applied sciences

Article

Implementing Many-Lights Rendering with IES-Based Lights

Davide Gadia *, Vincenzo Lombardo, Dario Maggiorini and Antonio Natilla

Department of Computer Science, University of Milan, 20133 Milan, Italy; dario@di.unimi.it (D.M.)
* Correspondence: gadia@di.unimi.it

Abstract: In recent years, many research projects on real-time rendering have focused on the introduction of global illumination effects in order to improve the realism of a virtual scene. The main goal of these works is to find a compromise between the achievable quality of the resulting rendering and the intrinsic high computational cost of global illumination. An established approach is based on the use of Virtual Point Lights, i.e., "fictitious" light sources that are placed on surfaces in the scene. These lights simulate the contribution of light rays emitted by the light sources and bouncing on different objects. Techniques using Virtual Point Lights are often called *Many-Lights* techniques. In this paper, we propose an extension of a real-time Many-Lights rendering technique characterized by the integration of photometric data in the process of Virtual Point Lights distribution. We base the definition of light sources and the creation of Virtual Point Lights on the description provided in the IES standard format, created by the Illuminating Engineering Society (IES).

Keywords: real-time rendering; global illumination; Many-Lights; Virtual Point Lights; photometric data; IES

check for updates

Citation: Gadia, D.; Lombardo, V.; Maggiorini, D.; Natilla, A. Implementing Many-Lights Rendering with IES-Based Lights. *Appl. Sci.* **2024**, *14*, 1022. https://doi.org/10.3390/app14031022

Academic Editor: Jorge Martin-Gutierrez

Received: 20 December 2023
Revised: 19 January 2024
Accepted: 24 January 2024
Published: 25 January 2024

1. Introduction

Global illumination (GI) is one of the most investigated topics in computer graphics. GI takes into account the light reaching a surface directly from the light sources (direct illumination) and also the light reflected or refracted by other surfaces in the scene (indirect illumination). Previously, several approaches were proposed in order to introduce highly realistic simulations of GI in the rendering of a virtual scene.

The rendering process is based on the simulation of the interaction between light and the surfaces in a scene. For a uniform non-emitting surface, the radiance reflected in a point $\mathbf{x}$ is provided by the *rendering equation* [1]:

$$L_o(\mathbf{x}, \omega_o) = \int_\Omega f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i)(\omega_i \cdot \mathbf{n})\, d\,\omega_i \tag{1}$$

where

- $\mathbf{n}$ is the surface normal;
- Ω is the hemisphere centered around $\mathbf{n}$;
- ω_i and ω_o are the directions of incoming and reflected light, respectively;
- $L_i(\mathbf{x}, \omega_o)$ and $L_o(\mathbf{x}, \omega_o)$ are the incoming and the reflected radiances, respectively;
- $f_r(\mathbf{x}, \omega_i, \omega_o)$ is the Bidirectional Reflectance Distribution Function (BRDF), which describes the amount of radiance reflected in the direction ω_o given the radiance coming from direction ω_i.

Every GI technique in the literature represents an approximation, at different levels of accuracy, of Equation (1). Techniques related to Physically Based Rendering (PBR) [2] are usually based on variants of ray tracing approaches and Monte Carlo methods, which can achieve very high levels of photorealism at the expense of high computational costs, thus limiting their full application only to *offline* photorealistic rendering.

In the context of real-time rendering, the inclusion of GI effects is particularly complex due to the time constraints of the rendering process. The standard approaches usually consider a limited set of GI effects (e.g., color bleeding, shadows, and reflections) and/or decouple the computation of GI from the actual main rendering process, which focuses only on the direct illumination component. The most common techniques rely on screen-based algorithms and the precomputation (baking) of shading data in texture maps to be used in the actual rendering process [3]. The recent availability of real-time ray tracing graphics hardware is slowly allowing the introduction of techniques that were limited only to offline rendering. However, the current capabilities of this new generation of graphics cards are still far from supporting a full ray tracing approach, forcing the use of hybrid solutions still based on screen-based techniques, combined with highly sophisticated and effective denoising algorithms [4,5].

With *Many-Lights rendering* [6], we refer to a family of rendering techniques that are based on the placement of several Virtual Point Lights (VPLs) in the scene. The VPLs' positions correspond to the intersection points of rays traced from the light sources and bouncing on the surfaces in the environment. With a similar approach, the indirect illumination is approximated using the direct contribution of all the light sources in the scene (both the actual light sources and VPLs). As reported in [7], this class of techniques enables efficient computation of GI, providing results comparable to path tracing methods but with lower rendering times. Several extensions and improvements were proposed in the literature starting from this original idea, which was also applied successfully in the context of real-time rendering.

In this paper, we analyze the state of the art of Many-Lights techniques for real-time rendering, and we investigate the feasibility of the integration of photometric data in the process of the generation of the VPLs. We base the definition of the light sources and the creation of the VPLs on photometric data provided in the standard IES format proposed by the Illuminating Engineering Society. In particular, our work extends the *Reflective Shadow Maps* technique [8] so that any kind of light source described through a well-formed IES file can be used to illuminate a virtual environment. The technique generates an "enhanced" version of the original Reflective Shadow Map, associated to an IES-based light source. During the rendering step, each pixel of the map can be viewed as a VPL and used to compute its contribution to the overall lighting. The proposed approach, as in the original work [8], focuses on diffuse reflections. Following the standard Many-Lights approach, gathering and normalizing the illumination contributions from N samples offers a reasonable approximation of color bleeding effects in the final rendering.

To our knowledge, there are no other examples of works that integrate a photometric description of light sources into a Many-Lights rendering technique. While IES-based lights are currently supported by several game engines, they are actually used to "weight" only the intensity of the emission component (thus mainly affecting the direct illumination component), and they are not actively considered in the computation of GI effects.

This paper is structured as follows. In Section 2, we provide an overview of GI models based on Many-Lights techniques. An extensive explanation is reserved for the Reflective Shadow Maps [8] technique, which our work is based on. In Section 3, we describe the photometric data that are contained in IES files, as defined by the Illuminating Engineering Society. Section 4 describes our proposed technique and how photometric data can be used to characterize a light source in a real-time rendering pipeline. In Section 5, we present the results of the proposed method on two test scenes. Finally, Section 6 draws conclusions from the results and presents future work.

2. State of the Art on Many-Lights Rendering Techniques

The concept at the basis of Many-Lights techniques is quite straightforward. The rendering process is split into two steps: the first step consists of the generation of a certain amount of VPLs in the scene. This process can follow established approaches for the generation of *light transport paths*, such as in the photon tracing pass in photon mapping [9] or other Monte Carlo methods [2]. In the Many-Lights approach, every time the light ray bounces on a surface in the scene, information related to the intersection point is saved:

position $\mathbf{x}$, normal $\mathbf{n}$, incident direction ω_{i}, incoming radiance $L_{\mathrm{i}}(\mathbf{x}, \omega_{\mathrm{i}})$ (appropriately reduced at each bounce), and bidirectional reflectance distribution function $f_{\mathrm{r}}(\mathbf{x}, \omega_{\mathrm{i}}, \omega_{\mathrm{o}})$. This is the core intuition of Many-Lights techniques: these data are sufficient to compute the irradiance coming from each point of the light path. As a consequence, each point of a light path can be considered as a *virtual* light source, contributing to the overall illumination of the scene. Thus, the Many-Lights approach *redefines* the classical concept of light transport paths used by other GI models [2] and enables dealing with GI without using a different technique for each component.

In the second step, the irradiance in each pixel is approximated by calculating the sum of the direct illumination contributions from all the light sources in the scene, i.e., the "actual" light sources and also the set of generated VPLs. Using as an example the seminal *Instant Radiosity* [10] method, the radiance reflected by a point $\mathbf{x}_1$ towards the camera in $\mathbf{x}_0$ is provided by [7]

$$\langle L(\mathbf{x}_1 \to \mathbf{x}_0) \rangle = \sum_{k=1}^{M} f_{\mathrm{r}}(\mathbf{x}_1) G(\mathbf{x}_1, \mathbf{x}_2^k) \hat{V}(\mathbf{x}_1, \mathbf{x}_2^k) \Phi_{\mathrm{k}} \tag{2}$$

where

- M is the number of generated VPLs
- $\mathbf{x}_2^k$ is the position of the k-th VPL
- $\hat{V}(\mathbf{x}_1, \mathbf{x}_2^k)$ is the visibility term between the k-th VPL and the point we are shading
- $f_{\mathrm{r}}(\mathbf{x_1})$ is the BRDFs of point $\mathbf{x_1}$. We use a simplified notation because only diffuse reflection is considered.
- $G(\mathbf{x}_1, \mathbf{x}_2^k)$ is the geometry term between the k-th VPL and the point we are shading
- Φ_{k} is the "flux" of the k-th VPL (i.e., the component of radiance "emitted" by the k-th VPL). It is calculated as: $\Phi_{\mathrm{k}} = f_{\mathrm{r}}(\mathbf{x}_2^k) L_{\mathrm{i}}(\mathbf{x}_2^k, \omega_{\mathrm{i}}^k)(\omega_{\mathrm{i}}^k \cdot \mathbf{n}^k)$.

Figure 1 shows a schematic description of the two stages of a standard Many-Lights technique.

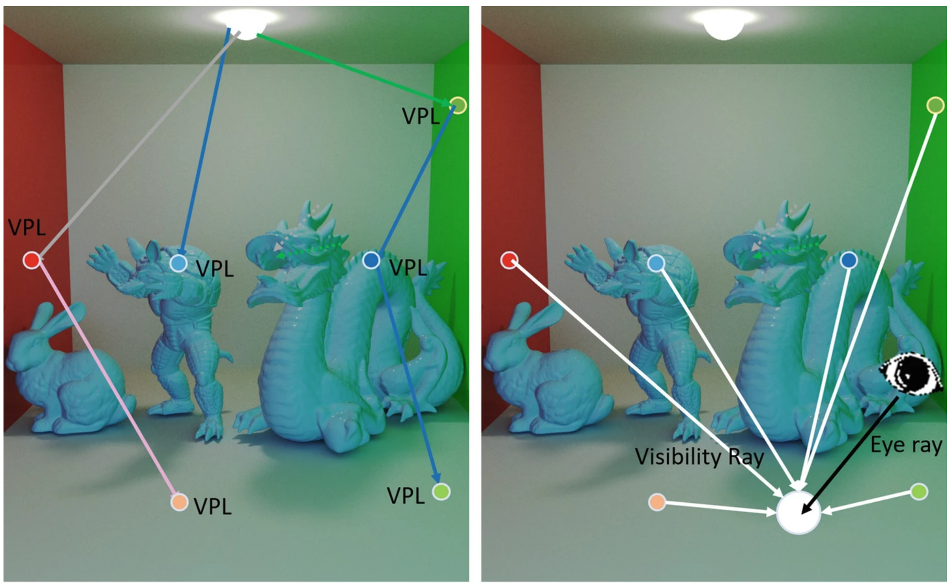


Figure 1. The two steps of a generic Multi-Lights rendering technique. On the (**left**): generation of VPLs on different surfaces of a scene. On the (**right**): rendering process taking into account the effect of the different VPLs. The image considers multiple light bounces, which are applied only in *offline* rendering. Many-Lights techniques applied in real-time rendering (like the proposed method) usually consider just a single light bounce (e.g., as in the case of the object in the center of the scene). Original image from [6].

A wide range of deriving algorithms have been devised starting from this idea, each one focused to address different needs with regard to performance or image quality. As an example, some techniques [11–13] focused on solving bias issues with the original formulation of Instant Radiosity, for example optimizing or redefining the generation process of the VPLs.

Many-Lights techniques scale according to the number of VPLs that are considered. A few light sources can roughly approximate the illumination of a scene, producing biased artifact-free images in a short rendering time, as demonstrated in [14,15]. On the other hand, though, using thousands of VPLs and evaluating them in an efficient and scalable way allows rendering of images with little or no bias. Dachsbacher et al. [7] provide a large overview of scalable algorithms developed in order to solve this challenge. Some examples are *Lightcuts* [16,17] or *Matrix Row–Column sampling* [18].

Moreover, the intrinsic scalability of Many-Lights techniques makes them also often suitable for GPU parallelization. While current GPUs are able to process any number of light sources, Many-Lights techniques in real-time applications are usually applied in a *deferred rendering* [19] pipeline, considering just a single light bounce. Some techniques are focused on the optimization of the *interleaved sampling* [20] approach, which uses disjoint subsets of VPLs to compute illumination for adjacent pixels, making the processing faster by organizing data in GPU-friendly structures [21], and addressing issues that led to aliasing artifacts and problems with glossy materials [22]. Other examples are techniques that shade "hierarchically" according to the level of geometric detail in the processed portion of the scene [23], or subdivide the image in tiles, each having a subset of VPLs employed in the shading process [24,25]. Visibility computation is another field that is researched in order to speed up Many-Lights methods [26]: defining whether or not a shading point can see a VPL can save rendering time. An example is the *Imperfect Shadow Maps* method [15,27].

Reflective Shadow Maps

We provide here a description of the *Reflective Shadow Maps* [8] method, which we have extended in this paper.

The technique is based on the consideration that all one-bounce indirect illumination for a light source is related to the surfaces "visible" via the light source during the generation of a standard shadow map. Thus, each pixel of a light source's shadow map can be considered a virtual light source for the rest of the scene, and it can simulate the bounce of light on that surface. On such a premise, virtual lights generation corresponds to rasterization from the point of view of the light source.

A Reflective Shadow Map (RSM), as described in [8], stores for each pixel p:

- a depth value, d_p, as in a standard shadow map;
- the world space position of the pixel $\mathbf{x}_p$;
- the normal in world space $\mathbf{n}_p$;
- the reflected radiant flux Φ_p, which corresponds to the amount of luminous power leaving a surface.

The technique follows a standard deferred rendering approach, saving each of these data in different dedicated G-buffers in a first render step and finalizing the rendering process in a final second step.

The following equation

$$E_p(\mathbf{x},\mathbf{n}) = \Phi_p \frac{\max(0,\mathbf{n}_p \cdot (\mathbf{x}-\mathbf{x}_p)) \max(0,\mathbf{n}\cdot(\mathbf{x}_p-\mathbf{x}))}{||\mathbf{x}-\mathbf{x}_p||^4} \tag{3}$$

describes how the virtual light source identified by the pixel p in the RSM illuminates the surface point $\mathbf{x}$ having a normal $\mathbf{n}$. Adding up the contributions of each pixel/virtual light allows the computation of the indirect lighting received by the point. However, the generated RSM is moderately large (512 × 512 pixels/virtual lights), and its use would not be compatible with a real-time rendering context. To reduce the number of sums, an *importance sampling* approach is applied in order to concentrate the sampling of the RSM to a subset of relevant lights. The approach is based on the consideration that, the closer a virtual light source p is to the point $\mathbf{x}$, the more it will contribute to its illumination. The distances between two points in image space can be considered a reasonable approximation for distances in world space; thus, the point $\mathbf{x}$ is projected to the image space, and then the texture is sampled using Poisson disk sampling.

The technique as presented in [8] considers only diffuse surfaces, and it is applied to directional lights and spotlights only, which can be treated using standard 2D texture images, due to the strong directionality of these kinds of light sources. The authors suggested that the technique could be extended to consider *omnidirectional* lights like point lights. In our proposed approach, we present an extension of the RSM technique characterized by the use of cube maps to manage the multiple emission directions of a VPL.

3. Photometric Description of Light Sources

Light sources in computer graphics are usually limited to extremely simplified approximations of real light sources. Point lights, directional lights, and spotlights are expressed just using a position in the world space, with no dimension, and with simple types of light emission. The only type of virtual light source that has an actual dimension is the area light. However, area lights are computationally expensive to calculate, thus limiting their actual use only to offline rendering. In real-time applications, illumination data from area lights are usually prebaked in *light maps* and used during the rendering process through texture sampling operations. Another texture-based approach commonly used in real-time rendering to introduce complex illumination features is Image-Based Lighting [28].

Real light sources may have complex characteristics of emission. Indeed, light emitters may have different dimensions, and they may emit light with different intensities at different directions of emission. The IES file format [29], proposed by the Illuminating Engineering Society [30], stores photometric data measured from real light sources (luminaires). IES files are largely used in the lighting design field and in PBR rendering engines [2], where they are used as input in the *importance sampling* process of Monte Carlo methods. In recent years, they have also been used for real-time applications. Examples are the integrations in the Frostbite [31,32] and Unreal [33] game engines. However, IES data are usually applied in game engines only as a sort of "mask" assigned to point lights in order to weight the intensity of light emission around the point light source in a non-uniform way. This process leads to a more realistic rendering of the surfaces close to the light source emission, which are primarily affected by the direct illumination component. However, the role of IES photometric data is usually not considered actively in the other processes related to the computation of GI effects.

In practice, an IES file is an ASCII file, which contains information about a specific luminaire, and measured photometric data. Listing 1 shows the content of an IES file we have used in our test scenes. The content can be divided into three parts:

- *Information about the luminaire product (line 1 to line 10)*: a list of metadata, providing information about the specific luminaire (e.g., model number, manufacturer, destination of use, model of the lamp(s) and ballast, type of mounting, etc.). The list of keywords is not fixed in type, order, or number of elements. Thus, IES files may differ in this section; however, it is not used during the rendering process.
- *Information about the measurement test (lines 11 and 12)*: a list of values related to the measurement test on the actual luminaire. The values in line *11* represent the *number of lamps* inside the luminaire, the *luminous flux* of each lamp, a *multiplication factor* of light intensity, the *number of vertical and horizontal angles* considered in the measurements, the *Photometric Type* of the luminaire, the *unit of measurement* (1 for feet, 2 for meters), and, finally, *length*, *width*, and *height* of the emitting surface of the luminaire. The values in line *12* provide information on the electrical ballast and the input power of the luminaire.

- *Measured photometric data (line 13 to line 19)*: the actual measured data, which consist of the core part of an IES file and the data actually used in the rendering process. The luminaire's photometric data are captured by locating them at the center of an imaginary sphere and measuring light intensity values at some points on the surface of this sphere. These points are expressed using polar coordinates relative to a grid called *photometric web*. The measured samples define a volume that is called *photometric solid*. Lines *13* and *14* show the lists of the vertical and horizontal angles on the photometric web representing the measured samples. The numbers of the considered angles are stated in line *11* (in our example, thirty-seven vertical angles and five horizontal angles). Lines *15* to *19* show the list of luminous intensity values (expressed in candelas) captured at each angle pair (a line for each horizontal angle, each consisting of 37 values).

Listing 1. The content of an IES file.

```
1  IESNA:LM-63-1995
2  [REPORT NUMBER]ITL36346
3  [DATE]11-28-1989
4  [MANUFAC]LEDALITE ARCHITECTURAL PRODUCTS, INC.
5  [CATALOG NUMBER]111621-PN-12HP-NN
6  [LUMINAIRE]EXTRUDED SQUARE ALUMINUM HOUSING, WHITE PAINTED REFLECTORS, SEMI
     -SPECULAR PARABOLIC LOUVER, OPEN TOP.
7  [LAMP]F40T12/CW
8  [BALLAST]PHILIPS HM-140-TPS
9  [MOUNTING]SUSPENDED-DIRECT/INDIRECT
10 TILT=NONE
11 3 3150 1 37 5 1 1 0.4063 4 0.5104
12 1 1 147.5
13 0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100 105 110 115
     120 125 130 135 140 145 150 155 160 165 170 175 180
14 0 22.5 45 67.5 90
15 1734 1738 1704 1651 1574 1480 1384 1254 1108 936 678.8 337.1 69.11 26.11
     13.05 7.68 3.84 3.07 0 23.8 93.68 177.4 264.9 354 436.1 511.4 582.8 646.5
      707.2 764 811.6 847 881.5 907.6 927.6 939.1 936.6
16 1734 1733 1704 1654 1577 1489 1388 1250 1075 892.3 673.4 376.3 109.4 31.1
     16.89 9.98 5.38 4.22 0 31.48 85.62 141.3 210.4 292.9 373.6 474.9 572.4
     593.9 693.4 759.4 806.3 838.9 871.5 896.5 919.9 933.7 936.6
17 1734 1731 1712 1677 1603 1468 1363 1226 1071 903 715.6 500.3 242.6 78.71
     24.19 12.29 6.91 4.61 0 18.81 80.63 135.9 195 261.5 332.5 402.4 474.5
     546.7 629.6 736.4 742.1 809.3 870.8 895.7 910.3 928.7 936.6
18 1734 1724 1711 1685 1571 1471 1359 1222 1077 927.2 761.3 557.9 323.3 156.3
     63.35 17.66 7.68 4.99 0 12.67 63.73 133.6 182.4 247.3 312.9 379.7 452.3
     522.1 591.6 665.4 766.3 805.5 842 896.1 910.7 923.4 936.6
19 1734 1728 1715 1687 1578 1481 1379 1252 1119 978.3 826.2 603.5 358.6 181.2
     76.79 23.04 8.45 5.38 0 10.75 59.89 127.5 179.7 242.6 308.7 371.6 438.5
     511.4 579 648.1 734.1 832.4 814.7 898.4 916.1 926 936.6
20
```

Figure 2 shows an example of visualization of the content of the IES file in Listing 1. The orientation of the luminaire's axes with respect to those of the photometric web is provided by the value of the *Photometric Type* (the sixth value in line *11* of Listing 1). In this paper, we consider the most common case, where Photometric Type corresponds to Type C luminaires (=1 in the IES file format), i.e., architectural and roadway luminaires. For this kind of luminaire, the polar axis of the photometric web coincides with the vertical axis of the luminaire. Finally, notice that, even if the actual luminaire may have multiple lamps (like the luminaire described in Listing 1), the results are expressed considering the light as emitted by a single position in space, thus making the data perfectly compatible with the point light source used in CG. A more comprehensive description of the IES file format can be found in [29].

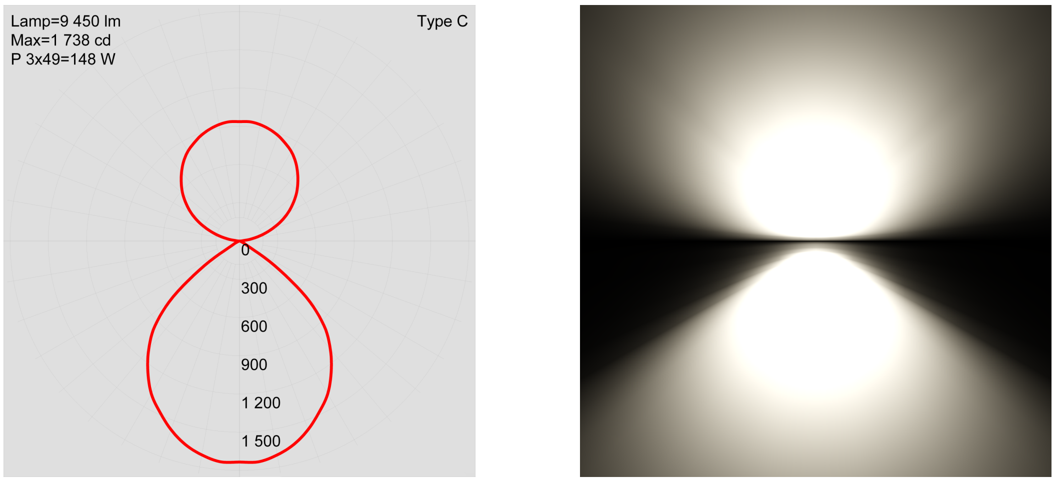


Figure 2. On the (**left**), a visualization of the photometric web of the data in Listing 1; on the (**right**), a rendering produced using the same IES file. Both images have been generated using IESviewer 3.6 software [34].

4. Method

The main goal of the paper is to investigate the feasibility of the introduction of photometric-based light sources in a real-time Many-Lights technique, thus considering the management and the effect of a non-uniform emission in the computation of both direct and indirect illumination components.

In this section, we describe the main contributions of the paper. In particular:

- we implemented a parser for the IES file format, which creates a polygon mesh representing the photometric solid of a luminaire. This mesh is then used in the rendering process to influence the direct and indirect illumination components;
- we extended the Reflective Shadow Maps technique [8] using cube maps in order to address the omnidirectional emission of point lights. The original technique as proposed in [8] considers only directional lights, using a standard 2D texture image to store the RSM data;
- we weighted the emission from the VPLs on the basis of the parsed data from the IES files. This affects not only the direct illumination component (leading to a more realistic rendering of the area and surfaces closer to the light source) but also the indirect illumination component because the placement of the VPLs is tuned by the actual volume of emission, which is no longer perfectly spherical but described by the data from a real luminaire (Figure 2).

4.1. Implementation Details

The project was implemented in C++20 using OpenGL API and GLSL shading language on a Windows machine equipped with Intel$^{®}$ Core i7-8750H CPU @ 2.20 GHz, 16.0 GB RAM, and a NVIDIA GeForce GTX 1050 Ti graphics card.

We have considered two test scenes:

- a *modified Cornell box* scene containing the *Stanford Lucy* model. The scene, which consists of 33,480 triangles, uses only simple diffuse materials without any texturing. Both the original models (downloaded from [35,36], respectively) were resized by the authors before use.
- the *Crytek Sponza*, which consists of 262,267 triangles. The scene considers only diffuse surfaces and also presents several textures. The original file (downloaded from [35]) was resized by the authors before use.

4.2. Applying an IES Light in a Virtual Scene

The first step for the introduction of IES data into a real-time rendering pipeline is the parsing of the content of an IES file. This is a trivial passage because IES format is

ASCII-based. The most relevant data that are required are the list of horizontal and vertical angles, the luminous intensities, and the photometric type of the luminaire.

Once the data have been parsed, a polygon mesh representing the photometric solid described in the IES file is built. The entire process is based on established mesh processing techniques [37]. Each angle pair (i.e., a horizontal and vertical angle describing a direction of sampling of the luminaire) expressed in the coordinate system of the photometric web, together with the corresponding measured intensity, is converted to a 3D Cartesian coordinate in space using the luminaire position as reference. The grid of points in the IES file usually represents only a portion of the photometric solid. Given the orientation of the luminaire (determined by the photometric type metadata), and exploiting symmetries, a complete solid can be obtained applying *reflection* operations. Triangulation and possibly subdivision operations are then applied in order to create the final polygon mesh of the photometric solid [37]. The last step consists of the computation of normals for every vertex of the generated mesh. Finally, all the data of the polygon mesh are saved in an OpenGL Vertex Array Object (VAO) [38].

The polygon mesh of the photometric solid is then used in the rendering module in order to extend the Reflective Shadow Maps technique. It can also be easily visualized using wireframe rendering for debug purposes (see Figures 3 and 4d).

4.3. Enhancing Reflective Shadow Maps with IES Data

We have extended the Reflective Shadow Maps technique [8], which we have presented in Section 2, in two ways.

First of all, we have extended the original technique in order to also consider point lights. The main approach remains the same: the rendering is based on deferred rendering, saving the same data (depth, position, normal, and flux) in different G-buffers. However, to manage the omnidirectional emission of a point light, we use cube maps instead of standard 2D textures. Cube maps are generated performing six different rendering steps, pointing the light sources in the front, back, up, down, left, and right directions. This way, we have a panoramic determination of what is "seen" by the light in every direction [3]. In OpenGL, the process of rendering to each face of the cube map can be completed in a single render pass using an ad hoc geometry shader [38]. Figure 5a–c show examples of the content of the cube maps version of the data required by the RSM technique.

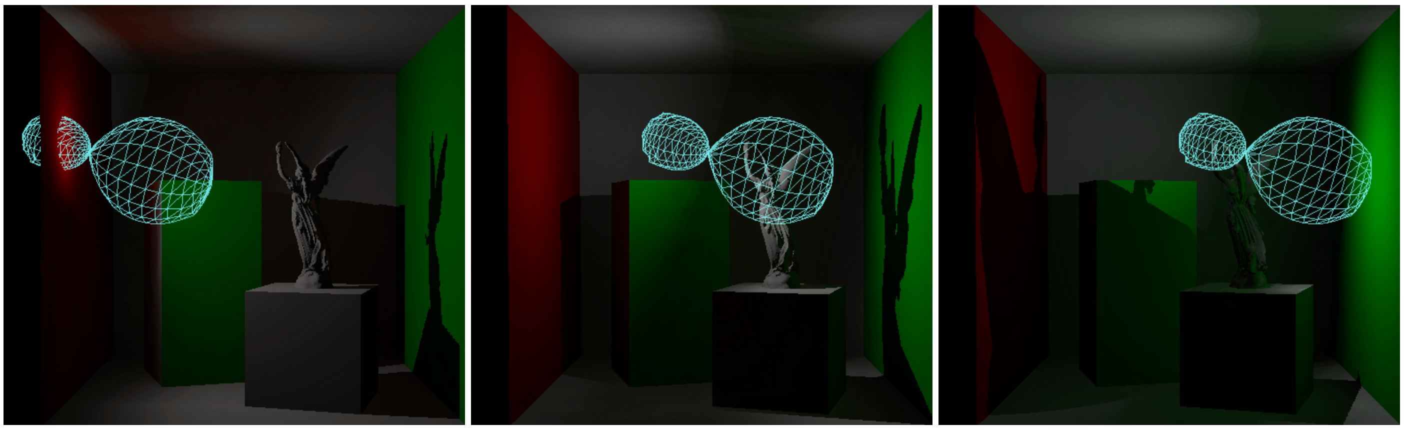


Figure 3. Some rendering results of the *modified Cornell box* using the proposed method. The scene presents a light source (described by the IES file in Listing 1 and Figure 2), moving from left to right. The photometric solid is rendered in wireframe. The results show the effect of the non-uniform emission of the light source (particularly evident on the ceiling in the image in the center) and the color bleeding effects provided by the Many-Lights approach, in particular on the solid on the left in all the frames and on the *Stanford Lucy* model in the right image.

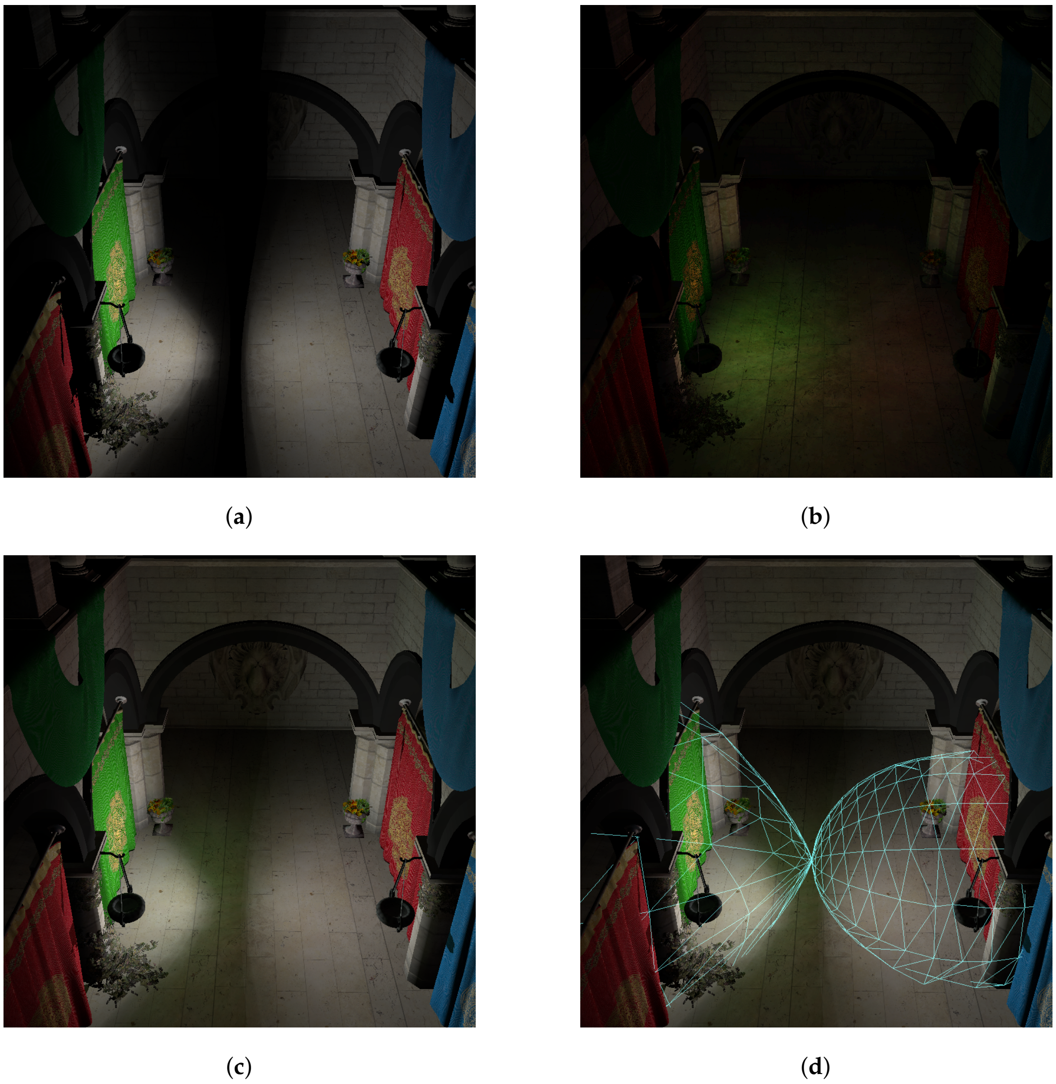


Figure 4. A render using the *Crytek Sponza* scene. The subfigures are (**a**) the scene rendered only with direct lighting using the IES light mask; (**b**) the scene illuminated only by indirect lighting (a 4× multiplication factor has been applied to make the component more visible); (**c**) the final render of the scene using the proposed method; (**d**) the final render with debug visualization of the photometric solid. The scene uses the light source described by the IES file in Listing 1 and Figure 2.

Standard point lights emit light in a uniform way in every direction (i.e., the volume of emission is perfectly spherical). With IES data, we want to "weight" the emission in each direction given the value provided by the photometric solid. In order to obtain such an effect, we can create an IES *light mask* starting from the photometric solid. In practice, we convert the value of the IES data, currently stored in the polygon mesh of the photometric solid, into a cube map whose values will represent how and how much light should be emitted from the light source towards each direction.

The photometric solid is positioned where the point light is placed, and oriented accordingly. Computing the associated light mask is completed through projecting its faces to the cube map. We can re-use the first part of the G-buffers creation step because the vertices of the polygon solid mesh need to be processed in the same way. As a consequence, we apply the same vertex and geometry shaders applied for the other cube maps of the RSM technique.

However, in order to achieve the desired result, we apply a fragment shader that saves the IES light mask information using a *false colors* approach. We present in Listing 2 an excerpt of the GLSL code relative to the final fragment color computation.

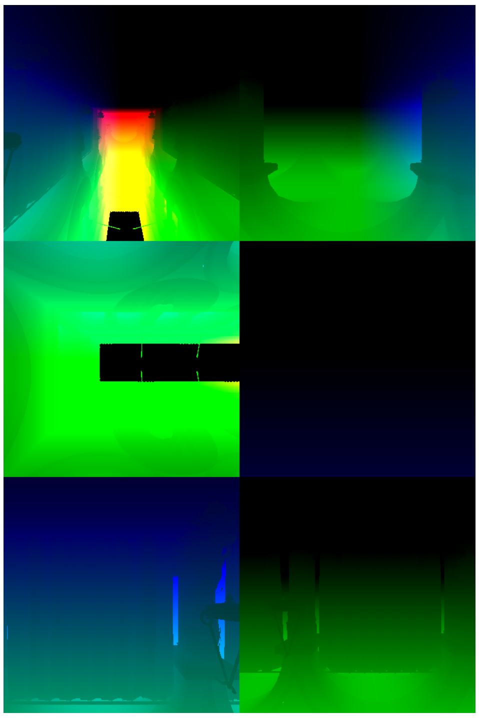
Listing 2. IES light mask generation: *false colors* computation in fragment shader.

```
vec3 light_to_fragment = fragment_position.xyz - scene_lights.positions[
  light_index].xyz;
float distance_from_light = length(light_to_fragment);

float scaled_distance = distance_from_light /
  distances_to_furthest_ies_vertex[light_index];
float emitting_along_direction_l = 1.0;

vec4 ies_mask = vec4(distance_from_light, scaled_distance,
  emitting_along_direction_l, 1.0);

```

Thus, the generated texture stores the following data:

- *Red* color channel: we store the length of the vector that goes from the position of the light to the processed fragment, i.e., the luminous intensity emitted from the light source in that direction.
- *Green* color channel: `distance_to_furthest_ies_vertex` is the original maximum distance between the origin and a vertex of the photometric solid. Dividing the distance of the fragment from the light by this value, we compute an intensity modifier based on the maximum intensity that the light source can emit in candelas. This will compensate scaling transformations applied to the original photometric solid once transformed to be used in the scene. The resulting value is always contained in the range $(0.0, 1.0]$.
- *Blue* color channel: since a fragment was generated, the solid emits light in that direction, so the variable `emitting_along_direction_l` is set to 1.0. Because the G-buffer is initialized with 0.0, the final cube map will have non-zero values only for fragments corresponding to actual direction of emission from the light source.

Figure 5d shows an example of the content of the IES light mask.


(**a**) Position map

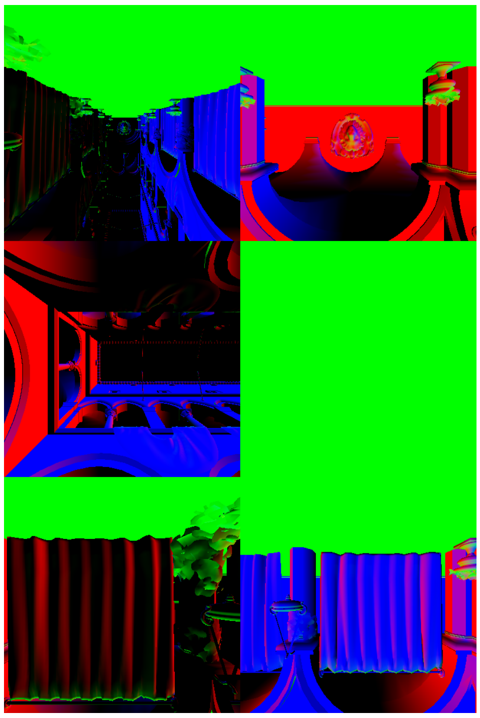
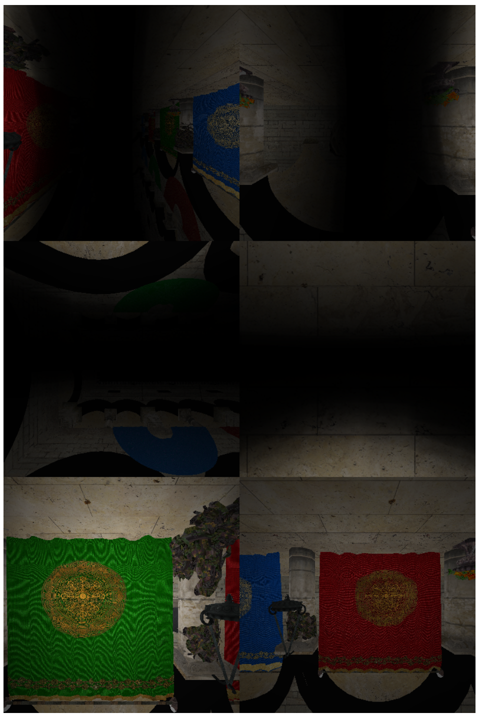
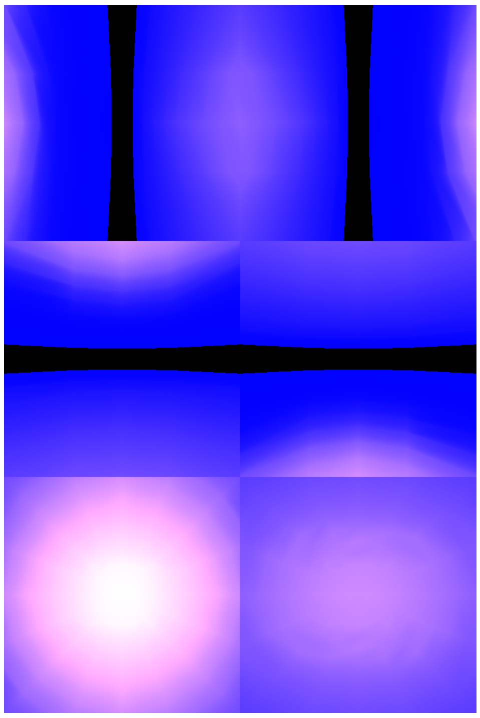

(**b**) Normal map

Figure 5. *Cont.*



(**c**) Flux map



(**d**) IES light mask

Figure 5. Reflective Shadow Map components and IES light mask for the light source in Listing 1 in the *Crytek Sponza* scene. Figure 4c shows the final render using these components.

The final rendering step must blend the direct illumination component and the indirect component obtained through the enhanced RSM technique. The computed IES light mask is applied in the computation of both the components:

- *direct illumination component*: for each light source present in the scene, the G-buffer storing the standard shadow map is checked to see if the considered fragment is illuminated. If not in shadow, the direct illumination component is computed following the standard screenspace approach of deferred rendering, but the resulting value is weighted by the values sampled in the IES light mask. The blue channel, corresponding to a boolean flag, tells the shader whether the light source emits light in that direction. If it does not, the contribution of that light is set to zero. On the other hand, if the contribution exists, the value in the red channel (possibly corrected by the component in the green channel) can be used as a multiplier for the intensity. An example of the rendering produced considering only this component is a scene that can be seen in Figure 4a.
- *indirect illlumination component*: the computation is based on the application of Equation (3) as discussed in Section 2 but computed using G-buffers consisting of cube maps, and weighting the final result using the values sampled in the IES light mask to take into account the light intensity actually emitted in a direction. Figure 4b shows a preview of the indirect illumination component created using our enhanced RSM technique. A multiplication factor of 4 is used for a better visualization.

Figure 4c shows the final render of the proposed technique.

5. Results and Discussion

Figures 3, 4c, 6 and 7 show some renderings obtained using the proposed approach on the two test scenes. The results seem to confirm the initial hypothesis: the original RSM technique can be extended to consider point lights, and the photometric data described in an IES file can be effectively integrated in the Many-Lights pipeline, not only in the direct illumination component (as usually considered in real-time applications), also playing an active role in the indirect illumination computation.

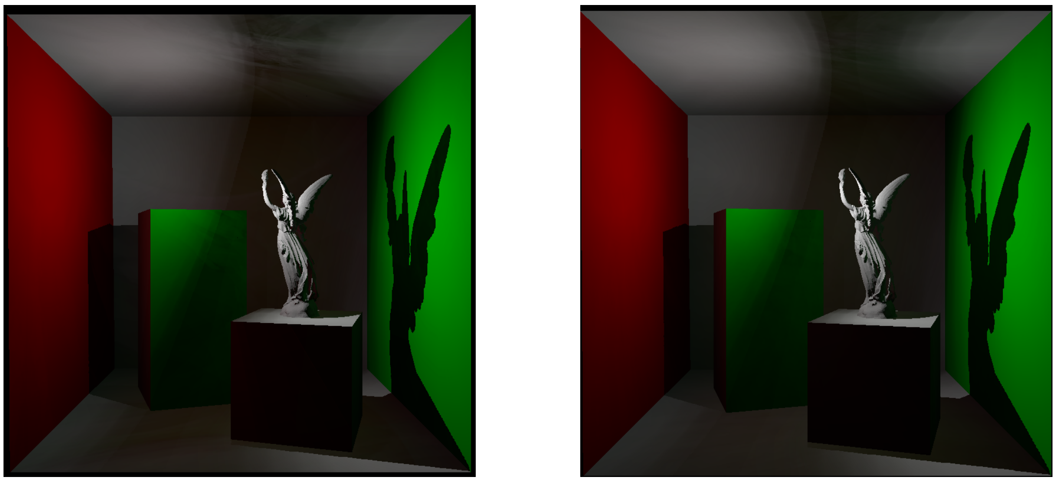


Figure 6. Rendering results of the *modified Cornell box* using 100 (**left**) and 400 (**right**) samples per fragment. In the image on the left, noticeable rendering artifacts are present on the ceiling and on the front face of the taller box. The scene uses the light source described in Listing 1.

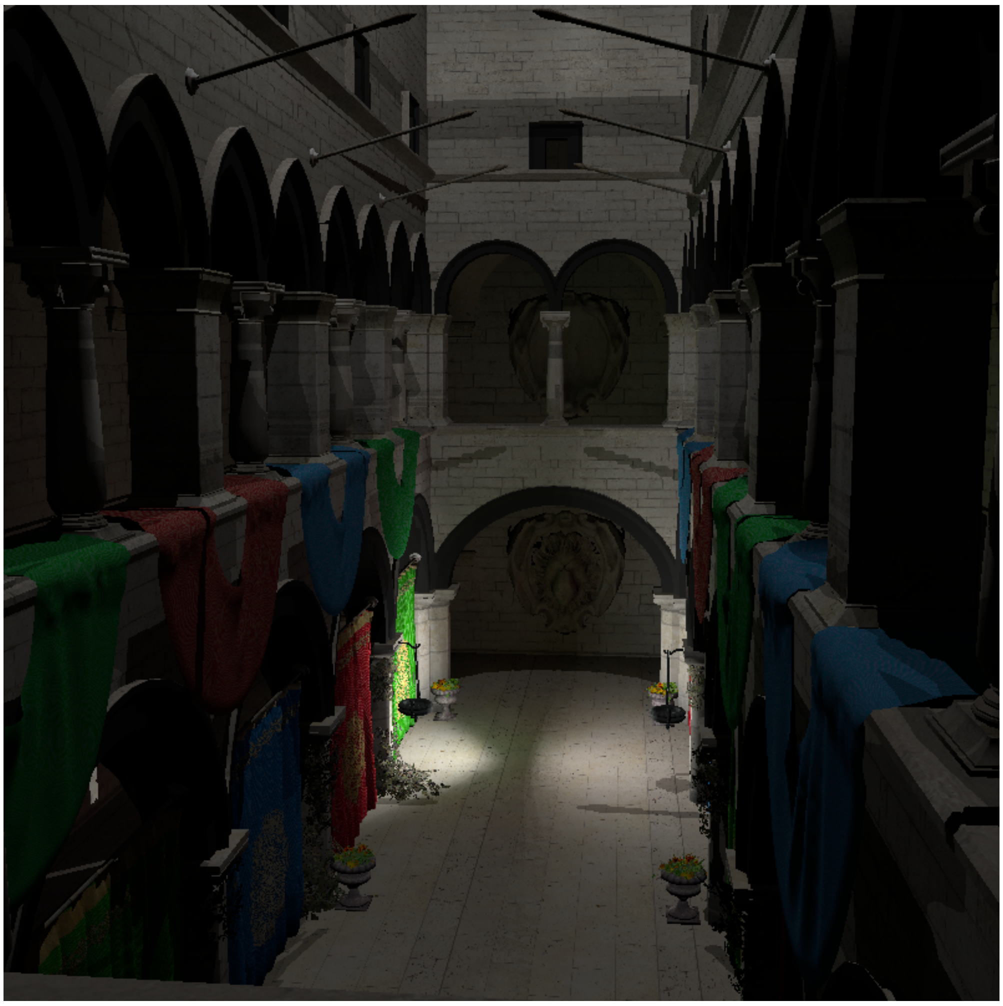


Figure 7. A render of the *Crytek Sponza* scene using the proposed technique. The scene uses the light source described in Listing 1. As in Figure 4c, the color bleeding effect is evident on the floor.

The two test scenes present only diffuse surfaces; as a consequence, the main GI effect introduced by the Many-Lights technique is color bleeding. The presented results show an evident and realistic color bleeding effect: this is particularly noticeable on the solids and the *Stanford Lucy* model in Figures 3 and 6, and on the floor in Figures 4c and 7.

Regarding the computational performance, the implemented technique is more expensive than the original RSM method. This is obviously caused by the use of several cube maps for each light source in the scene, while the original method used standard 2D textures. As a consequence, our method requires six times more memory to store each RSM component, and an additional cube map for the IES mask. Moreover, all the cube maps require recomputation in the case of dynamic objects moving in the scene. However, the technique is view-independent in the case of static scenes: moving the camera does not require recomputing either the RSMs or the IES masks for light sources; it only requires an update of the camera G-buffers, and thus of what is rendered to the screen in the last pass of a standard deferred rendering pipeline.

Given N IES lights and taking M samples from their RSMs, $N \times M$ texture random accesses are needed to render a single fragment. In our experimental evaluation of the proposed technique, we found that the best visual results are obtained with 400 samples taken in each RSM for each fragment. Fewer samples lead to evident visual artifacts. Figure 6 shows a comparison of the results obtained on the *modified Cornell box* scene using 100 and 400 samples per fragment. When using fewer samples, rendering artifacts are visible both on the ceiling and on the front face of the taller box.

Performances in terms of achieved framerate on our test machine (Section 4.1) when setting the rendering resolution to 800×800 pixels are summarized in Tables 1 and 2. Analysis has been performed on both test scenes considering a single light source and 100 and 400 samples per fragment. The application maintains an interactive framerate with a maximum of 4 lights in the scene. We forced a full recomputation of the RSM buffers at each frame, simulating a dynamic scene. By using *NVIDIA Nsight* [39] to debug the state of the GPU, we discovered that computing indirect illumination is the most computationally expensive step. Indeed, repeatedly sampling the RSM cube map is an actual bottleneck.

The application of an additional screenspace interpolation approach, as proposed in [8], could also be extremely advantageous in the proposed version of the RSM technique, offering relevant speedups while preserving a good final image quality.

Table 1. Framerates for the *modified Cornell box* scene with respect to the number of samples per fragment used in the indirect lighting computation.

Number of Samples	Min FPS	Max FPS	Average FPS
400	40	55	49
100	140	200	170

Table 2. Framerates for the *Crytek Sponza* scene with respect to the number of samples per fragment used in the indirect lighting computation.

Number of Samples	Min FPS	Max FPS	Average FPS
400	42	52	47
100	120	184	152

6. Conclusions and Future Work

In this paper, we introduced the use of photometric data in a Many-Lights technique so that a Virtual Point Lights distribution could be driven by the IES description of the light source. We proposed two extensions to the Reflective Shadow Maps (RSMs) technique while considering point lights and creating a *light mask* based on the data in IES files in order to “weight” the emission in different directions.

The results obtained with the proposed technique show how the introduction of photometric data describing real light sources can noticeably contribute to the level of realism in the generated images. The current state of the project allows for an adequate framerate with a limited number of light sources in the scene on the considered test machine.

Future work will focus on the optimization of the proposed technique. In the current implementation, we have not considered the optimization based on screen space interpolation, as also outlined in the original RSMs paper [8]. The introduction of a similar approach could surely enhance the performance of the overall rendering pipeline. Moreover, other possible approaches might involve encoding useful data in a more compact and efficient format, such as merging depth maps and IES light masks in a single cube map by saving more information in each component. The use of IES photometric data can also be considered in other Many-Lights techniques. Indeed, a different approach was proposed in [40]: instead of gathering energy from pixel lights on an ad hoc shadow map, energy might be "propagated" from the indirect light source, illuminating screenspace neighbors. This step can benefit from a description of actual photometric emission of a light source.

Author Contributions: Conceptualization, D.G. and A.N.; methodology, D.G. and A.N.; software, A.N.; validation, V.L. and A.N.; writing—original draft preparation, D.G. and V.L.; writing—review and editing, D.G., V.L. and D.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable

Informed Consent Statement: Not applicable

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kajiya, J.T. The Rendering Equation. In Proceedings of the SIGGRAPH '86: 13th Annual Conference on Computer Graphics and Interactive Techniques, Dallas, TX, USA, 18–22 August 1986; Association for Computing Machinery: New York, NY, USA, 1986; pp. 143–150. [CrossRef]
2. Pharr, M.; Jakob, W.; Humphreys, G. *Physically Based Rendering: From Theory to Implementation*, 4th ed.; The MIT Press: Cambridge, MA, USA, 2023.
3. Akenine-Möller, T.; Haines, E.; Hoffman, N.; Pesce, A.; Iwanicki, M.; Hillaire, S. *Real-Time Rendering*, 4th ed.; AK Peters/CRC Press: Boca Raton, FL, USA, 2018; p. 1200.
4. Haines, E.; Akenine-Möller, T. (Eds.) *Ray Tracing Gems*; Apress: New York, NY, USA, 2019.
5. Marrs, A.; Shirley, P.; Wald, I. (Eds.) *Ray Tracing Gems II*; Apress: New York, NY, USA, 2021.
6. Wang, T. High-Performance Many-Light Rendering. In *Encyclopedia of Computer Graphics and Games*; Springer International Publishing: Cham, Switzerland, 2020; pp. 1–6. [CrossRef]
7. Dachsbacher, C.; Křivánek, J.; Hašan, M.; Arbree, A.; Walter, B.; Novák, J. Scalable Realistic Rendering with Many-Light Methods. *Comput. Graph. Forum* **2014**, *33*, 88–104. [CrossRef]
8. Dachsbacher, C.; Stamminger, M. Reflective Shadow Maps. In Proceedings of the I3D '05: 2005 Symposium on Interactive 3D Graphics and Games, Washington, DC, USA, 3–6 April 2005; Association for Computing Machinery: New York, NY, USA, 2005; pp. 203–231. [CrossRef]
9. Jensen, H.W. *Realistic Image Synthesis Using Photon Mapping*; A. K. Peters Ltd.: Wellesley, MA, USA, 2001.
10. Keller, A. Instant Radiosity. In Proceedings of the SIGGRAPH '97: 24th Annual Conference on Computer Graphics and Interactive Techniques, Los Angeles, CA, USA, 3–8 August 1997; pp. 49–56. [CrossRef]
11. Hašan, M.; Křivánek, J.; Walter, B.; Bala, K. Virtual Spherical Lights for Many-Light Rendering of Glossy Scenes. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 Papers*; Association for Computing Machinery: New York, NY, USA, 2009. [CrossRef]
12. Hedman, P.; Karras, T.; Lehtinen, J. Sequential Monte Carlo Instant Radiosity. In Proceedings of the I3D '16: 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, Redmond, WA, USA, 26–28 February 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 121–128. [CrossRef]
13. Yuksel, C. Stochastic Lightcuts for Sampling Many Lights. *IEEE Trans. Vis. Comput. Graph.* **2021**, *27*, 4049–4059. [CrossRef] [PubMed]
14. Tabellion, E.; Lamorlette, A. An Approximate Global Illumination System for Computer Generated Films. *ACM Trans. Graph.* **2004**, *23*, 469–476. [CrossRef]

15. Ritschel, T.; Eisemann, E.; Ha, I.; Kim, J.D.K.; Seidel, H.P. Making Imperfect Shadow Maps View-Adaptive: High-Quality Global Illumination in Large Dynamic Scenes. *Comput. Graph. Forum* **2011**, *30*, 2258–2269. [CrossRef]
16. Walter, B.; Fernandez, S.; Arbree, A.; Bala, K.; Donikian, M.; Greenberg, D.P. Lightcuts: A Scalable Approach to Illumination. *ACM Trans. Graph.* **2005**, *24*, 1098–1107. [CrossRef]
17. Shi, X.; Wang, L.; Wu, J.; Fan, R.; Hao, A. Foveated Stochastic Lightcuts. *IEEE Trans. Vis. Comput. Graph.* **2022**, *28*, 3684–3693. [CrossRef] [PubMed]
18. Hašan, M.; Pellacini, F.; Bala, K. Matrix Row-Column Sampling for the Many-Light Problem. *ACM Trans. Graph.* **2007**, *26*, 26-es. [CrossRef]
19. Saito, T.; Takahashi, T. Comprehensible Rendering of 3-D Shapes. In Proceedings of the SIGGRAPH '90: 17th Annual Conference on Computer Graphics and Interactive Techniques, Dallas, TX, USA, 6–10 August 1990; Association for Computing Machinery: New York, NY, USA, 1990; pp. 197–206. [CrossRef]
20. Keller, A.; Heidrich, W. Interleaved Sampling. In *Eurographics Workshop on Rendering*; Gortle, S.J., Myszkowski, K., Eds.; The Eurographics Association: Limassol, Cyprus, 2001. [CrossRef]
21. Segovia, B.; Iehl, J.C.; Mitanchey, R.; Péroche, B. Non-Interleaved Deferred Shading of Interleaved Sample Patterns. In Proceedings of the GH '06: 21st ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics Hardware, Vienna, Austria, 3–4 September 2006; Association for Computing Machinery: New York, NY, USA, 2006; pp. 53–60. [CrossRef]
22. Segovia, B.; Wald, I. *Screen Space Spherical Harmonics Filters for Instant Global Illumination*; Technical Report; Intel Corporation Research: Santa Clara, CA, USA, 2010.
23. Nichols, G.; Wyman, C. Multiresolution Splatting for Indirect Illumination. In Proceedings of the I3D '09: 2009 Symposium on Interactive 3D Graphics and Games, Boston, MA, USA, 27 February–1 March 2009; Association for Computing Machinery: New York, NY, USA, 2009; pp. 83–90. [CrossRef]
24. Olsson, O.; Assarsson, U. Tiled Shading. *J. Graph. GPU Game Tools* **2011**, *15*, 235–251. [CrossRef]
25. Nabata, K.; Iwasaki, K. Adaptive Irradiance Sampling for Many-Light Rendering of Subsurface Scattering. *IEEE Trans. Vis. Comput. Graph.* **2022**, *28*, 3324–3335. [CrossRef] [PubMed]
26. Noh, G.; Choi, H.; Moon, B. Enhanced Direct Lighting Using Visibility-Aware Light Sampling. In *Advances in Computer Graphics*; Sheng, B., Bi, L., Kim, J., Magnenat-Thalmann, N., Thalmann, D., Eds.; Springer: Cham, Switzerland, 2024; pp. 187–198.
27. Ritschel, T.; Grosch, T.; Kim, M.H.; Seidel, H.P.; Dachsbacher, C.; Kautz, J. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Trans. Graph.* **2008**, *27*, 129. [CrossRef]
28. Debevec, P. Image-Based Lighting. In Proceedings of the SIGGRAPH '06: ACM SIGGRAPH 2006 Courses, Boston, MA, USA, 30 July–3 August 2006; Association for Computing Machinery: New York, NY, USA, 2006; p. 4-es. [CrossRef]
29. IES File Format. Available online: https://blog.ansi.org/2020/02/standard-file-photometric-data-ies-lm-63-19/ (accessed on 20 December 2023).
30. Illuminating Engineering Society Homepage. Available online: https://www.ies.org/ (accessed on 20 December 2023).
31. Lagarde, S.; Langlands, A. Moving Frostbite to PBR. In *Physically Based Shading in Theory and Practice*; ACM SIGGRAPH 2014 Courses; Association for Computing Machinery: New York, NY, USA, 2014.
32. Lagarde, S. Moving Frostbite to Physically Based Rendering 3.0. Available online: https://seblagarde.files.wordpress.com/2015/07/course_notes_moving_frostbite_to_pbr_v32.pdf (accessed on 20 December 2023).
33. Unreal Engine Documentation: IES Light Profiles. Available online: https://docs.unrealengine.com/5.0/en-US/using-ies-light-profiles-in-unreal-engine/ (accessed on 20 December 2023).
34. IESviewer Homepage. Available online: http://photometricviewer.com/ (accessed on 20 December 2023).
35. McGuire, M. Computer Graphics Archive. Available online: https://casual-effects.com/data (accessed on 20 December 2023).
36. Jacobson, A. Common 3D Test Models Repository. Available online: https://github.com/alecjacobson/common-3d-test-models (accessed on 20 December 2023).
37. Botsch, M.; Kobbelt, L.; Pauly, M.; Alliez, P.; Levy, B. *Polygon Mesh Processing*; CRC Press: Boca Raton, FL, USA, 2010.
38. Kessenich, J.; Sellers, G.; Shreiner, D. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.5 with SPIR-V*, 9th ed.; Addison-Wesley Professional: Boston, MA, USA, 2016.
39. NVIDIA Nsight Graphics. Available online: https://developer.nvidia.com/nsight-graphics (accessed on 20 December 2023).
40. Dachsbacher, C.; Stamminger, M. Splatting Indirect Illumination. In Proceedings of the I3D '06: 2006 Symposium on Interactive 3D Graphics and Games, Redwood City, CA, USA, 14–17 March 2006; Association for Computing Machinery: New York, NY, USA, 2006; pp. 93–100. [CrossRef]