IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 10, 2021, accepted August 3, 2021, date of publication September 1, 2021, date of current version September 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3109663

Comparative Analysis of Real-Time Global Illumination Techniques in Current Game Engines

CRISTIAN LAMBRU, ANCA MORAR, FLORICA MOLDOVEANU, (Member, IEEE), VICTOR ASAVEI, (Member, IEEE), AND ALIN MOLDOVEANU, (Member, IEEE)
Faculty of Automatic Control and Computers, POLITEHNICA University of Bucharest, 060042 Bucharest, Romania

Corresponding author: Cristian Lambru (andrei.lambru@upb.ro)

ABSTRACT The most important component of photorealism in Computer Graphics is given by a physically correct approximation of the light transport. Besides the direct illumination from light sources, there is an indirect illumination, produced by the reflections of the light rays on other surfaces of the scene. In Computer Graphics, the process of computing the illumination of a surface by considering both the direct and the indirect illumination is widely known as global illumination. This paper describes several classes of real-time global illumination techniques used in current game engines together with our own implementations of these approaches. All implementations were made in our own framework, specially designed with a multi-pass rendering architecture that allows fast implementation of rendering techniques and the reuse of functionalities. We analyze these classes based on the following criteria: the visual results produced by the indirect diffuse lighting, the ability to produce glossy reflections, shadows, ambient occlusion, subsurface scattering, translucency and volumetric lighting as well as the ability to simulate area lights. We present the quantitative results of our implementations, obtained with the same external parameters for all techniques, thanks to the unified implementations in the same framework. An important observation is that our analysis is focused on the techniques that are based on the rasterization pipeline, thus, the comparison does not include the techniques designed entirely for the ray-tracing pipeline.

INDEX TERMS Computer graphics, real-time global illumination, reflective shadow map, light propagation volumes, voxel-based representation, screen space.

I. INTRODUCTION

The rise of photorealism in real-time computer graphics took a big turn in recent years thanks to the advancements of the graphics processing units (GPUs). Along with this, relatively new techniques were adopted by the game industry to enhance the visual quality of their products, taking advantage of the new capabilities of GPUs.

There are several techniques of real-time global illumination with different degrees of flexibility. Most of them require a pre-computation pass which accumulates information about the scene geometry. A major drawback of this approach is the lack of indirect lighting influence on dynamic objects, which cannot be processed offline. In recent years there was a breakthrough in real-time global illumination techniques that don't require a pre-processing pass and work entirely with real-time computed data structures.

More and more commercially available game engines provide specialized implementations of global illumination techniques to support different degrees of photorealism, in real-time, at various performance costs. Unreal Engine 4 implements a monolithic lighting system which provides very good visual results, based on its own surface caching technique called Lightmass Global Illumination [1]. This built-in global illumination approach is completely pre-computed. Alternatively, it offers various implementations for real-time techniques, but the support for them is partially limited, requiring additional work for the developer. Unity 3D game engine offers various implementations for different use cases, but all of them are partially offline, requiring a scene pre-computation pass [2]. CryEngine offers full support for real-time global illumination and also provides tools for pre-computed techniques based on the user requirements [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Giambattista Gruosso.

Additionally to game engines, there are several frameworks called global illumination engines, which provide only the illumination part and support integration in most of the commercially available game engines or in proprietary ones. They usually offer solid implementations of pre-computed techniques but lack support for real-time ones. One such example is the framework called Enlighten [4], which uses a hybrid technique that offers partially dynamic global illumination results, but still requires a scene pre-processing pass.

In the following sections we analyze several classes of real-time global illumination techniques used in current game engines. We chose to analyze the following four classes, that we consider to cover most of the real-time global illumination techniques used in current game engines:

- Reflective shadow map-based techniques
- Discrete ordinate method-based techniques
- Techniques using the voxel representation of the scene
- Screen space techniques

We chose a representative technique from each class and made our own implementations. We present these implementations, together with our in-house framework specially designed for implementing rendering techniques, with a multi-pass rendering architecture. We chose to develop this framework in order to test all techniques using the same external parameters, such as scene geometry and material properties. Also, a more accurate comparison requires the same implementation at the level of the graphics API, objects management in the GPU memory and the same internal structure, objectives achieved with the developed framework. Moreover, it was designed for fast implementation of rendering techniques and allows the reuse of functionalities. We describe this framework in more detail and present the rendering pipeline of each real-time global illumination implementation.

Further, we analyze the visual results for the indirect diffuse illumination obtained with each class of techniques and also their ability to produce auxiliary photorealistic effects that are usually decoupled from the actual illumination process and require supplementary techniques. We chose as auxiliary effects the ability to produce: glossy reflections, shadows, ambient occlusion, subsurface scattering, translucency, volumetric lighting and the simulation of area lights. We provide visual results, when they exist, along with the technical details that were used to obtain them. Moreover, we explain why some effects cannot be produced. We provide and analyze the quantitative results of our implementations concluding with our recommendations for the best scenarios that suit each class of selected real-time global illumination techniques.

The rest of the paper is organized as follows. Section II reviews related comparative analyses and surveys in the field. In Section III we offer the details of the chosen classes of real-time global illumination techniques, followed by the description of our implementations in Section IV. In Sections V and VI we analyze each class of techniques based on qualitative and quantitative criteria, respectively. Finally, conclusions are presented in Section VII.

II. RELATED WORK

Over time, several global illumination techniques that were initially classified as offline or interactive solutions have achieved real-time performance due to the technological advances of GPUs. The game industry, which is the main beneficiary of these technological advances, slowly accepts techniques that were not initially intended for real-time applications. Therefore, a classification of global illumination techniques based solely on time performance is not always relevant, so several classifications on different criteria have been proposed.

Ritschel *et al.* [5] provided a very comprehensive classification of the interactive global illumination techniques. Moreover, they proposed a comparative analysis based on qualitative criteria. This analysis, even if quite old, offers an exceptional overview of the global illumination field. The authors of this analysis are the same authors of most of the techniques that are currently considered state of the art in the game industry. They are the authors of some of the techniques that are described and analyzed in this paper.

Other analyses were performed for particular scenarios and environments. Dachsbacher *et al.* [6] provided a comprehensive analysis for techniques that are based on many-lights rendering. Jönsson *et al.* [7] classified the global illumination techniques specialized for general volume rendering. They also proposed a comparative analysis based on theoretical criteria. Heidrich [8] offered a specialized analysis for interactive techniques based on the capability to provide good results in non-diffuse environments. These environments present an additional difficulty to obtain the indirect components of illumination. Damez *et al.* [9] described and analyzed the global illumination techniques specialized for high-quality animations and dynamic environments.

The book "Advanced Global Illumination" by Dutre *et al.* [10] offers a detailed analysis for most of the offline global illumination techniques. It also offers a comprehensive description of the theoretical basis which all of those techniques rely on. Also, the book "Real-Time Rendering" by Akenine-Möller *et al.* [11] presents a very good analysis for the current rendering techniques accepted by the game industry. They offer an analysis for current global illumination techniques with a focus on industry approaches.

There are several comprehensive analyses of various components of the global illumination effect. Szirmay-Kalos *et al.* [12] presented the techniques that can produce mirror reflections. A good description of shadow techniques was made by Woo *et al.* [13] and in particular for those in real-time by Hasenfratz *et al.* [14]. Also, several analyses were made for classes of shadow techniques, such as that of Scherzer *et al.* [15] for the shadow mapping technique or that of Kolivand and Sunar [16] for the shadow volumes one. A more comprehensive analysis of shadows techniques is presented in the book "Real-Time Shadows"

by Eisemann *et al.* [17]. An analysis of the ambient occlusion component is presented by Méndez-Feliu and Sbert [18]. Cerezo *et al.* [19] provided a comprehensive analysis of participating media rendering techniques.

All the analyses described above represent a very good source of documentation about the field of real-time global illumination. In this paper, we analyse the techniques in this field from a specific point of view, important for the visual results they offer. We propose a classification of these techniques and analyze the possibility that the proposed classes to produce autonomously and unitarily a chosen set of illumination effects and phenomena. All our conclusions are accompanied by visual and quantitative results obtained by us in a framework in which all classes of techniques have been implemented.

III. REAL-TIME GLOBAL ILLUMINATION TECHNIQUES

A. RENDERING EQUATION

Kajiya [20] states that light can be described as an electromagnetic radiation using the following equation:

$$L(\mathbf{x}, \boldsymbol{\omega}) = L_e(\mathbf{x}, \boldsymbol{\omega}) + \int_{\Omega^+} f_r(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}') L_i(\mathbf{x}, \boldsymbol{\omega}') \cos\theta \, d\boldsymbol{\omega}' \quad (1)$$

where $L(\mathbf{x}, \boldsymbol{\omega})$ represents the radiance leaving from point $\mathbf{x}$ in direction $\boldsymbol{\omega}$. $L_e(\mathbf{x}, \boldsymbol{\omega})$ is the emittance term, which represents the radiance directly emitted from point $\mathbf{x}$ in direction $\boldsymbol{\omega}$. $f_r(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}')$ is the scattering function of the surface in point $\mathbf{x}$ for the radiance which comes from direction $\boldsymbol{\omega}'$ and is reflected in direction $\boldsymbol{\omega}$. $L_i(\mathbf{x}, \boldsymbol{\omega}')$ represents the radiance which comes from direction $\boldsymbol{\omega}'$ to $\mathbf{x}$. Ω^+ is the upper hemisphere oriented around the normal vector $\vec{N}_\mathbf{x}$ and $\boldsymbol{\theta}$ is the angle made by the direction $\boldsymbol{\omega}'$ with $\vec{N}_\mathbf{x}$. Figure 1 illustrates the light transport approximation.

$f_r(\mathbf{x}, \boldsymbol{\omega}, \boldsymbol{\omega}')$ function represents the distribution of the radiance reflected over the upper hemisphere of point $\mathbf{x}$. The function is evaluated for a pair of directions $(\boldsymbol{\omega}, \boldsymbol{\omega}')$ in relation to the angle with the normal vector $\vec{N}_\mathbf{x}$. It is widely known as the bidirectional reflectance distribution function (BRDF) [21]. This surface material specific function can be described using various reflection models like the ones proposed by Lambert, Phong [22], Blinn [23], Cook and Torrance [24] or Walter *et al.* [25]. A comprehensive analysis of the reflection models was presented by Schlick [26].

The final objective of the global illumination techniques is to compute an approximation for the $L_i(\mathbf{x}, \boldsymbol{\omega}')$ factor using various radiance accumulation methods for all directions in the upper hemisphere of point $\mathbf{x}$.

Kajiya [20] proposed a statistical solution known as path tracing that uses the Monte Carlo integration to solve the equation. This technique extends the ray-tracing approach proposed by Whitted [27] by using several randomly traced rays at each intersection point, instead of a single ray traced in the reflected direction. As can be seen from the number of rays created recursively, this technique has a low efficiency and it is not suitable for real-time applications, even if over time several optimizations have been proposed [10].

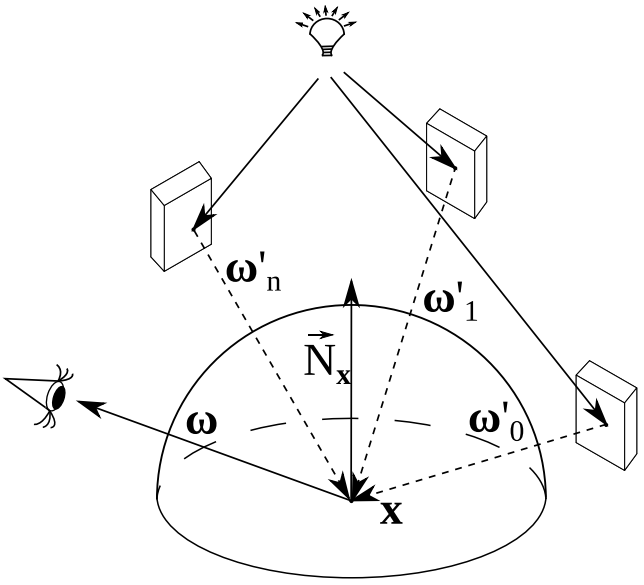


FIGURE 1. The light transport approximation described in Equation 1. The dotted paths represent the light coming indirectly from source to point x after the reflection on geometry. It should be noted that each point where light is reflected receives indirect light from the rest of the points in the scene. However, for the simplicity of the visual representation, the paths of these light rays have been omitted graphically.

Other approaches use the finite elements method to discretize the scene geometry in patches and compute the indirect illumination using the evaluation of those patches. Goral *et al.* [28] first proposed such an approach, known as radiosity. A comprehensive analysis of this class of techniques was presented by Cohen and Wallace [29]. Another class is represented by the photon mapping [30] technique. This approach computes the incoming radiance in two passes. First, a set of photons are spread from the light position into the scene using the ray-tracing principle and each bounce of rays is cached into a photon map. In the second pass, the intensity of the light in a point is computed based on the photon density estimation around that point. A class that extends this technique is instant radiosity [31]. It introduces the concept of virtual point light (VPL) to represent a point on a surface that emits light in a hemisphere around the normal to that point. This technique uses a first pass similar to that of the photon mapping technique to generate the positions of the VPLs, but for the second pass, a gathering approach is used to compute the indirect illumination.

Unfortunately, none of the techniques described above is suitable for real-time applications. However, several approaches have been proposed to cache the results of these techniques into various data structures and compute the indirect illumination using the cached information. Ward *et al.* [32] proposed to cache the results of the ray-tracing technique into a texture. However, with the advances of GPUs, it has been proven that the texture can also be incrementally generated in real-time [33]. Greger *et al.* [34] proposed to cache the irradiance at discrete points in world

space and use this data to compute the indirect illumination. Sloan *et al.* [35] proposed to pre-compute only the radiance transfer to allow changes such as light intensity or color in real-time.

Over time, several real-time techniques have been proposed that do not require pre-computed information. Such techniques, along with their variations, are described in the following sections.

B. REFLECTIVE SHADOW MAP-BASED TECHNIQUES

The reflective shadow map (RSM) was introduced by Dachsbacher and Stamminger [36] and represents a data structure with several buffers that stores information about the first bounce of light on the surfaces of the scene. It is obtained by rendering the scene from the light position with a projection specific to the type of the light source. The buffers store the position where the reflections take place, the normal vectors of the geometry and the reflected radiant flux, which represents the intensity of the photon flux emitted by the light source.

To produce the indirect diffuse illumination, an extended version of the instant radiosity technique is used, where the VPLs represent the pixels in the RSM. Unfortunately, this means that the evaluation time is directly proportional to the resolution of the RSM. This process has a final complexity of $\theta(Screen_w \cdot Screen_h \cdot RSM_w \cdot RSM_h)$, where $\{Screen_w, Screen_h\}$ is the resolution of the screen and $\{RSM_w, RSM_h\}$ is the resolution of the RSM. This complexity exceeds the capabilities of current GPUs and Dachsbacher and Stamminger [36] proposed a stochastic sampling to acquire the VPLs. The evaluation of each pixel in the screen is still done, but instead of processing all the pixels in the RSM, only a subset of pixels is chosen. Let $proj_{RSM}^{Screen_{i,j}}$ be the position of a screen pixel projected on the RSM. The pixels in the RSM that become VPLs are chosen around $proj_{RSM}^{Screen_{i,j}}$, through a Poisson sampling. Also, for temporal coherence, a constant set of samples is used for every screen pixel. This optimization results in a complexity of $\theta(Screen_w \cdot Screen_h \cdot K)$, where K is the constant number of samples. This approach produces a wide range of visual artifacts because the technique is no longer physically correct after including the stochastic sampling.

There are several techniques that use this approach as a pass of accumulating VPLs, which are later processed in different ways. The prevalent approach of these techniques is to use an importance selection strategy of the RSM pixels. Lensing and Broll [37] proposed a shading method that needs only a small set of VPLs, but requires an offline process of selection. Prutkin *et al.* [38] proposed a segmentation preprocess of the RSM to extract a set of virtual area lights which are used for the shading calculations.

There are certain limitations related to an importance selection of the samples, therefore several techniques were proposed to pre-process the RSM and to transform it in different data structures which are more suitable for the indirect lighting computations. Kaplanyan [39] used the RSM as the source for a spatial grid of spherical harmonics. This technique is described in the next section. Bischoff *et al.* [40] used a more complex process to extract and generate a set of proxy lights from the RSM. These lights are interpreted as direct light sources in a specialized graphics engine which is optimized for the evaluation of a large set of lights. Dachsbacher and Stamminger [41] later described a reverse approach. Instead of calculating the indirect illumination for the entire screen by gathering the VPLs, they used a splatting process for each VPL to compute the indirect illumination only for the screen pixels for which a VPL has a significant influence. This approach improves the performance and allows for additional effects.

The use of the RSM alone has certain limitations. Even if the indirect contribution of light can be computed using the VPLs, the occlusion for the indirect light is difficult to produce. Moreover, the VPLs can be used to obtain only one bounce for the indirect diffuse component. To overcome these limitations, Ritschel *et al.* [42] introduced the imperfect shadow maps (ISMs), which represent low resolution z-buffers acquired for each VPL. The acquisition process uses a coarse approximation of the scene as a set of points. The ISMs are used for the indirect light occlusion and they can be extended to imperfect reflective shadow maps (IRSMs) to acquire sequential bounces of light. Ritschel *et al.* [43] later improved the occlusion detection by using the screen space information in the process of selecting the RSM pixels.

C. DISCRETE ORDINATE METHOD-BASED TECHNIQUES

In order to avoid storing the reflected radiance after one or more light bounces for all surfaces in the scene, several approaches were proposed to pre-compute and store the radiance transfer at discrete positions in a grid that covers the entire scene. The information in the grid cells can be used to obtain the radiance that reaches any position for any direction on the scene surfaces. To store the radiance transfer at a certain position, a specialized data structure is required to store the radiance that reaches the specified position from all directions around it. This class of approaches is known as discrete ordinate methods [44] (DOMs) and was first introduced into the Computer Graphics field by Kajiya and Von Herzen [45]. They stored the radiance at discrete positions in the scene with spherical harmonics to compute the scattering of light in clouds. Geist *et al.* [46] have adopted a similar approach that uses the Lattice-Boltzmann method to store the radiance transfer in generic participating media. Unfortunately, they failed to achieve real-time performance. Fattal [47] proposed a faster approach to store the radiance at discrete positions using several two-dimensional data structures called light propagation maps.

In contrast to the techniques presented above, which have been designed for light scattering in participating media, Kaplanyan [39] applied this approach to produce real-time indirect illumination. He used a three-dimensional grid, called light propagation volume (LPV), that stores a spherical harmonic in each cell. Instead of storing the radiance transfer

of the light that comes directly from a light source, he stored in the LPV the radiance transfer from the light reflected by the scene surfaces. To compute the information in each cell, he initially introduced the reflected radiance from the pixels of an RSM and propagated this initial radiance between the cells for the entire grid. In the shading pass, each cell acts as a VPL.

This technique is divided into three passes: RSM generation, radiance injection and radiance propagation. The first pass generates an RSM for every light. For the radiance injection pass, a set of pixels are selected from the RSM and transformed into a set of VPLs in world space. Each VPL is stored inside the spatial grid in the nearest cell. The representation of the VPL's radiance is stored as a spherical harmonic with emitted radiance contribution only on the hemisphere described by the VPL. The third pass propagates the radiance from any filled cell to all its neighbors. This is an iterative process, every cell receiving in each iteration information from more distant cells. However, this pass can be described as an aggregation process.

There are certain extensions that can enhance this technique. One problem is represented by the required size of the grid which highly depends on the scene scale. Also, there is a lack of indirect light occlusion inside the grid at the propagation pass. To overcome these problems, Kaplanyan and Dachsbacher [48] presented several extensions to the original technique. A cascaded approach was proposed to store different levels of detail of the spatial grid according to the distance from the observer. Moreover, a coarse representation of the scene was proposed to cancel the light propagation. This is represented as an additional grid of spherical harmonics with positions and orientations according to the actual scene geometry. In addition, Franke [49] has extended this approach to be used for mixed reality environments.

D. VOXEL-BASED TECHNIQUES

Voxel-based techniques compute a simplification of the mesh representation of the geometry in the form of a voxel representation. A voxel is basically a three-dimensional version of the pixel and is visually represented as an axis aligned cube. The reason for choosing a voxel structure is the wide range of advantages it provides, ranging from memory usage, friendly packaging into the GPU as a built-in 3D texture to fast collision detection.

The field of scan conversion of the mesh representation into a voxel representation, known as voxelization, has several techniques developed to achieve real-time results. This field is very important because of the flexibility of the voxel representation which has multiple applications on different fields, like simulation, collision detection or rendering. After 1990, several techniques were focused on the accuracy of the voxelization process [50], but with the GPUs advances, those techniques have changed the focus on real-time performance. Fang and Chen [51] presented an implementation on early GPU generations that requires multiple drawings of the scene. Dong *et al.* [52] proposed a technique that requires a single draw of the scene to acquire the voxel representation, but this representation is limited to binary information per voxel. Eisemann and Décoret [53] proposed a similar but faster approach, specialized for watertight models. With the advances of GPUs, several implementations [54], [55] have taken advantage of the high parallel power with CUDA or OpenCL. Crassin *et al.* [56] proposed a technique that takes advantage of the rasterization process of the GPU, which is hardware accelerated. Moreover, they proposed a specialized data structure that can store arbitrary types of data in a sparse voxel octree (SVO) inside the GPU's memory. This data structure can be partially recreated without the need to rebuild the entire SVO.

Over time, several techniques were proposed to compute the indirect illumination in real-time using the voxel representation. Soler *et al.* [57] proposed to use the voxel volume only for indirect lighting occlusion and they have taken advantage of the fast techniques that produce only binary information per voxel. The albedo information is acquired from screen space, but the results are good because of the supplementary accuracy offered by the voxel representation. Thiedemann *et al.* [58] proposed a real-time implementation of the ray-tracing technique using the voxel representation of the geometry. Pantaleoni [55] also used the ray-tracing technique, but he has taken advantage of the CUDA technology for fast GPU voxelization.

Crassin *et al.* [59] proposed a cone-tracing technique which uses 3D mipmaps of the voxel volume and the trilinear interpolation between the voxels for better visual results. Panteleev [60] proposed a slightly modified version of this technique that uses 3D clipmaps of the voxel volume. Aherne *et al.* [61] took advantage of the hardware-accelerated sparse textures to compute indirect illumination. The cone-tracing approach has proven to be flexible and Franke [62] has extended it to be used in mixed reality environments. Several approaches have been proposed to use the cone-tracing process, but without the disadvantage of storing the reflected radiance information for the entire voxel volume, when only a limited subset is required. Sugihara *et al.* [63] used the voxel volume with binary information per voxel for occlusion and a pre-filtered layered RSM for indirect illumination. Chen and Chien [64] used a similar approach, but stored the radiance of the voxels in a compact list.

E. SCREEN SPACE TECHNIQUES

This class of global illumination techniques depends on the information available from the observer's position in the viewing direction. The data structure used for indirect lighting computation is represented by a set of buffers, known as geometric buffers (G-buffers) [65], which store various properties of the geometry visible on the screen. To produce global illumination, the minimum G-buffers required are those that store the positions, normal vectors and material properties. In addition, to compute the indirect illumination, the radiance reflected by the geometry visible on the screen is required.

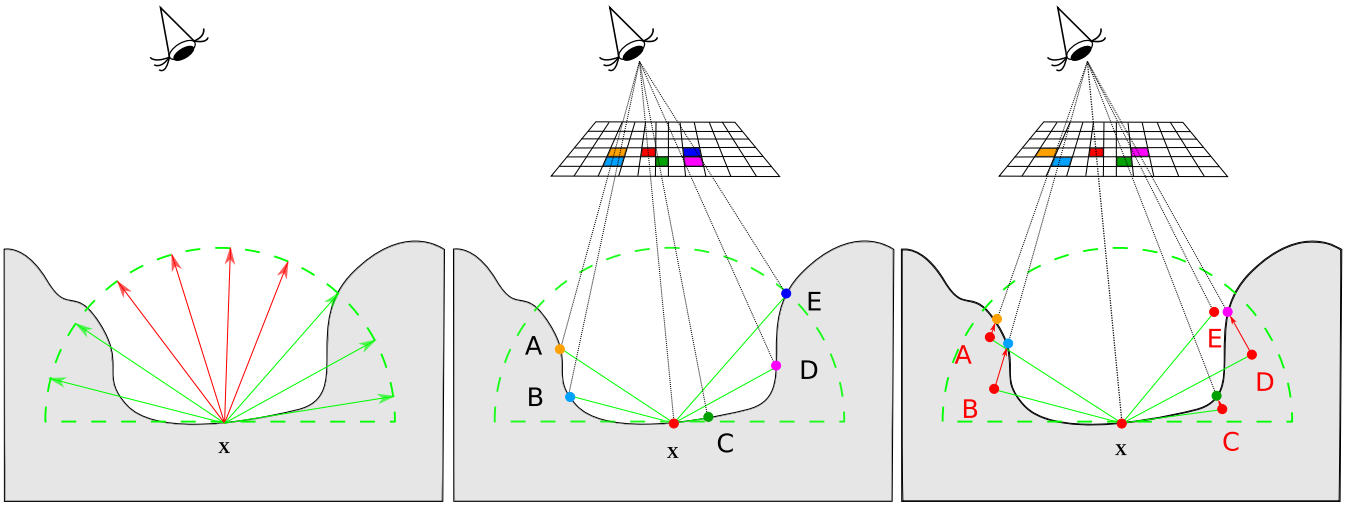


FIGURE 2. Left: The path-tracing technique. Middle: The intersection points of the rays with the geometry. These points are used for indirect lighting computations. Right: The intersection points using the SSDO technique.

This radiance is obtained by computing the direct illumination for each pixel of the screen. This additional buffer is usually produced after the G-buffers acquisition and is considered a separate data structure from the G-buffers. This geometry representation comes with its own set of advantages and problems. The most important advantage is that the information present in the G-buffers is independent of the geometry complexity. Also, the G-buffers can be structured in mipmap pyramids and the sampling is hardware accelerated. Therefore, the techniques that use this data structure can further reduce the amount of geometry processed. However, this is a coarse approximation of the scene geometry and depending on the resolution there may be several scalability problems.

A process similar to the path-tracing technique can be used directly in screen space to compute the indirect illumination. For every pixel, N 2D rays are cast into the G-buffers pixels. These rays are normally distributed inside the upper hemisphere of the pixel according to its normal. All rays are cast in screen space and march through the pixels of the G-buffers. Each pixel is tested for intersection with the ray. Intersecting pixels are used as a source of indirect illumination. This process is described in Figure 2 Left. The N diffuse rays are traced from the evaluated point $\mathbf{x}$. The green rays intersect the geometry and contribute to indirect diffuse illumination and the red rays, which don't intersect the geometry, are ignored. The intersection points can be seen in Figure 2 Middle. The positions and normal vectors of these points are obtained from the G-buffers.

A direct approach is impractical with current GPUs and several alternatives were proposed over time. Sloan *et al.* [66] used the screen space information to generate a set of spherical harmonics. These data structures were used to compute the indirect illumination. Soler *et al.* [57] proposed a similar approach to generate a sparse voxel volume with information from screen space. Nichols *et al.* [67], [68] proposed a segmentation process of the G-buffers to extract a set of VPLs that are splatted to compute indirect illumination. They applied this process on downsampled versions of the G-buffers to gather the VPLs into clusters and they proposed a hierarchical computation from bottom to top to accelerate the indirect lighting computations for the VPLs.

Ritschel *et al.* [69] proposed a more faithful approach to the path-tracing technique, called screen space directional occlusion (SSDO). They used a statistical approach to simulate the path-tracing process. This technique is described in Figure 2 Right. Each of the $\{A, B, C, D, E\}$ samples is randomly selected along a ray. The sampling rays are normally distributed inside the upper hemisphere of $\mathbf{x}$, similarly with the path-tracing process described above. They used these samples to approximate the 2D ray-tracing process inside the G-buffers information. As it can be seen from Figure 2 Right, the samples don't accurately represent the intersection points. Some samples are way further inside the geometry ($\{A, B, C, D\}$) and others ($\{E\}$) are outside the geometry. For this reason, the sampling pixels inside the G-buffers are not the same as in the path-tracing process. Each sample is projected into screen space and is tested whether it resides inside the geometry, otherwise it's discarded. The samples that are inside the geometry contribute to the indirect lighting computations.

F. RAY-TRACING HARDWARE ARCHITECTURE SOLUTIONS

Even if this paper doesn't analyze the ray-tracing hardware architecture solutions in depth, we decided to present their development and the current status of technology in the field of real-time global illumination.

The rasterization pipeline has long been the de facto technology used for real-time computer generated graphics. Moreover, the GPUs have incorporated the pipeline into their hardware and optimized and extended it at a fast pace. However, this pipeline has a major drawback because it does

not allow direct access to the global information of a scene. This problem is due to the restrictive and inflexible access to GPU memory. All the techniques presented in Section III-A offer superior results in terms of photorealism, but require access to global information from the scene. They are also based on calculations whose requirements exceed the capabilities of current GPUs. For this reason, the possibility of hardware acceleration of these calculations has been explored.

The first variants of hardware accelerators that proved practical were proposed for volumetric visualization [70]. These hardware accelerators were designed with a fixed pipeline and could query the intersection of rays with the geometry of the scene. They could simulate only the primary ray, without the possibility of recursive ray casts, required for the ray-tracing technique, and were used for the photon mapping technique. Subsequently, hardware accelerators with more flexible [71] and configurable pipelines [72] were proposed. Several hardware solutions have been proposed, but their analysis goes beyond the scope of this article.

The Nvidia company has recently developed a GPU architecture [73] capable of efficiently computing the intersection of rays with scene geometry. These GPUs have specialized cores for this process and use internal geometry management based on acceleration structures. From the point of view of the pipeline's API, a new set of shaders was introduced to create rays and to detect and process intersections.

The visual results of real-time global illumination techniques offered by these GPUs are superior to those of the rasterization pipeline. Also, the technology has proven that it can provide very good results for various graphic effects such as real-time shadows and reflections [74]. However, a real-time global illumination solution that uses entirely a path-tracing implementation has not proved practical for this generation of GPUs. Thus, hybrid approaches were chosen instead of a complete implementation of the path-tracing technique. These approaches use the rasterization pipeline to compute the direct illumination and use the ray-tracing hardware accelerators through various techniques to compute the indirect illumination. Several games are limited to cast a small set of short rays per pixel to produce the indirect diffuse illumination. This process is followed by a denoising approach, such as the one proposed by Chaitanya *et al.* [75] to eliminate the noise made by reduced sampling.

Hillaire [76] proposed another approach that uses an interactive update of textures in which the indirect illumination is cached. Andersson and Barré-Brisebois [77] proposed an adapted version of the radiosity technique. They used a dynamic set of surfels and computed the indirect diffuse lighting influence between each pair of those surfels. In order to obtain good performance, the surfels are created and deleted interactively and remain persistent when they are inside the camera frustum. Majercik *et al.* [78] proposed a DOM-based technique, with spherical harmonics in the grid cells. To obtain the radiance transfer in each cell, they used a set of ray casts into the scene geometry. They took advantage of the ray-tracing hardware accelerators developed by Nvidia to obtain a good performance for these ray casts. Several particular techniques have been used by companies from the game industry, but due to the performance of current GPUs [79], they all need to use hybrid approaches.

G. AUXILIARY PHOTOREALISTIC EFFECTS

In this section we present real-time techniques designed to generate particular effects in global illumination. We chose the following representative ones: the generation of glossy reflections (Section III-G1), shadows (Section III-G2), ambient occlusion (Section III-G3), subsurface scattering and translucency (Section III-G4), volumetric lighting (Section III-G5) and the simulation of area lights (Section III-G6).

For each chosen effect we present only the more relevant real-time techniques with an emphasis on those that have been designed or can be integrated with at least one of the four chosen classes of real-time global illumination.

1) GLOSSY REFLECTIONS

The generation of real-time reflections represents a vast field and many techniques have been proposed for different scenarios and at different levels of quality and performance. A comprehensive analysis of these techniques was performed by Szirmay-Kalos *et al.* [12].

The reflections can have different levels of glossiness, depending on the material properties of the surface on which the light is reflected. Ideally, a technique that produces glossy reflections should be able to produce perfect mirror reflections, because these results can be altered to obtain different levels of glossiness.

The de facto approach adopted by the game industry to generate reflections uses environment maps [80]. In particular localized environment maps are used, acquired by rendering the scene from the position of the object for which the reflections around it are computed. To obtain glossy reflections it is necessary to filter [81] the map. However, a major drawback of this approach is that the acquisition of the map is done from a single position. When the object for which the reflections are computed is large, a reflection direction corresponding to the surface of the object doesn't always produce the same result as the same direction from the position in which the map was acquired. To solve this problem, Brennan [82] and Bjorke [83] computed the intersection point of the reflection direction corresponding to the surface of the object with a proxy geometry, a sphere. Subsequently, they computed the direction from the acquisition point of the environment map to the previously obtained intersection point and used this direction to sample the map.

To obtain glossy reflections with DOM-based techniques, Kaplanyan [39] proposed to accumulate several spherical harmonics along the reflected direction. The voxel representation of the scene geometry can also be used to obtain reflections with a ray-casting process. For glossy reflections, Crassin *et al.* [59] showed that the voxel cone tracing technique is appropriate.

Another approach adopted by the game industry uses a ray-casting process in screen space. Sousa *et al.* [84] proposed for the first time to use a 2D ray-casting process to obtain the reflections, but the approach was discovered at the same time by several developers from the game industry. The class of these approaches is widely known as screen space reflections (SSR). Moreover, several optimizations were developed over time. McGuire and Mara [85] presented an efficient GPU implementation of the 2D ray-casting process. Uludag [86] proposed a special representation of the screen space information to accelerate the intersection test of the ray with the pixels. He created sequential downsampled maps of the depth information using a min filter. This structure was used to skip the pixels that are not intersected by the ray. To produce glossy reflections, Stachowiak and Uludag [87] used a Monte Carlo approach with several rays per pixel. To get a good performance, they used a small number of rays per frame and temporally filtered the result. The screen space information lacks significant geometry data, therefore several techniques that use multi-layer [88], [89] and multi-view [90]–[92] methods have been proposed. In particular, Lambru *et al.* [92] explored the possibility of using a 2D ray-casting process directly in the RSM pixels to produce reflections. A comprehensive analysis of the optimizations of this class of techniques was presented by Vasilakis *et al.* [93].

2) SHADOWS

The field of real-time shadows generation is vast and several comprehensive analyses of the proposed techniques have been performed [13], [14]. Eisemann *et al.* [17] also provided a very detailed analysis focusing on the techniques used by the game industry. Over time, the de facto techniques for real-time rendering have become the shadow mapping technique proposed by Williams [94] and the shadow volumes technique proposed by Crow [95]. A comprehensive analysis for all optimizations and variations of the shadow mapping technique was provided by Scherzer *et al.* [15]. For the shadow volumes technique, a similar analysis was done by Kolivand and Sunar [16].

The shadow generation techniques tend to fall into two major categories: those that can produce hard shadows or soft shadows. Both shadow mapping and shadow volumes techniques can produce hard shadows, but obtaining soft shadows is a big challenge for which different variations and adaptations of these techniques have been proposed.

For the particular cases of the chosen classes of global illumination, several techniques have been proposed. It is obvious that the RSM can be used to produce shadows, because it contains the z-buffer used in the shadow mapping technique. Unfortunately, the DOM-based techniques cannot produce shadows. These techniques use a data structure that stores information about the radiance reflected by the geometry of the scene. This information alone and at a low grid resolution cannot be used to extract information about the scene geometry. Therefore, there is no direct approach to producing shadows with this class of techniques. It is important to mention that some techniques in this class use an RSM, so for them, the shadows can be generated with the shadow mapping technique.

Shadows can be generated using the voxel representation of the scene with a ray-casting process [17]. However, this approach produces aliased shadows due to the coarse approximation of the scene geometry provided by the voxel volume. Nichols *et al.* [96] tried to solve this with an incremental refining process in screen space. They used a voxel representation of the geometry visible on the screen to produce shadows for area lights. Instead, Franke [62] used a cone-tracing process to produce shadows in mixed reality scenes. He traced a specialized shadow cone with a small aperture in the direction of the light to accumulate the amount of geometry in its path. Villegas and Ramírez [97] showed that this approach can produce soft shadows and solve the aliasing problem. Moreover, Kasyan [98] showed that this approach can also produce soft shadows for area lights.

Ritschel *et al.* [69] used a 2D ray-casting process directly in screen space to obtain the light occlusion with the geometry. This approach depends on the available information on the screen, which is not enough to provide shadows for all the geometry in the scene. The geometry that is not present on the screen cannot produce shadows on the screen, even if it should. In addition, there are temporal coherence problems for dynamic environments. To overcome these drawbacks, they explored the possibility of using multi-layer or multi-view approaches. However, this technique, as explained by them, can enhance the results of the shadow mapping technique. The latter introduces a visual artifact due to the discrete nature of the z-buffer which it is based on. The orientation of the light camera projection, used to acquire the z-buffer, introduces an error on the intersection process between the z-buffer pixels and the screen pixels. To solve this problem, a bias error is introduced to translate the z-buffer pixels further into the geometry, but this process introduces other types of artifacts according to the bias value. The 2D ray-casting process in screen space can solve this problem using a short ray to check the occluding geometry over short distances. The shadows produced by the shadow mapping technique can be enhanced for geometry details. The game industry adopted this approach known as contact shadows [17].

3) AMBIENT OCCLUSION

Ambient occlusion [99] offers an acceptable approximation of the rendering equation using a global method instead of a localized one. In a simple description, it computes how exposed the geometry is to the illumination of the scene. It darkens the enclosed parts of the scene according to the surrounding geometry and offers a more realistic image of the scene. This effect must be computed for the entire geometry around a point, to infinite distances, but such an approach has a significant impact on performance. Instead, Zhukov *et al.* [100] introduced the concept of obscurances, which computes occlusion with geometry only to finite distances.

Pharr and Green [101] proposed to pre-process the ambient occlusion for the geometry surfaces and cache the results in an additional texture. Unfortunately, this approach cannot take into account the occlusions received and made by dynamic objects. To solve this problem, Bunnel [102] introduced a real-time approach that uses an approximation of the scene geometry as a set of surfels. The ambient occlusion is computed using those surfels. Kontkanen and Laine [103] used a pre-computed discrete 3D grid to approximate the objects and introduced a real-time technique for inter-object ambient occlusion computation.

Shanmugam and Arikan [104] and Mittring [105] simultaneously developed techniques that use only the geometry visible on the screen. Several techniques [106]–[109] were later proposed to take advantage of the screen space information. However, these techniques provide results closer to the obscurances effect than to the ambient occlusion one. They use a sampling process in screen space, a process that is limited by the information available on the screen only in a small vicinity around the evaluated position. Shade *et al.* [110] introduced layered depth images that represent multiple renderings of the scene at specified distances along the observer's view direction. Bavoil and Sainz [111] used such a set of layers to compute the ambient occlusion. Nalbach *et al.* [112] introduced the concept of deep screen space which represents a set of surfels acquired from the observer position, that can be used to compute ambient occlusion. Vardis *et al.* [113] used a multi-view approach to improve the visual results. In particular, they also explored the possibility of using the RSM z-buffer to produce ambient occlusion.

Due to the flexibility of the voxel representation of the scene geometry, several techniques [59], [114]–[116] were proposed to compute the ambient occlusion using the voxel volume.

4) SUBSURFACE SCATTERING AND TRANSLUCENCY

To compute the subsurface scattering of the light inside the geometry of an object, Jensen *et al.* [117] introduced the bidirectional scattering-surface reflectance distribution function (BSSRDF) which represents a unified model for the approximation of the reflected and transmitted light. The BSSRDF depends on the topology and internal properties of the material, which requires information about the entire object, not just locally at the evaluated position.

To simulate this effect, Dachsbacher and Stamminger [118] introduced the concept of translucent shadow maps (TSMs). A TSM is similar to an RSM, but instead of the reflected flux, it stores the transmitted flux inside the geometry and is used at the shading pass to sample the lighting information. Børlum *et al.* [119] used a map similar to a TSM to inject the transmitted radiance inside a LPV. This modified LPV can be used to acquire the transmitted flux from inside an object. Di Koa and Johan [120] enhanced this technique to optimize the VPLs scattering with a ray marching process and used an additional voxel volume to accelerate the propagation pass.

A specialized solution for human skin was proposed by d'Eon *et al.* [121]. It approximates the subsurface transport of the light with a decoupled model that uses multiple layered Gaussian filters on texture space. Jimenez *et al.* [122] extended this technique in screen space. An extension for other translucent materials was later proposed by Jimenez *et al.* [123].

All the techniques previously presented are specialized for the subsurface scattering of the light, but cannot simulate translucent materials. For this effect, information from the whole scene is required. Jensen and Buhler [124] used the BSSRDF along with a ray-tracing sampling strategy to acquire such information. This approach follows a physically correct approximation and provides plausible results for both subsurface scattering and translucency. It represents the basis of the simulation that real-time global illumination techniques try to achieve. To approximate this approach, Mertens *et al.* [125] used a multi layered screen space to produce better results for subsurface scattering. Also, Nalbach *et al.* [112] proved that both subsurface scattering and translucency can be easily acquired with their proposal of deep screen space. Ganestam and Doggett [126] extended the SSR technique to simulate the refraction of light behind translucent objects with a 2D ray-casting process in the G-buffers pixels. Lambru *et al.* [92] proposed a similar process for RSM pixels. Eisemann and Décoret [53], [127] used a voxel representation of the scene with binary information per voxel to approximate the depth of geometry. This approach provides physically correct results for subsurface scattering of the light and can produce plausible results for semi-transparent materials like glass.

5) VOLUMETRIC LIGHTING

The volumetric lighting represents a particular case of a wider field that deals with simulating the scattering of light when it travels through different media that participate in the light transport. Such media are represented by, but not limited to, dust particles, smoke, atmosphere, fog or clouds. Cerezo *et al.* [19] conducted a comprehensive analysis of this field for offline techniques. The particular case of the volumetric lighting refers to the simulation of the visual effects known as crepuscular rays, sunbeams or light shafts. The transport medium in this simulation has an important impact on the scattering of the light, but the simulation of this effect through a general medium comes with its own difficulties.

Max [128] first proposed a technique to simulate the volumetric lighting. He intersected epipolar slices with shadow volumes to find the lit segments visible to the observer. Nishita *et al.* [129] extended this approach and proposed what is known as the air-light equation. They used light volumes in addition to shadow volumes and analytically integrated this equation with a ray-casting process. This model was later used by other approaches, adapted and improved for different scenarios. Wyman and Ramsey [130] proposed a ray marching process along the epipolar lines and checked the occlusion with geometry with the shadow volumes technique.

Tóth and Umenhoffer [131] proposed a faster approach that uses the shadow mapping technique to verify the occlusion with geometry. It is obvious that RSM-based techniques can also use this approach to produce volumetric lighting, because the RSM contains the z-buffer used in the shadow mapping technique.

Kajiya and Von Herzen [45] first used a spherical harmonics grid to compute the scattering of light in clouds, so it came naturally that this approach was used to produce volumetric lighting. Billeter *et al.* [132] extended the light propagation volumes technique to produce this effect. They used a modified LPV in which they injected the radiance that comes directly from the light source, together with a propagation strategy that incorporates a light scattering and decimation model. Unfortunately, because the LPV stores only low frequency light, the visual results provided have a lot of light bleeding outside the areas represented by the volumetric lighting, which creates a general fog effect.

Wyman [133] used a voxel volume to store binary information about the visibility of a voxel from the light source perspective. A ray marching process can be used to simulate the volumetric lighting. For performance reasons, Wyman built the voxel volume in epipolar space. This construction allows the placement of epipolar lines on a single row of voxels and facilitates the visibility check of a voxel. This approach proved to be flexible and was extended by Wyman and Dai [134] to simulate volumetric lighting for area light sources.

Several techniques [135], [136] have been proposed to use the screen space information to simulate volumetric lighting. All of them use a directional blur to extend an initial radiance introduced at the position of the light source. Even though these approaches are fast, they offer very little flexibility over the visual results provided and require the light source to be visible on the screen.

6) AREA LIGHTS

An area light is a light source characterized by a geometry. It emits light directly from all its surface. It might have more properties which describe the intensity of different zones of the geometry and the spreading behavior. The main focus of area light simulation is the emitted lighting coming from the surface itself to its surroundings. Heitz *et al.* [137] proposed an analytical approach to compute the emitted illumination when an area light is represented by a simple geometric shape, like a rectangle or a disk. Kuge *et al.* [138] used a layered approach to approximate more complex geometry, but they also used an initial approximation of the entire geometry in the form of simple shapes. Over time, several techniques have been proposed to approximate the geometry of area lights in various forms for analytical evaluation, but their analysis goes beyond the scope of this article.

Due to the fact that the emitted illumination of an area light behaves similarly to the light reflected by its surface, many techniques designed to compute indirect illumination can simulate the area lights. In particular, any approach based on the instant radiosity technique, which uses a gathering process of the VPLs, can simulate area lights. The only change required is the generation of additional VPLs for the emitted illumination of the area lights. A major problem with such an approach is the real-time generation of these VPLs, especially for dynamic area lights. One technique that partially solves this problem is SSDO, which samples the VPLs from the screen space. The only change required is to take into account the emissive component when computing the direct illumination of the geometry visible on the screen. Unfortunately, the RSM-based techniques cannot be used for such an approach because the RSM cannot be generated for area light sources.

The approach described above for instant radiosity techniques can be extended to other approaches, as long as the emitted illumination of the area lights is used as an additional source of information. Kaplanyan and Dachsbacher [48] showed that the light propagation volumes technique can simulate area lights by injecting in the LPV a set of VPLs that approximate the emitted illumination of these lights. Di Koa *et al.* [139] extended this approach and used a Poisson sampling process to obtain the initial set of VPLs. Villegas and Ramírez [97] showed that the voxel cone tracing technique can also simulate the area lights by inserting the light emitted by their geometry into the voxels during the voxelization process.

IV. IMPLEMENTATION

For each class of real-time global illumination techniques described in the previous section, except for the one based on ray-tracing hardware solutions, we have developed our own implementation. These implementations closely follow the original descriptions and where possible, introduce a certain degree of optimization. In addition to the initial descriptions that are usually limited to the production of indirect diffuse illumination and glossy reflections, we have tried to implement, where possible, the auxiliary photorealistic effects described above. In this section we present these implementations and highlight the key differences to the initial descriptions of the authors who proposed them.

A. THE IMPLEMENTATION FRAMEWORK

All techniques were implemented in the same framework. The decision to develop such a framework resulted from the necessity to compare the techniques using the same external parameters (scene geometry, material properties, camera information) for more accurate quantitative results. The framework was intended from the beginning to be flexible and to allow the rapid implementation of the global illumination techniques. Also, the possibility to reuse functionalities was an objective. Due to these requirements, a multi-pass rendering architecture was implemented, as shown in Figure 4, where every pass has an output that can be propagated further to any subsequent pass. This architecture allows the reuse of passes in several implementations. In Figure 4, the

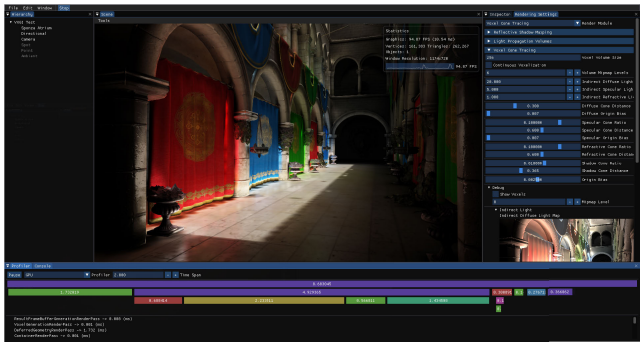


FIGURE 3. A screenshot of the framework used to implement all classes of real-time global illumination techniques discussed in this paper. The editor of the framework can be seen with most of its debug and development windows. The scene contains the Sponza Atrium model.

G-buffers Generation pass is the same for the first three global illumination techniques.

Also, the implementation framework provides real-time debug functionalities. It facilitates the possibility to view and modify the parameters in the GPU memory through the user interface (UI). This is very important from a development point of view, because it gives the possibility to quickly test various parameters. This can be seen in the right window in Figure 3. Moreover, it allows the visualization of debug information. It provides access to GPU texture visualization in the UI. Any output information from a rendering pass can be visualized. Furthermore, the framework facilitates performance evaluation and offers a visual CPU and GPU profiler. It can be seen in the bottom window in Figure 3. It also offers functionalities similar to those of a modern visual editor, like interactive object manipulation, information persistence and assets management.

Unfortunately, this approach has a disadvantage in terms of performance because the multi-pass rendering architecture requires a high level management. However, the initial requirements of the framework to implement all the techniques in the same sandbox and to compare them using the same external parameters have been met.

The framework was implemented using the C++ programming language with the OpenGL 4.6 graphics API.

B. REFLECTIVE SHADOW MAP-BASED TECHNIQUES

From the class of RSM-based techniques, we chose to implement the sampling strategy proposed by Dachsbacher and Stamminger [36] to compute the indirect diffuse illumination. The glossy reflections were obtained with the 2D ray-casting process into the RSM pixels used by Lambru *et al.* [92]. To produce shadows, we used the shadow mapping technique with the RSM z-buffer and for the ambient occlusion we applied the sampling strategy proposed by Mittring [105] on the RSM buffers. To render translucent objects we applied again the approach proposed by Lambru *et al.* [92] which uses a 2D ray-casting process in the RSM pixels. The volumetric lighting was produced with the technique proposed by Tóth and Umenhoffer [131].

Unfortunately, the disadvantage of using a single subset of samples instead of the entire RSM resolution is that the indirect lighting evaluation doesn't produce physically correct results. A large set of samples helps, but it has a high negative impact on performance, as previously discussed. To overcome this problem, we implemented the screen space interpolation approach proposed by Dachsbacher and Stamminger [36]. They used a preliminary pass that computes the indirect lighting contribution at a lower resolution than that of the screen with a large set of samples. This result is later used as a source of information when computing the indirect illumination at full screen resolution. The pixels which have similar normal vectors and are not far apart from the corresponding pixels in the lower resolution buffer use the already computed indirect illumination. This process proved to improve the performance at a cost of an acceptable degree of blurriness on the indirect illumination result.

Another problem was the difference between the visual results provided by the evaluation of two different sets of samples. Moreover, the intensity of the indirect illumination is directly influenced by the chosen samples, so that large discrepancies in brightness can be obtained between the results from two different sets of samples. To solve this problem, a quasi-random algorithm [140] was used to generate the same sample set every time and also to preserve the low-discrepancy between the samples.

To obtain soft shadows with the shadow mapping technique we used a filtering strategy [141]. In addition, we took advantage of the shadow sampler types offered by the OpenGL 4.6 standard to improve the performance.

The sampling strategy to produce ambient occlusion cannot be applied with a single sampling pattern for the entire screen because visual artifacts appear. To solve this problem, we rotated the sampling pattern, similar to Mittring [105], using a kernel of 4×4 rotations in screen space. To eliminate the noise provided by such an approach we applied a Gaussian blur with a 5×5 size kernel.

Another change was made to the RSM generation. The approach proposed by Lambru *et al.* [92] computes the light refracted by a translucent object after it has been reflected by the scene surfaces. This is done with a 2D ray-casting process in the RSM pixels. The translucent objects can cover a fairly large area in the RSM, so a lot of information about opaque objects, that are the source of light reflection isn't available in the RSM. To avoid such a problem, we didn't render the translucent objects when the RSM was generated. More details on this approach are described in Section V-E.

The rendering pipeline of the implementation is described in Figure 4a. The RSM Sampling Pattern represents a set of 2D samples with coordinates in the $[-1, 1]$ interval. For temporal coherency, the sample set is the same for all frames. The AO Sampling Pattern represents a set of samples uniformly distributed inside a hemisphere of radius 1. The same set was also used for all frames. The RSM Generation pass renders the scene from the position of the light and acquires

the buffers needed for the indirect lighting computation: world space positions, world space normal vectors, the radiance flux and the z-buffer. The G-buffers Generation pass is similar to the previous one and renders the scene from the observer position to acquire the following G-buffers: view space positions, view space normal vectors, material properties and the z-buffer. The G-buffers Mipmap Generation pass creates the mipmap pyramids of all the G-buffers obtained in the previous pass. The RSM AO Generation pass and the Gaussian Blur pass implement the ambient occlusion technique described above. All these passes produce the information source for the direct and the indirect lighting computations, which are done in the RSM Interpolated Indirect Lighting Computation pass and the RSM Lighting Computation pass. The last two passes implement the screen space interpolation approach previously presented. The RSM Interpolated Indirect Lighting Computation pass acquires the indirect illumination at a lower resolution than the screen with the previously produced mipmaps. The RSM Lighting Computation pass is used to compute both the direct and the indirect illumination at the full screen resolution using the low resolution version of the indirect lighting buffer computed in the previous pass. In addition, this pass computes glossy reflections, shadows and volumetric lighting and uses the AO Buffer to produce the final image.

C. DISCRETE ORDINATE METHOD-BASED TECHNIQUES

From the class of DOM-based techniques, we chose to implement the light propagation volumes technique to compute indirect diffuse illumination and glossy reflections. The implementation follows the initial description and the extensions proposed by Kaplanyan and Dachsbacher [48]. To render translucent objects we used an approach similar to Børlum *et al.* [119], but we used the already available LPV which contains the light reflected by the scene surfaces, used to compute the indirect illumination. To simulate the area lights, we initially inserted in the LPV a set of VPLs that describe the emitted illumination of these lights.

The initial description of the light propagation volumes technique introduces a visual artifact called *light bleeding*. The LPV Propagation pass, described in Figure 4b, attempts to propagate the initial radiance inside the whole volume, but doesn't take into account the occlusion with the geometry. This approach allows the radiance to be propagated through walls and introduces lighting in areas where it's not physically possible. However, even if this artifact can provide unpleasant visual results, it can sometimes provide plausible results represented by over-illumination of some areas of the scene. The proposed extensions of Kaplanyan and Dachsbacher [48] prove to partially solve this visual artifact. They used a geometry occlusion volume, which contains the scene geometry, probabilistically approximated as a grid of spherical harmonics. The process of constructing the occlusion volume from the whole scene geometry is not efficient enough and is usually not required. The areas that need the most attenuation are the ones which are visible by the observer, so the only scene geometry which was included in the geometry occlusion volume is the one from the RSM and the G-buffers.

The glossy reflections were obtained with a ray marching process in the reflected direction, as proposed by Kaplanyan [39].

To compute the light refracted by translucent objects after being reflected by the scene surfaces, we modified the RSM generation process so as not to render the translucent objects for the same reasons described in the previous section. At the shading pass, we used a ray marching process of spherical harmonics, similar to that of obtaining the glossy reflections, in the refracted direction.

To generate the initial set of VPLs that approximate the emitted illumination of area lights, we used a stochastic sampling process on the surface of these lights. For each sample, a VPL is created with the position, normal vector and radiance that correspond to the position of the sample on the surface. These VPLs are subsequently injected into the LPV.

One important problem was the LPV cell data precision. The LPV Radiance Injection pass, described in Figure 4b, requires atomic operations. Unfortunately, the OpenGL 4.6 standard doesn't accept atomic operations on float data types. For this reason, the LPV cell can't store an RGBA32F data type. It could store an R32F thanks to intelligent type castings, but this approach offers only 8 bits per spherical harmonic coefficient, which are not enough. To solve this problem, a total of 4 different LPVs were used, one LPV with R32F per cell for each coefficient.

The rendering pipeline of this technique is described in Figure 4b. The RSM Generation pass and the G-buffers Generation pass are described in the previous section. The Area Lights Geometry Voxelization pass implements the acquisition process of the VPLs that approximate the area lights, described above and injects these VPLs into the LPV. The LPV Radiance Injection pass extracts the VPLs from the RSM and inserts them into the LPV. The Geometry Injection pass implements the extension proposed by Kaplanyan and Dachsbacher [48], presented above. It takes the RSM and the G-buffers and extracts the attenuation spherical harmonics from the existing geometry in those buffers. The actual process is split in two sequential passes, one for the RSM and one for the G-buffers, but for simplicity it is described as one pass. The LPV Propagation pass is an iterative process of N iterations. It takes each cell from LPV and gathers the radiance from the neighboring cells. To occlude the propagation between two adjacent cells separated by geometry, the geometry volume is used. This pass uses the LPV created by the LPV Radiance Injection pass for the first iteration and for the next ones it uses the LPV created in the previous iteration. This process is implemented on the GPU and uses a compute shader to process the LPV. The LPV Lighting Computation pass takes the LPV and the G-buffers and computes both the direct and the indirect illumination. Also, at this pass, the translucent objects are rendered.

D. VOXEL-BASED TECHNIQUES

From the class of voxel-based techniques, we chose to implement the voxel cone tracing technique to produce indirect diffuse illumination, glossy reflections, shadows, ambient occlusion, subsurface scattering, translucency and to simulate area lights.

Using a single color information per voxel introduces the light bleeding artifact. Any voxel that is illuminated only on one face stores the lighting information for the entire voxel. This approach can bleed the light through thin walls that are illuminated only on one side. Any zone of the geometry that is voxelized into a single layer of voxels is potentially affected by this artifact. Moreover, the visual impact of this artifact increases as the resolution of the voxel volume decreases. A low resolution generates more areas affected by this effect and produces a higher intensity of indirect lighting. This effect appears as a better illumination result, but in fact it is just an error which provides a physically incorrect result. To solve this problem, Crassin *et al.* [59] proposed an anisotropic representation of the voxel volume with 6 color information per voxel, one for each main direction. In the shading pass, the voxel color is obtained with an interpolation between the three closest voxel directions to the cone direction. Unfortunately, this approach has a negative impact on the memory consumption, therefore it is not used for the highest mipmap level at full resolution. Instead, this approach is used for the second mipmap level to create 6 versions that are linearly filtered to obtain the subsequent mipmap levels.

To compute the light refracted by the translucent objects after it was reflected by the scene surfaces, we stored the transparency of the geometry in each voxel. At the shading pass, we accumulated the geometry from the refracted direction. Similar to the previous sections, the translucent objects were not rendered when the RSM was generated. Instead, the albedo of the voxelized geometry was associated with the voxels for a coarse approximation of the transmitted radiance inside the translucent objects. More details about this approach are presented in Section V-E. In addition, to simulate the area lights, the light emitted by the voxelized geometry was associated to each voxel.

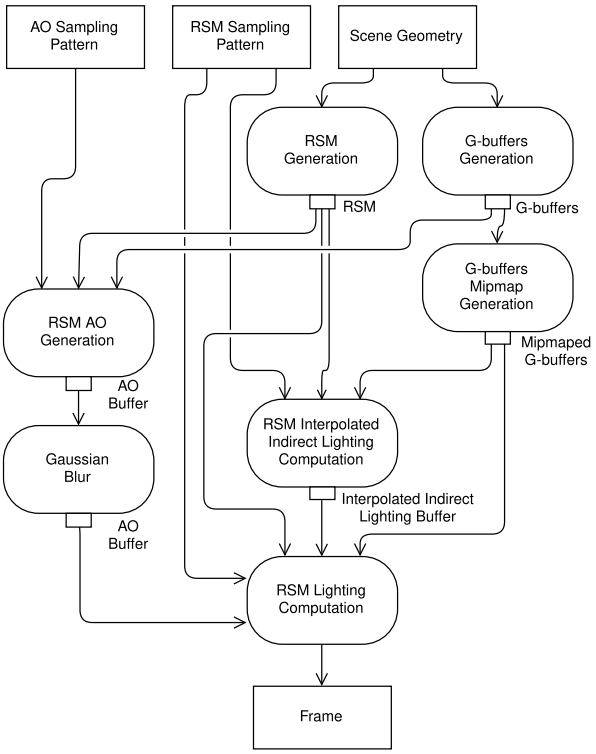
The pipeline of this technique is described in Figure 4c. The RSM Generation pass and the G-buffers Generation pass are described in the previous sections. The Voxelization pass uses the approach proposed by Crassin *et al.* [56] to acquire a voxel representation of the scene. The Voxel Volume generated at this pass represents the highest mipmap level of the anisotropic volume presented above. It is represented by a 3D texture with a single color information per voxel. The Voxel Radiance Injection pass inserts the reflected radiance flux of each pixel from the RSM in the corresponding voxel inside the Voxel Volume. The Voxel Anisotropic Mipmap Generation pass creates anisotropic downsampled versions of the Voxel Volume, as previously explained. This pass is implemented on the GPU and uses a compute shader to process the Voxel Volume. The Voxel Lighting Computation pass takes the Voxel Volume and the G-buffers and computes direct illumination, indirect diffuse illumination, glossy reflections, shadows, ambient occlusion, subsurface scattering and translucency.

E. SCREEN SPACE TECHNIQUES

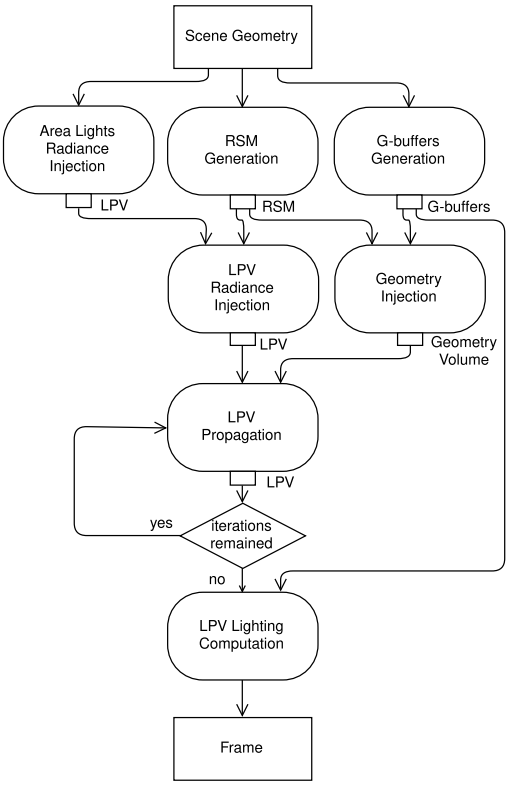
From the class of screen space techniques, we chose to implement the screen space directional occlusion technique to compute the indirect diffuse illumination and to simulate the area lights. To produce glossy reflections, we used the SSR technique and for the shadows we used a ray casting process in screen space. The ambient occlusion was produced using the screen space sampling strategy proposed by Mittring [105], previously detailed in Section IV-B. The translucent objects were rendered with an extension of the SSR technique, known as screen space refraction [126]. Volumetric lighting was produced with the technique proposed by Mitchell [135].

In order to compute the light refracted by translucent objects using the information available in screen space, it is necessary to obtain the geometry behind the translucent objects. A simple option for this approach is to generate two sets of G-buffers, one for opaque objects, which is used in the screen space refraction technique and one for translucent objects for which the refraction of light is computed. More details about this approach are presented in Section V-E.

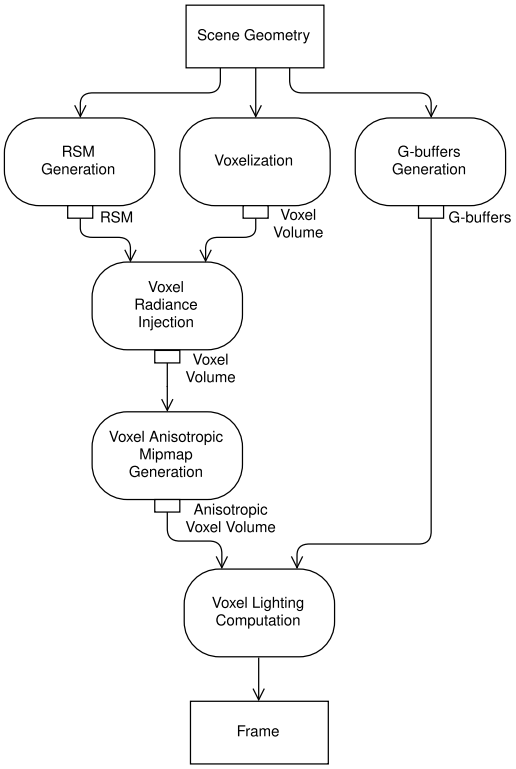
The pipeline of the implementation is described in Figure 4d. The SSDO Sampling Pattern represents a set of samples uniformly distributed inside a hemisphere of radius 1. For temporal coherency, the sample set is the same for all frames. The G-buffers + Transparency G-buffers Generation pass is similar to the G-buffers Generation pass, described in the previous sections, except that it generates two sets of G-buffers, one for opaque objects and one for translucent objects. The G-buffers Mipmap Generation pass is applied only to the set of G-buffers for the opaque objects and is the same as in the previous sections. The SSAO Sampling Pattern, the SSAO Generation pass and the Gaussian Blur pass are used to implement the ambient occlusion technique proposed by Mittring [105] and whose pipeline was previously described in Section IV-B. The Direct Lighting Computation pass generates a buffer with the direct illumination for the opaque geometry visible on the screen. This is the only global illumination implementation for which it is impossible to compute the direct and the indirect illumination at the same pass. The SSDO technique uses the direct lighting information as the source to calculate the indirect illumination of the second bounce of the light. The SSDO Interpolated Indirect Lighting Computation pass and the SSDO Lighting Computation pass implement the screen space interpolation approach proposed by Dachsbacher and Stamminger [36] and described in Section IV-B. The sampling process in both passes takes the Direct Lighting Buffer and the SSDO Sampling Pattern and acquires the indirect diffuse illumination for each pixel. Furthermore, the SSDO Lighting Computation pass uses the resulted indirect lighting information,



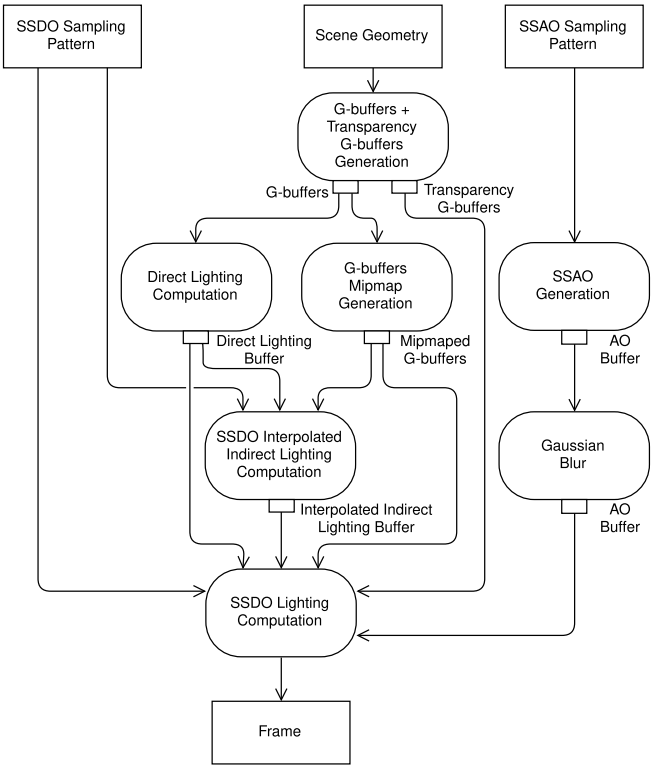
a) RSM-based global illumination



b) DOM-based global illumination



c) Voxel-based global illumination



d) Screen space global illumination

FIGURE 4. **Rendering pipelines of all real-time global illumination implementations.**

TABLE 1. Comparative analysis of the classes of real-time global illumination techniques, based on qualitative criteria.

Global illumination technique class	Produces indirect diffuse illumination	Produces glossy reflections	Produces shadows	Produces ambient occlusion	Produces subsurface scattering & translucency	Produces volumetric lighting	Simulates area lights
RSM-based GI	**Yes**	*Partially*[1,3]	**Yes**	*Partially*[1]	*Partially*[1,4]	**Yes**	No
DOM-based GI	**Yes**	*Partially*[3]	No	No	*Partially*[3,4]	No[4]	**Yes**
Voxel-based GI	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	No[4]	**Yes**
Screen space GI	*Partially*[1,2]	*Partially*[1]	*Partially*[1,2]	**Yes**	*Partially*[1,4]	*Partially*[5]	*Partially*[1,2]

[1] The technique class partially lacks the necessary information for the effect.
[2] The technique class has problems of temporal coherence for dynamic environments when the effect is produced.
[3] The technique class lacks high frequency information, required for the effect.
[4] There are several approaches available to produce the effect, but they require significant modifications of the technique class.
[5] The effect is produced with inflexible approximations designed for particular cases.

the Direct Lighting Buffer, the G-buffers and the Transparency G-buffers to compute indirect diffuse illumination, glossy reflections, shadows, translucency and volumetric lighting. Together with the AO Buffer, generated at a previous pass, it creates the final image.

V. COMPARATIVE ANALYSIS BASED ON QUALITATIVE CRITERIA

In this section, we analyze the possibility of the chosen classes of real-time global illumination techniques to produce the photorealistic effects that we have chosen as comparison criteria: indirect diffuse illumination (Section V-A), glossy reflections (Section V-B), shadows (Section V-C), ambient occlusion (Section V-D), subsurface scattering, translucency (Section V-E) volumetric lighting (Section V-F) and the simulation of area lights (Section V-G).

Our assessments for the ability of each class of techniques to produce these effects are presented in Table 1. We considered that a class of techniques can fully produce an effect, marked with **Yes** in the table, if it provides complete visual results at an acceptable level of detail, otherwise it can only partially produce the effect. For each scenario in which an effect can only be partially produced, we marked the problems in the table and described them in the sections below. Additionally, we offer visual results of our implementations for each approach and analyze them from a qualitative point of view.

A. INDIRECT DIFFUSE ILLUMINATION

The diffuse component of the indirect illumination doesn't require to be very precise, so it can be computed using a lower resolution than that of the screen. Unfortunately, this is not true for the high frequency details visible on the screen, where the blurriness created by a low resolution can produce unpleasant visual results. The screen space interpolation approach proposed by Dachsbacher and Stamminger [36] removes the blurriness for these high frequency details. This approach can be used with any shading technique at the cost of an acceptable blurriness only for low frequency details. However, even if this blurriness is visible for the visual results provided by the indirect diffuse component alone, when these results are combined with those provided by the direct illumination, the blurriness becomes less visible. Furthermore, another problem is that it's not always possible to achieve the desired level of light intensity, so a light intensity scale factor (f_s) is required.

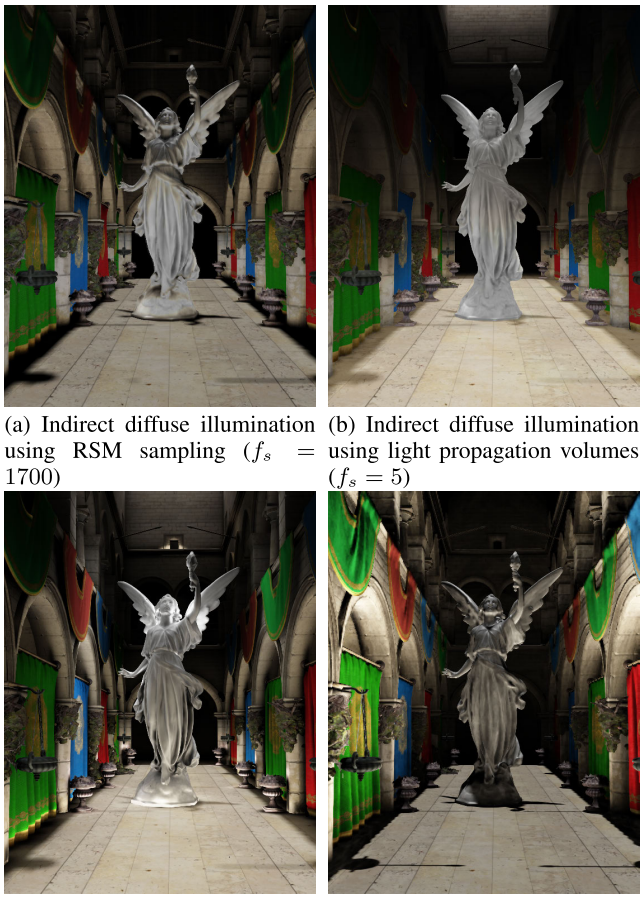
The RSM can be used to produce indirect diffuse illumination using any sampling approach. Several sampling strategies were proposed, as described in Section III-B. We chose to implement the sampling strategy proposed by Dachsbacher and Stamminger [36] along with the screen space interpolation approach. The visual results can be seen in Figure 5. The advantage of this technique is that it does not produce any light bleeding, as it can be seen, compared to the other techniques. Unfortunately, this technique provides only a single bounce of light, so it is not possible to indirectly illuminate the floor in Figure 5. However, it is possible to obtain further light bounces using the approach proposed by Ritschel *et al.* [42]. They extended the initial proposal of ISMs, used to compute the occlusion for VPLs. This approach requires the generation of low resolution RSMs for each VPL chosen from the initial RSM. These low resolution RSMs can be sampled to compute the indirect illumination for the second bounce of light. This process can be applied iteratively to obtain the i-th bounce of light by generating low resolution RSMs for the VPLs created and evaluated in step $i-1$. Unfortunately, this approach has a negative impact on performance, but could theoretically provide an infinite number of light bounces.

One important problem is the presence of a visual effect called *banding artifact*. This is described as the presence of several bands of pixels with similar values of indirect illumination, which appear along the directions from which the light comes. This phenomenon occurs because all pixels that are along the same direction of light are projected in the same position in the RSM. Therefore, for all these pixels the same RSM samples are chosen and evaluated, so that almost the same value of the indirect illumination is obtained. Moreover, a large discrepancy may occur between the indirect illumination values for two close bands when the sampling set is not large enough.

The light propagation volumes technique was designed to produce indirect diffuse illumination with a single spherical harmonic evaluation. The spherical harmonics can store information with any degree of precision, but due to memory constraints, the low frequency diffuse lighting is the only component suitable for this data structure. The visual results can be seen in Figure 5b. Unfortunately, the results are highly dependent on the resolution of the LPV and on the number of propagation iterations. A high resolution LPV requires additional iterations to propagate the radiance, compared to a low resolution one. However, even if a high resolution and a large number of propagation iterations have a negative impact on performance, they can greatly improve the visual results. The disadvantage of a low resolution LPV is represented by the presence of the light bleeding artifact. Due to a low resolution, the size of a cell is bigger, so that a spherical harmonic must approximate the light transport for a larger size. Thus, the accuracy of the light transport becomes lower and usually an over-illumination of the scene occurs because the radiance is introduced in areas where it shouldn't be present. It is possible to obtain multiple bounces of light. Kaplanyan and Dachsbacher [48] proposed a modification of the propagation pass in order to obtain a coarse approximation of the indirect illumination after several bounces of light. The geometry volume used for occlusion checks in the propagation pass is used as a bounce media inside the LPV. Unfortunately, this approach requires even more propagation iterations, as these bounces occur during the propagation process and require additional iterations to reach the rest of the cells in the LPV.

The voxel representation can be easily used to compute the indirect diffuse illumination. There are several techniques that gather information from the voxel volume, as presented in Section III-D. We chose to implement the voxel cone tracing technique. The visual results can be seen in Figure 5c. Unfortunately, the light bleeding artifact is present, due to the use of a single color information per voxel at the highest mipmap level. It is possible to obtain multiple bounces of light using an approach that accumulates the indirect illumination of several light bounces directly inside the voxel volume. This approach offers only a coarse approximation of indirect illumination, but it can theoretically provide an infinite number of light bounces.

The screen space directional occlusion technique was designed to produce indirect diffuse illumination. Unfortunately, the screen space nature of this approach produces several artifacts. It cannot provide acceptable visual results when the direct lighted areas, used as the sources of the indirect lighting, cover small areas on the screen. This artifact also appears for a large number of samples and produces different degrees of noise. This problem is visible in Figure 5d. However, this approach could produce acceptable visual results when the direct lighted areas are large enough. Unfortunately, a major drawback is the lack of temporal coherence. The indirect illumination changes when the camera moves and the direct lighted areas are not consistent in two consecutive frames. It is possible to theoretically obtain an infinite number of light bounces with multiple iterations of this technique, so that at each iteration, the input is the output of the previous iteration.



(a) Indirect diffuse illumination using RSM sampling ($f_s = 1700$)

(b) Indirect diffuse illumination using light propagation volumes ($f_s = 5$)

(c) Indirect diffuse illumination using voxel cone tracing ($f_s = 20$)

(d) Indirect diffuse illumination using SSDO ($f_s = 10000$)

FIGURE 5. Indirect diffuse illumination comparison. Stanford Lucy and Sponza Atrium models rendered with direct illumination together with the indirect diffuse illumination, shadows and ambient occlusion.

B. GLOSSY REFLECTIONS

As presented in Section III-G1, glossy reflections can be obtained by modifications of techniques that obtain mirror reflections. However, in order to obtain perfect mirror reflections, high frequency lighting must be used, which leads to a significant impact on performance. Moreover, the lower the lighting frequency, the lower the level of glossiness obtained and the visually closer it is to the diffuse reflections.

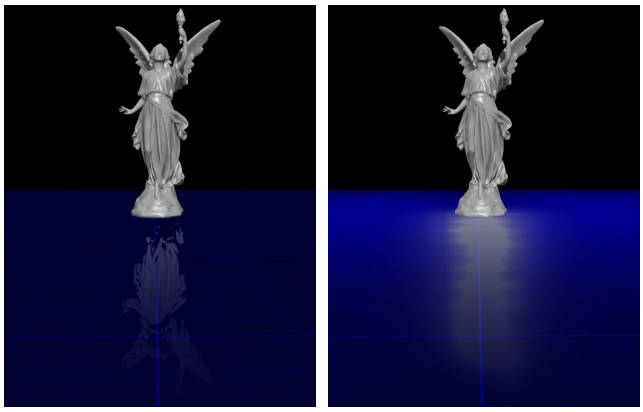
The RSM can be used to approximate the glossy reflections. Lambru *et al.* [92] explored the possibility of using a 2D ray-casting process directly on the RSM to acquire them. The visual results can be seen in Figure 6a. The disadvantage of this technique is that the results are highly dependent on the information visible from the position of the light. The geometry that is not present in the RSM cannot be reflected. This problem is visible in Figure 6a, where Lucy's reflection

is only partially obtained, because not all of its geometry is present in the RSM. It is theoretically possible to obtain an infinite number of bounces with successive 2D rays cast inside the RSM.

The LPV stores low frequency details, due to the use of spherical harmonics. However, it is possible to acquire a coarse approximation of the glossy reflections. Kaplanyan [39] proposed to accumulate several spherical harmonics along the reflected direction. The visual results of this approach can be seen in Figure 6b. However, the visual results of the glossy reflections are close to those of the diffuse reflections. Moreover, their calculation introduces a certain level of self illumination. These phenomena occur due to the fact that each spherical harmonic in the LPV covers a large volume and cannot provide a high level of accuracy for the entire geometry that is present inside this volume. To improve the visual results, a high resolution of the LPV or a layered approach [48] can be used, but they come with an impact on performance. Using the same approach described in the previous section, the approach that computes several bounces of light for the indirect diffuse illumination inside the LPV, it is possible to obtain multiple bounces of light for the indirect specular illumination.

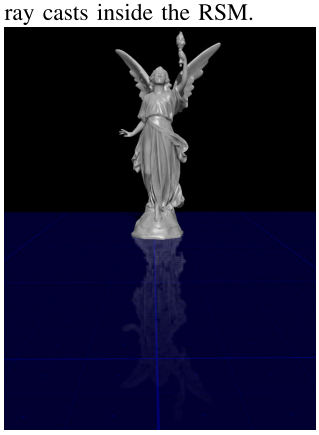
The voxel representation contains a detailed geometry. To obtain a mirror reflection, a ray-casting process can be used inside the voxel volume at the highest resolution. The visual results of this approach can be seen in Figure 6c. These results can be altered later to obtain different levels of glossiness. It is possible to obtain several light bounces using the same approach described in the previous section, the approach that accumulates the indirect illumination after several light bounces directly inside the voxel volume.

The screen space information offers the most detailed geometry compared to the other data structures. The glossy reflections can be computed using a 2D ray-casting process in the G-buffers pixels, after the computation of the direct lighting, as proposed by Sousa *et al.* [84]. The visual results of this approach can be seen in Figure 6d. Unfortunately, the results are highly dependent on the geometry that is visible on the screen. The geometry that is not visible from the observer position cannot be reflected. However, this approach could offer acceptable results and can be controlled when the camera moves, so that the temporal coherence is preserved. Also, it has a minor impact on performance. It is visible in Figure 6 that this approach provides the sharpest results compared to the other techniques. It is theoretically possible to obtain an infinite number of light bounces with several iterations of this technique, as described in the previous section.

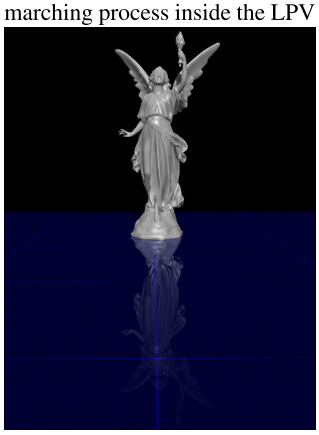


(a) Glossy reflections using 2D ray casts inside the RSM.

(b) Glossy reflections using a ray marching process inside the LPV



(c) Glossy reflections using a ray-casting process inside the voxel volume



(d) Glossy reflections using SSR

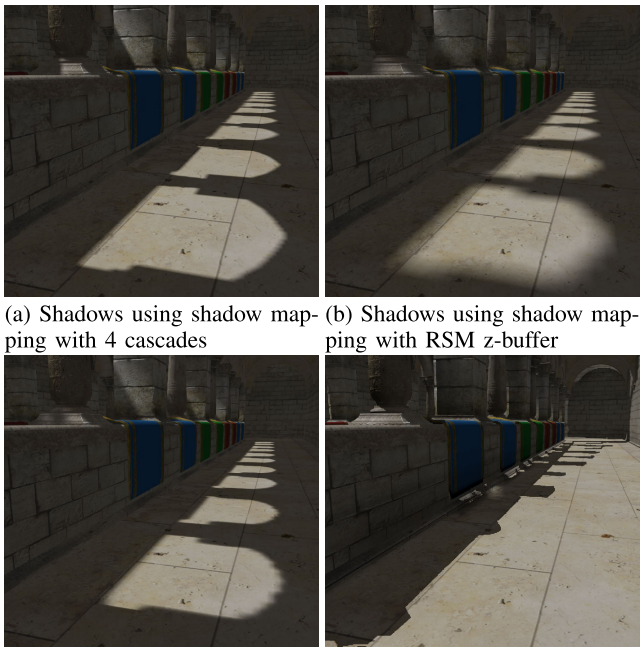
FIGURE 6. Glossy reflections comparison. The scene is rendered using direct illumination together with glossy reflections affected by the Fresnel effect.

C. SHADOWS

The z-buffer in the RSM can be used to obtain shadows with the shadow mapping technique. The visual results can be seen in Figure 7b. Unfortunately, for good shadows quality, the z-buffer needs to have a high resolution, therefore all maps of the RSM must have the same high resolution even if it is not required for all. However, a low z-buffer resolution may provide acceptable visual results. As presented in Section III-G2, the DOM-based techniques cannot produce shadows by themselves. In particular, the light propagation volumes technique uses the RSM as the source of information for the LPV, so it can produce the same results as the RSM-based techniques.

Shadows can be generated using the voxel representation of the scene as presented in Section III-G2. The visual results of using a cone-tracing process in the voxel volume can be seen in Figure 7c. Unfortunately, these results are highly dependent on the resolution of the voxel volume. Moreover, a high resolution provides good visual results, but has a negative impact on performance and memory consumption. It can be seen that the shadow obtained with this approach (7c) has a slightly different shape than the one obtained with shadow mapping (7a). This "boxy" shape occurs due to the low resolution of the voxel volume.

The screen space cannot always provide all the information needed to produce shadows. Ritschel *et al.* [69] proposed a ray-casting process in screen space, but this approach may produce incomplete and unusable results, as can be seen



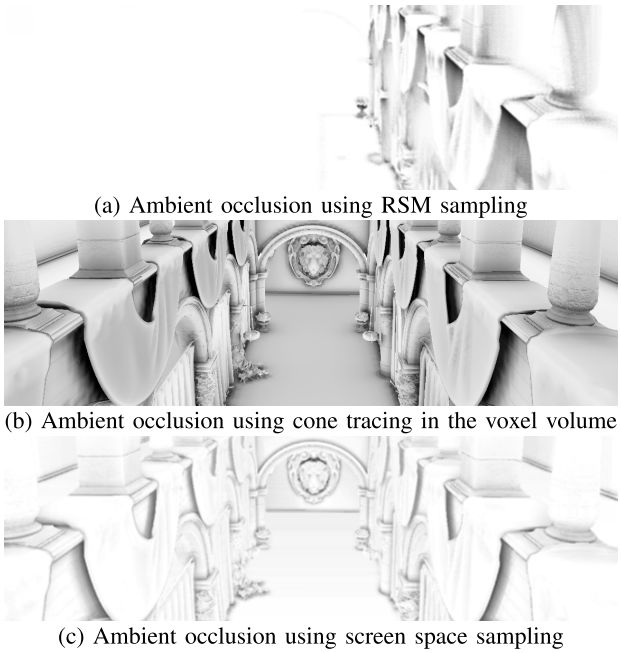
(a) Shadows using shadow mapping with 4 cascades

(b) Shadows using shadow mapping with RSM z-buffer

(c) Shadows using a cone-casting process in the voxel volume

(d) Shadows using 2D ray casts in screen space

FIGURE 7. Shadows comparison.



(a) Ambient occlusion using RSM sampling

(b) Ambient occlusion using cone tracing in the voxel volume

(c) Ambient occlusion using screen space sampling

FIGURE 8. Ambient occlusion map comparison.

in Figure 7d and explained in Section III-G2. However, this approach can be used to improve the shadow mapping technique.

D. AMBIENT OCCLUSION

The RSM can be used to provide limited results for the ambient occlusion. Vardis *et al.* [113] explored the possibility to use the RSM z-buffer to enhance the ambient occlusion map produced with screen space techniques. They proved that the geometry visible from the light position can offer an acceptable degree of precision to compute ambient occlusion, but the RSM cannot cover all areas visible to the observer, so it cannot provide the entire ambient occlusion map by itself. The results produced with the sampling strategy proposed by Mittring [105] on the RSM can be seen in Figure 8a. Similar to shadow generation, discussed in the previous section, the DOM-based techniques cannot provide information about the scene geometry. For this reason, the techniques in this class cannot produce ambient occlusion.

The voxel-based techniques can simply use a ray marching algorithm to approximate the amount of geometry that exists in the upper hemisphere of a point. Several variants of this approach were proposed, as presented in Section III-G3. The ambient occlusion map produced with the technique proposed by Crassin *et al.* [59] can be seen in Figure 8b. Their approach uses the cone-tracing strategy inside the voxel volume to obtain the amount of geometry that resides in the upper hemisphere. To approximate the entire hemisphere, 5 cones with wide apertures are enough. However, this approach cannot produce plausible results for short-distance occlusion due to the coarse approximation provided by the voxel volume. For this reason, it requires much longer distances and the results are considerably darker than the other approaches visible in Figure 8, which are limited to short distances.

Several techniques have been proposed to produce ambient occlusion using the screen space information, as presented in Section III-G3. The results obtained with the sampling strategy proposed by Mittring [105] can be seen in Figure 8c. Unfortunately, this particular technique has some shortcomings. It produces noise due to stochastic sampling, so it requires a subsequent filtering process. Because the ambient occlusion is not viewed directly, expensive filtering that preserves the edges, such as bilateral filtering, isn't required. Instead, a Gaussian blur is enough. The resulting blurriness can be seen in Figure 8c, but in general it is no longer visible in the final image. In addition, the sampling is limited to a very small vicinity around the evaluated position. Even if the vicinity is enlarged, the results don't improve because not all the geometry between the evaluated position and the sample is taken into account.

E. SUBSURFACE SCATTERING AND TRANSLUCENCY

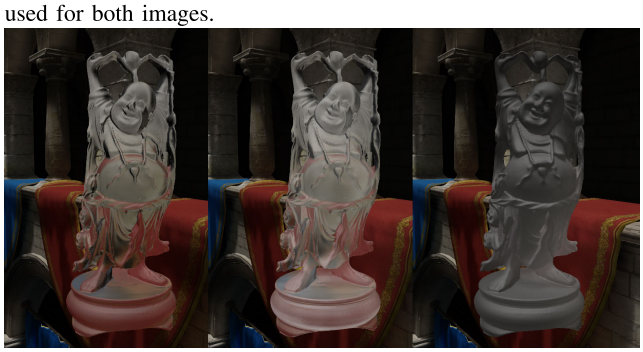
The RSM can be used to provide good visual results for the subsurface scattering of the light with a modified RSM that stores the transmitted light inside the geometry, as presented in Section III-G4. In fact, several variants of this approach were proposed, but none of these approximations can render translucent objects. Lambru *et al.* [92] proposed a 2D ray-casting process directly in the RSM to render them. Because of the potential low frequency of the geometry inside the RSM, it is necessary to use a very permissive

(a) Happy Buddha model without subsurface scattering or translucency

(b) Translucency using the RSM

(c) Translucency using the LPV. The same translucency properties and light intensities were used for both images.

(d) Translucency using the screen space refraction technique



(e) Subsurface scattering and translucency using the voxel volume. Left: $\rho = 0$, $\phi = 10^{\circ}$. Middle: $\rho = 0.1$, $\phi = 10^{\circ}$. Right: $\rho = 0.1$, $\phi = 90^{\circ}$.

FIGURE 9. Subsurface scattering and translucency comparison.

intersection check. Moreover, as detailed in Section IV-B, the translucent objects must not be rendered in the RSM. Unfortunately, this approach implies that light is no longer reflected by translucent objects. The results of this approach can be seen in Figure 9b.

The light propagation volumes technique has several extensions to compute the subsurface scattering of the light, as presented in Section III-G4. These approaches produce plausible results for subsurface scattering, but cannot render translucent objects. In addition, these approaches require the creation and use of a LPV that stores the radiance that comes directly from a light source. This LPV is different than the one used to compute the indirect illumination, which stores the radiance reflected by the scene surfaces. This implies that in order to produce both indirect illumination and subsurface scattering, it is necessary to create and store two LPVs at the same time, which leads to a significant impact on performance.

However, it is possible to render translucent objects with the LVP used for indirect illumination. Using the ray marching process proposed by Kaplanyan [39] to produce glossy reflections, it is possible to gather the VPLs along the refracted direction, inside the LPV. This approach provides plausible visual results (Figure 9c), but lacks flexibility. It is not possible to simulate a wide range of translucent materials because the LPV stores only low frequency lighting. This is visible in Figure 9c where the visual results from the left image provide a higher intensity of the refracted light compared to the intensity from the right image. This happens due to the larger lighted area on the ground, from behind the object on the left, that can be better approximated by the LPV compared to geometry details like the red decoration behind the object on the right.

Both subsurface scattering and translucency can be produced with the voxel representation of the scene geometry using a cone-tracing process. With color and opacity information per voxel, it is possible to use a specialized cone with a small aperture to accumulate both the direct and the indirect light behind the geometry. However, it is necessary to take into account the subsurface scattering phenomenon when the direct light is computed for the voxel volume. Still, a coarse approximation can be represented by the albedo of the voxelized geometry. The visual results can be seen in Figure 9e. The differences between the three results (ranging from low to high translucency) are represented by a variation in the opacity (ρ) of the geometry and the aperture (ϕ) of the cones used to accumulate the information in the refracted direction.

The screen space cannot provide enough information to obtain the subsurface scattering of the light. This happens because the required geometry is not present on the screen. Several multi-layer approaches were proposed to produce subsurface scattering and translucency, as presented in Section III-G4, but they no longer use only screen space information and require changes that have a significant impact on performance. However, for the particular case of rendering translucent objects, the screen space refraction technique was proposed with a minimum of modifications. It separates the opaque and the translucent objects into two sets of G-buffers. The implementation has been detailed in Section IV-E. This technique uses a 2D ray-casting process in the pixels of the G-buffers generated for opaque objects to compute the light refracted by the translucent objects in the other set of G-buffers. The results can be seen in Figure 9d. This approach is very limited and can't produce subsurface scattering. Moreover, the translucent objects can no longer reflect light, because they are not present in the G-buffers generated for the opaque objects, which are used to compute the indirect illumination.

(a) Volumetric lighting using shadow mapping

(b) Volumetric lighting using screen space information

FIGURE 10. Volumetric lighting comparison.

F. VOLUMETRIC LIGHTING

The RSM can be used to produce volumetric lighting, as presented in Section III-G5. The visual results of the approach proposed by Tóth and Umenhoffer [131] can be seen in Figure 10a.

A DOM-based approach can produce volumetric lighting, but requires a modified LPV than that used to compute the indirect illumination. To simulate the volumetric lighting, the radiance that comes directly from the light source is injected into the LPV, in contrast to the LPV used in the light propagation volumes technique, which uses the radiance reflected by the geometry. In addition, the propagation strategy requires to incorporate a light scattering and decimation model. Such an approach introduces a significant impact on performance, because the passes of radiance injection and propagation, the most consistent in terms of time performance, must be performed for each LPV.

Similar to the DOM-based techniques, those that use the voxel volume require substantial modifications to produce volumetric lighting. These approaches must store in the voxel volume information about the visibility of a voxel from the light source perspective. This is a different voxel volume than the one designed to compute the indirect illumination, which stores information about the geometry in the scene. For this reason, the use of this approach has a great impact on performance because both voxel volumes must be built.

As presented in Section III-G5, there are several techniques that use the screen space information to produce volumetric lighting. The visual results of the approach proposed by Mitchell [135] can be seen in Figure 10b. However, all of these approaches are limited and cannot produce results as flexible as the techniques presented above. Moreover, the light source must be visible on the screen.

G. AREA LIGHTS

As presented in Section III-G6, the techniques that produce indirect illumination can also simulate the area lights by treating the light emitted by them as direct illumination. Such approaches require a process of approximating the light emitted by the area lights. Unfortunately, as previously presented, the RSM-based techniques cannot simulate them. In contrast, all the other technique classes can simulate the area lights with little modification.

The DOM-based techniques only require that the emitted illumination of the area lights be injected in the LPV. We used a stochastic sampling process to approximate this illumination. More details about this approach were described in Section IV-C. The visual results obtained with the light propagation volumes technique can be seen in Figure 11a. For the voxel-based techniques, it is only necessary to introduce the emissive component into the voxels during the voxelization process of the scene geometry. The visual results obtained with the voxel cone tracing technique can be seen in Figure 11b.

The screen space techniques only need to take into account the emissive component when generating the direct illumination buffer. The visual results obtained with the SSDO and SRR techniques can be seen in Figure 11c. However, there are some observations to be made for this approach. The results depend on the amount of geometry of the area lights that is present on the screen. The problems caused by this limitation can be seen in Figure 11, where the back face of the quad emits light, but its influence is not present in Figure 11c. In contrast, this influence can be observed in the figures corresponding to the other techniques (Figure 11a, b).

Furthermore, all analyzed classes of real-time global illumination techniques, except for the one based on the RSM, produces both indirect diffuse illumination and glossy reflections for the emitted illumination of the area lights, without other changes. This is visible, with different degrees of glossiness of the reflections, in all cases in Figure 11.

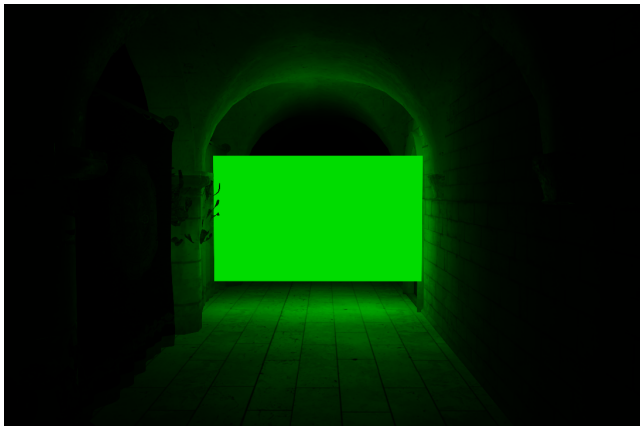
VI. COMPARATIVE ANALYSIS BASED ON QUANTITATIVE CRITERIA

We evaluated the implementations of the real-time global illumination techniques described in Section IV on a machine with a GTX1070 GPU at 1280×720 and 1920×1080 screen resolutions. The evaluation scene is Sponza Atrium, composed of 262k triangles. The comparison criterion is the frame rate, evaluated with different parameters that are particular for each technique. It is important to mention that for a more correct comparison between techniques, only the computing

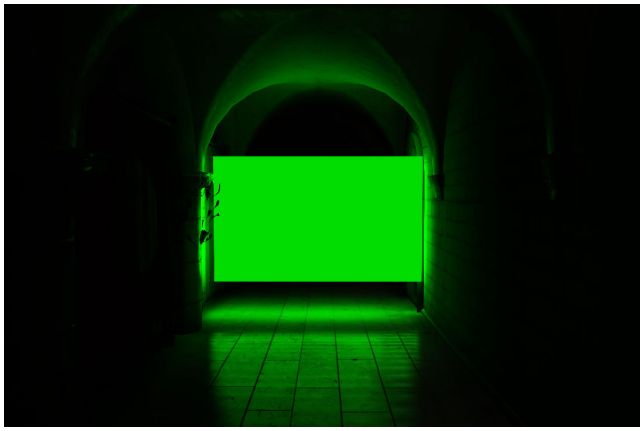
TABLE 2. The results of the performance evaluation for real-time global illumination techniques.

	Screen resolution	Criteria		Frame rates
		RSM resolution	Sample set size	
Reflective shadow map	1280×720	512^2	100	226
			200	187
			300	157
			400	136
		1024^2	100	190
			200	144
			300	116
			400	96
	1920×1080	512^2	100	122
			200	99
			300	84
			400	73
		1024^2	100	109
			200	84
			300	68
			400	58
	Screen resolution	Criteria		Frame rates
		LPV resolution		
Light propagation volumes	1280×720	32^3		196
		64^3		94
	1920×1080	32^3		111
		64^3		68
	Screen resolution	Criteria		Frame rates
		Continuous voxelization	Voxels resolution	
Voxel cone tracing	1280×720	off	64^3	283
			128^3	250
			256^3	216
			512^3	178
		on	64^3	214
			128^3	214
			256^3	170
			512^3	74
	1920×1080	off	64^3	145
			128^3	128
			256^3	109
			512^3	89
		on	64^3	138
			128^3	121
			256^3	96
			512^3	52
	Screen resolution	Criteria		Frame rates
		Sample set size		
Screen space directional occlusion	1280×720	100		85
		200		50
		300		35
		400		27
	1920×1080	100		36
		200		21.4
		300		15.0
		400		11.6

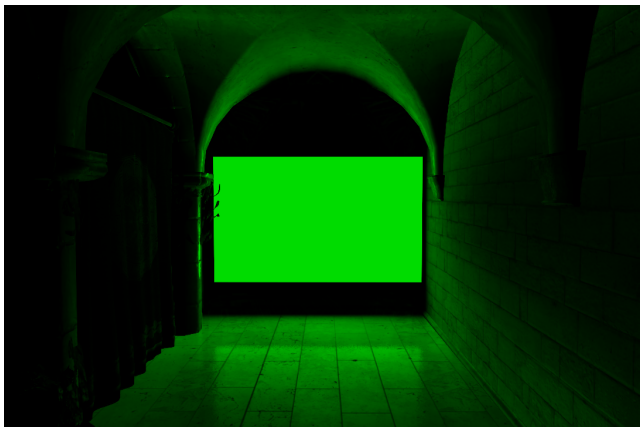
of the indirect diffuse illumination, the shadows and the ambient occlusion were considered. The reason for this decision is the fact that all initial descriptions of the techniques contain the computing of the indirect diffuse component. None of the approaches implemented for the calculation of glossy reflections or other effects was evaluated quantitatively because



(a) Area light using light propagation volumes



(b) Area light using voxel cone tracing



(c) Area light using SSDO and SSR

FIGURE 11. Area light comparison.

the qualitative results, where they exist, are not homogeneous between the approaches. The results are presented in Table 2 and are analyzed below.

The RSM-based technique has been evaluated together with the shadow mapping technique, with the RSM z-buffer, allowing shadows computation, and the sampling approach in screen space, proposed by Mittring [105] to obtain the ambient occlusion. We evaluated this technique with two parameters: the RSM resolution and the size of the sample set. We used the same 4 sets of samples for various screen and RSM resolutions. It is visible that a larger sample set has a negative impact on performance and the reasons are obvious. On the other hand, it is more difficult to explain the connection between a higher resolution and the big impact on performance. In addition to the time required to obtain the RSM at a higher resolution, there is a more subtle impact on the Texture Mapping Unit (TMU) of the GPU. The caching mechanism of the TMU has more misses when the sampling texture, the RSM, has a higher resolution. All those misses have a negative impact on performance and this impact is even more visible when the sample set is larger and there are more cache misses.

The light propagation volumes technique was evaluated together with the shadow mapping technique that uses the RSM generated at an intermediary pass and the ambient occlusion technique proposed by Mittring [105]. The only particular parameter is the LPV resolution. The impact on performance is represented by the radiance injection and propagation passes more than by the shading one, which has similar performance for both LPV resolutions.

The voxel cone tracing technique uses a shadow cone to obtain the light occlusion together with 5 short occlusion cones to obtain the ambient occlusion. The evaluation of this technique was done for various resolutions of the voxel volume in the scenario in which the volume is recreated at each frame (Continuous voxelization: on) and in the one in which it is created only once and after that it is only used. The voxel volume resolution has a big impact on performance due to the additional samplings along the traced cone. Also, the impact of continuous voxelization is visible in Table 2 where the recreation of the voxel volume uses more than the shading pass alone. However, the impact is lighter for smaller voxel volume resolutions.

The screen space directional occlusion technique was evaluated with the ambient occlusion approach proposed by Mittring [105] along with an additional shadow mapping technique. The only particular parameter for this technique is the size of the sample set. To obtain similar visual results as the other techniques, the SSDO have to accumulate the direct lighting information from big distances in world space. This process has a massive negative impact on performance due to a huge overload on the caching mechanism of the TMU. Also, it is visible that a higher screen resolution has a higher negative impact on performance. The results showcased in Table 2 prove that the SSDO technique is not suitable to obtain indirect diffuse illumination from huge distances and it only provides acceptable results for geometry details.

VII. CONCLUSION

We described several classes of real-time global illumination techniques used in current game engines and our own implementations with detailed rendering pipelines. We tested the capability of these classes of techniques to produce the indirect diffuse illumination along with a set of auxiliary photorealistic effects that are usually decoupled from the

illumination process. We offered visual results and technical details to obtain them where possible and explained why some effects cannot be produced. We offered and analyzed the performance of the implemented real-time global illumination techniques.

The real-time global illumination field is complex and provides a set of flexible techniques. Over time, several approaches that allowed only interactive times took advantage of the advances of GPUs to obtain real-time performance. Our results provide several conclusions about each class of techniques.

We start our discussions with the class of screen space techniques because it is the only one that was partly accepted by almost the entire game industry. It could provide the most correct results for some effects and scenarios, but it hasn't proven suitable for several other effects. The use of screen information as a simplified geometry representation offers good results for approaches that require high frequency details. It provides the sharpest results for glossy reflections, acceptable results for the ambient occlusion and subsurface scattering and it could also enhance the direct light occlusion results provided by decoupled shadow techniques. Unfortunately, it proved to be unsuitable to compute indirect diffuse illumination for long distances. It also introduces temporal coherence problems. However, the advantages offered by this class of techniques represent the reason why it was partly accepted by the game industry.

The RSM has proven to be an extremely flexible data structure which unfortunately provides information about only the first bounce of the light. It offers plausible results for several effects, including the diffuse component of the indirect lighting, but the previously mentioned limitation makes it a better choice as the source of photons for other classes of techniques. It was used in the implementations of both light propagation volumes and voxel cone tracing techniques. The sampling of the RSM has been shown to offer acceptable performance and could be used as a plausible source of radiance for other techniques.

The LPV with spherical harmonics coefficients stored in the cells has certain limitations. High frequency details cannot be stored, and therefore the technique cannot provide precise results. However, it has proven to provide plausible results for the indirect diffuse illumination, the effect for which it was designed. It produces brighter images overall, but also thanks to good performance, it has shown to be a good choice for effects that require low frequency details, like the diffuse component of lighting or the rendering of low translucent materials.

The voxel-based techniques represent the class that accommodates most of the chosen effects. Furthermore, in most scenarios, it has proven to provide the most plausible visual results. For this reason, it was recently accepted as a good choice to produce ambient occlusion. Unfortunately, it requires the voxelization of the scene geometry, which is a complex process with a significant impact on performance. The approach that produces ambient occlusion takes advantage of the use of occlusion voxels, which require less memory and have a faster voxelization time. The voxel representation of the scene has proven to be a very flexible data structure that can accommodate several purposes.

All classes of global illumination techniques analyzed in this paper present disadvantages from the point of view of the unitary production of all the chosen illumination effects and phenomena. Some classes may produce a larger set of these effects, but they aren't always practical and don't provide good visual results for all scenarios. Certain offline global illumination techniques produce superior visual results for many of the chosen effects. Therefore, one of the important research topics in the near future will be the implementation of these techniques in real-time. This topic has been of great interest so far, but with the growing computing power that has been presented by GPUs, it will become even more important.

ACKNOWLEDGMENT

Many appreciations to Alexandru Naiman for all technical assistance provided.

REFERENCES

[1] P. V. Satheesh, *Unreal Engine 4 Game Development Essentials*. Birmingham, U.K.: Packt, 2016, pp. 93–113.
[2] M. Menard and B. Wagstaff, *Game Development With Unity*. Toronto, ON, Canada: Nelson Education, 2015, pp. 338–339.
[3] A. Kaplanyan, "Real-time diffuse global illumination in cryengine 3," in *SIGGRAPH Course on 'Global Illumination Across Industries Course'*, 2010.
[4] S. Martin and P. Einarsson, "A real-time radiosity architecture for video games," in *ACM SIGGRAPH Courses: Advances in Real-Time Rendering*, 2010.
[5] T. Ritschel, C. Dachsbacher, T. Grosch, and J. Kautz, "The state of the art in interactive global illumination," *Comput. Graph. Forum*, vol. 31, pp. 160–188, Feb. 2012.
[6] C. Dachsbacher, J. Křivánek, M. Hašan, A. Arbree, B. Walter, and J. Novák, "Scalable realistic rendering with many-light methods," *Comput. Graph. Forum*, vol. 33, pp. 88–104, Feb. 2014.
[7] D. Jönsson, E. Sundén, A. Ynnerman, and T. Ropinski, "A survey of volumetric illumination techniques for interactive volume rendering," *Comput. Graph. Forum*, vol. 33, pp. 27–51, Feb. 2014.
[8] W. Heidrich, "Interactive display of global illumination solutions for non-diffuse environments—A survey," *Comput. Graph. Forum*, vol. 20, pp. 225–244, Dec. 2001.
[9] C. Damez, K. Dmitriev, and K. Myszkowski, "State of the art in global illumination for interactive applications and high-quality animations," *Comput. Graph. Forum*, vol. 22, no. 1, pp. 55–77, Mar. 2003.
[10] P. Dutre, P. Bekaert, and K. Bala, *Advanced Global Illumination*. Boca Raton, FL, USA: CRC Press, 2018.
[11] T. Akenine-Möller, E. Haines, and N. Hoffman, *Real-Time Rendering*. Boca Raton, FL, USA: CRC Press, 2019.
[12] L. Szirmay-Kalos, T. Umenhoffer, G. Patow, L. Szécsi, and M. Sbert, "Specular effects on the GPU: State of the art," *Comput. Graph. Forum*, vol. 28, no. 6, pp. 1586–1617, Sep. 2009.
[13] A. Woo, P. Poulin, and A. Fournier, "A survey of shadow algorithms," *IEEE Comput. Graph. Appl.*, vol. 10, no. 6, pp. 13–32, Nov. 1990.
[14] J.-M. Hasenfratz, M. Lapierre, N. Holzschuch, F. Sillion, and A. Gravirimag-Inria, "A survey of real-time soft shadows algorithms," *Comput. Graph. Forum*, vol. 22, no. 4, pp. 753–774, Dec. 2003.
[15] D. Scherzer, M. Wimmer, and W. Purgathofer, "A survey of real-time hard shadow mapping methods," *Comput. Graph. Forum*, vol. 30, no. 1, pp. 169–186, Mar. 2011.
[16] H. Kolivand and M. S. Sunar, "Survey of shadow volume algorithms in computer graphics," *IETE Tech. Rev.*, vol. 30, no. 1, pp. 38–46, 2013.
[17] E. Eisemann, M. Schwarz, U. Assarsson, and M. Wimmer, *Real-Time Shadows*. Boca Raton, FL, USA: CRC Press, 2011.

[18] À. Méndez-Feliu and M. Sbert, "From obscurances to ambient occlusion: A survey," *Vis. Comput.*, vol. 25, no. 2, pp. 181–196, Feb. 2009.
[19] E. Cerezo, F. Pérez, X. Pueyo, F. J. Seron, and F. X. Sillion, "A survey on participating media rendering techniques," *Vis. Comput.*, vol. 21, no. 5, pp. 303–328, Jun. 2005.
[20] J. T. Kajiya, "The rendering equation," *ACM SIGGRAPH Comput. Graph.*, vol. 20. no. 4, pp. 143–150, Aug. 1986.
[21] F. E. Nicodemus, "Directional reflectance and emissivity of an opaque surface," *Appl. Opt.*, vol. 4, no. 7, pp. 767–775, 1965.
[22] B. T. Phong, "Illumination for computer generated pictures," *Commun. ACM*, vol. 18, no. 6, pp. 311–317, Jun. 1975.
[23] J. F. Blinn, "Models of light reflection for computer synthesized pictures," *ACM SIGGRAPH Comput. Graph.*, vol. 11, pp. 192–198, Jul. 1977.
[24] R. L. Cook and K. E. Torrance, "A reflectance model for computer graphics," *ACM Trans. Graph.*, vol. 1, no. 1, pp. 7–24, 1982.
[25] B. Walter, S. R. Marschner, H. Li, and K. E. Torrance, "Microfacet models for refraction through rough surfaces," in *Proc. 18th Eurograph. Conf. Rendering Techn.*, 2007, pp. 195–206.
[26] C. Schlick, "A survey of shading and reflectance models," *Comput. Graph. Forum*, vol. 13, no. 2, pp. 121–131, May 1994.
[27] T. Whitted, "An improved illumination model for shaded display," *Commun. ACM*, vol. 23, no. 6, pp. 343–349, Jun. 1980.
[28] C. M. Goral, K. E. Torrance, D. P. Greenberg, and B. Battaile, "Modeling the interaction of light between diffuse surfaces," *SIGGRAPH Comput. Graph.*, vol. 18, no. 3, pp. 213–222, Jan. 1984.
[29] M. F. Cohen and J. R. Wallace, *Radiosity and Realistic Image Synthesis*. Amsterdam, The Netherlands: Elsevier, 2012.
[30] H. W. Jensen, "Global illumination using photon maps," in *Proc. Eurograph. Workshop Rendering Techn.* Vienna, Austria: Springer, 1996, pp. 21–30.
[31] A. Keller, "Instant radiosity," in *Proc. 24th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 1997, pp. 49–56.
[32] G. J. Ward, F. M. Rubinstein, and R. D. Clear, "A ray tracing solution for diffuse interreflection," in *Proc. 15th Annu. Conf. Comput. Graph. Interact. Techn.*, 1988, pp. 85–92.
[33] C. Luksch, M. Wimmer, and M. Schwärzler, "Incrementally baked global illumination," in *Proc. ACM SIGGRAPH Symp. Interact. 3D Graph. Games*, May 2019, pp. 1–10.
[34] G. Greger, P. Shirley, P. M. Hubbard, and D. P. Greenberg, "The irradiance volume," *IEEE Comput. Graph. Appl.*, vol. 18, no. 2, pp. 32–43, Mar. 1998.
[35] P.-P. Sloan, J. Kautz, and J. Snyder, "Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 527–536, Jul. 2002.
[36] C. Dachsbacher and M. Stamminger, "Reflective shadow maps," in *Proc. Symp. Interact. 3D Graph. Games (SID)*, 2005, pp. 203–231.
[37] P. Lensing and W. Broll, "Efficient shading of indirect illumination applying reflective shadow maps," in *Proc. ACM SIGGRAPH Symp. Interact. 3D Graph. Games (ID)*, 2013, pp. 95–102.
[38] R. Prutkin, A. Kaplanyan, and C. Dachsbacher, "Reflective shadow map clustering for real-time global illumination," in *Proc. Eurographics*, 2012, pp. 9–12.
[39] A. Kaplanyan, "Light propagation volumes in cryengine 3," *ACM SIGGRAPH Courses*, vol. 7, p. 2, Aug. 2009.
[40] D. Bischoff, T. Schwandt, and W. Broll, "A real-time global illumination approach for high resolution reflective shadow maps in open world scenes," in *Proc. VISIGRAPP*, 2017, pp. 116–126.
[41] C. Dachsbacher and M. Stamminger, "Splatting indirect illumination," in *Proc. Symp. Interact. 3D Graph. Games (SID)*, 2006, pp. 93–100.
[42] T. Ritschel, T. Grosch, M. H. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz, "Imperfect shadow maps for efficient computation of indirect illumination," *ACM Trans. Graph.*, vol. 27, no. 5, pp. 1–8, Dec. 2008.
[43] T. Ritschel, E. Eisemann, I. Ha, J. D. K. Kim, and H.-P. Seidel, "Making imperfect shadow maps view-adaptive: high-quality global illumination in large dynamic scenes," *Comput. Graph. Forum*, vol. 30, no. 8, pp. 2258–2269, Dec. 2011.
[44] S. Chandrasekhar, *Radiative Transfer*. New York, NY, USA: Dover, 1960.
[45] J. T. Kajiya and B. P. Von Herzen, "Ray tracing volume densities," *ACM SIGGRAPH Comput. Graph.*, vol. 18, no. 3, pp. 165–174, Jul. 1984.
[46] R. Geist, K. Rasche, J. Westall, and R. Schalkoff, "Lattice-Boltzmann lighting," in *Proc. 15th Eurograph. Conf. Rendering Techn. (EGSR)*. Norrköping, Sweden: Eurographics Association, 2004, pp. 355–362.
[47] R. Fattal, "Participating media illumination using light propagation maps," *ACM Trans. Graph.*, vol. 28, no. 1, pp. 1–11, Jan. 2009.
[48] A. Kaplanyan and C. Dachsbacher, "Cascaded light propagation volumes for real-time indirect illumination," in *Proc. ACM SIGGRAPH Symp. Interact. 3D Graph. Games (ID)*, 2010, pp. 99–107.
[49] T. A. Franke, "Delta light propagation volumes for mixed reality," in *Proc. IEEE Int. Symp. Mixed Augmented Reality (ISMAR)*, Oct. 2013, pp. 125–132.
[50] D. Cohen-Or and A. Kaufman, "Fundamentals of surface voxelization," *Graph. Models Image Process.*, vol. 57, no. 6, pp. 453–461, Nov. 1995.
[51] S. Fang and H. Chen, "Hardware accelerated voxelization," *Comput. Graph.*, vol. 24, no. 3, pp. 433–442, Jun. 2000.
[52] Z. Dong, W. Chen, H. Bao, H. Zhang, and Q. Peng, "Real-time voxelization for complex polygonal models," in *Proc. 12th Pacific Conf. Comput. Graph. Appl. (PG)*, 2004, pp. 43–50.
[53] E. Eisemann and X. Décoret, "Single-pass GPU solid voxelization for real-time applications," in *Proc. Graph. Interface (GI)*. Windsor, ON, Canada: Canadian Information Processing Society, 2008, pp. 73–80.
[54] M. Schwarz and H.-P. Seidel, "Fast parallel surface and solid voxelization on GPUs," *ACM Trans. Graph.*, vol. 29, no. 6, pp. 1–10, Dec. 2010.
[55] J. Pantaleoni, "VoxelPipe: A programmable pipeline for 3D voxelization," in *Proc. ACM SIGGRAPH Symp. High Perform. Graph. (HPG)*, 2011, pp. 99–106.
[56] C. Crassin, F. Neyret, S. Lefebvre, and E. Eisemann, "GigaVoxels: Ray-guided streaming for efficient and detailed voxel rendering," in *Proc. Symp. Interact. 3D Graph. Games (ID)*, 2009, pp. 15–22.
[57] C. Soler, O. Hoel, and F. Rochet, "A deferred shading pipeline for real-time indirect illumination," in *Proc. ACM SIGGRAPH Talks*, 2010, p. 1.
[58] S. Thiedemann, N. Henrich, T. Grosch, and S. Müller, "Voxel-based global illumination," in *Proc. Symp. Interact. 3D Graph. Games (ID)*, 2011, pp. 103–110.
[59] C. Crassin, F. Neyret, M. Sainz, S. Green, and E. Eisemann, "Interactive indirect illumination using voxel cone tracing," *Comput. Graph. Forum*, vol. 30, no. 7, pp. 1921–1930, Sep. 2011.
[60] A. Panteleev, "Practical real-time voxel-based global illumination for current GPUs," presented at the ACM SIGGRAPH, 2014.
[61] I. Aherne, R. Davison, G. Ushaw, and G. Morgan, "Adoption of sparse 3D textures for voxel cone tracing in real time global illumination," in *Proc. VISIGRAPP*, 2020, pp. 201–209.
[62] T. A. Franke, "Delta voxel cone tracing," in *Proc. IEEE Int. Symp. Mixed Augmented Reality (ISMAR)*, Sep. 2014, pp. 39–44.
[63] M. Sugihara, R. Rauwendaal, and M. Salvi, "Layered reflective shadow maps for voxel-based indirect illumination," in *Proc. High-Perform. Graph.* Lyon, France: Eurographics Association, 2014, pp. 117–125. [Online]. Available: https://dblp.org/rec/conf/egh/SugiharaRS14.bib, doi: 10.2312/hpg.20141100.
[64] Y.-Y. Chen and S.-Y. Chien, "Lighting-driven voxels for memory-efficient computation of indirect illumination," *Vis. Comput.*, vol. 32, nos. 6–8, pp. 781–789, Jun. 2016.
[65] T. Saito and T. Takahashi, "Comprehensible rendering of 3-D shapes," in *Proc. 17th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 1990, pp. 197–206.
[66] P.-P. Sloan, N. K. Govindaraju, D. Nowrouzezahrai, and J. Snyder, "Image-based proxy accumulation for real-time soft global illumination," in *Proc. 15th Pacific Conf. Comput. Graph. Appl. (PG)*, Oct. 2007, pp. 97–105.
[67] G. Nichols, J. Shopf, and C. Wyman, "Hierarchical image-space radiosity for interactive global illumination," *Comput. Graph. Forum*, vol. 28, no. 4, pp. 1141–1149, Jun. 2009.
[68] G. Nichols and C. Wyman, "Multiresolution splatting for indirect illumination," in *Proc. Symp. Interact. 3D Graph. Games (ID)*, 2009, pp. 83–90.
[69] T. Ritschel, T. Grosch, and H.-P. Seidel, "Approximating dynamic global illumination in image space," in *Proc. Symp. Interact. 3D Graph. Games (ID)*, 2009, pp. 75–82.
[70] H. Pfister, J. Hardenbergh, J. Knittel, H. Lauer, and L. Seiler, "The VolumePro real-time ray-casting system," in *Proc. 26th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 1999, pp. 251–260.
[71] J. Schmittler, I. Wald, and P. Slusallek, "SaarCOR: A hardware architecture for ray tracing," in *Proc. ACM SIGGRAPH/EUROGRAPHICS Conf. Graph. Hardw.*, 2002, pp. 27–36.
[72] S. Woop, J. Schmittler, and P. Slusallek, "RPU: A programmable ray processing unit for realtime ray tracing," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 434–444, 2005.
[73] C. Nvidia. (2018). *Nvidia Turing GPU Architecture*. [Online]. Available: https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/technologies/turing-architecture/NVIDIA-Turing-Architecture-Whitepaper.pdf

[74] A. Keller, T. Viitanen, C. Barré-Brisebois, C. Schied, and A. M. McGuire, "Are we done with ray tracing?" in *Proc. SIGGRAPH Courses*, 2019, pp. 1–3.
[75] C. R. A. Chaitanya, A. S. Kaplanyan, C. Schied, M. Salvi, A. Lefohn, D. Nowrouzezahrai, and T. Aila, "Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 1–12, Jul. 2017.
[76] S. Hillaire, C. de Rousiers, and D. Apers, "Real-time raytracing for interactive global illumination workflows in frostbite," presented at the Game Developers Conf., 2018.
[77] J. Andersson and C. Barré-Brisebois, "Shiny pixels and beyond: Real-time raytracing at SEED," presented at the Game Developers Conf., 2018.
[78] Z. Majercik, J.-P. Guertin, D. Nowrouzezahrai, and M. McGuire, "Dynamic diffuse global illumination with ray-traced irradiance fields," *J. Comput. Graph. Techn.*, vol. 8, no. 2, pp. 1–30, 2019.
[79] V. V. Sanzharov, V. A. Frolov, and V. A. Galaktionov, "Survey of Nvidia RTX technology," *Program. Comput. Softw.*, vol. 46, no. 4, pp. 297–304, Jul. 2020.
[80] J. F. Blinn and M. E. Newell, "Texture and reflection in computer generated images," *Commun. ACM*, vol. 19, no. 10, pp. 542–547, Oct. 1976.
[81] J. Kautz and M. D. McCool, "Approximation of glossy reflection with prefiltered environment maps," in *Proc. Graph. Interface*, 2000, pp. 119–126.
[82] C. Brennan, "Accurate environment mapped reflections and refractions by adjusting for object distance," in *Shader X*, W. Engel, Ed. Wordware Publishing, 2002, pp. 290–294.
[83] K. Bjorke, "Image-based lighting," in *GPU Gems*, R. Fernando, Ed. Reading, MA, USA: Addison-Wesley, 2004, pp. 307–321.
[84] T. Sousa, N. Kasyan, and N. Schulz, "Secrets of CryENGINE 3 graphics technology," in *ACM SIGGRAPH 2011 Courses, Advances in Real-Time Rendering in 3D Graphics and Games*, 2011.
[85] M. McGuire and M. Mara, "Efficient GPU screen-space ray tracing," *J. Comput. Graph. Techn.*, vol. 3, no. 4, pp. 73–85, 2014.
[86] Y. Uludag, "Hi-Z screen-space cone-traced reflections," *GPU Pro*, vol. 5, pp. 149–192, Dec. 2014.
[87] T. Stachowiak and Y. Uludag, "Stochastic screen-space reflections," in *ACM SIGGRAPH Courses: Advances in Real-time Rendering*, 2015.
[88] S. Widmer, D. Pająk, A. Schulz, K. Pulli, J. Kautz, M. Goesele, and D. Luebke, "An adaptive acceleration structure for screen-space ray tracing," in *Proc. 7th Conf. High-Perform. Graph. (HPG)*, 2015, pp. 67–76.
[89] N. Hofmann, P. Bogendörfer, M. Stamminger, and K. Selgrad, "Hierarchical multi-layer screen-space ray tracing," in *Proc. High Perform. Graph.*, 2017, pp. 1–10.
[90] K. Vardis, A. A. Vasilakis, and G. Papaioannou, "A multiview and multilayer approach for interactive ray tracing," in *Proc. 20th ACM SIGGRAPH Symp. Interact. 3D Graph. Games*, Feb. 2016, pp. 171–178.
[91] K. Vardis, A.-A. Vasilakis, and G. Papaioannou, "Dirt: Deferred image-based ray tracing," in *Proc. High Perform. Graph.*, 2016, pp. 63–73.
[92] C. Lambru, A. Morar, F. Moldoveanu, V. Asavei, and S. Ivaşcu, "Hybrid global illumination: A novel approach combining screen and light space information," Dept. C-Elect. Eng. Comput. Sci., Univ. Politehnica Bucharest Sci. Bull., Bucharest, Romania, Tech. Rep., 2021, pp. 3–20, vol. 83, no. 2.
[93] A. A. Vasilakis, K. Vardis, and G. Papaioannou, "A survey of multifragment rendering," *Comput. Graph. Forum*, vol. 39, no. 2, pp. 623–642, May 2020.
[94] L. Williams, "Casting curved shadows on curved surfaces," *ACM SIGGRAPH Comput. Graph.*, vol. 12, no. 3, pp. 270–274, Aug. 1978.
[95] F. C. Crow, "Shadow algorithms for computer graphics," *ACM SIGGRAPH Comput. Graph.*, vol. 11, no. 2, pp. 242–248, Aug. 1977.
[96] G. Nichols, R. Penmatsa, and C. Wyman, "Interactive, multiresolution image-space rendering for dynamic area lighting," *Comput. Graph. Forum*, vol. 29, no. 4, pp. 1279–1288, Aug. 2010.
[97] J. Villegas and E. Ramírez, "Deferred voxel shading for real-time global illumination," in *Proc. 42nd Latin Amer. Comput. Conf. (CLEI)*, Oct. 2016, pp. 1–11.
[98] N. Kasyan, "Playing with real-time shadows," in *SIGGRAPH Course on 'Efficient Real-Time Shadows'*, 2013.
[99] G. Miller, "Efficient algorithms for local and global accessibility shading," in *Proc. 21st Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 1994, pp. 319–326.
[100] S. Zhukov, A. Iones, and G. Kronin, "An ambient light illumination model," in *Proc. Eurograph. Workshop Rendering Techn.* Vienna, Austria: Springer, 1998, pp. 45–55.
[101] M. Pharr and S. Green, "Ambient occlusion," *GPU Gems*, vol. 1, pp. 279–292, Mar. 2004.
[102] M. Bunnell, "Dynamic ambient occlusion and indirect lighting," *GPU Gems*, vol. 2, no. 2, pp. 223–233, 2005.
[103] J. Kontkanen and S. Laine, "Ambient occlusion fields," in *Proc. Symp. Interact. 3D Graph. Games (SID)*, 2005, pp. 41–48.
[104] P. Shanmugam and O. Arikan, "Hardware accelerated ambient occlusion techniques on GPUs," in *Proc. Symp. Interact. 3D Graph. Games (ID)*, 2007, pp. 73–80.
[105] M. Mittring, "Finding next gen: Cryengine 2," in *Proc. ACM SIGGRAPH Courses*, 2007, pp. 97–121.
[106] M. McGuire, M. Mara, and D. Luebke, "Scalable ambient obscurance," in *Proc. 4th ACM SIGGRAPH/Eurograph. Conf. High-Perform. Graph.*, 2012, pp. 97–103.
[107] O. Mattausch, D. Scherzer, and M. Wimmer, "Temporal screen-space ambient occlusion," in *GPU Pro 2*, W. Engel, Ed. A.K. Peters, Feb. 2011. [Online]. Available: https://www.cg.tuwien.ac.at/research/publications/2011/matt2011/
[108] L. Bavoil, M. Sainz, and R. Dimitrov, "Image-space horizon-based ambient occlusion," in *Proc. ACM SIGGRAPH Talks*, 2008, p. 22.
[109] L. Szirmay-Kalos, T. Umenhoffer, B. Tóth, L. Szécsi, and M. Sbert, "Volumetric ambient occlusion for real-time rendering and games," *IEEE Comput. Graph. Appl.*, vol. 30, no. 1, pp. 70–79, Dec. 2010.
[110] J. Shade, S. Gortler, L.-W. He, and R. Szeliski, "Layered depth images," in *Proc. 25th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 1998, pp. 231–242.
[111] L. Bavoil and M. Sainz, "Multi-layer dual-resolution screen-space ambient occlusion," in *Proc. SIGGRAPH, Talks*, 2009, p. 1.
[112] O. Nalbach, T. Ritschel, and H.-P. Seidel, "Deep screen space," in *Proc. 18th Meeting ACM SIGGRAPH Symp. Interact. 3D Graph. Games (ID)*, 2014, pp. 79–86.
[113] K. Vardis, G. Papaioannou, and A. Gaitatzes, "Multi-view ambient occlusion with importance sampling," in *Proc. ACM SIGGRAPH Symp. Interact. 3D Graph. Games (ID)*, 2013, pp. 111–118.
[114] C. K. Reinbothe, T. Boubekeur, and M. Alexa, "Hybrid ambient occlusion," in *Proc. Eurographics*, 2009, pp. 51–57.
[115] G. Papaioannou, M. L. Menexi, and C. Papadopoulos, "Real-time volume-based ambient occlusion," *IEEE Trans. Vis. Comput. Graphics*, vol. 16, no. 5, pp. 752–762, Sep. 2010.
[116] R. Penmatsa, G. Nichols, and C. Wyman, "Voxel-space ambient occlusion," in *Proc. ACM SIGGRAPH Symp. Interact. 3D Graph. Games*, 2010, p. 1.
[117] H. W. Jensen, S. R. Marschner, M. Levoy, and P. Hanrahan, "A practical model for subsurface light transport," in *Proc. 28th Annu. Conf. Comput. Graph. Interact. Techn. (SIGGRAPH)*, 2001, pp. 511–518.
[118] C. Dachsbacher and M. Stamminger, "Translucent shadow maps," in *Proc. 14th Eurograph. Workshop Rendering (EGRW)*. Leuven, Belgium: Eurographics Association, 2003, pp. 197–201.
[119] J. Børlum, B. B. Christensen, T. K. Kjeldsen, P. T. Mikkelsen, K. Ø. Noe, J. Rimestad, and J. Mosegaard, "SSLPV: Subsurface light propagation volumes," in *Proc. ACM SIGGRAPH Symp. High Perform. Graph. (HPG)*, 2011, pp. 7–14.
[120] M. D. Koa and H. Johan, "ESLPV: Enhanced subsurface light propagation volumes," *Vis. Comput.*, vol. 30, nos. 6–8, pp. 821–831, Jun. 2014.
[121] E. d'Eon, D. Luebke, and E. Enderton, "Efficient rendering of human skin," in *Proc. 18th Eurograph. Conf. Rendering Techn.*, 2007, pp. 147–157.
[122] J. Jimenez, V. Sundstedt, and D. Gutierrez, "Screen-space perceptual rendering of human skin," *ACM Trans. Appl. Perception*, vol. 6, no. 4, pp. 1–15, Sep. 2009.
[123] J. Jimenez, K. Zsolnai, A. Jarabo, C. Freude, T. Auzinger, X.-C. Wu, J. von der Pahlen, M. Wimmer, and D. Gutierrez, "Separable subsurface scattering," *Comput. Graph. Forum*, vol. 34, no. 6, pp. 188–197, Sep. 2015.
[124] H. W. Jensen and J. Buhler, "A rapid hierarchical rendering technique for translucent materials," in *Proc. 29th Annu. Conf. Comput. Graph. Interact. Techn.*, 2002, pp. 576–581.
[125] T. Mertens, J. Kautz, P. Bekaert, F. Van Reeth, and H.-P. Seidel, "Efficient rendering of local subsurface scattering," in *Proc. 11th Pacific Conf. Comput. Graph. Appl.*, 2003, pp. 51–58.
[126] P. Ganestam and M. Doggett, "Real-time multiply recursive reflections and refractions using hybrid rendering," *Vis. Comput.*, vol. 31, no. 10, pp. 1395–1403, Oct. 2015.
[127] E. Eisemann and X. Décoret, "Fast scene voxelization and applications," in *Proc. Symp. Interact. 3D Graph. Games*, 2006, pp. 71–78.
[128] N. L. Max, "Atmospheric illumination and shadows," *ACM SIGGRAPH Comput. Graph.*, vol. 20, no. 4, pp. 117–124, Aug. 1986.

[129] T. Nishita, Y. Miyawaki, and E. Nakamae, "A shading model for atmospheric scattering considering luminous intensity distribution of light sources," *ACM SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 303–310, Aug. 1987.
[130] C. Wyman and S. Ramsey, "Interactive volumetric shadows in participating media with single-scattering," in *Proc. IEEE Symp. Interact. Ray Tracing*, Aug. 2008, pp. 87–92.
[131] B. Tóth and T. Umenhoffer, "Real-time volumetric lighting in participating media," in *Proc. Eurographics*, 2009, pp. 57–60.
[132] M. Billeter, E. Sintorn, and U. Assarsson, "Real-time multiple scattering using light propagation volumes," in *Proc. ID*, vol. 12, 2012, pp. 119–126.
[133] C. Wyman, "Voxelized shadow volumes," in *Proc. ACM SIGGRAPH Symp. High Perform. Graph.*, 2011, pp. 33–40.
[134] C. Wyman and Z. Dai, "Imperfect voxelized shadow volumes," in *Proc. 5th High-Perform. Graph. Conf.*, 2013, pp. 45–52.
[135] K. Mitchell, "Volumetric light scattering as a post-process," in *GPU Gems 3*, H. Nguyen, Ed. Reading, MA, USA: Addison-Wesley, 2008, pp. 275–285.
[136] T. Sousa, "Crysis next gen effects," presented at the Game Developers Conf., 2008.
[137] E. Heitz, J. Dupuy, S. Hill, and D. Neubelt, "Real-time polygonal-light shading with linearly transformed cosines," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 1–8, Jul. 2016.
[138] T. Kuge, T. Yatagawa, and S. Morishima, "Real-time indirect illumination of emissive inhomogeneous volumes using layered polygonal area lights," *Comput. Graph. Forum*, vol. 38, no. 7, pp. 449–460, Oct. 2019.
[139] M. Di Koa, H. Johan, and A. Sourin, "Voxel-based interactive rendering of translucent materials under area lights using sparse samples," in *Proc. Int. Conf. Cyberworlds (CW)*, Sep. 2017, pp. 56–63.
[140] J. Beck, P. Hellekalek, P. Hickernell, G. Larcher, P. L'Ecuyer, H. Niederreiter, S. Tezuka, and C. Xing, *Random and Quasi-Random Point Sets*, vol. 138. New York, NY, USA: Springer, 2012. [Online]. Available: https://www.springer.com/gp/book/9780387985541
[141] W. T. Reeves, D. H. Salesin, and R. L. Cook, "Rendering antialiased shadows with depth maps," *ACM SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 283–291, Aug. 1987.

CRISTIAN LAMBRU received the B.S. degree in informatics from the University of Bucharest, in 2015, and the M.S. degree in computer graphics, multimedia and virtual reality from the POLITEHNICA University of Bucharest, in 2017, where he is currently pursuing the Ph.D. degree with the Faculty of Automatic Control and Computers. He is an Assistant Professor with the Computer Science and Engineering Department, Faculty of Automatic Control and Computers, POLITEHNICA University of Bucharest. His research interests include computer graphics and real-time global illumination.

ANCA MORAR received the B.S. degree in computer science from the POLITEHNICA University of Bucharest, in 2009, and the Ph.D. degree in computer science, in 2012, with a focus on medical image analysis and visualization. She is currently an Associate Professor with the Computer Science and Engineering Department, Faculty of Automatic Control and Computers, POLITEHNICA University of Bucharest. Her research interests include computer graphics, GPGPU, computer vision, and e-health.

FLORICA MOLDOVEANU (Member, IEEE) is currently a Professor with the Department of Computer Science and Engineering, POLITEHNICA University of Bucharest. She coordinates the master program computer graphics, multimedia, and virtual reality at the Faculty of Automatic Control and Computers. She is also the President of the Health Level Seven Romania Association. Her research interests and teaching activities include computer graphics, computer vision, software engineering, and e-health.

VICTOR ASAVEI (Member, IEEE) received the Ph.D. degree in computer science and information technology from the POLITEHNICA University of Bucharest, in 2011.

He is currently an Assistant Professor (a Lecturer) with the Computer Science and Engineering Department, Faculty of Automatic Control and Computers, POLITEHNICA University of Bucharest. He has coauthored approximately 60 articles and four books in the fields of computer graphics, distributed computing, software engineering, and medical ITs. He has participated in numerous national and international research projects. His current teaching and research interests include real time computer graphics and general purpose computing on graphics processing units (GPGPU).

ALIN MOLDOVEANU (Member, IEEE) is currently a Professor and the Vice-Dean of the Faculty of Automatic Control and Computers, POLITEHNICA University of Bucharest, where he completed as a Valedictorian some 20 years ago. He is teaching virtual and augmented reality at master's and software engineering at bachelor's level. He is in charge of the 24 master programs of the faculty. His focus is on applied research using VR and AR (exploring and applying immersion, sensory substitution, and distorted reality), mainly in e-health, e-learning, and e-culture. As a Leader of the 3DUPB Research Team, he was/is the Director or a Team Leader for many national or European research projects in these areas, such as sound of vision, TRAVEE, HAI-OPS, and Lib2Life. His works received prestigious prizes, such as the Best "Tech for Society" Horizon 2020 Project awarded by the European Commission through Innovation Radar at ICT 2018–received by Sound of Vision (project conducted as a Technical Coordinator and UPB Team responsible).