occlusion (SSAO) [Mittring 2007] exploiting temporal coherence.

SSAO is an approximation in screen space and a very commonly used technique to create the effect of ambient occlusion. It has become quite popular in recent years and is used in many high-end games. The basic approach is to estimate the occlusion of every visible fragment in the current viewport using the depth values of surrounding pixels. According to this calculated occlusion the brightness of the corresponding pixel is then adjusted to create the characteristic shading effect.

In order to approximate AO using SSAO at real-time frame rates, certain simplifications are made. First of all, the geometry is represented by the depth buffer. Thus, only visible geometry is taken into account. Additionally the area surrounding the evaluated pixel is covered by only a few samples. This leads to noise and decreased quality compared to more accurate (but slower) AO techniques.

The authors proposed to reduce noise and quality issues by using temporal coherence. The basic principle is to accumulate samples over time. Using this approach it is possible to increase the overall quality of the result, while still calculating only a few samples per frame.

In order to reuse the samples from previous frames the information of the last frame is stored in a buffer. However, to retrieve the correct information from the buffer, it is necessary to know which pixel of the current frame corresponds to which pixel of the preceding frame. This problem is solved by using reverse reprojection. The basic principle is to apply the view-projecton-matrix of the preceding frame to the world position of a pixel in the current frame. However, in case of dynamic objects, the whole vertex transformation (including model matrix) would need to be considered. Due to this and the use of deferred shading, Mattausch et al. proposed to store 3D optical flow to calculate the pixel correspondences.

Due to dynamic changes in the scene, it is also necessary to know, if the AO value of a corresponding pixel can be reused or not. The solution, as presented by Mattausch et al., is to test for three cases in which the value of a re-projected pixel is considered invalid and is not reused. The first one is the case of disocclusion where seemingly corresponding pixels would actually occlude each other. This is decided by comparing the depth values. Secondly, it may be that the neighbourhood might have changed, thus making previous samples obsolete. Such a case is evaluated by using the samples calculated in the current frame to compute the relative change in position compared to the previous samples. The authors introduced a so-called smooth invalidation where the previous samples are not completely discarded, but weighted differently based on the magnitude of relative change. In the last case where a previous pixel was outside the view plane, it is considered invalid.

Special weighting is applied during combination of new and reused samples to ensure that the influence of older samples decreases over time. In case of low convergence (low number of total samples), filtering is applied to reduce noise.

The authors compared their method to normal SSAO. Their approach achieves better quality and performance compared to the standard approach and only degenerates to normal SSAO if the temporal coherence is low. It turns out that the smooth invalidation increases the visual quality, compared to basic SSAO, in the case of deforming objects. An image generated using this method can be seen in Figure 5.

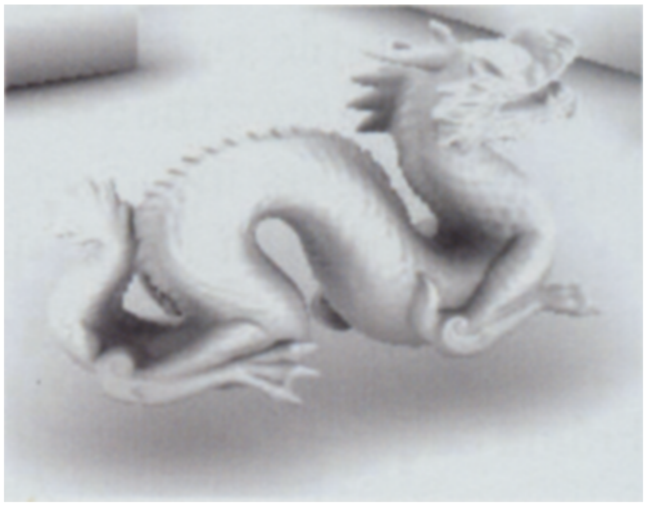

Figure 5: A rendering created using the algorithm described in the paper called 'Temporal Screen-Space Ambient Occlusion' (see Section 3.1.1). Image taken from Mattausch et al. [2011].

3.1.2 Approximating dynamic global illumination in image space

The approach presented by Ritschel et al. [2009] extends SSAO to approximate global illumination in real-time. The two main contributions of the algorithm are directional-occlusion and one indirect light bounce.

The method operates in image-space and uses positions and normals stored in a frame buffer as input. Direct illumination and indirect illumination are calculated in two separate render passes. The output is a frame buffer with illuminated pixels.

In order to calculate the direct illumination for every pixel, its normal aligned hemisphere is sampled. To calculate a sample location, a random vector is added to the 3D position fetched from the input frame buffer at the corresponding pixel location. The vector is generated by choosing a random direction of the hemisphere and scaling it by a random value between zero and a user defined radius. Each sample's 3D position is then back-projected into the image plane and compared to the 3D position of the corresponding pixel. In case the sample's position is further away than the corresponding 3D position of the pixel, it is considered as an occluder. The direct illumination is then calculated only for the sample directions of non-occluders. The authors assume that the direct illumination for a certain direction can be efficiently calculated from environment maps or point lights.

The direct illumination calculated in the preceding pass can now be used to compute the one-bounce indirect illumination. For each pixel the indirect illumination is basically the sum over incoming light reflected by a number of patches. These patches are defined by the position and the normal stored in the frame buffer at the pixel locations which correspond to samples classified as occluders. The indirect light contribution of a patch is determined by its orientation and the direct light at the position of the patch. The direct light is weighted by the geometric relation between the normal of the patch and the normal at the pixel being currently shaded. The strength of the indirect light effect can be adjusted by the user.

There are several cases where the screen-space approach leads to certain problems or drawbacks. The fact that only the visible front of the scene (geometry) is used for calculations can result in wrong occlusions. Additionally, colour bleeding from surfaces not visible or viewed at very flat angles is missed. In order to solve these limitations and capture invisible geometry, the authors proposed to use multiple depth layers and additional cameras.

An interesting fact described by the authors is, that their method can be used to enhance and support algorithms like shadow mapping and instant radiosity. In the case of shadow mapping, many algorithms need a certain geometry offset during generation of the shadow map. This offset often leads to the loss of perceptually important contact shadows. To add such missing contact shadows, a number of user defined samples (in screen space) are taken in the direction of the light source to calculate the occlusion.

The method is able to render at real-time frame rates. Without enhancements, like multiple depth layers or additional cameras, the performance of directional occlusion is nearly the same as for SSAO. Adding one bounce indirect illumination results in an increased computation time of 31.1 percent, for the test scene used by the authors. An image rendered using this method can be seen in Figure 6.

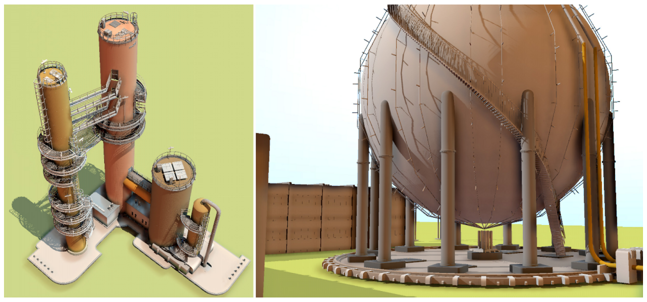

Figure 6: A scene rendered using the method described in the paper called 'Approximating dynamic global illumination in image space' (see Section 3.1.2). Image taken from Ritschel et al. [2009].

3.2 Indirect illumination

Although ambient occlusion significantly enhances a rendered image, it is a fairly big approximation. In order to get closer to the real solution and create more realistic images, indirect illumination has to be taken into account. The method described in Section 3.1.2 approximates indirect light. However, as it is closely related to SSAO it was presented in the corresponding section.

In this section various methods which approximate global illumination by calculating indirect illumination are presented. The method described in the next subsection (3.2.1) is used by some of the subsequent algorithms, and is therefore the first to be discussed.

3.2.1 Reflective shadow maps

Dachsbacher and Stamminger [2005] presented an algorithm to approximate one-bounce indirect illumination in real-time. The authors introduced so-called reflective shadow maps (RSMs), which extend classic shadow maps. In addition to depth, also world position, normal and reflected radiant flux are stored for each surface point (pixel), which is visible from the light source.

In order to calculate direct and indirect illumination, one RSM is calculated per light source. Each pixel of the RSM is then treated as a light source to approximate the indirect light. Indirect illumination for a surface point is calculated by sampling the RSM. The basic idea is to project the surface point into the RSM. Then the RSM is sampled several times (e.g. 400) around the projection location and the indirect light contributions are accumulated. With growing distance to the projection location the sampling density is decreased, and the sample weight is increased. Occlusions are not taken into account.

Due to the fact that processing a large number of RSM samples per pixel is impractical, the authors proposed to calculate the indirect illumination only in low resolution, and interpolate where possible. This is done by using deferred shading, and rendering the scene from the camera into a low resolution off-screen buffer. This buffer stores the indirect illumination. In an additional rendering pass the low resolution indirect illumination is then interpolated for pixels rendered in full resolution. In case the normals or world locations of currently shaded pixel and low resolution sample differ significantly, the corresponding sample is discarded. Only in case three or more samples are considered valid, they are used for bi-linear interpolation. For pixels where no interpolation was possible, the indirect illumination is calculated the same way as for the low resolution buffer.

The method was able to render one-bounce indirect illumination at up to 24 FPS and can handle dynamic scenes. An image (with additional ambient occlusion) rendered using this algorithm can be seen in Figure 7.

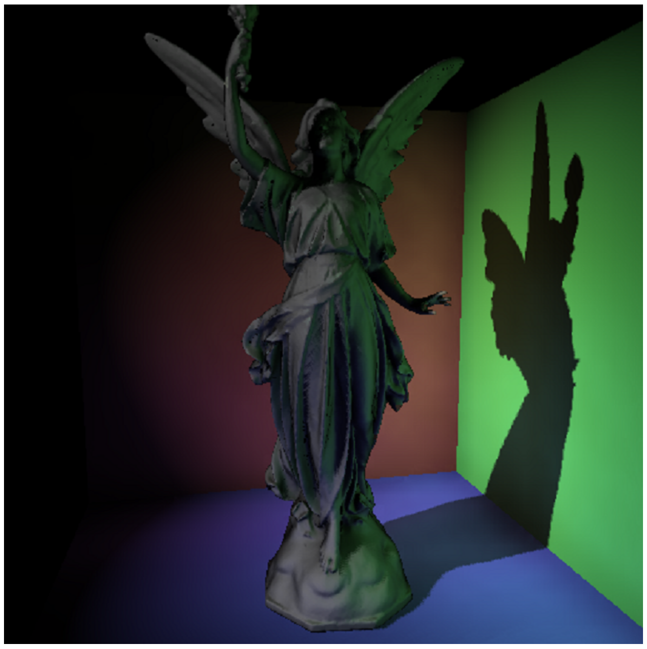

Figure 7: An image rendered using the algorithm described in the paper 'Reflective shadow maps' (see Section 3.2.1). Image taken from Dachsbacher and Stamminger [2005].

3.2.2 Interactive Indirect Illumination Using Voxel Cone Tracing

Crassin et al. [2011] presented a method for real-time indirect illumination of dynamic scenes. They use a voxel octree data-structure and cone tracing to generate the final image. The method is able to calculate two light bounces for diffuse and glossy surfaces at interactive rates.

The scene geometry is voxelized by rasterization (with deactivated depth test) along the three main axis using a low resolution. Material, texture colour and normal are stored for each voxel in a sparse octree on the GPU. This data structure is used for lighting calculations and dynamically updated in case of changes in the scene. The algorithm consists of three basic steps: (i) injections of direct light into the voxel tree, (ii) tree filtering, and (iii) rendering respectively lighting calculations from the viewpoint of the camera.

In the first step the scene is rasterized from the viewpoint of every light source. For each pixel a photon is splatted which injects the direct light intensity and light direction into the leaves of the octree. In the next step the information stored in the leaves is filtered and propagated into the upper tree nodes. In the last step the scene is rendered form the viewpoint of the camera using deferred shading. Indirect illumination is calculated using final gathering via approximate cone tracing. The hemisphere sampling is approximated by only a few cones. For example, in case of Phong-like materials, a small number of larger cones for the diffuse part and one narrow cone in direction of the reflection is used. Using the cones aperture, the appropriate octree level for sampling is chosen.

For the sponza scene (see Figure 1) the method was able to render an average of 30 FPS without a specular cone 20 FPS respectively using one specular cone. The screen resolution was 512 x 512 and the scene contained one moving light and a dynamic object. In comparison to light propagation volumes, the authors stated that their approach achieves better performance and quality. A mentioned drawback is the high memory consumption. Another example image produced by this algorithm can be seen in Figure 8.

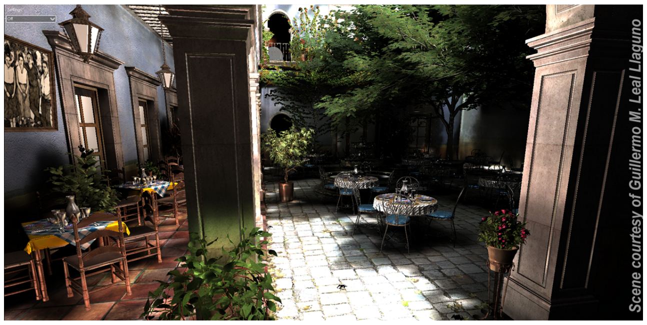

Figure 8: An image rendered using the method described in the paper called 'Interactive Indirect Illumination Using Voxel Cone Tracing' (see Section 3.2.2). Image taken from Crassin et al. [2011].

3.2.3 Making Imperfect Shadow Maps View-Adaptive: High-Quality Global Illumination in Large Dynamic Scenes

Ritschel et al. [2011] presented an approach, which extends instant radiosity-based methods to support large dynamic scenes and increase illumination quality. The authors improve two common techniques used for instant radiosity, reflective shadow mapping (RSM) and imperfect shadow mapping (ISM), to be view adaptive.

Instant radiosity approximates global illumination, and indirect illumination respectively, by using many virtual point lights (VPLs). Rendering a huge amount of lights is accomplished via a deferred renderer, which only computes shading for actual visible fragments. VPLs are placed using reflective shadow maps (see Section 3.2.1). Using the information stored in the RSMs, VPLs can be placed to approximate the indirect light bounce at surfaces. In order to calculate indirect shadows, a shadow map has to be calculated for each VPL. Due to a high number of lights, on the one hand and real-time requirements, on the other hand, classic shadow maps are not suitable and so-called imperfect shadow maps (ISM) are used. For the ISM generation the scene geometry is approximated by points which can be efficiently splatted into a huge texture that stores many low resolution shadow maps. Possible gaps are filled by a push-pull diffusion of neighbouring depth values.

The ideal placement of VPLs is not trivial and often uniform distribution is not optimal. Due to this fact, the authors proposed a non-uniform placement of VPLs, according to their influence on the final image. The influence of a VPL on a given view is represented via a so-called bidirectional reflective shadow map (BRSM). Furthermore, a regular distribution of points over the scene geometry during ISM generation may lead to artefacts for large scenes. Thus, the authors presented an algorithm for adaptive point placement.

For the non-uniform generation and distribution of the VPLs, the BRSMs have to be generated. At first the scene is rendered into a frame buffer (camera) and into a RSM (light). The RSM stores position, normal and reflectance. Each texel of the RSM is treated as a potential VPL. Using every texel as a actual VPL is not efficient. Thus, in practice, only subsets (of approx. 1000 texels) result in actual VPLs. In order to select a good subset, the contribution of each potential VPL on a few randomly selected view samples is computed and the average is stored in a BRSM. The visibility is not taken into account during this step. Additionally view samples near to the virtual point light are favoured, by projection of the VPL to screen-space and evaluation of the distance to the potential view sample. Finally the BRSM is used to guide the creation of the actual VPLs.

The goal of the adaptive placement of points during ISM generation is to have many points near to the viewer and less in the distance. The basic idea is to calculate the importance of every triangle in respect to the view samples and generates points accordingly. The importance of a triangle is represented by the solid angle in respect to a view sample. Using a cumulative density function (CDF) the importances of all triangles is combined. This CDF is used to control the placement of points and ensures that triangles near to the viewer are represented using more samples than triangles further away. The exact position of a point inside a triangle is determined using random barycentric coordinates. In practice it is not feasible to calculate the importance of every triangle in respect to all view samples. Thus, only a few random view samples, per triangle, are selected and considered.

In order to increase temporal stability, only 1/8 of the points for ISMs generation are updated every frame. Additionally, the projections used for the ISMs are adapted to the radiance of the VPLs. A computational speed-up can be achieved by using differing VPL-subsets for adjacent pixels shading and use geometry-aware filtering. For improved stability and to avoid popping artefacts, the BRSM is also smoothed using a geometry-aware filter. Further, VPL positions are not clamped to the texel centres of the BRSM, but are also placed in between.

The method was tested using large dynamic scenes, and was able to run at interactive to real-time frame rates using a resolution of 1600 x 800 pixels. The performance barely depends on the chosen view-port. However, a possible limitation is the lack of scene-size independence. An image rendered using this method can be seen in Figure 9.

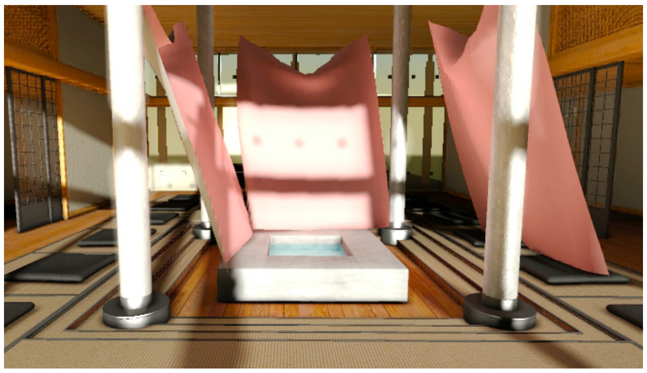
Figure 9: A scene rendered using the method described in the paper called 'Making Imperfect Shadow Maps View-Adaptive: High-Quality Global Illumination in Large Dynamic Scenes' (see Section 3.2.3). Image taken from Ritschel et al. [2011].

3.2.4 Real-time diffuse global illumination using radiance hints

Papaioannou [2011] presented an algorithm that approximates multiple diffuse light bounces. It uses reflective shadow maps (see Section 3.2.1) and stores the calculated radiance in a grid based data structure. A grid element is referred to as radiance hint (RH). The algorithm can be divided into four steps: (i) RSM generation, (ii) initialisation of the RH gird, (iii) accumulation of multiple bounces and (iv) irradiance estimation per fragment.

In the first step an RSM is calculated for each light source. The RSMs are sampled by each RH in order to construct the approximate radiance field. In each grid cell corresponding to one RH multiple positions are chosen. From each of those positions a circle shaped area of the RSM is sampled and the incoming radiance in respect to the corresponding location is computed. The radiance acquired at each position of the grid cell is combined and encoded using spherical harmonics. RSM samples with a normal pointing away from the RH are rejected and not used. Additionally, the minimum and maximum distances to the RSM samples are stored for each RH.

In order to approximate multiple bounces each RH accumulates the radiance from other RHs which lie within a certain radius around the RH cell centre. Due to the fact that reflected radiance from surrounding geometry is of special interest, RHs in proximity to surfaces are favoured. The previously stored minimum distances are used to identify RHs which are close to geometry. Using the minimum and maximum values it is also possible to estimate the visibility between RHs. In case the distance between two RHs is smaller than the sum of both minimum RSM sample distances, the visibility is considered unblocked. The visibility is treated as blocked in case the distance between the RHs is greater than the sum of both maximum RSM sample distances. Intermediate distances respectively visibility values are linearly interpolated. According to the visibility, the radiance exchange between RHs is modulated.

In order to evaluate the radiance of each fragment for the final image the RH grid is sampled. A small number of sample positions are placed on the normal aligned hemisphere around the currently shaded point. Then the radiance of the four nearest RHs is interpolated for each sample location. The combined result is then used for the final deferred shading of the corresponding fragment.

Only occlusion information captured in the RSM (normal orientation) affects the RSM sampling. This can lead to incorrect illumination, in case the path between the RH and a corresponding RMS sample is blocked by geometry not represented in the RSM. To solve this problem the authors proposed to attenuate the radiance of certain RSM samples based on the current view. The basic idea is to compare the depth values of equidistant steps along the path between RH and corresponding RSM sample with the camera depth map. According to the number of samples which are behind the depth map of the camera, the radiance of the RMS sample is attenuated.

The method is able to approximate multiple diffuse light bounces and was tested using various scenes. The global illumination computation for a Cornell Box including a dragon and a happy Buddha model took between 4 and 13 ms, depending on various parameters. A possible limitation mentioned by the authors is, that indirect light blocking geometry may not be captured by the RSM, thus leading to artefacts and wrong illumination. It is stated, that although the solution (mentioned in the previous paragraph) does not work in every situation, it is able to handle cases in which light propagation volumes fail. In Figure 10 two example images rendered using this method can be seen.

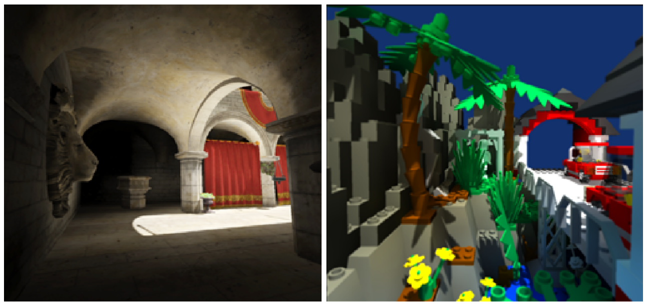

Figure 10: Two images rendered using the method described in the paper called 'Real-time diffuse global illumination using radiance hints' (see Section 3.2.4). Images taken from Papaioannou [2011].

3.2.5 Cascaded light propagation volumes for real-time indirect illumination

Kaplanyan and Dachsbacher [2010] presented a method for real-time approximation of low-frequency indirect illumination. This approach is able to handle large and fully dynamic scenes and is used in a state of the art game engine called CryEngine by Crytek.

For the calculation of indirect illumination the scene is represented via two low resolution grid data structures. The first grid, called light propagation volume (LPV), stores the indirect illumination. The second grid represents the light blocking geometry volume (GV). At each grid cell the illumination and a so-called blocking potential are encoded using low-order spherical harmonics (SH) coefficients. The algorithm consists of four basic steps: (i) initialisation of the LPV by injecting light, (ii) injection of blocking geometry into the GV grid, (iii) light propagation and (iv) final rendering. Direct light and shadows of point light sources are rendered using traditional methods.

In the first step, the direct illumination of area lights and indirect illumination through surfaces is approximated using a large number of so-called virtual point lights (VPL). This VPL are not directly used for rendering, but their illumination is injected into the LPV. In order to generate those VPLs which represent the indirect light, a reflective shadow map (see Section 3.2.1) is generated for every light source in the scene. The RSM texels are treated as virtual light sources with a light distribution determined by the stored normal and intensity. The light distributions of all VPLs are converted into their SH representation and the coefficients are injected into the LPV and accumulated in the corresponding grid cells. For direct light from area lights, additional VPLs are created and injected in the same way.

In the second step, the geometry volume is initialized. In each cell the coefficients for the SH encoding of the blocking potential are stored. The prior computed RSMs and the deferred rendering framebuffer of the current view-port represent the sampling of the scene geometry. Each sample is treated as a small surface element. The blocking probability of a surface element is determined by its size, its normal and the size of the corresponding grid cell. For each surface element the blocking probability is converted into the SH representation, injected into the GV and accumulated similar to the first step.

In the third step, the light is iteratively propagated using the previously calculated information stored in the LPV and the GV. In every iteration the light of each grid cell is propagated along the main axes to its neighbours. Propagation from one cell to another is done by computing received light intensity for each face of the destination cell. The intensities of each face are then used to construct light sources with an intensity distribution that mimics the illumination of the faces. The light distribution of these lights is then encoded, using SH, and accumulated in the corresponding grid cell. To account for blocking geometry, the calculated light distribution is modulated by the blocking potential stored in the GV to produce indirect shadows. The results of all iterations are accumulated into a separate grid which represents the final light distribution.

In the last step, the illumination is evaluated, using SH coefficients stored in the LPV, and used for the final rendering. In order to handle large scenes, the authors proposed the use of nested grids which move with the camera. This approach enables the use of grids with higher resolution near the viewer.

In case of the Sponza scene (see Figure 11) the method is able to render at 60 FPS. The calculation of the indirect illumination took about 10 ms. Possible problems of the method are light bleeding, in case of low grid resolution or incomplete scene sampling, and strong light diffusion due to low-order SH representation. The authors also discussed extensions of the method to support multiple bounces, glossy reflections and single scattering in participating media.

4 Discussion

In the previous sections several methods have been described, which approximate global illumination to some extent. All algorithms are capable to render at interactive to real-time frame rates and are able to handle dynamic scenes.

Ritschel et al. [2012] presented a comprehensive comparison of different global illumination algorithms in respect to various properties, e.g. speed or quality. An excerpt of this comparison, according to the methods described in this paper, can be seen in Table 1. The table shows that performance, scalability and the capability to handle dynamic scenes are quite good compared to the quality of the resulting global illumination solution. The methods presented

Method	Speed	Quality	Dynam.	Scalab.	Transport
Ritschel et al. [2009]	★★★★★	★☆☆☆☆	★★★★☆	★★★★☆	LDDE
Dachsbacher and Stamminger [2005]	★★★★☆	★☆☆☆☆	★★★★★	★★★☆☆	LDDE
Ritschel et al. [2011]	★★★★☆	★★★☆☆	★★★★☆	★★★★★	$\mathrm{LD}\{\mathrm{S}\vert\mathrm{D}\}\mathrm{E}$
Kaplanyan and Dachsbacher [2010]	★★★★★	★★☆☆☆	★★★★☆	★★★★★	$\mathrm{LD}^{+}\{\mathrm{D}\vert\mathrm{V}\vert\mathrm{S}\}\mathrm{E}$

Table 1: Comparison of the different methods (excerpt) [Ritschel et al. 2012].

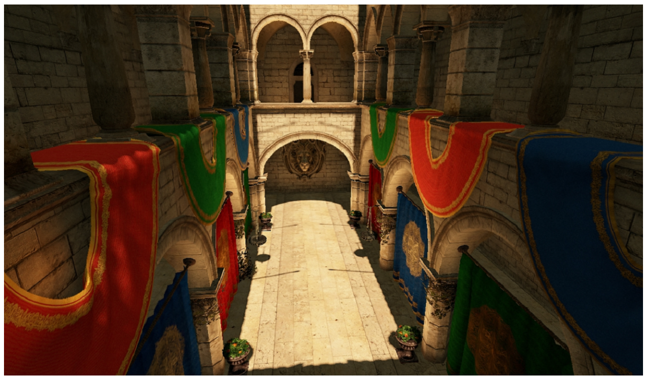

Figure 11: The Sponza scene rendered using the algorithm described in the paper called 'Cascaded light propagation volumes for real-time indirect illumination' (see Section 3.2.5). Image taken from Kaplanyan and Dachsbacher [2010].

by Mattausch et al. [2011], Crassin et al. [2011] and Papaioannou [2011] were not discussed in the paper from Ritschel et al. [2012] and are therefore not present in the comparison table.

The method described in the paper called 'Temporal Screen-Space Ambient Occlusion' (see Section 3.1.1) improves SSAO, which is already in use in many real-time applications, especially games. Thus extending existing SSAO implementations to use temporal coherence as described in the corresponding paper, could improve the visual quality without performance loss. The second method described in the paper called 'Approximating dynamic global illumination in image space' (see Section 3.1.2) is also based on SSAO. Using this method it is possible to extend an existing SSAO to support one-bounce indirect illumination. Due to the computation of the additional indirect bounce, it is slower than basic SSAO. Like most screen space approaches the method is, to some extent, independent of the scene complexity and handles dynamic scenes.

Another approach to calculate indirect illumination was described in the paper called 'Reflective shadow maps' (see Section 3.2.1). The quality of the calculated indirect illumination and the performance, even for rather simple scenes, are quite low. However, compared to the other methods, it is quite old and it is likely that on today's graphic hardware the performance would be better. It is used by the methods described in the papers 'Real-time diffuse global illumination using radiance hints' (see Section 3.2.4) and 'Cascaded light propagation volumes for real-time indirect illumination' (see 3.2.5) and improved by the algorithm described in the paper called 'Making Imperfect Shadow Maps View-Adaptive: High-Quality Global Illumination in Large Dynamic Scenes' (see Section 3.2.3).

The algorithm described in the paper called 'Interactive Indirect Illumination Using Voxel Cone Tracing' (see Section 3.2.2) is able to compute relatively high quality illumination and is even able to render glossy surfaces. However, this comes at the cost that its rendering speed is relatively low. A limitation in case of large scenes may be the high memory consumption.

The authors of the paper called 'Making Imperfect Shadow Maps View-Adaptive: High-Quality Global Illumination in Large Dynamic Scenes' (see Section 3.2.3) developed an algorithm especially suited for large and dynamic scenes based on an improved instant radiosity approach. Although the distribution of VPLs and the ISM generation is improved for larger scenes, the method is not totally scene-size independent.

The method presented by Papaioannou [2011] in the paper called 'Real-time diffuse global illumination using radiance hints' (see Section 3.2.4) is to some extent similar to the method described by Kaplanyan and Dachsbacher [2010] in the paper 'Cascaded light propagation volumes for real-time indirect illumination' (see Section 3.2.5), due to the fact that both encode the illumination using spherical harmonics stored in a regular grid data-structure. Papaioannou [2011] claims that his method handles specific geometry configurations better than the algorithm developed by Kaplanyan and Dachsbacher [2010]. However, the latter is already implemented in a state-of-the-art game engine used for major commercial games.

One common property of all discussed methods is that they all use quite a lot of simplifications in order to render an approximation of global illumination in real-time. Except for the AO method, all algorithms compute at least one diffuse indirect light bounce (LDDE) and use more or less accurate approaches to handle occlusion. Specular surfaces are only supported, to some extent, by the methods presented by Crassin et al. [2011], Kaplanyan and Dachsbacher [2010] and Ritschel et al. [2011].

Due to the fact that computing full global illumination is a hard task, all real-time methods only calculate a subset of full global illumination effects, e.g. a limited number of indirect diffuse light bounces. However, as the performance and memory size of graphic hardware increases, newer and better algorithms may be developed. Such methods may compute more accurate or even full global illumination in real-time. Furthermore, they may be able to render effects like accurate reflections/refractions and even high quality caustics.

5 Conclusion

In this paper a variety of methods which are able to approximate global illumination in dynamic scenes at real-time frame rates where presented and discussed. In Section 1 and 2 motivation for using global illumination was given and the basics were explained. In Section 3 two methods regrading ambient occlusion and five algorithms for approximation of indirect illumination where presented. The different methods and their features were compared and discussed in Section 4.

It turns out that current real-time global illumination methods use quite a lot approximations and the quality of the computed illumination is still rather low compared to offline renderings. However, due to the fast hardware development and increase in processing power, it is likely that in coming years high quality real-time global illumination will be possible.

References

BENGTSSON, M. 2010. *Real time global illumination using the GPU*. Master's thesis, Linkping University, Department of Science and Technology.

CRASSIN, C., NEYRET, F., SAINZ, M., GREEN, S., AND EISEMANN, E. 2011. Interactive indirect illumination using voxel cone tracing. *Computer Graphics Forum 30*, 7, 1921–1930.

DACHSBACHER, C., AND STAMMINGER, M. 2005. Reflective shadow maps. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, ACM, New York, NY, USA, I3D '05, 203–231.

DUTRÉ, P., BALA, K., AND BEKAERT, P. 2006. *Advanced global illumination*. Ak Peters Series. AK Peters.

HECKBERT, P. S. 1990. Adaptive radiosity textures for bidirectional ray tracing. *SIGGRAPH Computer Graphics 24*, 4 (Sept.), 145–154.

KAPLANYAN, A., AND DACHSBACHER, C. 2010. Cascaded light propagation volumes for real-time indirect illumination. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, New York, NY, USA, I3D '10, 99–107.

KNECHT, M. 2009. *Real-Time Global Illumination Using Temporal Coherence*. Master's thesis, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstrasse 9-11/186, A-1040 Vienna, Austria.

MATTAUSCH, O., SCHERZER, D., AND WIMMER, M. 2011. Temporal screen-space ambient occlusion. In *GPU Pro 2*, W. Engel, Ed. A.K. Peters, Feb.

MITTRING, M. 2007. Finding next gen: Cryengine 2. In *ACM SIGGRAPH 2007 courses*, ACM, New York, NY, USA, SIGGRAPH '07, 97–121.

PAPAIOANNOU, G. 2011. Real-time diffuse global illumination using radiance hints. In *Proceedings of the ACM SIGGRAPH Symposium on High Performance Graphics*, ACM, New York, NY, USA, HPG '11, 15–24.

RITSCHEL, T., GROSCH, T., AND SEIDEL, H.-P. 2009. Approximating dynamic global illumination in image space. In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, ACM, New York, NY, USA, I3D '09, 75–82.

RITSCHEL, T., EISEMANN, E., HA, I., KIM, J. D. K., AND SEIDEL, H.-P. 2011. Making imperfect shadow maps view-adaptive: High-quality global illumination in large dynamic scenes. *Computer Graphics Forum 30*, 8, 2258–2269.

RITSCHEL, T., DACHSBACHER, C., GROSCH, T., AND KAUTZ, J. 2012. The state of the art in interactive global illumination. *Computer Graphics Forum 31*, 1, 160–188.

WATT, A. 2002. *3D-Computergrafik*. Pearson Studium.